Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ЕКСПЕРТНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИБОРУ СПЕЦІАЛЬНОСТЕЙ В ЗАКЛАДАХ ВИЩОЇ ОСВІТИ УКРАЇНИ

Виконав: студент II курсу
групи 2ПІ-22м спеціальності
121 – Інженерія програмного забезпечення

Кубай М. О.

Керівник: к.т.н., доц. каф. ПЗ Катєльніков Д. І.

«18» грудня 2023 р.

Опонент: к.т.н., доц. каф. Колесник І. С.

«18» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«18» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти другий (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«19» вересня 2023 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кубаю Максиму Олексійовичу

1. Тема роботи – «Розробка методів та програмних засобів експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України.»

Керівник роботи: Катєльніков Денис Іванович, к. т. н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи : 9 грудня 2023 р.

3. Вихідні дані: інформація про спеціальності з бази ЄДЕБО; методи статистичного аналізу; методи розв'язання багатокритеріальних задач оптимізації; середовище розробки – Visual Studio 2022; мова розробки – C#; веб-фреймворк – ASP.NET.

4.Зміст текстової частини: вступ; аналіз стану розробки інформаційної системи та задач дослідження; розробка методів системи; розробка основних модулів системи; тестування системи; економічна частина; висновки; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): мета і задачі роботи; наукова новизна та практична цінність роботи; загальний алгоритм системи рекомендації спеціальностей; алгоритм методу

експертної рекомендації спеціальностей; алгоритм методу статистичного аналізу результатів рекомендації спеціальностей; основні результати роботи.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завдання прийняв
1–4	Катєльніков Д. І., доц. каф. ПЗ, к.т.н	19.09.2023	05.12.2023
5	Кавецький В. В., доц. каф. ЕПВМ, к.е.н.	17.11.2023	25.11.2023

7. Дата видачі завдання 19 вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач	20.09.23 – 02.10.23	вик
2	Розробка архітектури та алгоритмів програмного продукту	03.10.23 – 23.10.23	вик
3	Аналіз і вибір мови програмування та середовища розробки	16.10.23 – 23.10.23	вик
4	Розробка програмного продукту	24.10.23 – 13.11.23	вик
5	Тестування програми	14.11.23 – 21.11.23	вик
6	Розробка економічної частини	17.11.23 – 25.11.23	вик
7	Оформлення матеріалів до захисту МКР	22.11.23 – 01.12.23	вик

Студент _____ **Кубай М. О.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Катєльніков Д. І.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кубай М. О. Розробка методів і програмних засобів експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти україни: магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 95 с.

На укр. мові. Бібліогр. : 35 назв ; рис. : 20; табл. 13.

У магістерській кваліфікаційній роботі проведено аналіз сервісів для експертної рекомендації спеціальностей. Було проведено аналіз аналогів, визначено переваги та недоліки кожного.

Розроблено метод експертної рекомендації спеціальностей на основі балів ЗНО/НМТ з врахуванням побажань учня чи студента, популярності та актуальності конкретних спеціальностей і за допомогою використання минулих результатів прогнозування.

Виконано обґрунтування мови програмування та середовища розробки, які були використані. Розроблено загальний алгоритм системи. Розроблено програмний додаток з веб-інтерфейсом. Проведено тестування системи та її окремих модулів.

Робота реалізована у вигляді програмного додатку з веб-інтерфейсом за допомогою мови програмування C# та веб-платформи ASP.NET.

Ключові слова: учень, студент, спеціальність, прогнозування, заклади вищої освіти.

ABSTRACT

Kubai M. O. Development of methods and software tools of an expert recommendation system for the selection of specialties in higher education institutions of Ukraine: master's qualification thesis on specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 95 p.

In Ukrainian speech Bibliogr. : 35 titles; Fig. : 20; table 13.

In the master's qualification work, an analysis of services for expert recommendation of specialties was carried out. An analysis of analogues was carried out, the advantages and disadvantages of each were determined.

A method of expert recommendation of majors based on the scores of ZNO/NMT has been developed, taking into account the wishes of the pupil or student, the popularity and relevance of specific majors and using past forecasting results.

A justification of the programming language and development environment that were used is done. A general system algorithm has been developed. A software application with a web interface has been developed. The system and its individual modules have been tested.

The work is implemented in the form of a software application with a web interface using the C# programming language and the ASP.NET web platform.

Key words: pupil, student, specialty, forecasting, institutions of higher education.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У житті кожного із нас виникає немало моментів, коли нам доводиться приймати важливе рішення, що визначатиме подальший розвиток. Одним із таких моментів є вибір майбутньої професії ще під час навчання у школі. Слід ставитися до цього вибору як до початку нового захоплюючого етапу життя, а не як до обов'язковості.

Той, хто обирає свій професійний шлях, повинен ретельно вивчити різноманітність сучасних професій, розуміти їх сутність і об'єктивно оцінювати свої можливості, здібності та інтереси. Неправильний вибір професії у молодому віці може призвести до проблем як для самої особи, так і для суспільства в цілому.

Хоча можна змінити сферу професійної діяльності в майбутньому, це виявляється складним кроком. Зміна професії часто викликає хворобливі переживання, сумніви, а також втрату часу і енергії. Однак при правильному початковому виборі людина навчається і працює з насолодою, не відчуваючи втоми чи втрати часу. Тоді професійний успіх гарантовано [1].

Допомога абітурієнтам у визначенні найвідповіднішої спеціальності є важливим завданням, особливо в сучасному світі, де це може визначити не лише майбутню кар'єру, але й задоволеність студента життям. Сучасний ринок праці постійно змінюється, і деякі напрямки можуть бути більш вигідними з точки зору працевлаштування та заробітку. Визначення цих тенденцій може суттєво сприяти успішному кар'єрному шляху. Вибір вірної спеціальності допомагає студентам більше зосередитися на навчанні, оскільки вони будуть мотивовані та зацікавлені у предметах, пов'язаних із їхньою майбутньою професією. Заплутаність у великій кількості доступних спеціальностей може бути перешкодою для абітурієнтів.

Це робить актуальною задачу розробки експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України, яка допоможе широким верствам абітурієнтів забезпечити оптимальний вибір

майбутньої спеціальності з врахуванням їх індивідуальних інтересів, навичок та сильних і слабких сторін, що дозволить знизити ризик помилки.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення [2].

Мета та завдання дослідження. Метою роботи є підвищення рівня задоволеності та успішності студентів за рахунок оптимального вибору спеціальності шляхом використанням експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України.

Основними задачами дослідження є:

– розробка методу експертної рекомендації спеціальностей на основі власних побажань користувача, балів з документів про освіту та психологічних тестів;
– розробка методу статистичного аналізу результатів рекомендацій для покращення подальшої роботи системи;
– розробка програмного додатку з веб-інтерфейсом для користування системою;
– проведення тестування системи.

Об'єктом дослідження є процес вибору абітурієнтом оптимальної, з його точки зору, спеціальності.

Предмет дослідження – методи та засоби розробки експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України.

Методи дослідження. У процесі досліджень використовувались: теорія баз даних при організації зберігання інформації; методи багатокритеріальної оптимізації та методи прогнозування для отримання оптимальної рекомендації; теорія розподілених систем для побудови веб-сервісу експертної рекомендаційної системи; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів:

1. Подальшого розвитку отримав метод експертної рекомендації спеціальностей, в якому, на відміну від існуючого, додатково враховуються результати психологічних тестів, що дозволяє оптимізувати вибір спеціальності, використовуючи інформацію про особисті вподобання користувача.
2. Подальшого розвитку отримав метод статистичного аналізу результатів рекомендацій спеціальностей, в якому, на відміну від існуючого, враховуються зміни у персональних даних користувачів, що дозволяє накопичувати інформацію в моделі прийняття рішень і, таким чином, поступово підвищувати точність рекомендацій.

Практична цінність отриманих результатів. Практична цінність полягає в тому, що на основі отриманих в роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися на:

1. III Всеукраїнської науково-технічної конференції молодих вчених, аспірантів і студентів «КОМП'ЮТЕРНІ ІГРИ І МУЛЬТИМЕДІА ЯК ІННОВАЦІЙНИЙ ПІДХІД ДО КОМУНІКАЦІЇ - 2023», Одеса, 28-29 жовтня 2023 р. [3].
2. Міжнародній науково-практичній Інтернет конференції "Електронні інформаційні ресурси: створення, використання, доступ", Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 20-21 листопада 2023 р. [4].

Публікації. Основні результати досліджень опубліковано в матеріалах двох наукових конференцій.

1 АНАЛІЗ СТАНУ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Актуальність питання вибору спеціальності учням та студентам

Професія – це визначена група робіт, обов'язків, та компетенцій, пов'язаних з певною сферою діяльності чи видом занять, що передбачає спеціалізовану підготовку та володіння конкретними навичками для виконання роботи на високому рівні. Основні ознаки професії включають наявність спеціалізованої експертизи, систематичну діяльність у виконанні певних обов'язків, відповідність організованим стандартам та нормам, залучення до освіти і навчання, а також можливість отримання винагороди у вигляді заробітної плати чи інших матеріальних та соціальних вигід. Приклади різноманітних професій включають лікаря, вчителя, інженера, програміста, адвоката, архітектора та інших фахівців.

Готовність до свідомого вибору професії визначається складною мозаїкою внутрішніх чинників, які формують особистісний розвиток. Цей процес обумовлений взаємодією переконань, морально-вольових якостей, знань про різні професії та практичних вмінь і навичок. Зазначена готовність розглядається у трьох ключових аспектах: моральному, де важливим є усвідомлення цінності праці та позитивне ставлення до різноманітних видів трудової діяльності; психологічному, який передбачає обґрунтований вибір професії у відповідності до індивідуальних здібностей та можливостей; і практичному, де ключовим є наявність не лише загальних, але й спеціальних знань, а також вмінь і навичок.

Компоненти готовності до свідомого вибору професії включають у себе комплекс аспектів, які допомагають особистості усвідомити і обґрунтувати свої професійні переваги. Загальне позитивне ставлення до праці є важливим елементом, адже воно стимулює бажання вкладати зусилля у власний професійний розвиток. Сформованість професійних інтересів дозволяє особистості чітко визначити сферу, яка викликає найбільше зацікавлення.

Адекватна самооцінка, мотиваційна сфера та наявність спеціальних здібностей стають ключовими факторами, які визначають успішність вибору професійного шляху. Збалансованість інтересів, здібностей і нахилів, врахування їх відповідності вимогам конкретної професії до особистості, а також фізичне і психічне здоров'я грають ключову роль у формуванні повноцінного професійного самовизначення.

Важливою передумовою успішного професійного самовизначення є також розвиненість у особистості тих якостей, які мають значення для конкретної професії та сприяють ефективному володінню нею. Підготовка до свідомого вибору професії включає активний процес формування психологічних аспектів, таких як здібності, інтереси, ціннісні орієнтації, прагнення, професійні плани та переконання.

Актуальність проблеми професійного самовизначення серед учнів полягає у тому, що випадковий вибір професії може призвести до небажаних наслідків, таких як низька продуктивність праці, помилки та брак у виконанні обов'язків, відчуття незадоволення та пригніченості, що може спричинити психічні розлади, а також економічні втрати на процес переучення та перекваліфікації. З іншого боку, правильний та своєчасний вибір професії в шкільному віці виявляється в суттєвому зменшенні текучості кадрів, підвищенні продуктивності праці та зниженні вартості підготовки кадрів.

Вибір професії – це одне з найважливіших стратегічних рішень у житті людини, що стає складною проблемою, що вимагає глибокого самовизначення та свідомого вибору. Процес професійного самовизначення представляє собою складний акт, що визначає життєву позицію суб’єкта, що має вирішальне значення для вибору шляхів вирішення життєвих завдань. Робота над професійним самовизначенням молоді є складною задачею, яка охоплює соціально-економічний, медико-фізіологічний та психолого-педагогічний аспекти, і впливає на соціальні та економічні аспекти суспільства.

Процес самовизначення завершується досягненням стабільного становища у сфері соціального життя та формуванням відповідних переконань,

принципів, цінностей і мотивації. Такий підхід дозволяє особистості зручно і ефективно впоратися з вибором професії, створюючи підґрунтя для її подальшого успішного володіння.

Мотивація свідомого вибору професії складається з системи мотивів, спрямованих на задоволення потреби в певній сфері професійної діяльності. Мотив – це те, що підштовхує людину до вчинків та дій. Формування мотивації пов'язане із свідомим розумінням суспільної важливості обраної діяльності та об'єктивною оцінкою власних індивідуальних нахилів і здібностей. Таким чином, важливо розглядати мотивацію вибору професії як динамічне явище, що змінюється з віком, і при цьому не тільки розуміти природу змін у мотивації, але й визначати, які аспекти вимагають виховного впливу на конкретному етапі розвитку.

Мотиви вибору професії можна класифікувати на три основні комплекси:

1. Інтерес.

Професійні інтереси учнів поділяються на дві основні групи: безпосередні та опосередковані. Безпосередні інтереси виникають з привабливості конкретної діяльності та процесів праці, тоді як опосередковані інтереси залежать від організаційних, соціальних та інших характеристик професії. Безпосередні професійні інтереси включають професійно-специфічний інтерес, загальнопрофесійний інтерес, романтичний інтерес та ситуативний інтерес. Опосередковані професійні інтереси охоплюють професійно-пізнавальний інтерес, інтерес до самовиховання, престижний інтерес, інтерес супутніх можливостей та невизначений інтерес.

2. Потреби.

Вибір професії також може бути обумовлений задоволенням конкретних потреб, таких як потреба в творчості, соціальному визнанні, матеріальному забезпеченні, спілкуванні з людьми тощо.

3. Орієнтації та цінності.

Цей комплекс включає професійні орієнтації, що базуються на прагненні до пізнання певних процесів, інтерес до самовиховання, престижність обраної

професії, прагнення задовольнити різноманітні духовні та життєво-побутові потреби, а також невизначений емоційний потяг до певної професії.

Для досягнення ефективного вибору професії важливо враховувати та розвивати ці компоненти мотивації, забезпечуючи таким чином більш глибоке розуміння і вибір учнями своєї майбутньої професійної шляху.

Мотивом суспільного обов'язку при виборі професії стає свідоме розуміння учнем реальної суспільної користі, яку йому принесе участь у конкретній сфері діяльності. Це включає в себе відчуття особистої відповідальності за успішну працю та готовність подолати можливі моральні та фізичні труднощі. П'ять основних груп мотивів обов'язку можна виділити: відповідальність по відношенню до повсякденних професійних обов'язків і вимог, прагнення до вдосконалення майстерності у вибраній справі, новаторство у праці та її організації, загально-альтруїстичні прагнення та загальногромадянські ідеали.

По-друге, важливим етапом у виборі професії є самооцінка професійної придатності. Цей процес неоднаково розвивається, може проявлятися у дефіциті самопізнання або дефіциті професійної інформації. Наприклад, учень може не вдалим чином асоціювати властивості професії зі своїми особистими якостями або затруднятися у визначенні ідеальної професії. З віком, зміст самооцінки розширюється, але цей процес не завжди протікає послідовно та інтенсивно. Важливо сприяти розвитку цього етапу, щоб учні могли більш повно розуміти свої професійні можливості та здібності.

Допомогти вступнику вибрати спеціальність можна через систематичне та індивідуальне консультування. Спеціалісти з вибору професії можуть провести детальний аналіз індивідуальних інтересів, здібностей, цінностей та життєвих цілей абітурієнта. Важливо враховувати його попередню освітню та досвідчену базу, а також здійснювати відстеження та обговорення його академічних та особистісних досягнень. У процесі консультацій слід також представляти реальний образ майбутньої професійної діяльності, використовуючи відповідні інформаційні ресурси та можливості професійної

орієнтації. Такий індивідуальний підхід сприяє не лише правильному вибору спеціальності, але й формуванню в учня реалістичних очікувань та мотивації для успішного навчання та подальшої кар'єри [5].

Крім того, важливо акцентувати увагу на розвитку ключових навичок, які будуть важливими у обраній сфері. Надавати можливості для участі в професійних заходах, стажуваннях або курсах допоможе абітурієнту зрозуміти реалії вибраної галузі та набути необхідний практичний досвід. Консультанти також повинні бути відкриті до обговорення можливих змін у виборі спеціальності та готові надавати підтримку на кожному етапі професійного самовизначення. Цей індивідуальний підхід сприяє не лише ефективному вибору спеціальності, але й побудові міцного фундаменту для успішної кар'єри та особистісного розвитку студента.

Продовжуючи процес професійного консультування, важливо також розглядати перспективи розвитку в обраній сфері та знайомити абітурієнта з сучасними тенденціями та інноваціями. Посилення уваги на важливості неперервного самовдосконалення та можливостях участі у професійних спільнотах сприятиме формуванню у студента позитивного ставлення до постійного навчання та адаптації до змін у світі праці. Крім того, консультанти повинні забезпечити абітурієнтів реальними відгуками від студентів, які вже обрали схожі спеціальності, та створювати можливості для зустрічей із представниками вищих навчальних закладів та представниками відомих компаній. Це допоможе учням отримати більше інсайтів та зробити інформований вибір щодо своєї майбутньої кар'єри. Наприкінці консультаційного процесу важливо підтримати абітурієнта в розробці особистого плану кар'єрного розвитку та надати рекомендації для подальших кроків у професійному самовизначенні.

Після завершення консультаційного процесу, акцент слід робити на наданні додаткових ресурсів та підтримці для абітурієнта у процесі вступу. Це може включати підтримку при підготовці до вступних іспитів, розробку портфоліо або резюме, а також надання рекомендацій щодо додаткових

освітніх курсів або стажувань, які можуть збагатити їхні знання та навички. Важливо також висвітлювати можливості фінансової підтримки та стипендій, щоб зробити вибір вищої освіти більш доступним. Проведення інформаційних сесій та відкритих днів на кафедрах і факультетах також може сприяти кращому розумінню абітурієнтами освітнього процесу та життя в університеті. Крім того, створення платформи для обміну інформацією між абітурієнтами, які обирають схожі спеціальності, може забезпечити підтримку спільноти та обмін корисним досвідом. Усі ці етапи спрямовані на те, щоб забезпечити абітурієнтам не лише необхідну інформацію для вибору професійного шляху, але й підтримку та настанови протягом усього процесу вступу та навчання.

1.2 Порівняльний аналіз аналогів

Існує багато веб-ресурсів, які допомагають з вибором спеціальності. Ось декілька популярних і корисних ресурсів:

1. MyNextMove.org [6]. Цей ресурс, створений Бюро статистики праці США, допомагає користувачам визначити свої інтереси та знайти спеціальності, які відповідають їхнім навичкам. На сайті є завдання, навички, інформація про зарплату та багато іншого для понад 900 різних професій. Користувачі можуть знайти кар’єру за допомогою пошуку за ключовими словами; переглядаючи галузі, в яких працюють різні типи працівників; або через O*NET Interest Profiler, інструмент, який пропонує персоналізовані пропозиції щодо кар’єри на основі інтересів особи та рівня досвіду роботи.

У той час як O*NET OnLine пропонує найширший спектр варіантів пошуку та кар’єрних звітів, My Next Move – це оптимізована програма з ключовою інформацією O*NET для тих, хто шукає роботу. Керований підхід допомагає дослідникам кар’єри знаходити потрібну інформацію, не перевантажуючись. Звіти про кар’єру також посилаються на O*NET OnLine для тих, хто хоче дізнатися більше про певну кар’єру.

Консультанти з питань кар’єри та інші спеціалісти, які працюють із шукачами роботи, знайдуть обидві програми цінними ресурсами, оскільки вони призначені як додаткові інструменти для різних аудиторій [7].

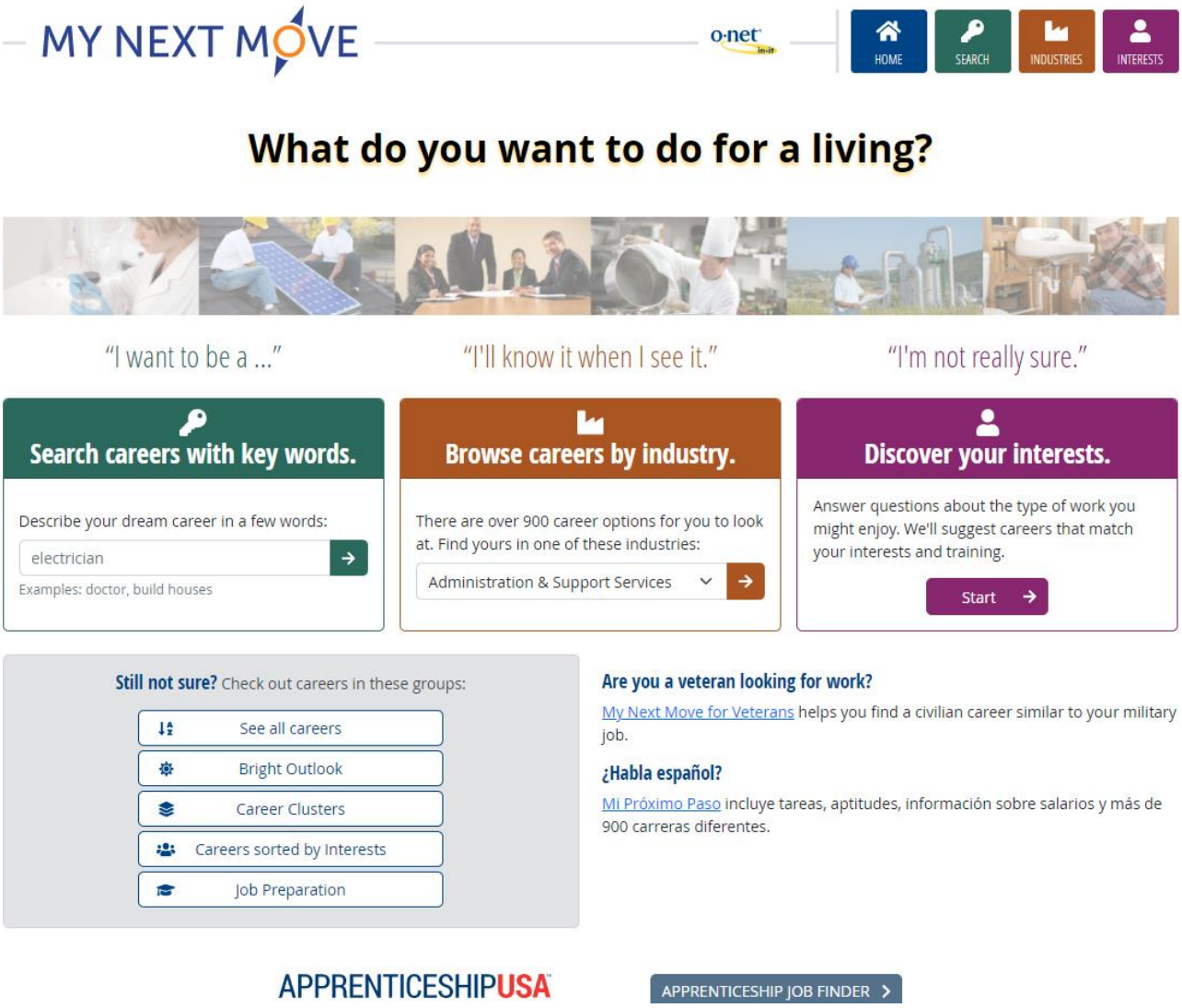

Рисунок 1.1 – Інтерфейс платформи MyNextMove.org

2. CareerExplorer пропонує безкоштовний тест особистості та інтересів, який допомагає визначити ідеальну спеціальність. Це найповніше у світі джерело кар’єрних даних та інформації, що містить понад 1000 профілів професій і ступенів, актуальні дані ринку праці, інструкції, інтерв’ю експертів тощо. Вони співпрацюють з провідними освітніми, тренінговими та

навчальними організаціями, щоб надати користувачам навички, необхідні для наступного кроку у своїй кар’єрі [8].

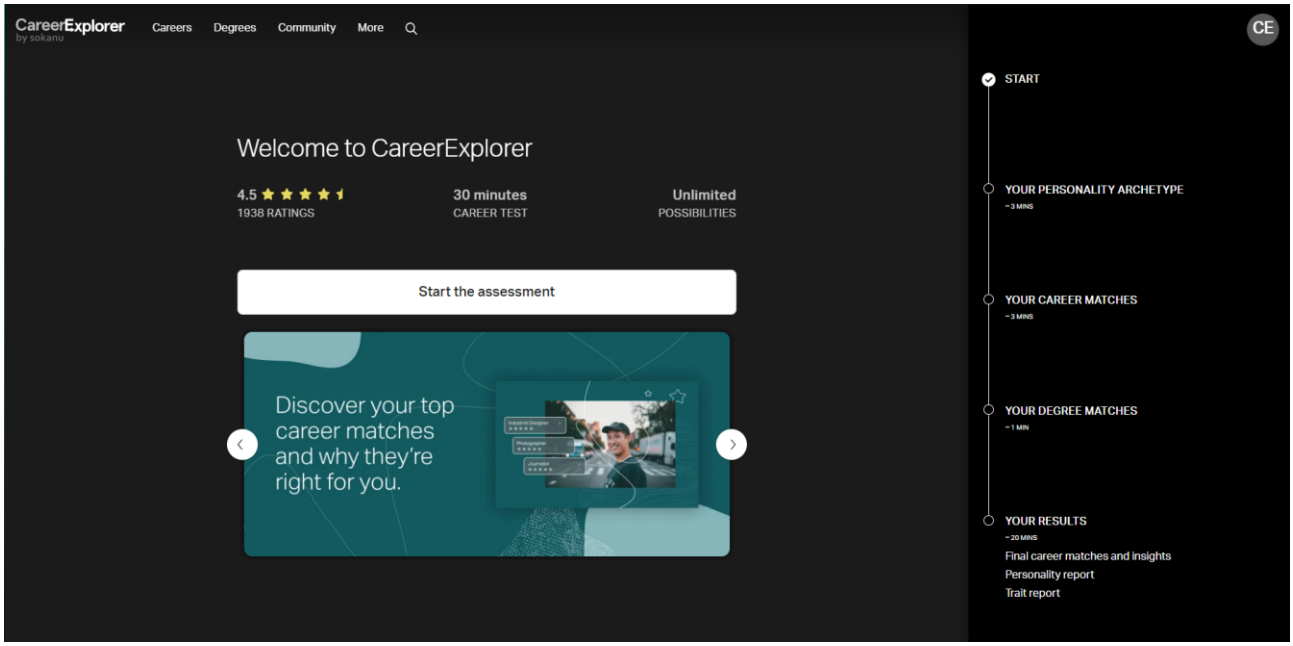

Рисунок 1.2 – Інтерфейс платформи CareerExplorer

3. CAREERwise Education надає онлайн-ресурси для вибору професії, включаючи інтерактивні тести та інформацію про різні кар'єрні шляхи. Інструменти оцінювання кар’єри ставлять запитання, які допоможуть вам дізнатися про себе та кар’єру, яка вам підходить. Ці інструменти не скажуть вам, що робити. Вони допоможуть вам вивчити варіанти та вирішити для себе. Знайдіть інформацію про оцінювання інтересів і навичок. Порівняйте та вибирайте з кількох популярних інструментів, включаючи оцінку інтересів, оцінку інтересів кластеру кар’єри та оцінку інтересів MnCareers [9].

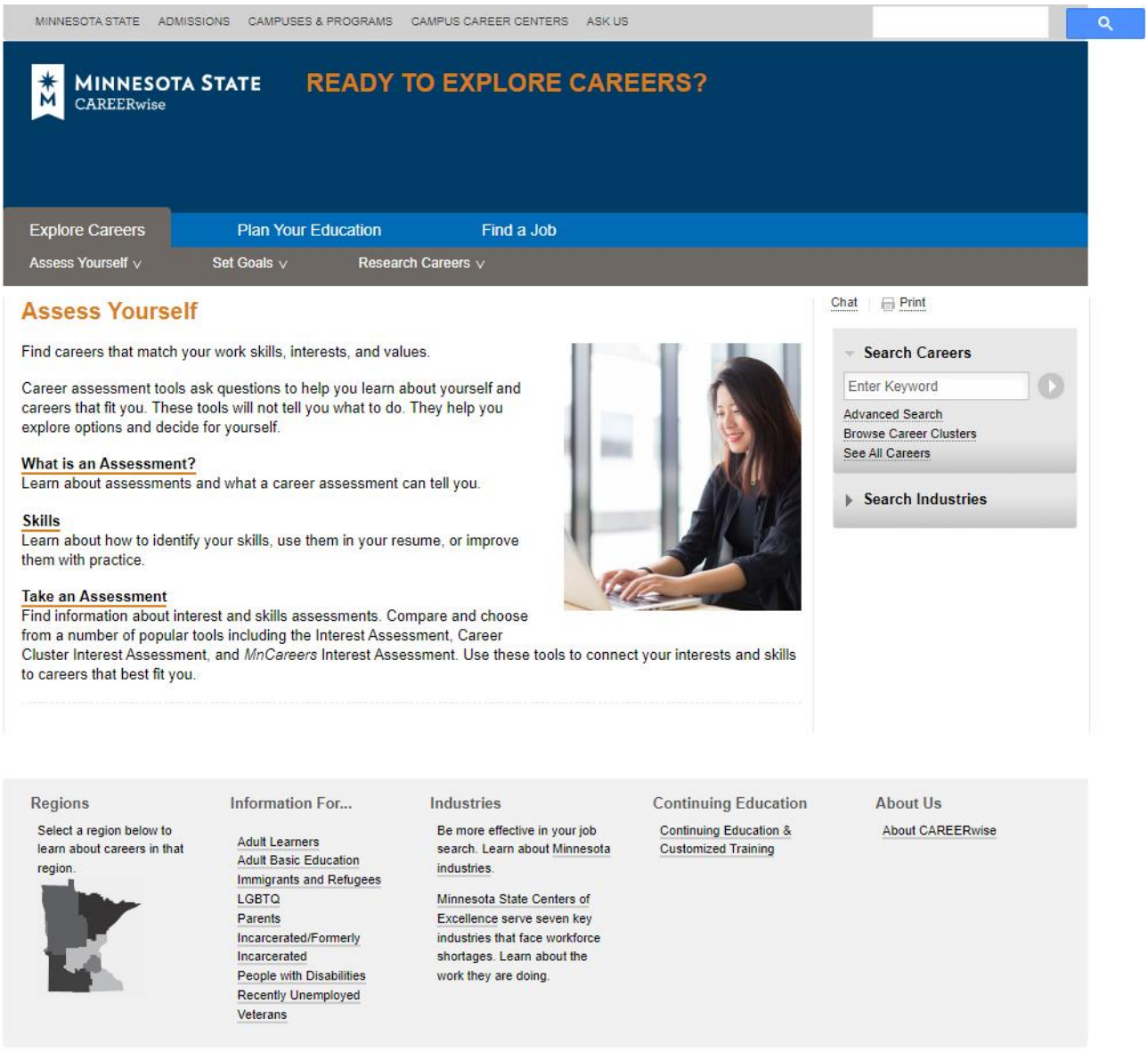

Рисунок 1.3 – Інтерфейс платформи CAREERwise Education

4. Big Future by the College Board містить інструменти для вибору коледжу та спеціальностей, які відповідають вашим інтересам та навичкам.

Це безкоштовний онлайн-посібник із планування, який допомагає всім учням зробити правильний перший крок після закінчення середньої школи.

Ви можете переглянути професії, які вас цікавлять. Ви можете знайти коледж на основі того, що для вас важливо. Ви можете знайти способи оплати навчання в коледжі [10].

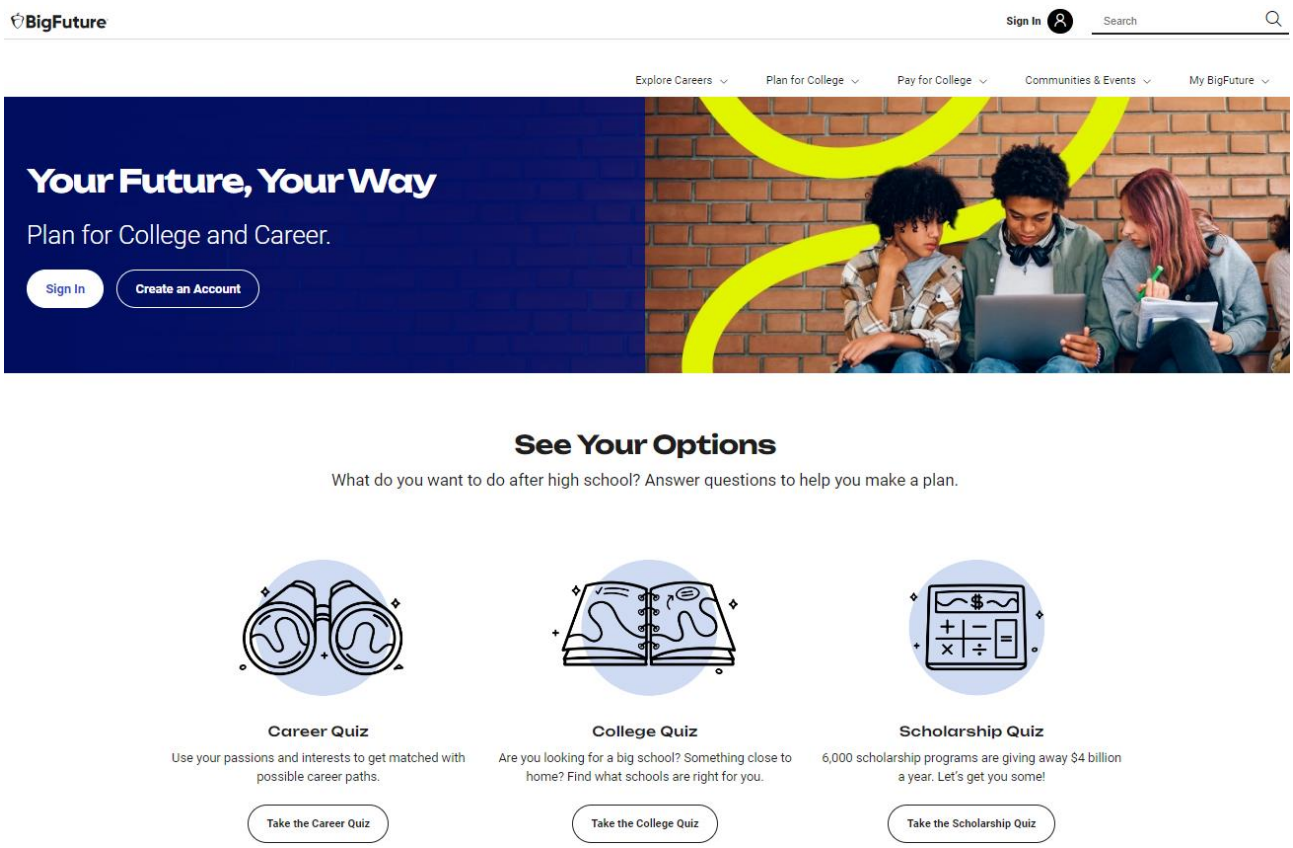

Рисунок 1.4 – Інтерфейс платформи Big Future by the College Board

5. Strong Interest Inventory допомагає людям визначити свою робочу особистість, досліджуючи їхні інтереси в шести широких сферах: реалістичній, мистецькій, дослідницькій, соціальній, заповзятливій та звичайній (часто називають абревіатурою RIASEC). Потім він розбиває області RIASEC на конкретні сфери інтересів, які називаються базовими шкалами інтересів, які можуть бути безпосередньо пов’язані зі сферами навчання, кар’єрою та дозвіллям. Крім того, він описує особисті стильові уподобання людини в різних сферах, таких як середовище навчання та готовність до ризику, які слід враховувати під час вивчення кар’єрних шляхів. Залежно від вибраного вами звіту, він ранжує 5 або 10 найбільш сумісних професій людини зі списку з 260 конкретних вакансій [11].

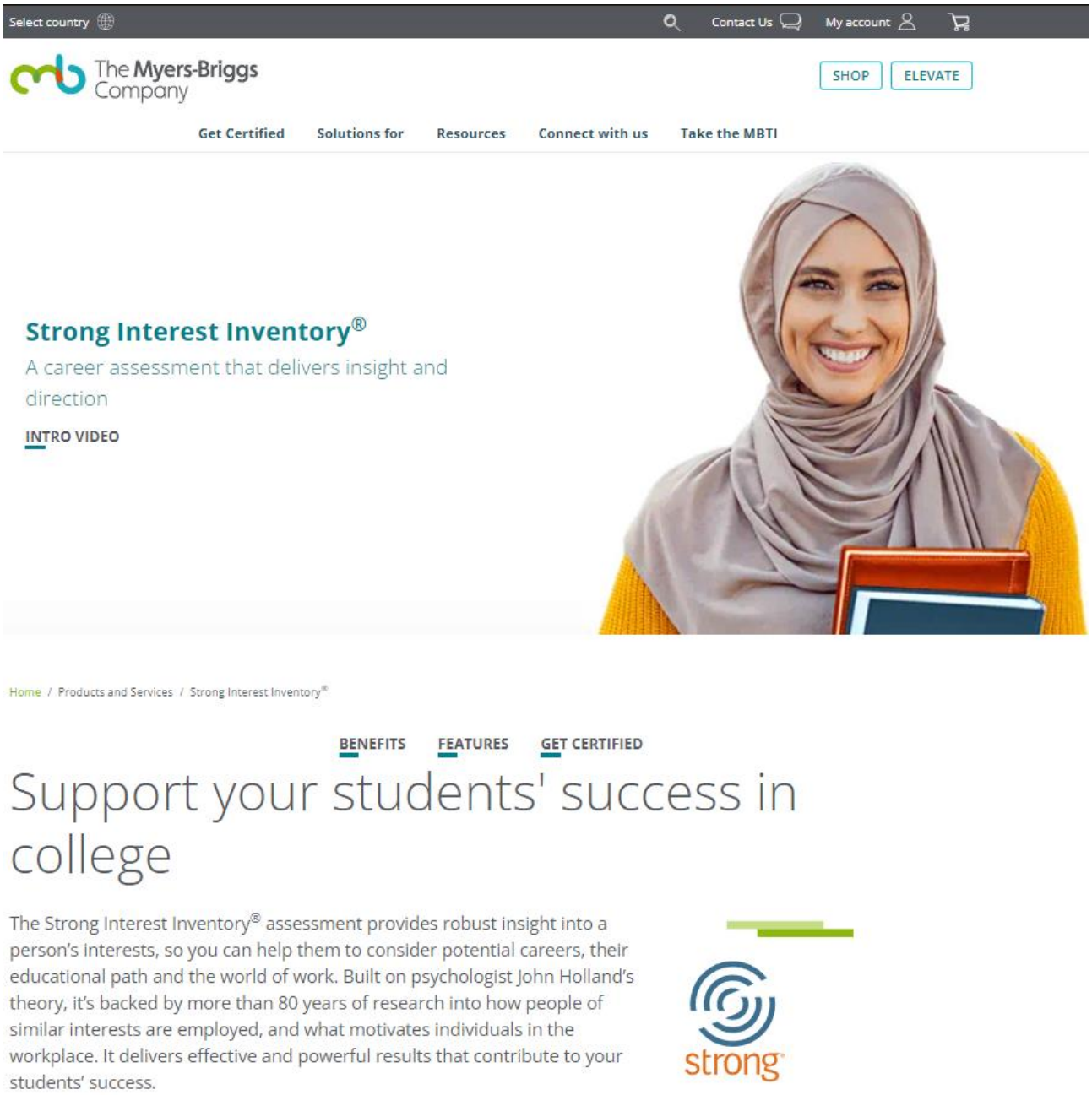

Рисунок 1.5 – Інтерфейс платформи Strong Interest Inventory

6. SAPA Project – це інструмент спільного дослідження для вивчення моделей людської поведінки. Мета – знайти закономірності серед величезної кількості способів, якими люди відрізняються один від одного з точки зору їхніх думок, почуттів, інтересів, здібностей, бажань, цінностей і уподобань. Психологи особистості протягом десятиліть теоретизували про ці сфери відмінностей, але дані, які надаються надаєте, допомагають емпірично об’єднати ці області в єдину комплексну структуру.

Тест на цьому сайті дасть вам відгук про вашу особистість на двох рівнях. По-перше, ви отримаєте бали за 27 параметрами особистості, які були визначені шляхом статистичного аналізу 300 000 учасників SAPA. Тоді також будуть надані вам бали за так званою великою п’ятіркою факторів особистості плюс загальний бал когнітивних здібностей. Більшість людей витрачає від 15 до 25 хвилин на повний тест (ви можете вийти посередині, якщо хочете, але відгук буде не таким точним). Тест є повністю безкоштовним (без жодних умов) і повністю анонімним (без файлів cookie, відстеження, нічого) [12].

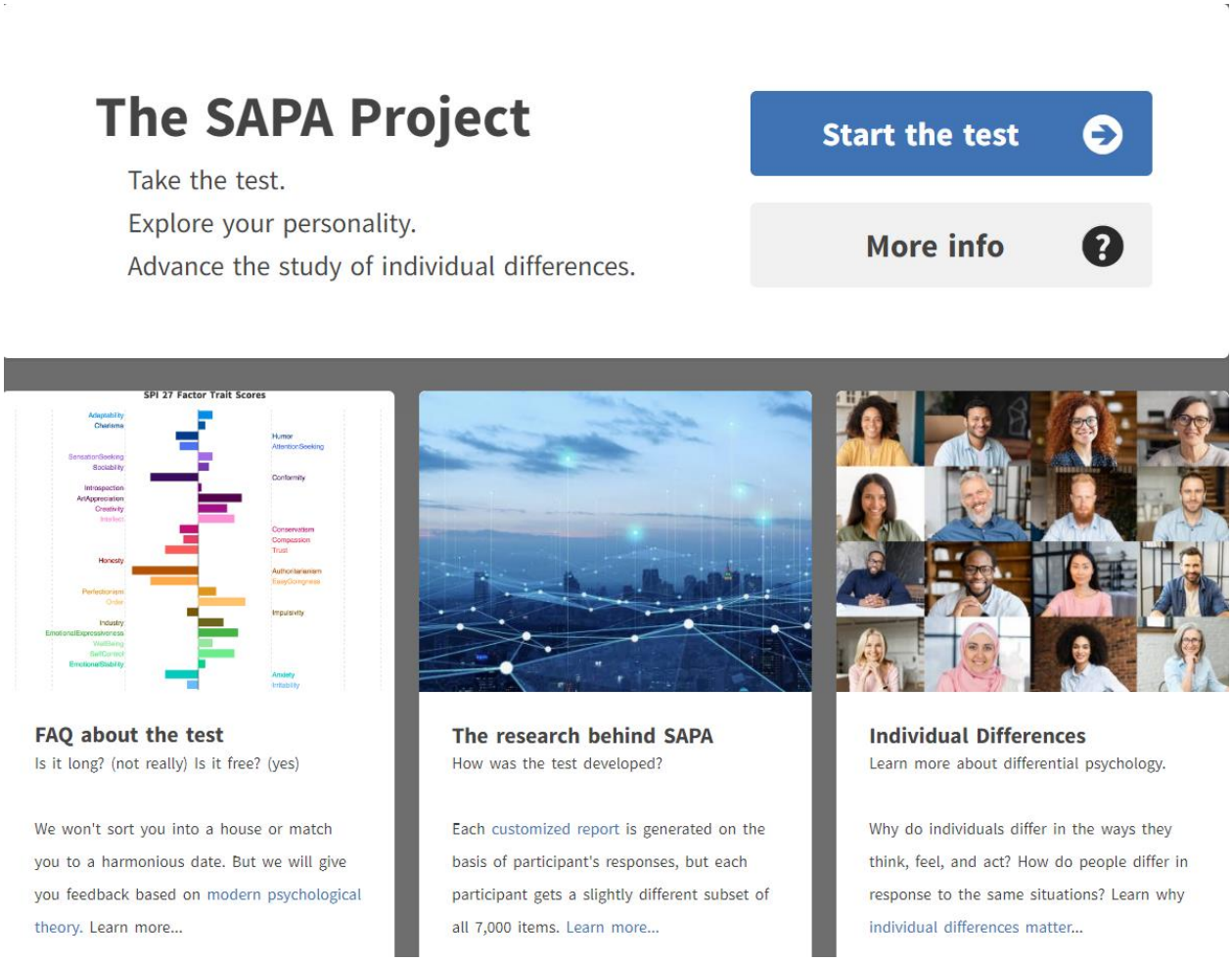

Рисунок 1.6 – Інтерфейс платформи SAPA Project

7. Myprofession.com.ua – Платформа державної служби зайнятості з профорієнтації та розвитку кар'єри створена з метою надання послуг з профорієнтації у дистанційному режимі [13]. Зареєстрованому користувачу

надається можливість безкоштовно отримати послуги з профорієнтації у онлайн форматі без відвідування центру зайнятості, якщо існує потреба у:

– виборі (зміні) професії;
– виборі майбутнього напряму професійного навчання;
– визначенні схильності до підприємницької діяльності;
– оцінці власних здібностей;
– саморозвитку.

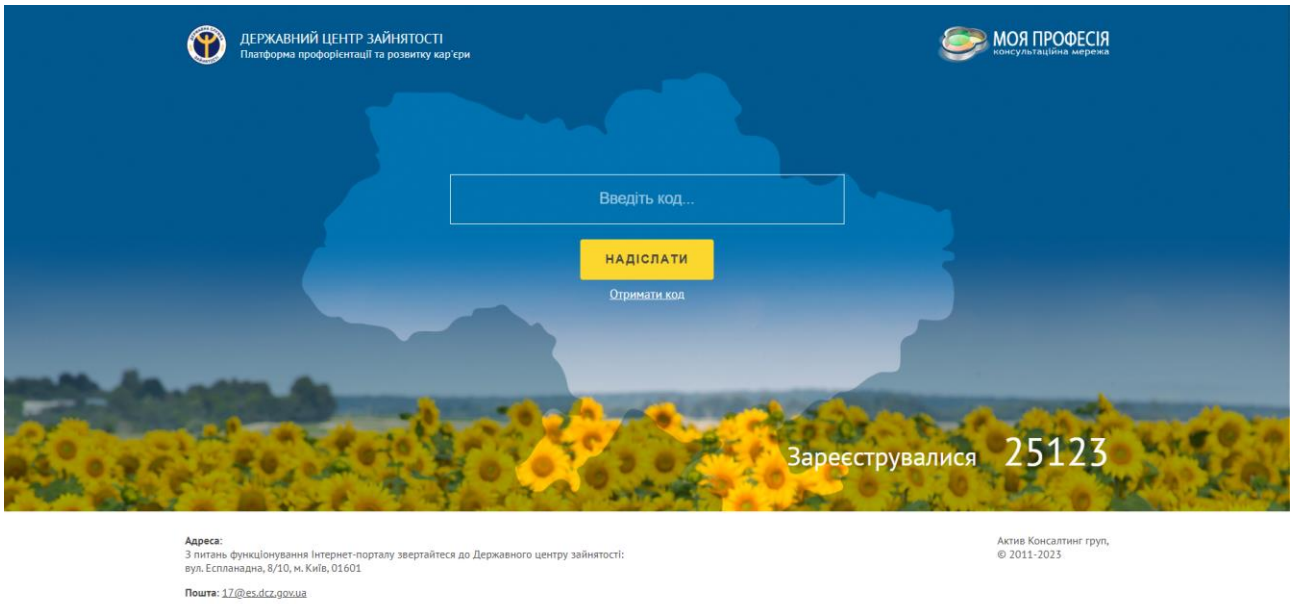

Рисунок 1.7 – Інтерфейс платформи Myprofession.com.ua

8. HR You – це команда експертів, яка допомагає українському бізнесу та державі формувати ринок людського капіталу в Україні [14]. Професійні консультанти в галузі методології навчання, human resources, профорієнтації, кар’єрного консультування та тестології зможуть бути корисними у таких сферах:

– HR-консалтинг (оцінка персоналу, аудит бізнес-процесів, створення ефективної оргструктури компанії, консультації щодо оптимізації роботи персоналу, профорієнтаційне та кар’єрне консультування);
– навчання персоналу (проведення офлайн- та онлайн тренінгів з напрямів: soft skills, менеджмент, продажі тощо);

– створення систем тестування щодо профорієнтації, а також визначення рівнів володіння українською та англійською мовами;
– розробка освітніх інноваційних онлайн-інструментів (повний цикл, від методології дистанційного навчання до технічної реалізації електронних курсів та відеопродакшену);
– створення екосистеми e-learning в державних структурах та комерційних компаніях (інтеграція онлайн-платформи для навчання, LMS та допомога при впровадженні).

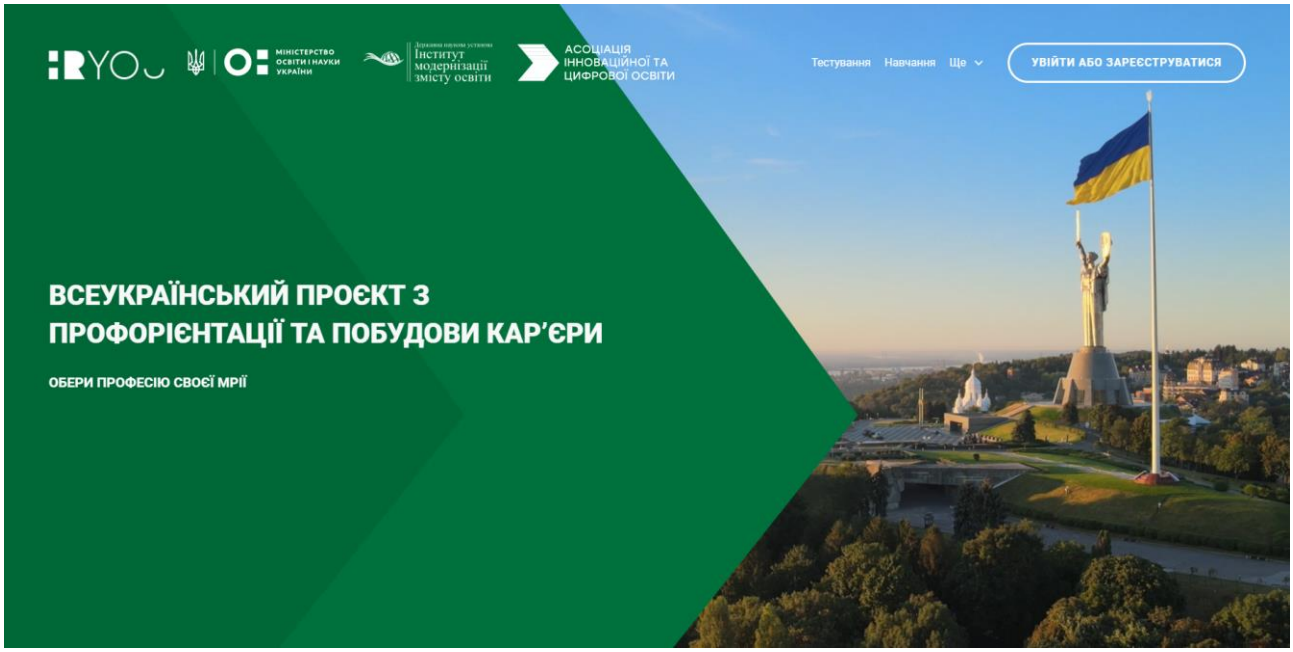

Рисунок 1.8 – Інтерфейс платформи HR You

Проаналізувавши вище наведені аналоги, було визначено переваги і недоліки кожного з них, які в подальшому враховувались при розробці власної системи для експертної рекомендації спеціальності (таблиця 1.1).

Таблиця 1.1 – Порівняльний аналіз аналогів

Критерії	MyNextMove.org	CareerExplorer	CAREERwise Education	Big Future by the College Board	Strong Interest Inventory	SAPA Project	Myprofession.com	HR You
Тестування на виявлення особистих інтересів	+	+	+	–	+	+	+	=
Перегляд актуальної інформації про професії	+	+	+	+	–	–	–	–
Допомога у виборі навчального закладу для вступу	–	–	–	+	–	–	–	–
Підбір найбільш підходящої професії	–	–	+		+	–	+	+
Безкоштовність	+	+	+	+	+	+	+	+
Підбір професій на основі документів про закінчення освіти	–	–	–	–	–	–	–	–
Українська локалізація	–	–	–	–	–	–	+	+

Таблиця порівняльного аналізу аналогів показала, що розробка власної системи є доцільною, в результаті чого були враховані недоліки існуючих аналогів, а також функцію рекомендації спеціальностей на основі документів про освіту.

1.3 Постановка задач розробки

Провівши аналіз стану питання актуальності вибору спеціальності та порівняння існуючих аналогів за переліком визначених критеріїв, були визначені завдання, які необхідно виконати для розробки експертної рекомендації спеціальностей:

- розробка загальної структури системи;
- розробка методу експертної рекомендації спеціальностей, який буде враховувати особисті вподобання користувача, його бали з документів про освіту та результати психологічних тестів;
- розробка методу статистичного аналізу результатів рекомендацій для покращення подальшої роботи системи;
- розробити веб-інтерфейс системи;
- провести тестування системи.

1.4 Висновки

У першому розділі було розглянуто аналіз стану питання важливості вибору спеціальності учням та студентам. Було проведено порівняльний аналіз аналогів систем з рекомендації спеціальностей та тестування особистості. Порівняння проводилось за такими критеріями, як можливість тестування особистості, рекомендація закладів освіти для вступу на основі результатів тестування, безкоштовність платформи та іншими критеріями.

Серед аналогів були виділені такі платформи, як MyNextMove.org, CareerExplorer, CAREERwise Education, Big Future by the College Board, Strong Interest Inventory, SAPA Project, Myprofession.com.ua та HR You. Основними недоліками цих платформ є орієнтованість на англомовну аудиторію, відсутність української локалізації та відсутність врахування документів про освіту.

Враховуючи недоліки аналогів, були поставлені задачі розробки, а саме побудова загальної структури системи, розробка методу експертної рекомендації спеціальностей на основі документів про освіту, з врахуванням минулих результатів рекомендацій та особистих інтересів користувача.

У ході порівняння було виявлено необхідність створення магістерської кваліфікаційної роботи та сформульовано перелік задач для створення системи для експертної рекомендації спеціальностей.

2 РОЗРОБКА МЕТОДІВ СИСТЕМИ

2.1 Аналіз методів, на яких основана система

Основною функцією системи експертної рекомендації спеціальностей є оцінка введених користувачем даних стосовно його вподобань, оцінок з документів про освіту та психологічних тестів, на основі чого користувачеві пропонуються спеціальності. Також система проводить аналізує всі отримані результати, за допомогою чого підвищується якість рекомендації. Всі ці функції основані на методах статистичного аналізу, прогнозування, теорії оптимального вибору та багатокритеріальних задач оптимізації. Оскільки система пов’язана з освітніми процесами, розглянемо їх детальніше.

Освітня система в Україні – це комплексна структура закладів, що забезпечують навчання та виховання громадян згідно з конституційними та законодавчими нормами. Згідно із статтею 53 Конституції України, держава гарантує доступність і безоплатність освіти на всіх її рівнях, включаючи дошкільну, загальну середню, професійно-технічну, вищу освіту у державних і комунальних навчальних закладах. Завдання та структура системи освіти визначаються численними законами, зокрема "Про освіту", "Про загальну середню освіту", "Про вищу освіту". Освітня система включає дошкільну, загальну середню, позашкільну, професійно-технічну, вищу, післядипломну освіту, а також аспірантуру, докторантуру та самоосвіту. Різноманітні навчальні заклади, від дошкільних до вищих навчальних закладів, надають різні види освіти, враховуючи індивідуальні особливості та потреби учнів.

Система освіти в Україні є складною та різноманітною, включаючи різні рівні: дошкільну освіту, загальну середню освіту та вищу освіту. Дошкільна освіта охоплює дітей віком від 1,5 до 6 років і має на меті розвиток загальних навичок та підготовку до навчання в школі. Загальна середня освіта включає базову та повну середню школу і триває 12 років. На вищому рівні система освіти включає бакалаврські, магістерські та докторські програми. Однак система стикається із викликами, такими як неоднаковий рівень доступу до

якісної освіти в різних регіонах, потреба у модернізації навчальних програм та підвищення професійної кваліфікації вчителів. Останні роки характеризуються ініціативами для адаптації системи освіти до вимог сучасного ринку праці та впровадження нових підходів до навчання, включаючи використання технологій та розвиток навичок критичного мислення та творчості серед учнів.

Заклади позашкільної освіти та виховання входять у структуру освітньої системи і спрямовані на розвиток та розкриття здібностей та талантів дітей. Ці установи ґрунтуються на принципі вільного вибору типів закладів та видів діяльності. До них відносяться палаци, будинки, центри, станції дитячої та юнацької творчості, учнівські та студентські клуби, дитячо-юнацькі спортивні школи, школи мистецтв, студії, початкові спеціалізовані мистецькі заклади освіти, бібліотеки, оздоровчі та інші.

Професійно-технічна освіта надає можливість отримати громадянам професії відповідно до їхніх покликань, інтересів та здібностей, а також проводить перепідготовку та підвищення їхньої професійної кваліфікації. Ця освіта може здійснюватися на базі повної загальної середньої освіти або базової, надаючи можливість отримати повну загальну середню освіту. Закладами професійно-технічної освіти є професійно-технічні училища, професійно-художні училища, професійні училища спеціальної реабілітації, училища-агрофірми, училища-заводи, вищі професійні училища, навчально-виробничі центри, центри підготовки і перепідготовки робітничих кадрів, навчально-виробничі комбінати та інші типи закладів, які забезпечують здобуття робітничих професій.

Вища освіта забезпечує комплексну наукову, професійну та практичну підготовку студентів. У системі вищої освіти існують різноманітні заклади, такі як технікуми (училища), коледжі, інститути, консерваторії, університети, академії і інші. Відповідно до статутів цих закладів встановлено IV рівні акредитації, які визначають рівень їхньої діяльності та кваліфікації, яку може отримати випускник.

І рівень охоплює технікуми, училища та інші подібні заклади, які є вищими освітніми установами або структурними підрозділами університетів, академій чи інститутів. Ці заклади проводять підготовку фахівців з вищою освітою за освітньо-професійними програмами молодшого спеціаліста, маючи відповідний кадровий потенціал та матеріально-технічну базу.

ІІ рівень охоплює коледжі та інші аналогічні установи, які також можуть бути структурними підрозділами університетів, академій чи інститутів. Коледжі проводять підготовку фахівців з вищою освітою за освітньо-професійними програмами бакалавра або молодшого спеціаліста з одного чи кількох споріднених напрямів підготовки або спеціальностей, маючи відповідний кадровий потенціал та матеріально-технічну базу.

ІІІ та IV рівні визначаються результатами акредитації і стосуються університетів, академій, інститутів та консерваторій. Вони відзначаються вищим рівнем академічної та наукової активності, а також розширеними можливостями для студентів у галузі досліджень та спеціалізацій.

Університет – це багатопрофільний вищий навчальний заклад, який спеціалізується на підготовці висококваліфікованих фахівців у різних галузях науки, техніки, гуманітарних та інших напрямках, надаючи освітньо-професійні програми на всіх рівнях. Університет проводить як фундаментальні, так і прикладні наукові дослідження, є визначальним науково-методичним центром та активно сприяє поширенню знань через культурно-просвітницьку діяльність серед населення. Цей вид закладу має велику наукову та виробничу інфраструктуру, а також високий рівень кадрового та матеріально-технічного забезпечення. Університети можуть бути різних спеціалізацій, включаючи технічні, технологічні, економічні, медичні, сільськогосподарські та інші.

Академія – це вищий навчальний заклад, який спеціалізується на підготовці фахівців у визначеній галузі знань або виробництва. Академії також проводять як фундаментальні, так і прикладні наукові дослідження, функціонують як ведучі науково-методичні центри та володіють високим рівнем кадрового та матеріально-технічного забезпечення.

Інститут – це вищий навчальний заклад або структурний підрозділ університету чи академії, який надає підготовку фахівців в певній галузі науки, виробництва, освіти, мистецтва та культури. Інститути також активно здійснюють наукову та науково-виробничу діяльність, мають високий рівень кадрового та матеріально-технічного потенціалу.

Консерваторія (музична академія) – це вищий навчальний заклад, спеціалізований у галузі мистецтва і культури, який готує висококваліфікованих фахівців, таких як музиканти, композитори, музикознавці та викладачі музичних дисциплін. Консерваторії проводять наукові дослідження, виступаючи як ведучі центри у своїй галузі, і мають високий рівень кадрового та матеріально-технічного забезпечення. Навчання в консерваторії передбачає всебічну теоретичну і практичну підготовку музикантів для професійної виконавської та педагогічної діяльності.

Післядипломна освіта є спеціалізованою формою вдосконалення освіти та професійної підготовки особи, спрямованою на поглиблення, розширення та оновлення її професійних знань, умінь і навичок або на отримання іншої спеціальності. Цей вид освіти створює можливість для безперервності та прогресу в освіті і включає такі напрями, як перепідготовка (здобуття іншої спеціальності), спеціалізація (розвиток здатностей виконувати конкретні завдання в рамках спеціальності), розширення профілю (підвищення кваліфікації - набуття навичок для виконання додаткових завдань в межах спеціальності) та стажування (отримання досвіду виконання завдань певної спеціальності) [15].

Цей вид освіти реалізується вищими навчальними закладами післядипломної освіти або через структурні підрозділи вищих навчальних закладів відповідного рівня акредитації. Далі розглянемо принципи та види управління:

Управління навчально-виховними закладами ґрунтується на нормативних актах, таких як Конституція України, Закон "Про освіту" та Положення про загальноосвітній навчальний заклад. Ця діяльність, подібно до навчальної та

виховної, базується на виконанні ряду принципів, які визначають директор та його заступники у процесі виконання всіх управлінських функцій.

Принцип управління є ключовим положенням, що виникає з закономірностей управлінської діяльності. Закономірності в управлінні навчальним закладом означають стійкі взаємозв'язки та взаємозалежності між процесом управління та зовнішніми суспільними системами та умовами. Це включає взаємодію між компонентами процесу управління та педагогічного процесу, а також між самими компонентами процесу управління.

До закономірностей в управлінні школою належать: залежність ефективності внутрішкільного управління від урахування особливостей зовнішнього середовища, яке впливає на неї; відповідність механізмів управління загальнодержавним механізмам; створення умов для реалізації мети школи; взаємодія керуючої та керованої підсистем, з визначеною роллю першої; оптимальне співвідношення управлінських впливів, самоорганізації і саморегуляції; надійність та достатність інформаційного забезпечення; цілісність всіх функцій управління в кожному управлінському циклі.

До основних принципів належать:

– принцип прогностичності внутрішньо-шкільного управління;
– єдності державних і внутрішньо-шкільних механізмів управління;
– демократизації і гуманізації управління;
– раціонального поєднання централізації і децентралізації;
– єдності єдиноначальства і колегіальності;
– гласності, відкритості управління;
– об'єктивності та інформаційної достатності;
– плановості, перспективності;
– компетентності;
– оптимізації
– системності в управлінні.

Принцип прогностичності внутрішньо-шкільного управління є однією з ключових характеристик загальноосвітніх навчальних закладів, оскільки вони

взаємодіють з зовнішнім середовищем в найтісніших специфічних зв'язках. Жодна школа не може існувати самостійно, вона повністю залежить від впливу зовнішніх факторів, таких як економічні умови, соціальні, культурні та політичні фактори і т.д. Цей принцип передбачає, що школа повинна аналізувати вплив зовнішнього середовища, розробляти стратегії взаємодії з ним, адаптувати навчально-виховний процес і, водночас, впливати на середовище для досягнення своєї мети [16].

Принцип єдності державних і внутрішньо-шкільних механізмів управління стверджує, що внутрішнє управління школою повинно повністю використовувати умови, які створені державою для її функціонування. Ці умови включають в себе закони, постанови, нормативні вимоги, установлені стандарти, що охоплюють навчальні плани, програми та вимоги до рівня освіченості. Крім того, вони охоплюють положення таких наук, як прогностика, педагогіка, психологія, теорія управління, шкільна гігієна, юридична наука, медицина та економічна теорія.

Принцип демократизації і гуманізації управління включає в себе забезпечення громадського самоврядування разом із єдиноначальністю директора та враховує наступне:

1. Делегування частини прав та повноважень адміністрації громадсько-державним структурам, таким як Рада школи, педагогічна рада, методичні об'єднання, учнівське самоврядування, забезпечення виборності та звітності цих управлінських структур.
2. Максимальне використання самодіяльності та ініціативи вчителів, учнів, батьків і широкої громадськості, сприяння розвитку їх творчих сил.
3. Установлення, функціонування та розвиток демократичних стосунків між керівниками, учителями, вихователями, учнями і батьками.
4. Визначення оптимального співвідношення цілеспрямованих управлінських впливів, співробітництва, співуправління і самоуправління.

Співпраця між адміністрацією і вчителями може складатися лише на основі гласності, яка вимагає наявності повної, об'єктивної та всебічної інформації про все, що відбувається у школі. Таким чином, демократизація внутрішньо-шкільного управління передбачає не лише звіти вчителів перед адміністрацією, але і звіти адміністрації перед вчителями, а також звіти педагогічної ради перед загальношкільними зборами наприкінці навчального року. Здійснення спеціальної системи внутрішньо-шкільної інформації, доступної для широкого використання (такої як стінна преса, радіо, телебачення тощо), також є складовою цього принципу.

Головним органом самоврядування є конференція представників громадськості, педагогів, батьків, учнів старших і середніх класів, на якій вибирається рада школи. Основні питання роботи школи директор зобов'язаний узгоджувати з радою. Положення про загальноосвітній навчальний заклад чітко формулюють права та обов'язки вчителів, учнів і працівників школи. Статут школи та правила внутрішнього трудового розпорядку виступають демократичною правовою базою. Школа є малим "острівцем" правової держави, де все регулюється нормами та правилами.

Принцип гуманізації ґрунтується на визнанні людини як найвищої цінності та необхідності поваги до неї. Цей принцип передбачає створення гуманних відносин та оптимальних умов для повноцінної життєдіяльності дитячого і педагогічного колективів.

Принцип поєднання централізації і децентралізації управління внутрішньо-шкільними процесами сприяє оптимальному збалансуванню контролю та ініціативи на різних рівнях. Умови надмірної централізації призводять до перевантаження адміністративних функцій, обмеження творчої ініціативи керівників, учителів і учнів, перетворюючи їх на виконавців рішень, які приймаються без їхньої участі. З іншого боку, децентралізація може призвести до втрати ефективності, адже зменшення ролі керівника та адміністрації може викликати конфлікти і невиправдане протистояння внутрішніх та громадських органів управління.

Поєднання централізації і децентралізації в управлінні навчальним закладом спрямоване на уникнення дублювання функцій, підвищення координації між структурними підрозділами та розробку професійних управлінських рішень. Це забезпечує ефективне співробітництво між адміністративними і громадськими органами в інтересах всього колективу.

Принцип єдиноначальності і колегіальності в управлінні передбачає персональну відповідальність директора перед вищими органами освіти та громадськістю. Цей принцип розглядає директора як особистість, що об'єднує в собі право самостійно приймати рішення та вимагати їх виконання, а також обов'язок співпрацювати з колегіальними органами, такими як рада та педагогічна рада. Реалізація цього принципу забезпечує збалансований підхід до управління, враховуючи інтереси всіх учасників педагогічного процесу.

Принцип гласності та відкритості в управлінні навчальним закладом передбачає відкрите обговорення рішень педагогічної ради, а також регулярні наради для вирішення поточних питань. Ці принципи вберігають від соціально-психологічної напруженості і сприяють позитивному розвитку в середовищі школи.

Принцип об'єктивності і повноти інформації управління загальноосвітнім закладом наголошує на необхідності наявності об'єктивної та повної інформації щодо всіх аспектів функціонування школи. Це охоплює оцінку успішності учнів, якість викладання предметів, виховну роботу, роботу з педагогічними кадрами, батьками та громадськістю, а також стан матеріально-технічної бази школи.

На першому етапі впровадження системи інформаційного забезпечення включає моделювання та організацію простої системи інформаційного надання. Другий етап передбачає подальшу модернізацію, використовуючи електронно-обчислювальні машини та створюючи банк даних для зберігання сумарної інформації. Неперервний моніторинг педагогічного процесу, який базується на комп'ютерних технологіях збору та обробки інформації, дозволяє постійно відстежувати прогрес, результати та ефективність навчального процесу.

Інформація управління поділяється за різними критеріями:

1. За часом:

– щоденна;

– щомісячна;

– четвертна;

– семестрова;

– щорічна.

2. За функціями управління:

– аналітична;

– оцінна;

– конструктивна;

– організаційна;

3. За джерелами надходження:

– внутрішньо-шкільна;

– відомча;

– позавідомча.

4. За цільовим призначенням:

– директивна;

– ознайомча;

– рекомендаційна.

На всіх етапах управлінського циклу – від аналізу та планування до організації виконання, контролю і регулювання – керівнику навчального закладу необхідна об'єктивна та повна інформація. Від навичок організації її збору, обробки, аналізу та використання залежить ефективність управління та нормальне функціонування навчально-виховного процесу.

Принцип плановості передбачає встановлення чіткої системи перспективного і щоденного планування у всіх сферах навчально-виховного та організаційно-господарського процесу з урахуванням об'єктивних умов та соціально-економічних можливостей конкретного освітнього закладу. Всі плани повинні спрямовуватися на вирішення основних завдань школи.

Принцип перспективності випливає з необхідності передбачення та прогнозування діяльності школи не лише на семестр чи навчальний рік, але й на весь період навчання учнів у школі.

Принцип компетентності вимагає високого рівня науково-педагогічної підготовки, загальної ерудиції та професіоналізму директора та вчителів.

Принцип оптимізації спрямований на створення в навчально-виховному закладі найсприятливіших соціально-психологічних та економічних умов для ефективної діяльності учасників педагогічного процесу.

Принцип системності в управлінні передбачає розуміння системної природи педагогічного процесу. Системний підхід в управлінні школою спонукає керівника мати чітку уяву про школу як систему: її основні компоненти, структуру, інтегративну властивість системи, яка є результатом взаємодії компонентів. Системність передбачає також цілісність усіх функцій управління в кожному із управлінських циклів.

Управління навчальним закладом поділяється на:

– органи колегіального управління школою (конференція, рада школи, педагогічна рада, нарада при директорі, його заступниках);
– адміністрація школи (директор, його заступники);
– органи громадського самоврядування (учнівське самоврядування, учителів (профком, методична рада), батьків (батьківський комітет).

З метою демократизації управління, встановлення зворотного зв'язку та проведення поточних коригувань управлінських рішень у навчальному закладі діють органи громадського самоврядування. Серед них – учнівський комітет школи (класу), профспілковий комітет, методична рада для учителів, а також батьківський комітет школи (класу). Повноваження цих органів визначає статут школи. Вони виступають як ефективний механізм формування громадської думки і сприяють розвитку діалогу з адміністрацією [17].

В Україні спеціальності виокремлюються у вищому та середньому освітніх закладах. У вищому освітньому секторі існують різноманітні спеціальності, такі як інженерія, медицина, право, економіка, гуманітарні науки

і багато інших. Система вищої освіти базується на присудженні кваліфікаційних рівнів: бакалавр, магістр, кандидат наук, доктор наук. Щодо середнього освітнього рівня, існують професійно-технічні навчальні заклади, які готують фахівців на різні професії, включаючи технічні, медичні, економічні та інші спрямування. В Україні велика увага приділяється актуальності та відповідності освіти вимогам ринку праці, щоб забезпечити студентам і випускникам не лише теоретичні знання, але й практичні навички для успішної кар'єри.

Статистична методологія - це галузь науки, що вивчає методи та принципи збору, обробки та аналізу статистичних даних. Ця область включає в себе розробку та застосування різноманітних статистичних методів, що допомагають розкривати закономірності в наборах даних, вести прогнозування та приймати обґрунтовані рішення на основі отриманих результатів.

Статистична методологія включає ряд ключових аспектів, починаючи від створення правильного експериментального дизайну для збору даних та визначення відповідних статистичних методів аналізу. Важливою складовою цієї галузі є розвиток математичних моделей та алгоритмів, які допомагають взаємодіяти з реальними статистичними даними.

Статистична методологія широко використовується у багатьох галузях, включаючи економіку, медицину, соціологію, психологію та природничі науки. Вона є ключовим інструментом для вивчення і визначення тенденцій, роботи зі складними даними та підтримки наукових висновків. Сучасні обчислювальні техніки значно розширили можливості статистичної методології, надаючи дослідникам можливість ефективно обробляти великі обсяги даних та використовувати складні моделі для аналізу.

У статистичній методології важливу роль відіграє імовірність, дослідження різноманітних розподілів, регресійний аналіз та багато інших підходів, спрямованих на отримання точних та достовірних результатів. Постійний розвиток цієї галузі визначає її важливість у сучасному науковому

дослідженні та практиці, сприяючи точнішому та обґрунтованішому використанню статистичних методів у різних сферах життя [18].

Філософський підхід, як методична основа статистичної науки, вимагає розгляду явищ і процесів у їх русі, постійних змінах і розвитку. Для досягнення цієї мети використовується система показників, яка дозволяє охарактеризувати варіації рівнів явищ, визначити тенденції і закономірності їх розвитку. Важливим аспектом філософського підходу є визначення меж та переходу від кількісних явищ до якісних форм їх прояву, що може вирішуватися статистичними методами, такими як групування, дисперсійний метод тощо.

Філософська основа статистики знаходить втілення в її специфічних методах, які можна класифікувати за стадіями дослідження. Методи статистичного спостереження, що використовуються на початковому етапі, забезпечують збір та оцінку якості первинних статистичних даних. Масове статистичне спостереження дозволяє вивчати загальні риси, процеси та закономірності великих сукупностей.

Методи зведення і групування первинного статистичного матеріалу включають в себе перевірку, систематизацію, обробку та підсумовування даних. Зведення забезпечує систематизацію інформації, розподіл її за ознаками відмінності, що дозволяє краще розуміти характеристики сукупності. Методи групування, зокрема, використовуються для класифікації та аналізу статистичних даних відповідно до завдань дослідження та якості первинної інформації.

Методи групування в статистичному дослідженні дозволяють класифікувати досліджувану сукупність за певними ознаками, наприклад, за формою виробничої діяльності господарств у конкретному регіоні, що дозволяє отримати більш деталізовану інформацію. На стадії узагальнення і аналізу зведеного матеріалу виявляються характерні властивості та закономірності соціально-економічних явищ, використовуючи широкий спектр статистичних прийомів і методів. Розрахунок узагальнюючих статистичних показників, таких як абсолютні, відносні та середні величини, вимірювання варіації дозволяють

здобути глибоке розуміння умов, в яких відбуваються масові процеси. Аналіз тенденцій та закономірностей у часовому розрізі виконується за допомогою рядів динаміки та індексного методу аналізу, особливо важливого для вивчення складних економічних явищ. Методи прогнозування, використовуючи вербальні і письмові тексти, включають структурно-морфологічний аналіз, визначення публічної активності, аналіз груп патентних документів, термінологічний та лексичний аналіз, що дозволяє отримувати обґрунтовані варіанти тенденцій розвитку об'єкта в часі і просторі [19].

Для ефективного прогнозування в практичній діяльності використовують різноманітні кількісні та якісні методи. Кількісні методи ґрунтуються на аналізі тенденцій зміни параметрів або вивченні статистично достовірних залежностей, що характеризують виробничу діяльність об'єкта управління. До цих методів відносять аналіз тимчасових рядів та каузальне (причинно-наслідкове) моделювання. З іншого боку, якісні методи базуються на експертних оцінках фахівців у сфері прийняття рішень, таких як методи експертних оцінок, висновок та моделі очікування споживача через опитування клієнтів.

При прогнозуванні складних об'єктів, таких як економічна кон'юнктура, застосовують різноманітні комбінації кількісних і якісних методів. Наприклад, прогноз економічної кон'юнктури ґрунтується на прогнозах у сфері обмежень на захист навколишнього середовища, міжнародної торгівлі, попиту та пропозиції на ринку. Кожен з цих прогнозів ґрунтується на проміжних прогнозах різних процесів. До відомих методів прогнозування економічної кон'юнктури входять "мозкова атака", метод Дельфі, екстраполяція тенденцій, морфологічний аналіз, імітаційне динамічне моделювання, структурний аналіз та інші. Важливо враховувати, що існують різні класифікації методів прогнозування, які враховують особливості конкретних завдань прогнозування.

Цілі, час та умови прогнозування визначають набір методів і прийомів, які можуть бути використані в розробці прогнозів. Різноманіття методів отримання інформації включає:

1. Метод структурно-морфологічний: Спрямований на виявлення

внутрішнього складу предметної області і фіксацію новаторських розробок. Цей метод дозволяє обґрунтовано формулювати стратегію науково-технічного прогресу підприємства.

2. Метод визначення публікаційної активності: Дозволяє відстежити циклічність документів у різних галузях знань, виявляючи стан розробки проблем в країні та на підприємствах. Цей метод допомагає коригувати стратегії науково-технічного прогресу в організації.
3. Метод виявлення групи латентних документів: Патенти, які видані провідними фірмами, можуть вказувати на спрямованість їх діяльності та рівень розв'язання проблем, що цікавлять вашу організацію.
4. Метод показників: Аналізуючи динаміку зміни показників технічної системи, можна зробити висновок про тенденції її розвитку. Кожна технічна система характеризується певними характеристиками, які можна використовувати для прогнозування.
5. Метод термінологічного і лексичного аналізу: Зміна термінологічного апарату в різних галузях знань може вказувати на народження інновацій. Лексичний аналіз текстів допомагає виявити зародження принципових інновацій і спрогнозувати дії організації на ранніх етапах.

Точність та перевірка прогнозів є суттєвим аспектом у процесі прогнозування. Ускладненість регулярного прогнозування майбутніх значень виникає внаслідок складності змінних. Тому важливо враховувати показник можливого відхилення значень змінної при формуванні прогнозу. Вибір еталону точності обумовлюється:

– різними показниками отриманих прогнозів;
– необхідністю наявності кількох показників у прогнозі;
– важливістю контролю помилок прогнозу порівняно із еталоном.

Помилка (П) в прогнозі означає різницю між фактичним і спрогнозованим значеннями:

$$П = ФЗ\text{-}ПГ, \quad (2.1)$$

де ФЗ – фактичне значення; ПГ – прогноз.

Позитивна помилка виникає, коли прогноз занижений, тоді як негативна помилка спостерігається, коли він завищений. Помилки в прогнозі впливають на прийняття рішень при виборі різних варіантів прогнозу та визначають результат використання конкретного методу прогнозування [20].

Визначення точної помилки у виробленому прогнозі неможливе, оскільки дійсне значення прогнозу невідоме. Однак існує ймовірність того, що обчислена помилка прогнозу не перевищить певну величину або максимальну допустиму помилку прогнозу, яку можна очікувати із заданою ймовірністю.

– математичне, очікування (МО):

$$МО = \frac{\sum |ДЗ - ПГ|}{n}, \quad (2.2)$$

де ДЗ – дійсне значення.

– середньоквадратичне відхилення (СКВ):

$$СКВ = \frac{\sum (ДЗ - ПГ)^2}{n-1}. \quad (2.3)$$

Контроль прогнозу здійснюється порівнянням помилок прогнозу із заздалегідь визначеними значеннями (межами). На практиці контроль здійснюється за допомогою розрахунку відношення сукупної помилки прогнозу до відповідного значення МО і використовується для спостереження за прогнозом:

$$П_0 = \frac{\sum (ДЗ - ПГ)}{МО}, \quad (2.4)$$

де $П_0$ – показник відхилення.

Значення показників відхилення порівнюються із межами значень показників, що засновані на судженнях і досвіді [21].

Згідно з теорією ймовірності та новими законами розсіювання, відхилення випадкової величини від центру групування не перевищує трьох середніх квадратних відхилень. Це призводить до того, що значення показника відхилення повинне знаходитися в межах ± 4, що відповідає межам трьох стандартних відхилень [22].

При прийнятті рішень, зокрема виборі найбільш прийнятної спеціальності, виникає складність через велику кількість критеріїв, які не завжди узгоджені між собою. Це передбачає необхідність побудови математичних моделей і застосування математичних методів для прийняття рішень при розгляді багатьох критеріїв.

Розглянемо методи рішення, які полягають в зведенні початкової багатокритеріальної задачі до скалярної шляхом введення деякого узагальненого критерію. Всі ці методи виконуються так:

1. Нормування всіх критеріїв для приведення їх до порівнянного безрозмірного вигляду.
2. "Згортання" їх в одну цільову функцію, відому як узагальнений критерій, з врахуванням їхньої відносної важливості за допомогою вагових коефіцієнтів.

В результаті вихідна багатокритеріальна задача зводиться до звичайної задачі оптимізації за одним критерієм. Найбільш поширеними видами згортки є:

1. Узагальнені критерії на основі середньозваженої функції, особливо лінійна згортка критеріїв.
2. Мультиплікативна згортка.
3. Відношення суми критеріїв, які максимізуються, до суми критеріїв, які мінімізуються.

Важливо враховувати, що метод відношення суми критеріїв має свій недолік, оскільки базується на припущенні, що нестача в одному показнику може бути компенсована іншим.

Також, для забезпечення більш ефективних рекомендацій стосовно вибору спеціальності, існує можливість проходження психологічного тесту для визначення особистих інтересів [23].

Як один із ключових напрямів профорієнтації, діагностика професійного самовизначення спрямована на вивчення особливостей процесів та властивостей, які є характерними для психіки конкретної особи, а отже, визначає ступінь придатності до виконання конкретної професійної діяльності. З цієї причини важливо розрізняти різні компоненти психодіагностики для профорієнтації, такі як:

- психологічний (виявлення індивідуально-психологічних властивостей особистості).
- психофізіологічний (виявлення та аналіз типологічних та психофізіологічних показників).
- медичний (вивчення стану здоров'я та схильності до захворювань).
- соціальний (виявлення нахилів, здібностей, професійної спрямованості, впливу сім'ї, оточення, умов життя та вибору професії).

Організація психодіагностичного обстеження вимагає від організаторів дотримання певних принципів роботи:

1. Принцип довготривалості (виявлення здібностей і обдарувань з метою встановлення профпридатності, що є довготривалим процесом і вимагає системності).
2. Принцип використання тренінгових методів і завдань (для розвитку встановлених у процесі психодіагностики професійно важливих індивідуальних якостей та властивостей особистості).
3. Принцип комплексності (всебічного і комплексного характеру діагностики).

4. Принцип участі різних спеціалістів (залучення до психодіагностичних досліджень педагогів, соціальних працівників, батьків).

Враховуючи, що успішний вибір професії залежить від правильної самооцінки схильностей, здібностей, інтересів, спрямованості, можливостей та обмежень, а також відсутності знань про те, що саме необхідно для успішної діяльності в обраній сфері, важливо відзначити наступне. По-перше, під час використання діагностичного інструментарію для допомоги у виборі професії необхідно враховувати три ключові фактори:

– визначення відповідності індивідуальних характеристик особистості вимогам майбутньої діяльності.
– встановлення рівня готовності особистості як майбутнього суб'єкта праці до подальшої організації професійної підготовки з урахуванням особливостей конкретного виду діяльності.
– врахування ринкового попиту на працівників відповідних професій та кваліфікацій.

По-друге, процес професійного самовизначення особистості визначається впливом різноманітних соціальних та особистісних факторів (економічних, політичних, соціокультурних, психологічних тощо). Ефективність цих факторів залежить від суспільних та особистих обставин, у яких відбувається базова соціалізація учнів та студентів. Таким чином, психологічне діагностування для профорієнтації має враховувати аналіз структури загальних та спеціальних здібностей, потенційної обдарованості учнів та студентів, дослідження їх професійних інтересів, що відповідають певному виду діяльності, і провідного типу мотивації у процесі вибору професії.

2.2 Розробка загального алгоритму роботи системи рекомендації спеціальностей

Першим етапом розробки системи є проєктування її загального алгоритму. Сама система має виконувати такі функції:

– реєстрація та авторизація користувачів;

– рекомендація спеціальностей на основі документів про освіту, психологічних тестів та власних побажань користувача;
– аналіз попередніх результатів рекомендацій;

Загальний алгоритм системи зображений на рисунку 2.1.

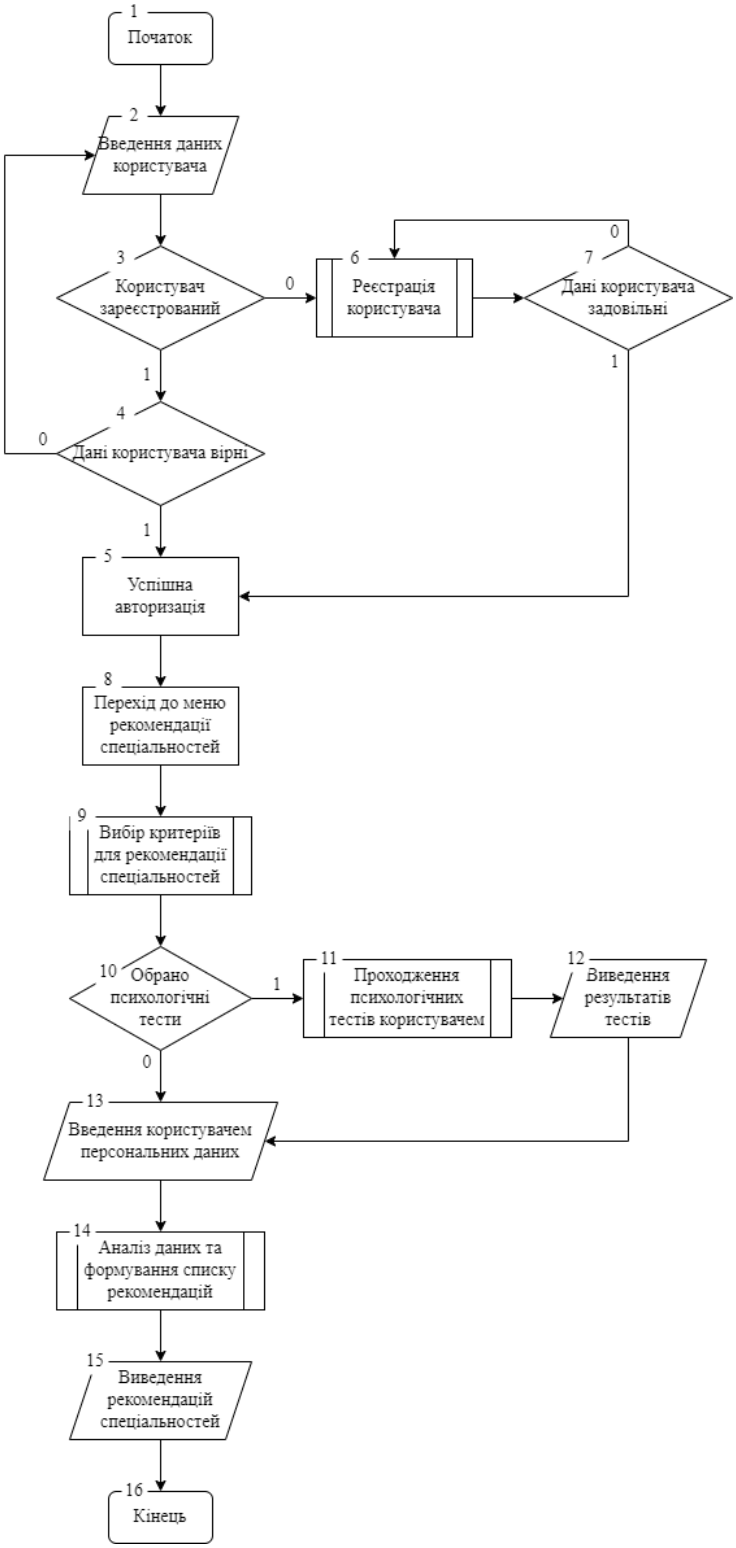

Рисунок 2.1 – Загальний алгоритм системи рекомендації спеціальностей

Опис загального алгоритму системи:

1. Користувач відкриває веб-сторінку системи.
2. Користувач вводить свої дані для авторизації на сайті, а саме електронну адресу та пароль.
3. Система перевіряє чи користувач з такими даними вже зареєстрований.
4. Якщо користувач з такою електронною адресою зареєстрований то система перевіряє чи пароль введено вірно.
5. Після введення правильного пароля користувач успішно переходить у основне меню системи.
6. Якщо користувача з такою електронною адресою не зареєстровано, система пропонує створити новий аккаунт.
7. Система перевіряє чи введена електронна адреса ще не зареєстрована та взагалі існує, і чи введено достатньо надійний пароль. Після успішної реєстрації користувач переходить у основне меню системи.
8. Користувач може обрати меню для рекомендації спеціальностей.
9. Користувач обирає за якими саме критеріями система буде рекомендувати спеціальності. Він обов’язково має ввести особисті вподобання та необов’язково може ввести дані документів про освіту та обрати проходження психологічних тестів.
10. Система пропонує користувачеві пройти психологічний тест для більш точної рекомендації спеціальностей.
11. Якщо користувач погодився проходити психологічний тест – система генерує перелік питань, на які потрібно відповісти.
12. Після проходження тестів система виводить на екран результати тестів.
13. Користувач вводить свої персональні вподобання стосовно майбутньої професії та опціонально вводить дані документів про освіту.

14. Після введення всіх даних та врахування результатів психологічного тесту, якщо користувач погодився його проходити, система на основі цих даних обирає найбільш прийнятні для користувача спеціальності.
15. На екран виводяться результати рекомендації спеціальностей для користувача.
16. Кінець роботи системи.

2.3 **Розробка методу експертної рекомендації спеціальностей**

Метод експертної рекомендації спеціальностей повинен виконувати наступні функції:

– зчитування даних, введених користувачем. Цими даними є особисті вподобання користувача, дані з документів про освіту та результати психологічних тестів.
– аналіз даних користувача, визначення пріоритетних критеріїв та їх оцінка.
– врахування минулих результатів прогнозування для більшої точності та прийнятності рекомендації спеціальностей.

Для визначення найбільш прийнятних спеціальностей для конкретного користувача система збирає всі введені дані та групує їх. З таких груп можна віднести особисті вподобання користувача, оцінки з технічних дисциплін, соціальних дисциплін, результати психологічного тесту і т. д.. Після групування даних обраховуться коефіцієнти по кожній групі та відбувається розстановка пріоритетів для кожного коефіцієнту. Далі враховуються такі критерії, як, наприклад, найвищі бали з певних дисциплін. Якщо у користувача великі бали з соціальних дисциплін, то переваги будуть надаватись саме їм.

Після визначення всіх коефіцієнтів та визначення їх пріоритетності, система починає враховувати особисті вподобання користувача. Якщо розглянути вище наведений приклад, коли у користувача високі бали з соціальних дисциплін, але в його вподобання вони не входять, то по пріоритетності обираються інші наукові дисципліни. Після отримання

результату система записує його у базу даних для подальшого аналізу з метою покращення майбутніх результатів прогнозування.

Загальний алгоритм методу експертної рекомендації спеціальностей зображений на рисунку 2.2.

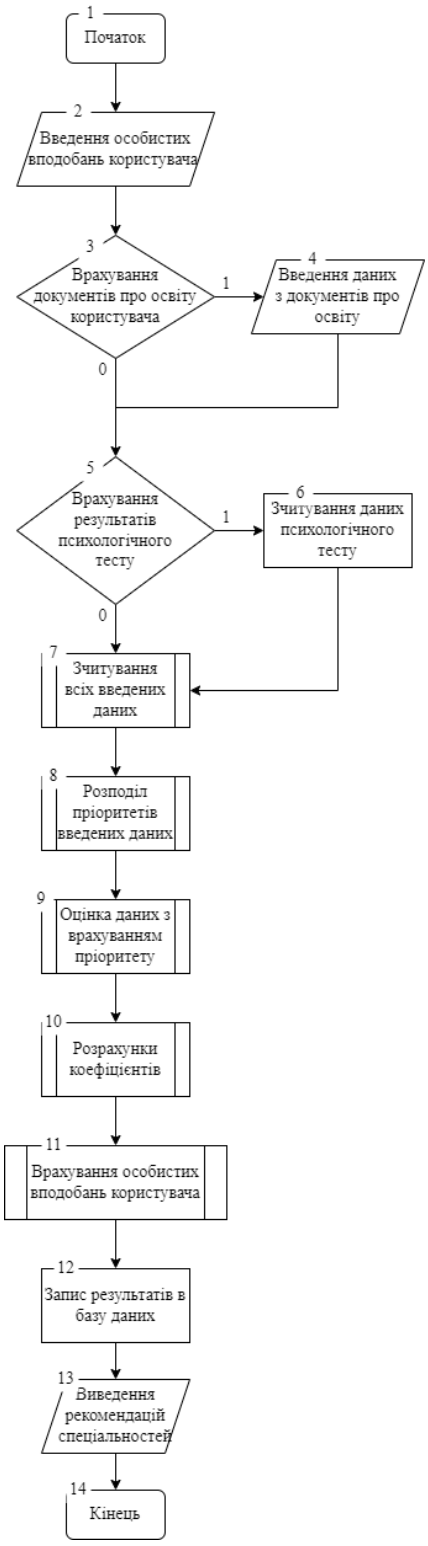

Рисунок 2.2 – Алгоритм методу експертної рекомендації спеціальностей

Опис алгоритму роботи методу експертної рекомендації спеціальностей:

1. Початок роботи підпрограми.
2. Користувач вводить особисті вподобання щодо майбутньої професії.
3. Користувач обирає чи проводити рекомендацію спеціальностей з врахуванням документів про освіту.
4. Користувач вводить дані з документів про освіту.
5. Користувач обирає чи враховувати результати психологічних тестів для рекомендації спеціальностей.
6. Система зчитує результати психологічного тесту користувача, якщо він його проходив.
7. Система зчитує всі введені раніше користувачем дані.
8. Проходить групування, розподіл даних на групи.
9. Система розставляє пріоритети для кожного з наявних критеріїв.
10. Розраховуються коефіцієнти, за якими в подальшому будуть визначатись найбільш прийнятні спеціальності.
11. Система враховує особисті вподобання користувача на розраховані результати та звіряє їх з коефіцієнтами для кожної окремої спеціальності.
12. Результат прогнозування записується у базу даних
13. На екран виводиться список рекомендованих спеціальностей для користувача.
14. Кінець роботи підпрограми.

2.4 Розробка методу статистичного аналізу результатів рекомендацій спеціальностей

Для більшої точності рекомендації спеціальностей система збирає всі результати роботи та аналізує їх. Вираховуються приблизні тенденції серед користувачів у вподобаннях та оцінках. Якщо, наприклад, за останній рік у користувачів підвищились бали з технічних дисциплін, то в подальшому

система буде рекомендувати користувачам з подібними даними саме технічні спеціальності.

Алгоритм методу статистичного аналізу результатів рекомендацій спеціальностей зображений на рисунку 3.3.

Рисунок 2.3 – Алгоритм методу статистичного аналізу результатів рекомендації спеціальностей

Опис алгоритму методу статистичного аналізу:

1. Початок роботи підпрограми.
2. Система зчитує всі результати рекомендацій з бази даних.
3. Проводиться аналіз всіх наявних даних за весь час та за останній період.
4. Система відслідковує закономірності у результатах на предмет схожості та зміни відносно попереднього періоду.
5. Проводиться розрахунок коефіцієнтів, який буде дораховуватись до результатів основного методу рекомендацій спеціальностей.
6. Кінець роботи підпрограми.

2.5 Висновки

У другому розділі розроблено та описано загальний та основні алгоритми системи експертної рекомендації спеціальностей. Було розроблено алгоритм методів експертної рекомендації спеціальностей, який враховує особисті вподобання користувача, дані з його документів про освіту та результати психологічних тестів. Також для вдосконалення майбутніх результатів роботи системи було розроблено алгоритм методу статистичного аналізу результатів рекомендацій, який буде враховувати основні зміни у останніх результатах користувачів та покращувати роботу основного методу рекомендації спеціальностей.

3 РОЗРОБКА ОСНОВНИХ МОДУЛІВ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Серед мов програмування для розробки програмного додатку розглядались такі мови як C++ , C#, Java та Python.

C++ є універсальною мовою програмування, яка підтримує різні парадигми, включаючи об’єктно-орієнтовану, узагальнену та процедурну.

Ця мова програмування має обширний функціонал, що дозволяє розробникам створювати складні програми. C++ підтримує різноманітні типи даних, такі як цілі числа, дійсні числа, символи, рядки та булеві значення. Крім того, вона надає розширені вбудовані функції для введення та виведення даних, роботи з файлами, рядками, обробки помилок тощо.

Однією з основних переваг C++ є висока швидкодія. Це робить мову однією з найшвидших у програмуванні, дозволяючи розробникам створювати продуктивні програми. Більше того, програми, написані на C++, можуть без проблем працювати на різних операційних системах, таких як Windows, Linux, macOS та інші.

Незважаючи на свої переваги, C++ має кілька недоліків. Серед них складність мови, вимагаюча великих зусиль для написання програм. Також важливо відзначити відсутність автоматичного збирача сміття, що може призвести до помилок в програмах. Крім того, відсутність чітких стандартів може викликати проблеми з переносом програм між різними операційними системами.

Також важливо відзначити, що незважаючи на здатність працювати на різних операційних системах, важливо враховувати, що програми на C++ можуть вимагати додаткових налаштувань для забезпечення сумісності [24].

Однією з істотних рис C++ є його велика ефективність та можливість оптимізації коду для досягнення високої продуктивності. Це особливо важливо

для розробників, які працюють над великими проектами або вимагають високої швидкодії в обчислювально інтенсивних застосунках.

З іншого боку, недоліки мови C++ полягають у її складності, що може стати перешкодою для новачків у програмуванні. Написання коду на C++ вимагає ретельного управління пам'яттю та уникнення потенційних помилок, пов'язаних з відсутністю автоматичного збирача сміття.

Також варто відзначити, що розробники C++ повинні бути обізнані з конкретними особливостями кожної платформи, на якій вони планують використовувати свої програми. Це може бути викликано відсутністю чітких стандартів, які дозволяють однозначно писати код, сумісний з будь-якою операційною системою.

У підсумку, C++ є потужним інструментом для розробників, які цінують швидкодію та високий рівень контролю над системою, але використання його вимагає глибокого розуміння мови та уважності до деталей.

Мова програмування C#, розроблена компанією Microsoft, визначається своєю великою популярністю у сфері розробки програмного забезпечення, охоплюючи широкий спектр застосувань від веб-сайтів до мобільних додатків та ігор. Ця мова підтримує різні парадигми програмування, включаючи об’єктно-орієнтовану, узагальнену та процедурну.

В C# існує розгалужений набір функціоналу, який дозволяє розробникам створювати складні програми. Мова підтримує різноманітні типи даних, такі як цілі числа, дійсні числа, символи, рядки та булеві значення. Крім цього, вона має обширний перелік вбудованих функцій, які спрощують роботу з введенням та виведенням даних, опрацюванням файлів та рядків, а також обробкою помилок та іншими завданнями [25].

Простий синтаксис C# є однією з ключових переваг цієї мови, що дозволяє розробникам швидко створювати програми. Натомість, вбудована підтримка для асинхронного програмування робить його ефективним інструментом для створення програм, які ефективно взаємодіють з мережевими запитами та іншими асинхронними операціями. Крім того, інтегрована

підтримка для LINQ (Language Integrated Query) робить працю з базами даних та іншими джерелами даних більш зручною для розробників.

Основні функції та можливості C#:

– об'єктно-орієнтована парадигма: C# є повністю об'єктно-орієнтованою мовою, що сприяє створенню модульних та легко розширюваних програм.
– інтегрована середовище розробки (IDE): C# розроблено для використання в середовищі Visual Studio, що надає розробникам різні інструменти для швидкого написання, налагодження та аналізу коду.
– широкий функціонал: мова має розгалужений функціонал, що дозволяє розробникам створювати різноманітні застосунки - від веб-сайтів і мобільних додатків до ігор.
– простий синтаксис: C# володіє лаконічним та читабельним синтаксисом, що полегшує написання коду та прискорює розробку.
– підтримка асинхронного програмування: інтегрована підтримка асинхронного коду дозволяє створювати ефективні програми для роботи з мережевими операціями.
– LINQ (Language Integrated Query): дозволяє розробникам взаємодіяти з базами даних та іншими джерелами даних з використанням декларативного синтаксису.

Переваги C#:

– простота вивчення: спрощений синтаксис робить C# доступною для новачків у програмуванні;
– висока продуктивність: забезпечує високу швидкодію в порівнянні з іншими мовами програмування;
– інтеграція з платформою Microsoft: взаємодіє з іншими технологіями в екосистемі Microsoft, такими як .NET і Azure.

Недоліки C#:

– обмежена підтримка платформ: хоча існують ініціативи, такі як .NET Core, але спочатку C# була пов'язана з платформою Windows;

– неявний збірник сміття (Garbage Collection): відсутність повного контролю над збором сміття може впливати на продуктивність у деяких сценаріях;
– менша підтримка для розробки під вбудовані системи: деякі вбудовані системи можуть вимагати мов, таких як C або C++, які надають більший рівень контролю над апаратним забезпеченням.

Java – це одна з найпопулярніших мов програмування, яка використовується для розробки різноманітних додатків, від веб-сайтів до мобільних додатків та ігор. Java є об’єктно-орієнтованою мовою програмування, що означає, що вона базується на концепції об’єктів, які містять дані та функції, які працюють з цими даними.

Java має багатий функціонал, який дозволяє розробникам створювати складні програми. Мова програмування Java підтримує різні типи даних, такі як цілі числа, дійсні числа, символи, рядки та булеві значення. Крім того, Java має багато вбудованих функцій, які дозволяють розробникам працювати зі звичайними завданнями, такими як введення та виведення даних, робота з файлами та рядками, обробка помилок та інші [26].

Однією з основних переваг мови Java є її кросплатформенність. Програми, написані на Java, можуть працювати на різних операційних системах, таких як Windows, Linux, macOS та інші. Крім того, Java має вбудовану підтримку для мережевого програмування, що дозволяє розробникам створювати програми, які працюють з мережевими запитами та іншими асинхронними операціями.

Java має простий синтаксис, який дозволяє розробникам швидко створювати програми. Крім того, Java має вбудовану підтримку для багатопоточності, що дозволяє розробникам створювати програми, які можуть виконувати кілька завдань одночасно. Нарешті, Java має вбудовану підтримку для використання бібліотек, що дозволяє розробникам використовувати готові рішення для різних завдань.

Незважаючи на свої переваги, мова програмування Java має деякі недоліки. Наприклад, Java може бути повільною порівняно з іншими мовами програмування, такими як C++. Крім того, Java може бути складною мовою програмування для новачків, оскільки вона вимагає ретельного використання концепцій об'єктно-орієнтованого програмування та інших аспектів.

Переваги Java:

– кросплатформенність: програми, написані на Java, можуть працювати на різних операційних системах, що забезпечує високий рівень переносимості.
– об'єктно-орієнтованість: сприяє модульності, гнучкості та повторному використанню коду.
– багатопоточність: вбудована підтримка багатопоточності для ефективної обробки паралельних завдань.
– широка спільнота та екосистема: велика та активна спільнота розробників, багато готових бібліотек та фреймворків для розробки.
– простий синтаксис: лаконічний та читабельний синтаксис полегшує розробку та утримування коду.
– мережеве програмування: інтегрована підтримка мережевого програмування для розробки розподілених систем.

Недоліки Java:

– швидкодія: порівняно з деякими мовами (наприклад, C++), Java може бути менш ефективною у виконанні вимог щодо швидкодії.
– високе споживання пам'яті: деякі додатки на Java можуть вимагати значної кількості пам'яті.
– серйозність: для новачків Java може здатися складною мовою через велику кількість концепцій та правил.
– недоліки у веб-розробці: у порівнянні з деякими модними фреймворками, розробка веб-застосунків на Java може виглядати менш гнучкою.

Python – це високорівнева, інтерпретована мова програмування загального використання, яка визначається своєю читабельністю та простотою синтаксису. з ключових рис мови є філософія «зробити код читабельним», що робить її ідеальним вибором для початківців та розробників з різних областей. Python підтримує об'єктно-орієнтований, функціональний та імперативний стилі програмування, що надає йому велику гнучкість.

Мова має величезну екосистему бібліотек і модулів, включаючи популярні такі як NumPy для роботи з науковими обчисленнями, Pandas для обробки та аналізу даних, або Flask та Django для веб-розробки. Python також є мовою вибору для штучного інтелекту, машинного навчання та обробки природної мови [27].

Ще однією сильною стороною Python є його спільнота, яка активно сприяє розвитку та підтримці мови. Оголошення принципів «PEP» (Python Enhancement Proposals) дозволяють структуровано внести нові функції та покращення в мову. Python є платформою-нейтральним інструментом, що робить його доступним для різних операційних систем, від Windows до Linux та macOS.

Переваги Python:

1. Простий синтаксис: Однією з головних переваг є легкий та зрозумілий синтаксис, що робить мову доступною для новачків та забезпечує швидке написання коду.
2. Об'єктно-орієнтованість: Python базується на концепції об'єктно-орієнтованого програмування, що сприяє модульності та повторному використанню коду.
3. Багатофункціональність: Python дозволяє розробникам створювати різноманітні програми, від веб-сайтів і мобільних додатків до аналітики даних та штучного інтелекту.
4. Багатофункціональні бібліотеки: Розширені бібліотеки, такі як NumPy, Pandas та Matplotlib, роблять Python потужним інструментом для обробки даних та створення графіків.

5. Багатопоточність та асинхронність: Вбудована підтримка багатопоточності і асинхронного програмування для ефективної обробки паралельних завдань.

Недоліки Python:

1. Швидкодія: У порівнянні з іншими мовами, такими як C++ або Java, Python може виявитися менш продуктивним у виконанні великих обчислювальних завдань.
2. Обмежена мобільність: Хоча Python використовується для розробки мобільних додатків, його використання не таке поширене, як, наприклад, у випадку мови Swift для iOS.
3. Високий рівень витрат пам'яті: Деякі програми на Python можуть вимагати більше пам'яті порівняно з іншими мовами програмування.
4. Менша швидкість у порівнянні з C++: Хоча Python зручний, він може бути менш ефективним для деяких задач, де потрібна велика швидкість виконання.

Проаналізувавши мови програмування, було складено таблицю (табл. 1.2), яка демонструє відмінності Java, Python, C#, C++.

Таблиця 3.1 – Функціональні характеристики мов програмування

Критерії	Java	Python	C#	C++
Швидкість обробки	0	0	1	1
Простота	0	0	1	0
Крос-платформеність	1	1	0	1
2D графіка	1	1	1	1
Графічний інтерфейс користувача	1	1	1	1
Спец. процесор	0	0	1	0
Сумарний коефіцієнт	2	2	6	5

Проаналізувавши дані таблиці, C# краща за Java, Python та C++ на 50% (100%-3/6*100% = 50%)

Оскільки мова програмування C# одна з найбільш вживана в світі, для неї існує широкий спектр середовищ розробки. Для порівняння було обрано такі середовища як, Visual Studio 2022, Visual Studio Code, Rider IDE.

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і ігри та програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight [28].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

Visual Studio визнана своєю потужністю та розширюваністю в області розробки програмного забезпечення. Забезпечуючи редактор вихідного коду з інтегрованою технологією IntelliSense і функціоналом для легкого рефакторінгу коду, ця інтегрована середовище розробки дозволяє зручно працювати з програмним кодом. Вбудований відладчик надає можливість відлагодження як

на рівні вихідного коду, так і на рівні машинного коду, що робить процес відлагодження більш ефективним. Помітними елементами Visual Studio є редактор форм для спрощення створення графічного інтерфейсу, веб-редактор, дизайнер класів і схеми баз даних. Крім того, розширюваність Visual Studio виходить за межі базового функціоналу, дозволяючи створювати та підключати сторонні додатки, такі як плагіни, для розширення можливостей розробки. Це може включати підтримку систем контролю версій, нові інструменти для редагування коду та візуального проектування, а також інші інструменти, що полегшують роботу розробників на різних етапах розробки програмного забезпечення.

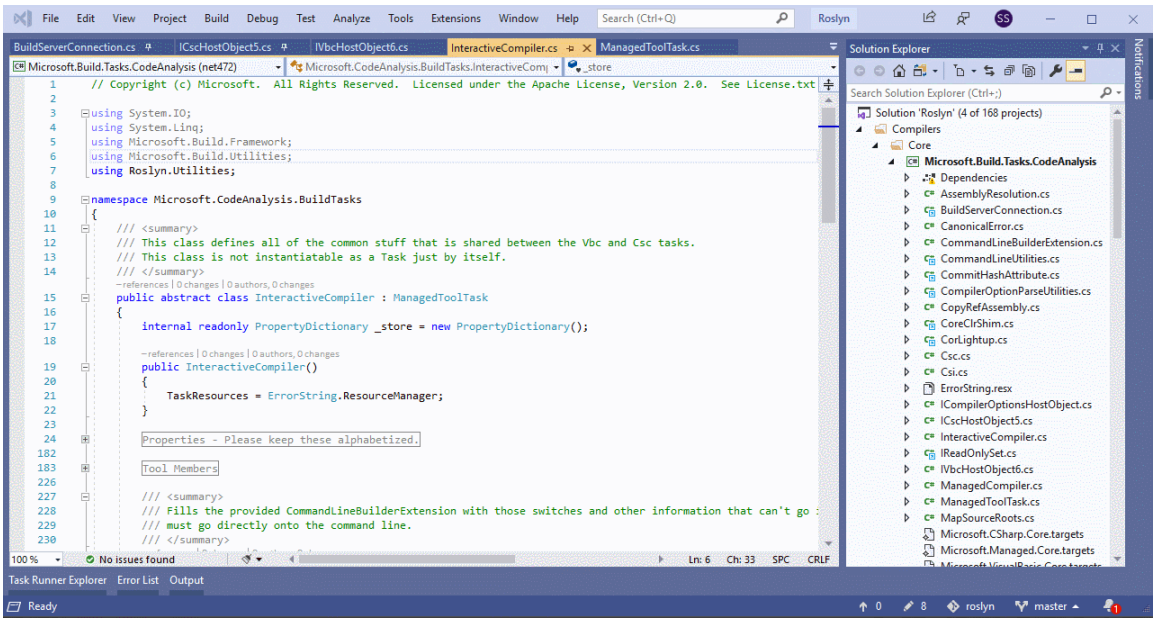

Рисунок 3.1 – Інтерфейс програмного додатку Visual Studio 2022

Visual Studio Code – це редактор вихідного коду, який можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. Він базується на структурі Electron, яка використовується для розробки веб-додатків Node.js, які працюють на механізмі компонування Blink. Visual Studio Code використовує той самий компонент редактора (під кодовою назвою «Monaco»), який використовується в

Azure DevOps (раніше називався Visual Studio Online і Visual Studio Team Services).

З коробки Visual Studio Code містить базову підтримку для більшості поширених мов програмування. Ця базова підтримка включає підсвічування синтаксису, зіставлення дужок, згортання коду та настроювані фрагменти. Visual Studio Code також постачається з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML, а також підтримує налагодження Node.js. Підтримка додаткових мов може бути забезпечена безкоштовними розширеннями на VS Code Marketplace.

Замість системи проектів вона дозволяє користувачам відкривати один або кілька каталогів, які потім можна зберегти в робочих областях для подальшого використання. Це дозволяє йому працювати як мовно-агностичний редактор коду для будь-якої мови. Він підтримує багато мов програмування та набір функцій, який відрізняється для кожної мови. Небажані файли та папки можна виключити з дерева проекту за допомогою налаштувань. Багато функцій Visual Studio Code не доступні через меню чи інтерфейс користувача, але доступ до них можна отримати через палітру команд.

Код Visual Studio можна розширити за допомогою розширень, доступних через центральне сховище. Це включає доповнення до редактора та підтримку мови. Примітною функцією є можливість створювати розширення, які додають підтримку нових мов, тем, налагоджувачів, налагоджувачів подорожей у часі, виконують статичний аналіз коду та додають лінтери коду за допомогою протоколу Language Server.

Керування джерелом є вбудованою функцією Visual Studio Code. Він має спеціальну вкладку всередині панелі меню, де користувачі можуть отримати доступ до налаштувань керування версіями та переглянути зміни, внесені до поточного проекту. Щоб використовувати цю функцію, Visual Studio Code має бути зв’язано з будь-якою підтримуваною системою керування версіями (Git, Apache Subversion, Perforce тощо). Це дозволяє користувачам створювати

репозиторії, а також робити запити push і pull безпосередньо з програми Visual Studio Code [29].

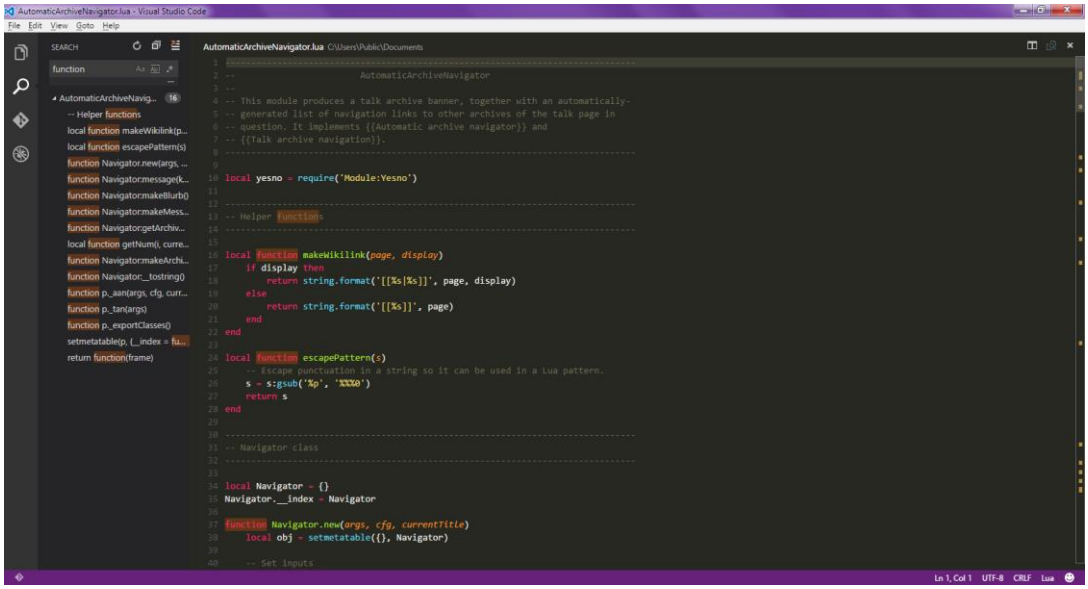

Рисунок 3.2 – Інтерфейс програмного додатку Visual Studio Code

Rider IDE від JetBrains – це потужне та інноваційне середовище розробки для платформи .NET, яке надає розробникам широкий спектр інструментів для створення високоякісних програм на мовах, таких як C#, VB.NET, F# та інші. Заснована на технології ReSharper, Rider пропонує вдосконалене автодоповнення, аналіз коду в реальному часі, а також інтеграцію з різними системами керування версіями, включаючи Git та SVN. Його інтуїтивний та дружній інтерфейс робить розробку більш продуктивною, а підтримка платформи .NET Core робить його ідеальним вибором для роботи з сучасними, кросплатформовими проектами. Забезпечуючи розширені можливості рефакторингу, підтримку тестування, інтеграцію з популярними фреймворками та інші продуктивні функції, Rider IDE виступає як важливий інструмент для професійних розробників, що працюють в екосистемі .NET.

Інтеграція з популярними серверами веб-розробки, такими як IIS та Kestrel, дозволяє зручно виконувати та тестувати веб-додатки. Rider також надає велику кількість плагінів та розширень для налаштування середовища розробки згідно із специфічними потребами розробника. Із спрощеним

процесом встановлення та постійними оновленнями, Rider IDE стала однією з популярних виборів для .NET розробки, об'єднуючи в собі високу продуктивність та комфортність роботи [30].

Rider IDE має кілька вагомих переваг, які роблять його привабливим вибором для розробників. Перш за все, це кросплатформенність, що дозволяє працювати на операційних системах Windows, macOS та Linux. Висока продуктивність та широкий спектр функцій для розробки .NET додатків, включаючи ASP.NET, також роблять Rider привабливим для спеціалістів. Велика кількість плагінів та можливість інтеграції з іншими інструментами, такими як системи контролю версій, сприяють зручності роботи. Окрім того, активна підтримка від компанії JetBrains, регулярні оновлення та спільнота користувачів роблять Rider популярним серед розробників.

Деякі користувачі можуть вважати, що інтерфейс не настільки інтуїтивно зрозумілий, особливо для тих, хто перейшов з інших інтегрованих середовищ розробки. Також варто врахувати, що, хоча Rider підтримує багато мов програмування, його найбільша сила полягає в розробці на платформі .NET, тому для проектів на інших мовах можуть існувати кращі альтернативи.

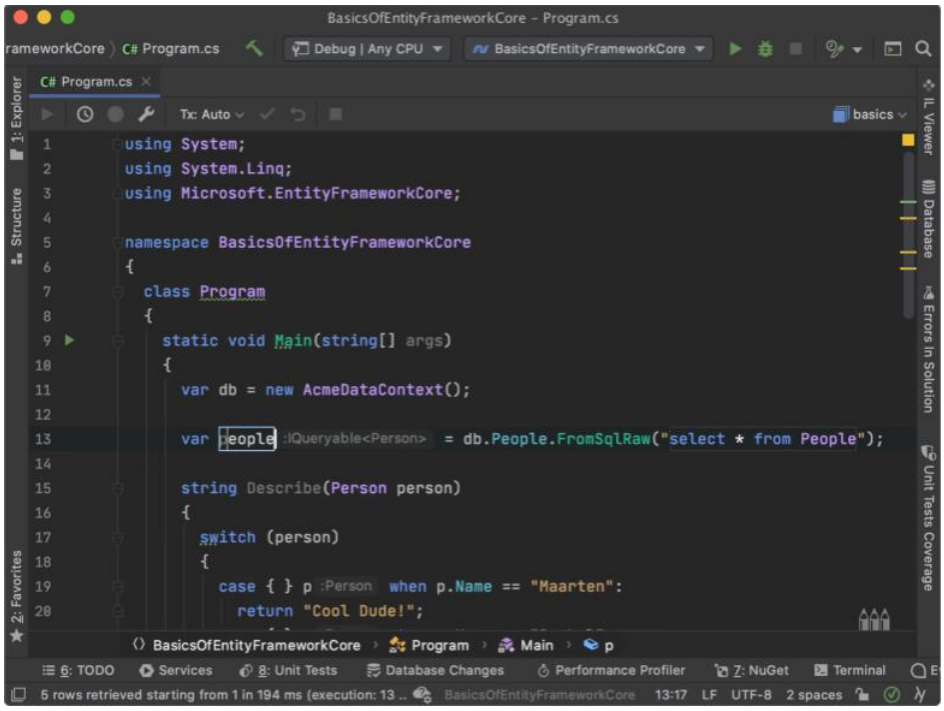

Рисунок 3.3 – Інтерфейс програмного додатку Rider IDE

MonoDevelop – відкрите інтегроване середовище розробки для платформ Linux, Mac OS X та Microsoft Windows, передусім націлене на розробку програм, які використовують і Mono, і Microsoft .NET Framework. На даний момент підтримуються мови C#, Java, Boo, Visual Basic.NET, CIL, Python, Vala, C та C++. Також MonoDevelop підтримує такі технології, як Gtk#, ASP.NET MVC, Silverlight, MonoMac и MonoTouch.

MonoDevelop включає можливості подібні до NetBeans та Microsoft Visual Studio, такі як автоматичне доповнення, інтеграція контролю коду, графічний користувацький інтерфейс і вебдизайнер. В MonoDevelop інтегрований Gtk# GUI дизайнер під назвою Stetic.

Можливості MonoDevelop:

– підсвітка синтаксису;
– згортання коду;
– автодоповнення коду;
– браузер класів;
– підтримка плагінів;
– вбудований відлагоджувач;
– візуальний конструктор форм (GTK#);
– модульне тестування.

MonoDevelop дозволяє розробникам швидко писати настільні та веб-програми для Linux, Windows і macOS. Це також дозволяє розробникам легко переносити програми .NET, створені за допомогою Visual Studio, на Linux і macOS, зберігаючи єдину кодову базу для всіх платформ [31].

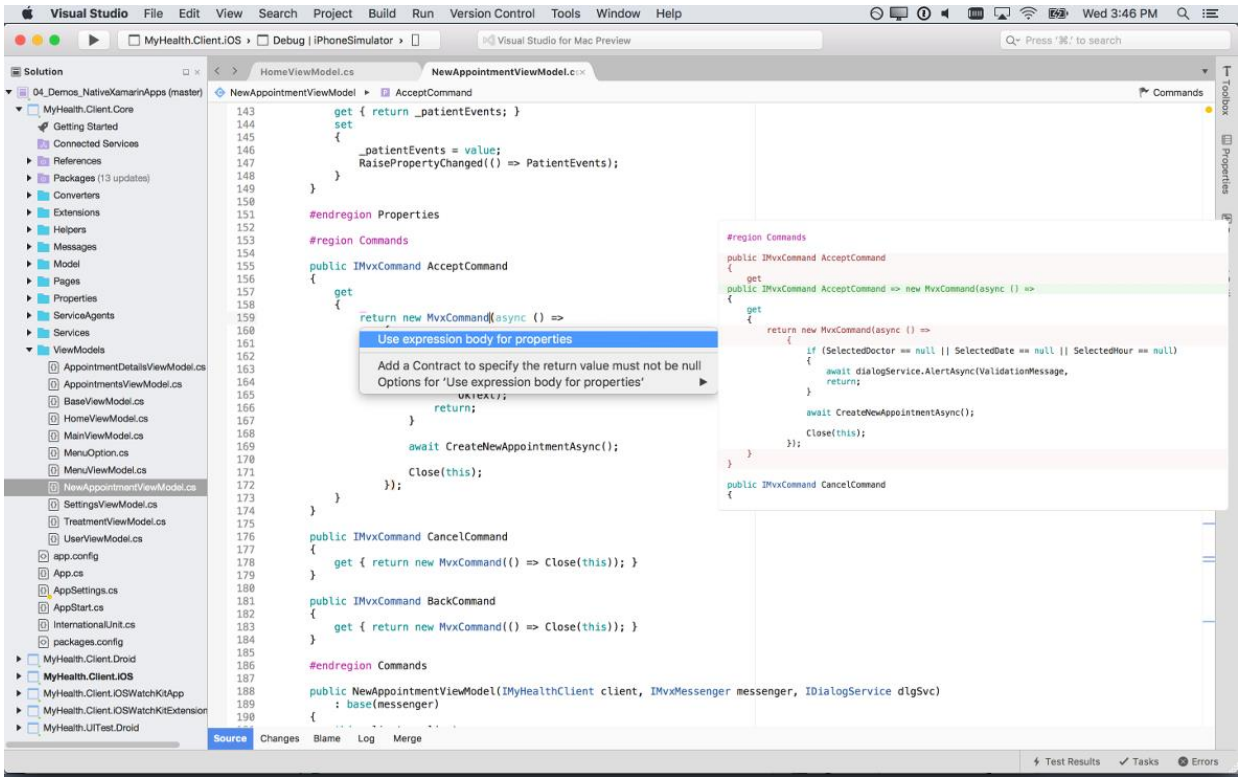

Рисунок 3.4 – Інтерфейс програмного додатку MonoDevelop

Порівняння всіх вище вказаних середовищ розробки наведено в таблиці 1.3.

Таблиця 3.2 – Функціональні характеристики середовищ розробки

	Visual Studio 2022	Visual Studio Code	Rider IDE	MonoDevelop
Підтримка Windows	1	1	1	1
Розробка GUI	1	0	1	1
Статистичний аналіз коду	1	1	1	1
Профілювання	1	0	0	0
Покриття коду	1	0	0	1
Сумарний коефіцієнт	5	2	3	4

Аналізуючи вищенаведені дані, Visual Studio має вищий сумарний коефіцієнт відносно аналогів: Visual Studio Code на 60% (100%-2/5*100% = 60%), Rider IDE на 40% (100%-3/5*100% = 40%) та MonoDevelop на 20% (100%-4/5*100%).

Отже, в результаті проведеного аналізу середовищ розробки, для створення додатку було обрано середовище Visual Studio 2022.

Оскільки для розробки системи експертної рекомендації спеціальностей мовою програмування було обрано C#, то найкращим рішенням для оформлення веб-платформи була технологія ASP.NET.

ASP (Active Server Pages) – це технологія, що дозволяє створювати динамічні веб-сайти та веб-додатки. З основних функцій можна виділити можливість генерації динамічних веб-сторінок на сервері, інтеграцію з різними базами даних, включаючи Microsoft SQL Server, та контроль засобів безпеки, таких як автентифікація та авторизація. ASP також використовує .NET Framework для розширення функціональності веб-додатків. Синтаксис ASP досить простий і базується на використанні вбудованого коду на мові VBScript або JScript, який виконується на стороні сервера.

ASP.NET, розроблене Microsoft на основі платформи .NET, є потужним інструментом для веб-розробки. Завдяки своєму широкому функціоналу, розробники можуть створювати різноманітні веб-застосунки, від корпоративних систем до електронних магазинів. Платформа підтримує різні мови програмування, такі як C#, VB.NET та F#. З тісною інтеграцією з продуктами Microsoft, такими як Azure, Visual Studio та SQL Server, ASP.NET спрощує розробку, розгортання та управління веб-застосунками, забезпечуючи вбудовані механізми безпеки та ефективне керування сесіями. Незважаючи на свої переваги, такі як масштабованість та висока продуктивність, ASP.NET також має свої виклики, такі як складність для новачків та високі витрати на ресурси для великих веб-застосунків. З метою подолання цих викликів, можливі стратегії для майбутнього розвитку включають збільшення простоти використання, оптимізацію витрат ресурсів та розширення інтеграції з

новітніми технологіями. Мета – підтримка високого стандарту веб-розробки та розширення співпраці зі спільнотою користувачів для постійного вдосконалення та розвитку ASP.NET [32].

3.2 Розробка основного модулю системи

Модель-Вигляд-Контролер (MVC) – це архітектурний шаблон проектування програмного забезпечення, який розділяє програму на три основні компоненти: Модель, Вигляд і Контролер. Модель відповідає за управління даними та бізнес-логікою, представляючи об'єкти, які можуть бути використані для зберігання та обробки інформації. Вигляд відповідає за відображення даних із Моделі користувачу та взаємодію з ним, створюючи інтерфейс для взаємодії. Контролер відповідає за обробку введення користувача та виклик методів Моделі або Вигляду відповідно до цього введення. Одна з основних переваг MVC - це розділення відповідальностей, що полегшує розробку, тестування та підтримку програмного забезпечення. Цей шаблон також забезпечує гнучкість і перевикористання коду, оскільки компоненти можуть функціонувати незалежно один від одного, забезпечуючи відокремленість та прозорість системи [33].

За виведення даних відповідають контролери Home (основні функції системи), Admin (адмін-панель), Speciality (модуль рекомендації спеціальностей та статистичного аналізу) та Psychology (психологічні тести).

Для відображення даних використовуються моделі MainViewModel (основне меню системи) та SpecialityViewModel (інформація про рекомендовані спеціальності).

3.3 Розробка модулів експертної рекомендації спеціальностей та статистичного аналізу результатів

Модуль SpecialityRecommend відповідає за рекомендацію спеціальностей на основі вподобань користувача, даних з документів про освіту та результатів психологічних тестів. Користувач може обирати на основі чого буде

проводитись рекомендація, але обов’язковим критерієм є вподобання користувача. Вони всі представлені у вигляді списку, тому користувач обирає для себе декілька критеріїв, які для нього важливі. При виборі психологічних тестів система пропонує користувачеві пройти психологічний тест орієнтовно на 10 хв для визначення параметрів, які важливі для врахування майбутньої професії.

Також є можливість обрати дані з документів про освіту. Користувач вводить бали з кожної дисципліни і на основі цих даних модуль визначає з яких саме напрямів у користувача найвищі бали, після чого додає їх в порівняння з особистими вподобаннями та результатами психологічного тесту. На основі цього порівняння користувач отримує список спеціальностей, які система вважає найбільш прийнятними для поточного користувача.

Також на роботу модуля SpecialityRecommend впливає інший модуль StatAnalysis, який відповідає за статистичний аналіз результатів рекомендацій. Він враховує всі результати за весь час та за поточний період, відслідковує закономірні зміни у результатах користувачів, після чого формує коефіцієнти, які впливають на подальші результати рекомендації спеціальностей.

3.4 Розробка інтерфейсу користувача

Розробка зручного та сучасного інтерфейсу вимагає комплексного підходу, спрямованого на задоволення потреб користувачів. Першочерговим завданням є створення інтуїтивно зрозумілої навігації, яка дозволяє швидко та легко отримувати доступ до ключових розділів. Графічний дизайн повинен бути чистим та естетично приємним, з врахуванням сучасних тенденцій у веб-дизайні. Важливо забезпечити адаптивність до різних пристроїв та екранних розмірів, щоб забезпечити однаково комфортний досвід використання як на комп'ютерах, так і на мобільних пристроях. Застосування яскравих та візуально привабливих елементів, разом із заповненими розділами контенту, сприяє залученню користувача та створює позитивний імідж веб-сайту. Регулярні оновлення та аналіз поведінки користувачів дозволяють адаптувати інтерфейс

до змінних потреб аудиторії, роблячи його найбільш зручним та відповідним сучасним стандартам.

У процесі розробки зручного та сучасного інтерфейсу, ключовим є комплексний підхід, орієнтований на задоволення потреб користувачів. Починаючи з створення інтуїтивно зрозумілої навігації, завданням є забезпечення швидкого та легкого доступу до ключових розділів. Графічний дизайн має бути чистим і естетично приємним, враховуючи останні тенденції у веб-дизайні. Адаптивність до різних пристроїв та екранних розмірів є важливою, щоб забезпечити однаково комфортний досвід використання як на комп'ютерах, так і на мобільних пристроях. Використання відмітних та візуально привабливих елементів, спільно з заповненими розділами контенту, сприяє залученню користувача та формує позитивний імідж веб-сайту. Регулярні оновлення та аналіз поведінки користувачів дозволяють адаптувати інтерфейс до змінних потреб аудиторії, роблячи його максимально зручним та відповідним сучасним стандартам.

Загалом, для системи були розроблені такі веб-сторінки:

1. Головна сторінка сайту (містить основну інформацію про систему) (рисунок 3.5).
2. Сторінка з інформацією про систему.

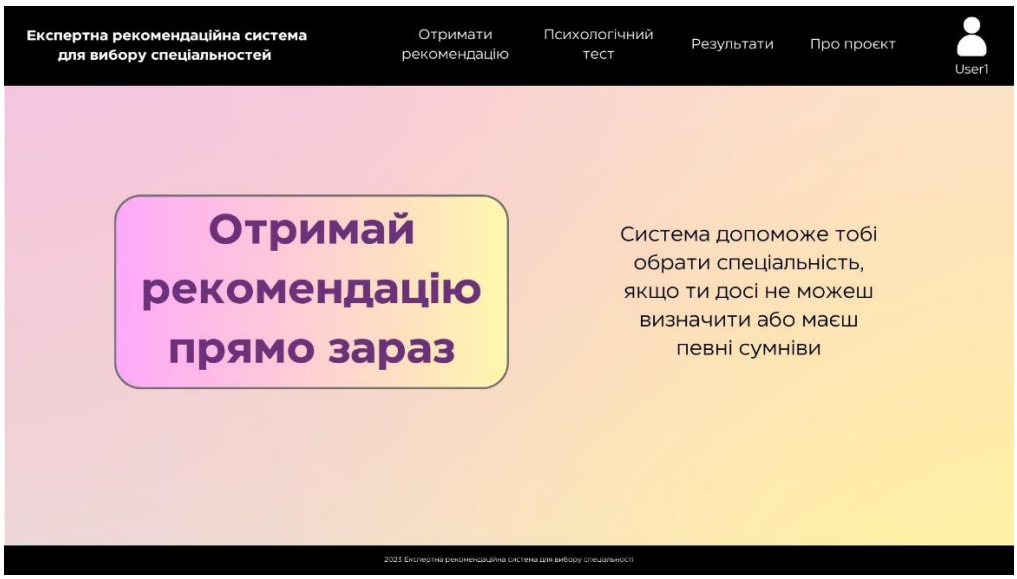

Рисунок 3.5 – Головна сторінка системи

3. Сторінка з вибором параметрів для рекомендації спеціальностей (тут можна обрати за якими саме параметрами буде проводитись прорахування спеціальності для користувача та, відповідно, вказати ці параметри) (рисунок 3.6).

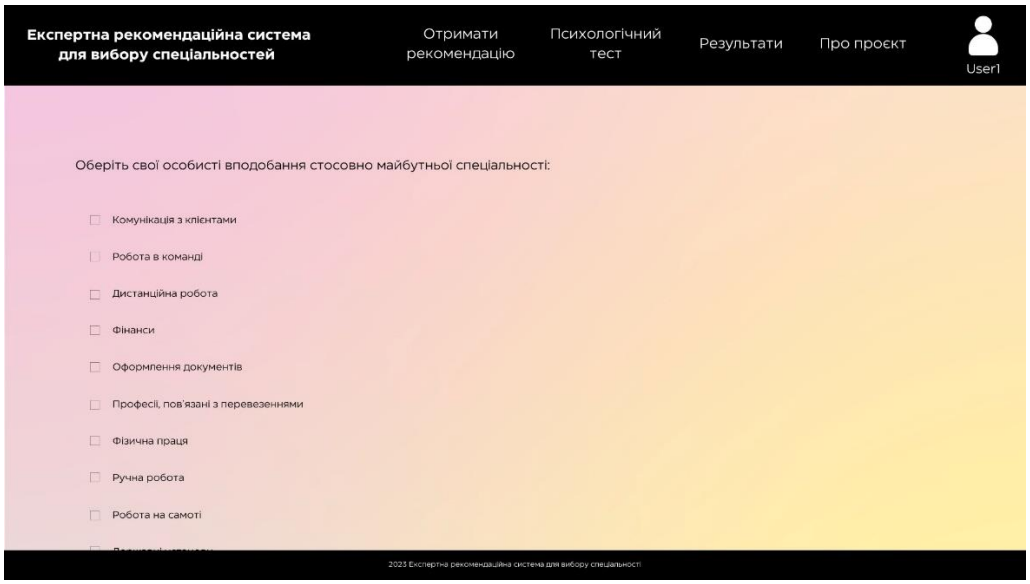

Рисунок 3.6 – Сторінка з вибором параметрів для рекомендації спеціальності

4. Сторінка з психологічним тестом (рисунок 3.7).

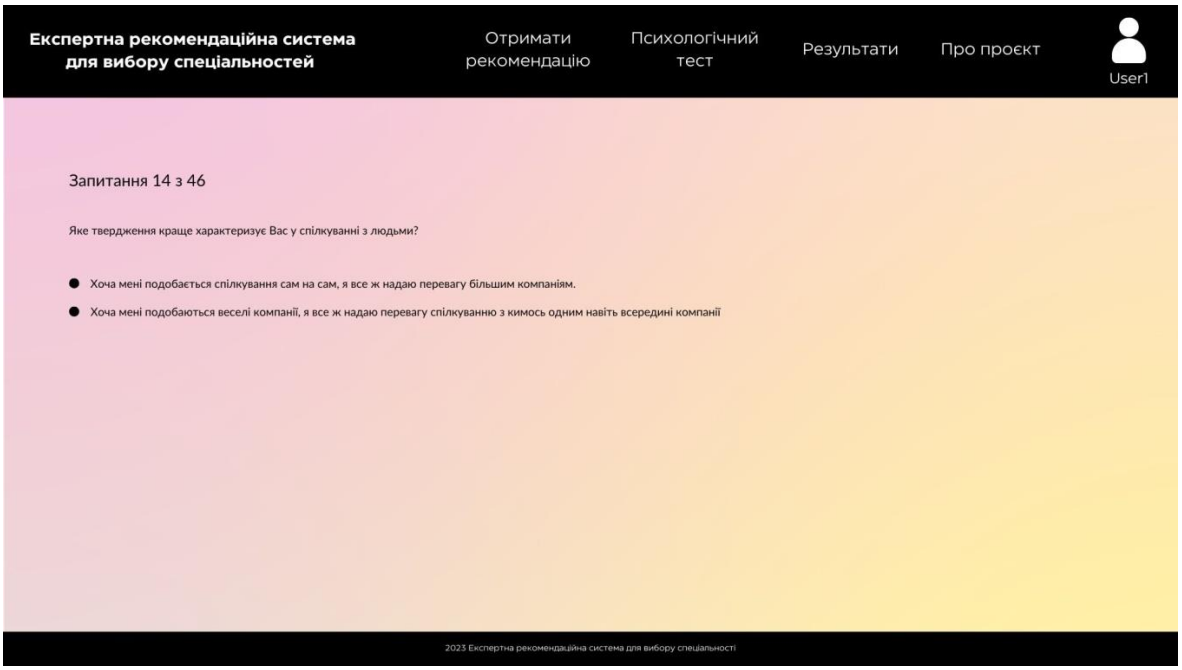

Рисунок 3.7 – Сторінка з психологічним тестом

5. Сторінка з результатами роботи системи (містить інформацію про рекомендовані спеціальності для поточного користувача) (рисунок 3.8).

Рисунок 3.8 – Сторінка з результатами роботи системи

3.5 Висновки

У третьому розділі було проведено варіантний аналіз різноманітних мов програмування, середовищ, засобів та технологій розробки системи експертної рекомендації спеціальностей. У результаті аналізу було прийнято рішення використовувати мову програмування C#, середовище розробки Visual Studio 2022 та веб-платформа ASP.NET.

Розроблено основний функціонал системи експертної рекомендації спеціальностей.

Розроблено модуль рекомендації спеціальностей за різними критеріями для точного прогнозування майбутньої професії та модуль статистичного аналізу результатів для покращення результатів майбутніх прогнозувань.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Аналіз методів та засобів тестування

У сучасному інформаційному суспільстві, де роль програмного забезпечення вирішальна, забезпечення якості стає стратегічним завданням для будь-якого проекту розробки програм. Методи тестування є ключовою складовою цього процесу, спрямованою на визначення якості та надійності програмного продукту перед його впровадженням у виробництво.

Серед методів тестування були розглянуті наступні:

1. Модульне тестування (Unit Testing): Модульне тестування спрямоване на перевірку окремих частин (модулів або функцій) програми. Розробники створюють тестові випадки, щоб перевірити правильність роботи окремих блоків коду. Це сприяє виявленню та виправленню помилок на ранніх етапах розробки, підвищуючи загальну стабільність програми.
2. Інтеграційне тестування (Integration Testing): Під час інтеграційного тестування використовуються тести для перевірки взаємодії між різними частинами системи, такими як модулі чи компоненти. Це допомагає виявляти проблеми, що можуть виникнути при об'єднанні окремих частин програми та забезпечує їх коректну взаємодію.
3. Функціональне тестування (Functional Testing): Функціональне тестування ставить за мету перевірку відповідності програми функціональним вимогам. Тести виконуються для переконання, що всі функції та опції програми працюють вірно та задовольняють вимоги користувачів.
4. Автоматизоване тестування (Automated Testing): Автоматизоване тестування використовує інструменти та скрипти для автоматизації виконання тестів. Це дозволяє збільшити ефективність тестування, зменшити кількість ручного втручання та швидше виявляти помилки.

5. Стрес-тестування (Stress Testing): Стрес-тестування проводиться для визначення та перевірки стійкості системи під великими навантаженнями. Тестується максимальна працездатність та реакція системи на екстремальні умови.
6. Тестування відмов (Fuzz Testing): Під час тестування відмов вводяться некоректні або випадкові дані з метою виявлення несправностей та вразливостей програми при неочікуваному введенні.
7. Тестування безпеки (Security Testing): Тестування безпеки спрямоване на виявлення потенційних вразливостей та заходів безпеки програми. Це включає тестування на проникнення, аналіз коду на вразливості та інші методи для забезпечення безпеки програмного продукту.
8. Регресійне тестування (Regression Testing): Регресійне тестування полягає в повторному виконанні тестів для виявлення нових помилок або змін у програмі, які можуть впливати на існуючий функціонал. Це сприяє виявленню та виправленню помилок, які можуть з'явитися після внесення змін.
9. Користувацьке тестування (User Acceptance Testing – UAT): Користувацьке тестування виконується користувачем або групою користувачів перед випуском продукту на ринок. Це дозволяє здобути зворотний зв'язок від реальних користувачів та переконатися, що програма відповідає їхнім очікуванням.
10. Тестування відмінностей (Compatibility Testing): Тестування відмінностей визначає, як програма працює на різних платформах, операційних системах та пристроях. Мета полягає в тому, щоб забезпечити, що програма сумісна з різними середовищами користувачів.
11. Альфа-тестування та бета-тестування: Альфа-тестування виконується в розробника або внутрішньому колективі для виявлення та виправлення проблем. Бета-тестування вже проводиться на етапі

після альфа-тестування та включає залучення зовнішніх користувачів для отримання зворотного зв'язку [34].

Кожен метод тестування має свою унікальну роль у забезпеченні якості програмного продукту. Поєднання цих методів в тестовій стратегії дозволяє виявляти різні види помилок та забезпечує повнокруглий підхід до тестування програмного забезпечення.

4.2 Етапи тестування програмного додатку

Ефективне тестування вимагає системного підходу та застосування методів, які дозволяють вичерпно перевірити функціональність. Для тестування системи рекомендації спеціальностей було застосовано метод TestCase.

Було розроблено перелік тест-кейсів для перевірки працездатності системи та перевірки її на можливі помилки. Перелік тест-кейсів та результати тестування наведені в таблиці 4.1.

Таблиця 4.1 – Перелік тест-кейсів

№	Назва етапу	Кроки	Очікуваний результат	Результат
1	Реєстрація	Натиснути «Реєстрація» Ввести логін та пароль Натиснути «Зареєструвати аккаунт»	Повідомлення про успішну реєстрацію та перехід до основного меню системи	Успіх
2	Авторизація	Натиснути «Авторизація» Ввести відповідний логін та пароль Натиснути «Увійти»	Перехід до основного меню системи	Успіх
3	Вибір параметрів для рекомендації спеціальностей	Натиснути «Рекомендація» Вибрати параметри, за якими будуть підбиратись спеціальності Натиснути «Отримати рекомендацію»	Вивід на екран списку рекомендованих спеціальностей	Успіх
4	Проходження психологічних тестів	Натиснути «Психологічний тест» Обрати відповіді на всі питання	Вивід на екран результатів тестів	Успіх

4.3 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Для початку користування системою спочатку потрібно в ній зареєструватись. Для цього потрібно ввести свою електронну адресу та придумати пароль до аккаунту. Після реєстрації користувач може обрати за якими параметрами він хоче визначити спеціальність для себе. Це можуть бути особисті вподобання, взяті за основу документи про освіту або визначення особистих якостей на основі психологічних тестів. Після проходження тестів користувач отримує результат, а саме список рекомендованих спеціальностей.

Деталі щодо мінімальної та рекомендованої конфігурації комп'ютера можна знайти в таблицях 4.2 та 4.3.

Таблиця 4.2 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) двоядерний процесор з тактовою частотою 1 ГГц
Об'єм оперативної пам'яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-розрядної системи
Операційна система	Windows 7 та вище

Таблиця 4.3 – Рекомендована конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) двоядерний процесор з тактовою частотою 2 ГГц
Об'єм оперативної пам'яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-разрядної системи
Операційна система	Windows 10 та вище

4.4 Висновки

У четвертому розділі було проведено тестування системи експертної рекомендації спеціальностей. Було перевірено реєстрацію користувачів, підсистему психологічних тестів та модулю рекомендації спеціальностей за різними критеріями. Також було розроблено інструкцію користувача для початку експлуатації програмного продукту.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного аудиту є оцінювання комерційного потенціалу впровадження методів та засобів розробленої системи експертної рекомендації спеціальностей.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету к.т.н., доцента Войтко В. В., к.т.н., доцента Ракитянську Г. Б., к.т.н., доцента Рейди О.М з кафедри програмного забезпечення.

Аудит науково-технічної розробки та її комерційного потенціалу проведено за допомогою таблиці 5.1, застосовуючи п’ятибальну шкалу оцінювання за 12-ма критеріями оцінки.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
#	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

1	2	3	4	5	6
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

В таблиці 5.2 наведено результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Войтко В. В.	2. Ракитянська Г. Б.	3. Рейда О. М.
	Бали, виставлені експертами:		
1	3	4	2
2	2	2	3
3	4	3	3
4	2	3	2
5	3	2	4
6	2	4	3
7	4	3	3
8	2	3	2
9	4	2	4
10	3	4	2
11	2	2	3
12	4	2	2
Сума балів	$СБ_1$=35	$СБ_2$=34	$СБ_3$=33
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{35 + 34 + 33}{3} = 34$		

В таблиці 5.3 наведено шкалу оцінки комерційного потенціалу розробки.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

За результатами розрахунків, наведених в таблиці 5.2, та шкалою оцінки, наведеної в таблиці 5.3, можна зробити висновок щодо рівня комерційного потенціалу розробки. Середньоарифметична сума балів, виставлених експертами, склала 34, що відповідає рівню «вище середнього».

Такий рівень комерційного потенціалу досягнуто за рахунок значного спрощення учням та студентам процесу пошуку майбутньої спеціальності. Використання додатку допоможе зручно та швидко визначити найбільш прийнятну спеціальність за особистими вподобаннями, на основі документів про освіту, особистих психологічних аспектів та з врахуванням раніше отриманих результатів роботи додатку.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- матеріали;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- програмне забезпечення для наукових робіт;
- інші витрати;
- накладні витрати.

5.2.1 Основна заробітна плата

Заробітна плата кожного із залучених осіб визначається за формулою (5.1)

$$З_0 = \frac{М}{Т_р} * t \text{ (грн)} \qquad (5.6)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для проектування і розробки системи експертної рекомендації спеціальностей було залучено наступних робітників: Frontend Developer та Backend Developer. Посадові оклади, число днів роботи та витрати на компенсацію наведено в таблиці 5.4.

Таблиця 5.4 – Компенсація спеціаліста в дослідницькій установі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Project Manager	30000	1363	36	49068
Frontend Developer	25000	1136	27	30672
Backend Developer	28000	1272	32	40704
Всього				120444

5.2.2 Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх робітників, які приймали участь в розробці нового технічного рішення розраховується за формулою (5.2) як 10-15% від основної заробітної плати робітників як премія [35].

На даному підприємстві додаткова заробітна плата нараховується в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.2)$$

$$З_д = 0{,}1 * 120444 = 12044 \text{ грн}$$

5.2.3 Нарахування на заробітну плату

Нарахування на заробітну плату $Н_{ЗП}$ робітників, які брали участь у виконанні роботи, розраховуються за формулою (5.3):

$$З_н = (З_о + З_р + З_д) * \frac{Н_{ЗП}}{100} \text{ (грн)} \quad (5.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$Н_{ЗП}$ – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %.

Основна ставка єдиного внеску на загальнообов'язкове державне соціальне страхування на 2023 рік – 22%, тоді:

$$З_н = (120444 + 12044) * 0.22 = 29147(\text{грн.})$$

5.2.4 Витрати на матеріали та комплектуючі вироби

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.4)$$

де Нj – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

Цj – вартість матеріалу j-го найменування, грн/кг;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj – маса відходів j-го найменування, кг;

Цвj – вартість відходів j-го найменування, грн/кг.

Потрібно закладати витрати на доставку у вигляді коефіцієнту транспортних витрат – 1.1. Інформацію про використані матеріали та комплектуючі наведено у таблиці 5.5.

Таблиця 5.5 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір офісний А4 білий 500 аркушів	180	1	180
Картридж для принтера HP LaserJet 1018	450	1	450
Всього			630
З врахуванням коефіцієнта транспортування			699

5.2.5 Витрати на програмне забезпечення

До даної статті входять витрати на програмне забезпечення, необхідне для проектування та розробки адаптивної системи тестування знань. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot C_{пргі} \cdot К_і, \quad (5.4)$$

де $Ц_{іпрг}$ – ціна придбання/використання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_і$ = 1,10…1,12).

k – кількість найменувань програмних засобів.

Отримані результати наведено в таблиці 5.6.

Таблиця 5.6 – Витрати на використання програмних засобів

Найменування устаткування	Час використання, місяців	Ціна за місяць, грн	Вартість, грн
Підписка Microsoft Visual Studio Professional 2022	2	1710	3420
Всього			3420

5.2.6 Амортизація обладнання

До даної статті включаються амортизаційні відрахування по кожному виду обладнання, устаткування яке використовувалось для проектування та розробки системи адаптивного тестування знань.

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (5.5)$$

де $Ц_б$ – балансова вартість даного виду обладнання (приміщень), грн.;

$t_{вик}$ – час користування;

$Т_в$ – термін використання обладнання (приміщень), цілі місяці.

Для розробки продукту використовувався персональний комп’ютер вартістю 56000 грн. Для тестування використовувався ноутбук вартістю 15000 грн. Амортизаційні відрахування наведено в таблиці 5.7.

Таблиця 5.7 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп’ютер	56000	5	2	1866
Ноутбук	15000	4	2	625
Офісне приміщення	1200000	20	2	10000
Всього				12491

5.2.7 Енергія для науково-виробничих цілей

До даної статті відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i} \quad (5.6)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Вартість електроенергії становить 7.6 грн за кВт в 2023 році. При розробці системи використовувались комп’ютер та ноутбук з сумарною потужністю 0.75 кВт. Сумарні витрати на електроенергію становлять:

$$В_е = \frac{0.75 \cdot 260 \cdot 7.6 \cdot 0.5}{0.86} = 1137 \text{ (грн.)}$$

5.2.8 Витрати на службові відрядження

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень. Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = \left(З_о + З_р\right) \cdot \frac{Н_{св}}{100}, \quad (5.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 120444 \cdot 0.22 = 26497 \text{ грн}$$

5.2.9 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Дана стаття витрат включає витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками. Такі витрати розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.9.

$$В_{сп} = \left(З_о + З_р\right) \cdot \frac{Н_{сп}}{100} \quad (5.9)$$

де $Н_{пв}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = 120444 * 0.30 = 36133 \text{ грн}$$

5.2.10 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у попередніх статтях витрат і можуть бути віднесені безпосередньо на собівартість розробки за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати робітників за формулою:

$$І_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100} \quad (5.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = 120444 * 0.7 = 84310 \text{ грн}$$

5.2.8 Загальновиробничі витрати

Дана стаття витрат охоплює витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати $В_{нзв}$ можна прийняти як 130% від суми основної заробітної плати розробників та робітників, які виконували МКР, тобто:

$$В_{нзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100}, \quad (5.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Загальновиробничі витрати».

$$В_{нзв} = 120444 * 1.3 = 156577 \text{ грн}$$

5.2.9 Загальні витрати

Сума всіх статей витрат дає в результаті витрати на проведення дослідження та розробку адаптивної системи тестування знань і розраховується за формулою:

$$В = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (5.9)$$

$$В = 120444 + 12044 + 29147 + 699 + 3420 + 12491 + 1137 + 26497 + 36133 + 84310 + 156577 = 482899 \text{ грн}$$

Загальні витрати ЗВ на завершення роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.10)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії дослідного зразка, то коефіцієнт $\beta = 0.5$.

Звідси:

$$ЗВ = \frac{323978}{0.5} = 965798 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

Економічна ефективність дозволяє спрогнозувати чистий прибуток, який може бути отриманий від впровадження розробленої системи.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу.

Для розрахунку збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки використовується формула формулою 5.11:

$$\Delta П_i = \sum_{1}^{n} \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100} \right) \quad (5.11)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження нової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_0$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

ϑ – ставка податку на прибуток. У 2023 році – 18%.

Впровадження результатів розробки дозволяє ефективніше рекомендувати спеціальність. Припустимо, сервіс рекомендацій спеціальностей проводить опитування особистих вподобань учнів та студентів, отримуючи 200 грн за одного користувача. Одне опитування займає приблизно годину, враховуючи що кількість запитань дорівнює 100. Щодня проходять тестування 50 користувачів в середньому, таким чином щорічно виходить 18250 тестів. Припустимо, що покращена система рекомендації спеціальностей скорочує час тестування користувача в середньому на 50%, компанія зможе отримувати

вдвічі більше прибутку. Важливо, щоб компанія просувала сервіс на ринку, тому припустимо що щорічно кількість користувачів буде збільшуватись на 10%. До того ж, вартість послуг може збільшитись на 10% щорічно.

$$\Delta П_1 = (200 \cdot 18250 + (200 + 20) \cdot 1825) \cdot 0.833 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right)$$
$$= (365000 + 73000) \cdot 0.170765 = 4051500 \text{ грн}$$
$$\Delta П_2 = (200 \cdot 18250 + (200 + 40) \cdot 3650) \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right)$$
$$= (365000 + 219000) \cdot 0.170765 = 4526000 \text{ грн}$$
$$\Delta П_3 = (200 \cdot 18250 + (200 + 60) \cdot 5475) \cdot 0.833 \cdot 0.25 \cdot \left(1 - \frac{18}{100}\right)$$
$$= (365000 + 438000) \cdot 0.170765 = 5073500 \text{ грн}$$

Далі за формулою 5.12 розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.12)$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15. В Україні рівень інфляції за підсумком 2023 року склав 10.6%, прогнозований рівень на 2024 рік – 8.5% ;

t – період часу (в роках) від моменту початку впровадження розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{4051500}{1+0.1} + \frac{4526000}{(1+0.085)^2} + \frac{5073500}{(1+0.085)^3}$$
$$= 3683181 + 3844634 + 3972084 = 11499899 \text{ грн}$$

За формулою 5.13 необхідно розрахувати величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації розробки.

$$PV = k_{\text{інв}} \cdot ЗВ \quad (5.13)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$ =2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 4 \cdot 965798 = 3863192 \text{ грн}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації розробки становитиме:

$$E_{\text{абс}} = ПП - PV \quad (5.14)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

$$E_{\text{абс}} = 11499899 - 3863192 = 7636707 \text{ грн}$$

Оскільки величина економічного ефекту має велике додатне значення, це свідчить про потенційну зацікавленість інвесторів у впровадженні та комерціалізацію розробки.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для остаточного прийняття рішення про впровадження розробки та виведення її на ринок необхідно розрахувати внутрішню економічну дохідність $E_в$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою 5.15:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.15}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн; PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[3]{1 + \frac{7636707}{3863192}} - 1 = 1.44 - 1 = 0.44 = 44\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою 5.16:

$$\tau = d + f, \tag{5.16}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0.17 + 0.07 = 0.24$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Далі розраховується період окупності інвестицій, вкладених у реалізацію проекту за формулою 5.17.

$$T_{ок} = \frac{1}{E_в} \tag{5.17}$$

де Ев – внутрішня економічна дохідність вкладених інвестицій.

$$\text{Т}_{\text{ок}} = \frac{1}{0.44} = 2.27 \text{ роки} = 27 \text{ міс} = 2 \text{ роки та 3 місяці}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування наукової розробки є доцільним.

5.5 Висновки

Було проведено оцінку комерційного потенціалу розробки програмного забезпечення для експертної рекомендації спеціальностей. Економічний потенціал склав значення, яке є вище середнього.

Також було спрогнозовано витрати на проектування та реалізацію системи за необхідними статтями витрат, які склали 482899 грн. Загальна величина витрат склала 965798 грн.

Обрахунок економічної ефективності та терміну окупності вкладених інвестицій показали, що розробка є економічно доцільною та привабливою для потенційних інвесторів. Термін окупності інвестицій складає 2 роки та 3 місяці, а прогнозований прибуток за 3 роки 11499899 грн.

ВИСНОВКИ

У магістерській кваліфікаційній роботі була розроблена експертна рекомендаційна система для вибору спеціальностей в закладах вищої освіти України. Процес виконання роботи включав аналіз сучасного стану проблеми, де детально розглядались основні аналоги програмного продукту. У результаті порівняльного аналізу були ідентифіковані переваги та недоліки конкурентів порівняно з розробленим програмним продуктом. На основі цього порівняння були сформульовані основні завдання та цілі магістерської кваліфікаційної роботи, які служили орієнтиром під час впровадження та вдосконалення розробленої системи.Під час аналізу технологій розробки було обґрунтовано вибір мови програмування C# та веб-платформи ASP.NET MVC.

У магістерській кваліфікаційній роботі було зроблено значний обсяг робіт, а саме:

- розроблено метод експертної рекомендації спеціальностей. Цей метод ґрунтується на врахуванні особистих побажань користувача, а також на аналізі балів з документів про освіту та психологічних тестів.
- розроблено метод статистичного аналізу результатів рекомендацій, що спрямований на постійне покращення ефективності системи.
- розроблено програмний додаток з веб-інтерфейсом, який дозволяє взаємодіяти з системою відразу та ефективно.
- проведено тестування системи, щоб підтвердити її функціональність та готовність до використання в реальних умовах.

Було розроблено схему загального алгоритму роботи системи, включаючи модуль експертної рекомендації спеціальностей, модуль статистичного аналізу результатів рекомендацій та інші ключові компоненти. Під час тестування програми підтверджено повну працездатність програмного продукту та його відповідність поставленому технічному завданню. В рамках розробки також створено інструкцію користувача, яка дозволяє зручно та

ефективно використовувати систему. Ці кроки сприяють впровадженню розробленого рішення в практику та забезпечують його готовність до використання в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Полтавський обласний центр зайнятості. Вибір професії - важливий крок в житті. [Електронний ресурс]. URL: https://pol.dcz.gov.ua/publikaciya/vybir-professiyi-vazhlyvyy-krok-v-zhytti
2. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 50 с.
3. Кубай М.О. Розробка експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України //Матеріали III Всеукраїнської науково-технічної конференції молодих вчених, аспірантів і студентів “Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації - 2023”, Одеса, 28-29 жовтня 2023 р. Одеса, Видавництво ОНТУ, 2023 р. С. 62-63.
4. Кубай М.О. Розробка методів і програмних засобів експертної рекомендації спеціальностей в закладах вищої освіти України //Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. С. 137-138.
5. Освіта.ua. Проблема професійного самовизначення молоді. [Електронний ресурс]. URL: https://osvita.ua/vnz/reports/sociology/30006
6. MyNextMove. What do you want to do for a living? [Електронний ресурс]. URL: https://www.mynextmove.org
7. O*Net Online. My Next Move. [Електронний ресурс]. URL: https://www.onetonline.org/help/onet/mynextmove
8. CareerExplorer. Uncover new beginnings in the world of work. [Електронний ресурс]. URL: https://www.careerexplorer.com/about/

9. Minnesota State CareerWise. Assess Yourself. [Електронний ресурс]. URL: https://careerwise.minnstate.edu/careers/assessyourself.html
10. Big Future Collegeboard. Your Future, Your Way. Plan for College and Career. [Електронний ресурс]. URL: https://bigfuture.collegeboard.org
11. The Myers-Briggs Company. Strong Interest Inventory. A career assessment that delivers insight and direction. [Електронний ресурс]. URL: https://www.themyersbriggs.com/en-US/Products-and-Services/Strong
12. The SAPA Project. Take the test. Explore your personality. Advance the study of individual differences. [Електронний ресурс]. URL: https://www.sapa-project.org
13. ПЛАТФОРМА ДЕРЖАВНОЇ СЛУЖБИ ЗАЙНЯТОСТІ З ПРОФОРІЄНТАЦІЇ ТА РОЗВИТКУ КАР'ЄРИ. [Електронний ресурс]. URL: https://myprofession.com.ua
14. ВСЕУКРАЇНСЬКИЙ ПРОЄКТ З ПРОФОРІЄНТАЦІЇ ТА ПОБУДОВИ КАР'ЄРИ. [Електронний ресурс]. URL: https://hryoutest.in.ua
15. Поняття системи освіти, її структура. Studentam.net. [Електронний ресурс]. URL: https://studentam.net.ua/content/view/10868/86/
16. Як визначитися із спеціальністю у виші? Proforientator.com.ua. [Електронний ресурс]. URL: https://proforientator.com.ua/ua/vopros-otvet-o-proforientatcii/kak-opredelitsya-so-speczialnostyu-v-vuze.html
17. Структура та органи управління навчального закладу. Германівський ліцей імені братів Гетьманів. [Електронний ресурс]. URL: http://germanivka-school.edukit.kiev.ua/struktura_ta_organi_upravlinnya_navchaljnogo_zakladu/
18. Галузь знань. Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Галузь_знань

19. Дніпропетровський державний університет внутрішніх справ. Методи статистики та етапи статистичного дослідження. [Електронний ресурс]. URL: https://dduvs.in.ua/files/student/lectures/eib
20. Метод статистики. Buklib. [Електронний ресурс]. URL: https://buklib.net/books/35937/
21. Основні методи прогнозування: класифікація прогнозів. [Електронний ресурс]. URL: https://pidru4niki.com/16400221/menedzhment/osnovni_metodi_prognozuvannya_klasifikatsiya_prognoziv
22. Методи розв'язання багатокритеріальних задач оптимізації. Studfiles. [Електронний ресурс]. URL: https://studfile.net/preview/7185500/page:15/
23. Психодіагностика професійного самовизначення особистості: навч.-метод. посіб. / І. М. Щербакова, Г. А. Стадник – Суми: Вид-во СумДПУ імені А. С. Макаренка, 2012. – 324 с.
24. Огляд і основи мови програмування C++. Портал знань. [Електронний ресурс]. URL: http://www.znannya.org/?view=Cpp_basics
25. C Sharp. Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/C_Sharp
26. Огляд Java. Hillelblog. [Електронний ресурс]. URL: https://blog.ithillel.ua/articles/vybir-pershoi-movy-prohramuvannia-ohliad-java
27. Що таке Python. Plone. [Електронний ресурс]. URL: http://www.plug.org.ua/documentation/about-python
28. Visual Studio 2022. Microsoft. [Електронний ресурс]. URL: https://visualstudio.microsoft.com/
29. Visual Studio Code. Code editing. Redefined. Visual Studio Code. [Електронний ресурс]. URL: https://code.visualstudio.com
30. Rider IDE. JetBrains. [Електронний ресурс]. https://www.jetbrains.com/rider/

31. Cross platform IDE for C#, F# and more. MonoDevelop. [Електронний ресурс]. URL: https://www.monodevelop.com
32. Огляд ASP.NET. Learn Microsoft. [Електронний ресурс]. URL: https://learn.microsoft.com/aspnet/overview
33. Модель-вид-контролер. Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер
34. ОГЛЯД ВИДІВ ТЕСТУВАННЯ. QA Test Lab. [Електронний ресурс]. URL: https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/
35. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

Додаток А

(обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою ПЗ

д.т.н., проф.

_____ О. Н. Романюк

"20" вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доцент Д. І. Катєльніков

«20» вересня 2023 р.

Виконав:

_____ студент гр. 2ПІ-22м М. О. Кубай

«20» вересня 2023 р.

Вінниця – 2023

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України».

Галузь застосування – навчальні заклади.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від «18» вересня 2023 р. ректора ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення рівню задоволеності та успішності студентів завдяки допомозі у виборі спеціальності та навчального закладу для вступу.

Основними задачами дослідження є:

- розробка методу експертної рекомендації спеціальностей на основі власних побажань користувача, балів з документів про освіту та психологічних тестів;
- розробка методу статистичного аналізу результатів рекомендацій для покращення подальшої роботи системи;
- розробка програмного додатку з веб-інтерфейсом для користування системою;
- проведення тестування системи.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Освіта.ua. Проблема професійного самовизначення молоді. [Електронний ресурс]. URL: https://osvita.ua/vnz/reports/sociology/30006
2. Поняття системи освіти, її структура. Studentam.net. [Електронний ресурс]. URL: https://studentam.net.ua/content/view/10868/86/

3. Структура та органи управління навчального закладу. Германівський ліцей імені братів Гетьманів. [Електронний ресурс]. URL: http://germanivka-school.edukit.kiev.ua/struktura_ta_organi_upravlinnya_navchaljnogo_zakladu/
4. Дніпропетровський державний університет внутрішніх справ. Методи статистики та етапи статистичного дослідження. [Електронний ресурс]. URL: https://dduvs.in.ua/files/student/lectures/eib
5. Методи розв’язання багатокритеріальних задач оптимізації. Studfiles. [Електронний ресурс]. URL: https://studfile.net/preview/7185500/page:15/
6. Психодіагностика професійного самовизначення особистості: навч.-метод. посіб. / І. М. Щербакова, Г. А. Стадник – Суми: Вид-во СумДПУ імені А. С. Макаренка, 2012. – 324 с.
7. Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації - 2023 / Матеріали III Всеукраїнської науково-технічної конференції молодих вчених, аспірантів і студентів, Одеса, 28-29 жовтня 2023 р. - Одеса, Видавництво ОНТУ, 2023 р. – 270 с.
8. Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – 336 с.

5. Технічні вимоги

- середовище розробки – Microsoft Visual Studio 2022;
- мова програмування – C#;

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до МКР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів дипломного проєкту	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задач	20.09.23 – 02.10.23
2	Розробка архітектури та алгоритмів програмного продукту	03.10.23 – 23.10.23
3	Аналіз і вибір мови програмування та середовища розробки	16.10.23 – 23.10.23
4	Розробка програмного продукту	24.10.23 – 13.11.23
5	Тестування програми	14.11.23 – 21.11.23
6	Розробка економічної частини	17.11.23 –25.11.23
7	Оформлення матеріалів до захисту МКР	22.11.23 – 01.12.23

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка методів і програмних засобів експертної рекомендаційної системи для вибору спеціальностей в закладах вищої освіти України.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Катєльніков Д. І.

Оригінальність	85.6%
Схожість	14.4%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Кубай М. О.

Керівник роботи _____ Катєльніков Д. І.

Додаток В

(обов'язковий)

Лістинг програмного коду системи експертної рекомендації спеціальностей

Програмний код класу DataContextIO

```
using UniversitiesInfo.Domain;
using System.IO;
using System.Text;
using System.Xml;

namespace UniversitiesInfo.Data
{
   partial class DataContext
   {
      public static void Save()
      {
         XmlTextWriter writer = null;
         writer = new XmlTextWriter(fileName, Encoding.Unicode);
         writer.WriteStartDocument();
         writer.WriteStartElement("UniversitiesInfo");
         WriteUniversities(writer);
         WriteOffers(writer);
         writer.WriteEndElement();
         writer.WriteEndDocument();
         writer.Close();
      }

      private static void WriteUniversities(XmlTextWriter writer)
      {
         writer.WriteStartElement("Universities");
         foreach (var obj in universities)
         {
            writer.WriteStartElement("University");
            writer.WriteAttributeString("Id", obj.Id.ToString());
            writer.WriteAttributeString("UniversityName", obj.UniversityName);
            writer.WriteAttributeString("Code", obj.Code);
            writer.WriteAttributeString("Type", obj.Type);
            writer.WriteAttributeString("Area", obj.Area);
            writer.WriteAttributeString("Settlement", obj.Settlement);
            writer.WriteAttributeString("Address", obj.Address);
            writer.WriteAttributeString("Phone", obj.Phone);
            writer.WriteAttributeString("Email", obj.Email);
            writer.WriteAttributeString("Website", obj.Website);
            writer.WriteEndElement();
         }
         writer.WriteEndElement();
      }

      private static void WriteOffers(XmlTextWriter writer)
      {
         writer.WriteStartElement("Offers");
         foreach (var obj in offers)
         {
            writer.WriteStartElement("Offers");
            writer.WriteAttributeString("Id", obj.Id.ToString());
            writer.WriteAttributeString("UniversityName", obj.UniversityName);
            writer.WriteAttributeString("SpecialityCode", obj.SpecialityCode);
            writer.WriteAttributeString("NumberofPlaces", obj.NumberofPlaces.ToString());
            writer.WriteAttributeString("EducationLevel", obj.EducationLevel);
```

```
            writer.WriteEndElement();
        }
        writer.WriteEndElement();
    }

    public static void Load()
    {
        if (!File.Exists(fileName)) return;
        XmlTextReader reader = null;
        reader = new XmlTextReader(fileName)
        {
            WhitespaceHandling = WhitespaceHandling.None
        };
        while (reader.Read())
        {
            if (reader.NodeType == XmlNodeType.Element)
            {
                if (reader.Name == "University")
                {
                    ReadUniversity(reader);
                }
                else if (reader.Name == "Offer")
                {
                    ReadOffer(reader);
                }
            }
        }
        reader.Close();
    }

    private static void ReadUniversity(XmlTextReader reader)
    {
        University obj = new University();
        string s = reader.GetAttribute("Id");
        if (!string.IsNullOrEmpty(s)) obj.Id = int.Parse(s);
        obj.UniversityName = reader.GetAttribute("UniversityName");
        obj.Code = reader.GetAttribute("Code");
        obj.Type = reader.GetAttribute("Type");
        obj.Area = reader.GetAttribute("Area");
        obj.Settlement = reader.GetAttribute("Settlement");
        obj.Address = reader.GetAttribute("Address");
        obj.Phone = reader.GetAttribute("Phone");
        obj.Email = reader.GetAttribute("Email");
        obj.Website = reader.GetAttribute("Website");
        universities.Add(obj);
    }

    private static void ReadOffer(XmlTextReader reader)
    {
        Offer obj = new Offer();
        string s = reader.GetAttribute("Id");
        if (!string.IsNullOrEmpty(s)) obj.Id = int.Parse(s);
        obj.UniversityName = reader.GetAttribute("UniversityName");
        obj.SpecialityCode = reader.GetAttribute("SpecialityCode");
        s = reader.GetAttribute("NumberofPlaces");
        if (!string.IsNullOrEmpty(s)) obj.NumberofPlaces = int.Parse(s);
        obj.EducationLevel = reader.GetAttribute("EducationLevel");
        offers.Add(obj);
    }
  }
}
```

Програмний код класу RouteConfig

```
namespace UniversitiesInfo.Web
{
   public class RouteConfig
   {
      public static void RegisterRoutes(RouteCollection routes)
      {
         routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

         routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
         );
      }
   }
}
```

Програмний код класу AdminController

```
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
   public class AdminController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }
   }
}
```

Програмний код класу HomeController

```
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
   public class HomeController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }

      public ActionResult EntitiesList()
      {
         return View();
      }
   }
}
```

Програмний код класу NavigationController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
```

```
{
   public class NavigationController : Controller
   {
      public IEnumerable<University> Universities { get; set; }

      public NavigationController()
      {
         Universities = DataContext.universities;
      }

      public const string ALL_CATEGORIES = "...";

      [ChildActionOnly]
      public PartialViewResult UniversitiesByAreasMenu(
      string categoryName = ALL_CATEGORIES)
      {
         ViewBag.SelectedCategoryName = categoryName;
         List<string> categoryNames = new List<string>();
         categoryNames.Add(ALL_CATEGORIES);
         categoryNames.AddRange(Universities.Select(e => e.Area).Distinct().OrderBy(e => e));
         return PartialView(categoryNames);
      }
   }
}
```

Програмний код класу OffersController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
   public class OffersController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }

      private IEnumerable<Offer> objects;
      private IEnumerable<OfferViewModel> viewModelObjects;

      public IEnumerable<Offer> Objects
      {
         get { return objects; }
         set
         {
            objects = value;
            viewModelObjects = objects.Select(e => (OfferViewModel)e)
               .OrderBy(e => e.UniversityName);
         }
      }

      public OffersController()
      {
         Objects = DataContext.offers;
      }

      public ActionResult Selection()
```

```
        {
            return View(viewModelObjects);
        }

        public PartialViewResult _SelectData(string selUniversityName, string selSpecialityCode,
            int? PlacesFrom, int? PlacesTo, string selEducationLevel)
        {
            var model = viewModelObjects;
            if (!string.IsNullOrWhiteSpace(selUniversityName))
                model = model.Where(e => e.UniversityName.ToLower()
                    .StartsWith(selUniversityName.ToLower()));
            if (!string.IsNullOrWhiteSpace(selSpecialityCode))
                model = model.Where(e => e.SpecialityCode.ToLower()
                    .StartsWith(selSpecialityCode.ToLower()));
            if (PlacesFrom.HasValue)
                if (PlacesFrom.HasValue)
                    model = model.Where(e => e.NumberofPlaces >= PlacesFrom.Value);
            if (PlacesTo.HasValue)
                model = model.Where(e => e.NumberofPlaces <= PlacesTo.Value);
            if (!string.IsNullOrWhiteSpace(selEducationLevel))
                model = model.Where(e => e.EducationLevel.ToLower()
                    .StartsWith(selEducationLevel.ToLower()));
            System.Threading.Thread.Sleep(1000);
            return PartialView("_TableBody", model);
        }

        public ViewResult List()
        {
            return View(viewModelObjects);
        }
    }
}
```

Програмний код класу OffersCrudController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class OffersCrudController : Controller
    {
        private IList<Offer> objects;
        private IEnumerable<OfferViewModel> viewModelObjects { get; set; }
        private IEnumerable<OfferEditingModel> editingModelObjects { get; set; }

        public IList<Offer> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects
                    .Select(e => (OfferViewModel)e).OrderBy(e => e.UniversityName);
                editingModelObjects = objects
                    .Select(e => (OfferEditingModel)e).OrderBy(e => e.UniversityName);
            }
        }

        public OffersCrudController()
```

```
{
   Objects = DataContext.offers;
}

public ActionResult Index()
{
   return View(viewModelObjects);
}

public ViewResult Create()
{
   return View(new OfferEditingModel());
}

[HttpPost]
public ActionResult Create(OfferEditingModel model)
{
   if (!ModelState.IsValid)
   {
      return View(model);
   }
   Objects.Add((Offer)model);
   DataContext.Save();
   TempData["message"] = string.Format(
      "Дані \"{0}\" збережено", model.UniversityName);
   return RedirectToAction("Index");
}

[HttpGet]
public ActionResult Edit(int id)
{
   OfferEditingModel model = editingModelObjects.First(e => e.Id == id);
   return View(model);
}

[HttpPost]
public ActionResult Edit(OfferEditingModel model)
{
   if (!ModelState.IsValid)
   {
      return View(model);
   }
   var entityObject = Objects.First(e => e.Id == model.Id);
   UpdateEntityObject(entityObject, model);
   DataContext.Save();
   TempData["message"] = string.Format(
      "Зміни даних \"{0}\" збережено", model.UniversityName);
   return RedirectToAction("Index");
}

private void UpdateEntityObject(Offer entityObject,
      OfferEditingModel model)
{
   entityObject.UniversityName = model.UniversityName;
   entityObject.SpecialityCode = model.SpecialityCode;
   entityObject.NumberofPlaces = model.NumberofPlaces;
   entityObject.EducationLevel = model.EducationLevel;
}

public ActionResult Delete(int id)
{
   var model = editingModelObjects.First(e => e.Id == id);
   return View(model);
```

```
        }

        [HttpPost]
        public ActionResult Delete(Offer model)
        {
            Offer entityObject = Objects.First(e => e.Id == model.Id);
            Objects.Remove(entityObject);
            DataContext.Save();
            return RedirectToAction("Index");
        }

        public ActionResult Details(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }
    }
}
```

Програмний код класу UniversitiesController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class UniversitiesController : Controller
    {
        private IEnumerable<University> objects;
        private IEnumerable<UniversityViewModel> viewModelObjects;

        public IEnumerable<University> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects.Select(e => (UniversityViewModel)e)
                    .OrderBy(e => e.UniversityName);
            }
        }

        public UniversitiesController()
        {
            Objects = DataContext.universities;
        }

        public ActionResult Index()
        {
            return View();
        }

        public ViewResult UniversitiesByAreasInfo(
        string categoryName = NavigationController.ALL_CATEGORIES)
        {
            IEnumerable<University> models = Objects.OrderBy(e => e.UniversityName);
            if (!string.IsNullOrEmpty(categoryName) &&
                categoryName != NavigationController.ALL_CATEGORIES)
            {
```

```
        models = models
            .Where(e => e.Area == categoryName);
    }
    ViewBag.SelectedCategoryName = categoryName;
    return View(models);
}

public ActionResult Selection()
{
    ViewBag.selArea = CreateAreasSelectList();
    return View(viewModelObjects);
}

const string ALL_VALUES = "...";

List<SelectListItem> CreateAreasSelectList()
{
    List<string> values = new List<string>();
    values.Add(ALL_VALUES);
    values.AddRange(Objects.Select(e => e.Area).Distinct());
    List<SelectListItem> list = new List<SelectListItem>();
    foreach (string e in values)
    {
        list.Add(new SelectListItem
        {
            Text = e,
            Value = e
        });
    }
    return list;
}

public PartialViewResult _SelectData(string selUniversityName, string selCode,
    string selType, string selArea, string selSettlement)
{
    var model = viewModelObjects;
    if (!string.IsNullOrWhiteSpace(selUniversityName))
        model = model.Where(e => e.UniversityName.ToLower()
            .StartsWith(selUniversityName.ToLower()));

    if (!string.IsNullOrWhiteSpace(selCode))
        model = model.Where(e => e.Code.ToLower()
            .StartsWith(selCode.ToLower()));

    if (!string.IsNullOrWhiteSpace(selType))
        model = model.Where(e => e.Type.ToLower()
            .StartsWith(selType.ToLower()));

    if (selArea != null && selArea != ALL_VALUES)
        model = model.Where(e => e.Area == selArea);

    if (!string.IsNullOrWhiteSpace(selSettlement))
        model = model.Where(e => e.Settlement.ToLower()
            .StartsWith(selSettlement.ToLower()));

    System.Threading.Thread.Sleep(1000);
    return PartialView("_TableBody", model);
}

public const string ALL_PAGE_LINK_NAME = "...";

public ActionResult BrowseByAreas()
{
```

```
            ViewBag.Areas = new[] { ALL_PAGE_LINK_NAME }
                .Concat(Objects.Select(e => e.Area.ToString()).Distinct().OrderBy(e => e));
            return View(viewModelObjects);
        }

        public PartialViewResult _GetDataByArea(string selArea)
        {
            var model = viewModelObjects;
            if (selArea != null && selArea != ALL_PAGE_LINK_NAME)
                model = model.Where(e => e.Area == selArea);
            System.Threading.Thread.Sleep(2000);
            return PartialView("_BrowseData", model);
        }

        public ViewResult List()
        {
            return View(viewModelObjects);
        }

        public ViewResult _Details(int id)
        {
            var obj = viewModelObjects.First(e => e.Id == id);
            return View(obj);
        }

        public ActionResult Recommendation()
        {
            return View();
        }

    }
}
```

Програмний код класу UniversitiesCrudController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class UniversitiesCrudController : Controller
    {
        private IList<University> objects;
        private IEnumerable<UniversityViewModel> viewModelObjects { get; set; }
        private IEnumerable<UniversityEditingModel> editingModelObjects { get; set; }

        public IList<University> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects
                    .Select(e => (UniversityViewModel)e).OrderBy(e => e.UniversityName);
                editingModelObjects = objects
                    .Select(e => (UniversityEditingModel)e).OrderBy(e => e.UniversityName);
            }
        }
```

```
public IEnumerable<University> Areas { get; set; }

public UniversitiesCrudController()
{
   Objects = DataContext.universities;
   Areas = DataContext.universities;
}

public ActionResult Index()
{
   return View(viewModelObjects);
}

public ViewResult Create()
{
   ViewBag.Area = CreateAreasSelectList();
   return View(new UniversityEditingModel());
}

[HttpPost]
public ActionResult Create(UniversityEditingModel model)
{
   if (!ModelState.IsValid)
   {
      ViewBag.Area = CreateAreasSelectList();
      return View(model);
   }
   Objects.Add((University)model);
   DataContext.Save();
   TempData["message"] = string.Format(
      "Дані \"{0}\" збережено", model.UniversityName);
   return RedirectToAction("Index");
}

[HttpGet]
public ActionResult Edit(int id)
{
   UniversityEditingModel model = editingModelObjects.First(e => e.Id == id);
   ViewBag.Area = CreateAreasSelectList(model.Area);
   return View(model);
}

[HttpPost]
public ActionResult Edit(UniversityEditingModel model)
{
   if (!ModelState.IsValid)
   {
      ViewBag.Area = CreateAreasSelectList(model.Area);
      return View(model);
   }
   var entityObject = Objects.First(e => e.Id == model.Id);
   UpdateEntityObject(entityObject, model);
   DataContext.Save();
   TempData["message"] = string.Format(
      "Зміни даних \"{0}\" збережено", model.UniversityName);
   return RedirectToAction("Index");
}

private void UpdateEntityObject(University entityObject,
      UniversityEditingModel model)
{
   entityObject.UniversityName = model.UniversityName;
   entityObject.Code = model.Code;
```

```
            entityObject.Type = model.Type;
            entityObject.Area = model.Area;
            entityObject.Settlement = model.Settlement;
            entityObject.Address = model.Address;
            entityObject.Phone = model.Phone;
            entityObject.Email = model.Email;
            entityObject.Website = model.Website;
        }

        public ActionResult Delete(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }

        [HttpPost]
        public ActionResult Delete(University model)
        {
            University entityObject = Objects.First(e => e.Id == model.Id);
            Objects.Remove(entityObject);
            DataContext.Save();
            return RedirectToAction("Index");
        }

        public ActionResult Details(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }

        List<SelectListItem> CreateAreasSelectList(string selectedValue = "")
        {
            List<string> values = new List<string>();
            values.AddRange(Objects.Select(e => e.Area).Distinct());
            List<SelectListItem> list = new List<SelectListItem>();
            foreach (string e in values)
            {
                list.Add(new SelectListItem
                {
                    Text = e,
                    Value = e
                });
            }
            return list;
        }
    }
}
```

Програмний код класу OfferEditingModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
    public class OfferEditingModel
    {
        public int Id { get; set; }

        [Display(Name = "Назва закладу")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Назва закладу\'")]
        public string UniversityName { get; set; }
```

```
        [Display(Name = "Код спеціальності")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Код спеціальності\'")]
        [RegularExpression(@"[0-9]{3}", ErrorMessage =
           "Код спеціальності має складатись з трьох цифр")]
        public string SpecialityCode { get; set; }

        [Display(Name = "Кількість місць")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Кількість місць\'")]
        [Range(1, 50000, ErrorMessage = "Кількість місць"
           + "має бути в межах від 1 до 50000")]
        public int NumberofPlaces { get; set; }

        [Display(Name = "Освітньо-кваліфікаційний рівень")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Освітньо-кваліфікаційний рівень\'")]
        public string EducationLevel { get; set; }

        public static explicit operator OfferEditingModel(Offer obj)
        {
           return new OfferEditingModel()
           {
              Id = obj.Id,
              UniversityName = obj.UniversityName,
              SpecialityCode = obj.SpecialityCode,
              NumberofPlaces = obj.NumberofPlaces,
              EducationLevel = obj.EducationLevel
           };
        }

        public static explicit operator Offer(OfferEditingModel obj)
        {
           return new Offer()
           {
              Id = obj.Id,
              UniversityName = obj.UniversityName,
              SpecialityCode = obj.SpecialityCode,
              NumberofPlaces = obj.NumberofPlaces,
              EducationLevel = obj.EducationLevel
           };
        }
     }
}
```

Програмний код класу OfferViewModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
   public class OfferViewModel
   {
      public int Id { get; set; }

      [Display(Name = "Назва закладу")]
      public string UniversityName { get; set; }

      [Display(Name = "Код спеціальності")]
      public string SpecialityCode { get; set; }

      [Display(Name = "Кількість місць")]
```

```
        public int NumberofPlaces { get; set; }

        [Display(Name = "Освітньо-кваліфікаційний рівень")]
        public string EducationLevel { get; set; }

        public static explicit operator OfferViewModel(Offer obj)
        {
            return new OfferViewModel()
            {
                Id = obj.Id,
                UniversityName = obj.UniversityName,
                SpecialityCode = obj.SpecialityCode,
                NumberofPlaces = obj.NumberofPlaces,
                EducationLevel = obj.EducationLevel,
            };
        }
    }
}
```

Програмний код класу UniversityEditingModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
    public class UniversityEditingModel
    {
        public int Id { get; set; }

        [Display(Name = "Назва закладу")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Назва закладу\'")]
        [StringLength(512, MinimumLength = 2,
            ErrorMessage = "Назва закладу має містити"
            + "від 2 до 512 символів")]
        public string UniversityName { get; set; }

        [Display(Name = "Ідентифікаційний код")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Ідентифікаційний код\'")]
        [RegularExpression(@"[1-99999999]{1}", ErrorMessage =
            "Потрібно ввести число з 8-ми цифр")]
        public string Code { get; set; }

        [Display(Name = "Тип закладу")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Тип закладу\'")]
        [StringLength(128, MinimumLength = 6,
            ErrorMessage = "Тип закладу має містити"
            + "від 6 до 128 символів")]
        public string Type { get; set; }

        [Display(Name = "Область")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Область\'")]
        [StringLength(17, MinimumLength = 4,
            ErrorMessage = "Область має містити"
            + "від 4 до 17 символів")]
        public string Area { get; set; }

        [Display(Name = "Населений пункт")]
        [Required(ErrorMessage =
            "Потрібно заповнити поле \'Населений пункт\'")]
```

```
[StringLength(128, MinimumLength = 2,
    ErrorMessage = "Населений пункт має містити"
    + "від 2 до 128 символів")]
public string Settlement { get; set; }

[Display(Name = "Адреса")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Адреса\'")]
[StringLength(512, MinimumLength = 2,
    ErrorMessage = "Населений пункт має містити"
    + "від 2 до 512 символів")]
public string Address { get; set; }

[Display(Name = "Телефон")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Телефон\'")]
[StringLength(64, MinimumLength = 2,
    ErrorMessage = "Телефон має містити"
    + "від 2 до 64 символів")]
public string Phone { get; set; }

[Display(Name = "Електронна адреса")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Електронна адреса\'")]
[StringLength(64, MinimumLength = 2,
    ErrorMessage = "Електронна адреса має містити"
    + "від 2 до 64 символів")]
public string Email { get; set; }

[Display(Name = "Веб-сайт")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Веб-сайт\'")]
[StringLength(64, MinimumLength = 2,
    ErrorMessage = "Веб-сайт має містити"
    + "від 2 до 64 символів")]
public string Website { get; set; }


public static explicit operator UniversityEditingModel(University obj)
{
   return new UniversityEditingModel()
   {
      Id = obj.Id,
      UniversityName = obj.UniversityName,
      Code = obj.Code,
      Type = obj.Type,
      Area = obj.Area,
      Settlement = obj.Settlement,
      Address = obj.Address,
      Phone = obj.Phone,
      Email = obj.Email,
      Website = obj.Website
   };
}

public static explicit operator University(UniversityEditingModel obj)
{
   return new University()
   {
      Id = obj.Id,
      UniversityName = obj.UniversityName,
      Code = obj.Code,
      Type = obj.Type,
```

```
            Area = obj.Area,
            Settlement = obj.Settlement,
            Address = obj.Address,
            Phone = obj.Phone,
            Email = obj.Email,
            Website = obj.Website
        };
    }
  }
}
```

Програмний код класу UniversityViewModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
  public class UniversityViewModel
  {
    public int Id { get; set; }

    [Display(Name = "Назва закладу")]
    public string UniversityName { get; set; }

    [Display(Name = "Ідентифікаційний код")]
    public string Code { get; set; }

    [Display(Name = "Тип закладу")]
    public string Type { get; set; }

    [Display(Name = "Область")]
    public string Area { get; set; }

    [Display(Name = "Населений пункт")]
    public string Settlement { get; set; }

    [Display(Name = "Адреса")]
    public string Address { get; set; }

    [Display(Name = "Телефон")]
    public string Phone { get; set; }

    [Display(Name = "Електронна адреса")]
    public string Email { get; set; }

    [Display(Name = "Веб-сайт")]
    public string Website { get; set; }

    public static explicit operator UniversityViewModel(University obj)
    {
      return new UniversityViewModel()
      {
        Id = obj.Id,
        UniversityName = obj.UniversityName,
        Code = obj.Code,
        Type = obj.Type,
        Area = obj.Area,
        Settlement = obj.Settlement,
        Address = obj.Address,
        Phone = obj.Phone,
        Email = obj.Email,
        Website = obj.Website
      };
```

```
        }
    }
}
```

Програмний код скрипта Admin.Index

```
@{
    ViewBag.Title = "Index";
}

<h2>Редагування даних</h2>

<div class="row panel">
    @Html.ActionLink("Редагувати дані про заклади освіти",
            "Index", "UniversitiesCrud", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Редагувати дані про конкурсні пропозиції",
            "Index", "OffersCrud", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("На головну",
              "Index", "Home", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
```

Програмний код скрипта HomeEntitiesList

```
<h2></h2>
<div class="row panel">
    @Html.ActionLink("Заклади вищої освіти",
              "Index", "Universities", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Конкурсні пропозиції",
              "Index", "Offers", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Рекомендації стосовно вибору спеціальності за результатами ЗНО",
              "Recommendation", "Universities", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    <p>
      @Html.ActionLink("Повернутись на головну сторінку", "Index", null, new { @class = "btn btn-block
btn-default btn-lg" })
    </p>
</div>
```

Програмний код скрипта HomeIndex

```
@{
    ViewBag.Title = "Index";
}

<h2>Інформаційна система закладів вищої освіти України</h2>

<div>
    Система призначена для абітурієнтів, які хочуть отримати вищу освіту.
    Також буде корисною для працівників технікумів та коледжів: директорів, методистів, викладачів.
```

Використання бази даних дасть можливість в зручній та доступній формі обрати спеціальність, освітньо-кваліфікаційний рівень та навчальний заклад.

```
</div>
```

Програмний код скрипта Navigation

```
@model IEnumerable<string>

<h3>Виберіть область</h3>

@foreach (var link in Model)
{
   @Html.RouteLink(
         link.Length < 20 ? link : link.Substring(0, 17) + "...",
         new
         {
            controller = "Universities",
            action = "UniversitiesByAreasInfo",
            categoryName = link
         },
         new
         {
            @class = "btn btn-block btn-default btn-lg btn-nav_menu"
               + (link == ViewBag.SelectedCategoryName ? " btn-primary" : "")
         })
}
```

Програмний код скрипта Offers _Details

```
@model UniversitiesInfo.Web.Models.OfferViewModel

@{
   ViewBag.Title = "Details";
}

<h2>Детальна інформація</h2>

<div>
   <h4>Конкурсна пропозиція</h4>
   <hr />
   <dl class="dl-horizontal">

      <dt>
         @Html.DisplayNameFor(model => model.UniversityName)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.UniversityName)
      </dd>

      <dt>
         @Html.DisplayNameFor(model => model.SpecialityCode)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.SpecialityCode)
      </dd>

      <dt>
         @Html.DisplayNameFor(model => model.NumberofPlaces)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.NumberofPlaces)
```

```
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EducationLevel)
        </dd>
    </dl>
</div>

<p>
    @Html.ActionLink("Назад", "List")
</p>
```

Програмний код скрипта Offers _SelectionForm

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))
{
    <p>
        Початок назви закладу<br />
        <input type="text" name="selUniversityName" />
    </p>
    <p>
        Код спеціальності<br />
        <input type="text" name="selSpecialityCode" />
    </p>
    <p>
        Кількість місць<br />
        від <input type="text" name="PlacesFrom" />
        до <input type="text" name="PlacesTo" />
    </p>
    <p>
        Освітньо-кваліфікаційний рівень<br />
        <input type="text" name="selEducationLevel" />
    </p>
    <p>
        <input type="submit" value="Відібрати" />
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Програмний код скрипта Offers _TableBody

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>


@foreach (var item in Model)
{
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.UniversityName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.SpecialityCode)
```

```
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.NumberofPlaces)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.EducationLevel)
    </td>
</tr>
}
```

Програмний код скрипта Offers _TableHead

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

<tr>
    <th>
        @Html.DisplayNameFor(model => model.UniversityName)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.SpecialityCode)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.NumberofPlaces)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.EducationLevel)
    </th>
</tr>
```

Програмний код скрипта Offers Index

```
@{
    ViewBag.Title = "Index";
}

<h2>Конкурсні пропозиції</h2>

<div class="row panel">
    @Html.ActionLink("Список конкурсних пропозицій",
                "List", "Offers", null,
                new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Відбір даних про конкурсні пропозиції",
                "Selection", "Offers", null,
                new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Назад",
                "EntitiesList", "Home", null,
                new { @class = "btn btn-block btn-default btn-lg" })
</div>
```

Програмний код скрипта Offers List

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

@{
    ViewBag.Title = "List";
}

<h2>Конкурсні пропозиції</h2>
<h3>
    @Html.ActionLink("Назад", "Index")
</h3>
```

```
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UniversityName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.UniversityName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.SpecialityCode)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.NumberofPlaces)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.EducationLevel)
    </td>
</tr>
    }
</table>
```

Програмний код скрипта Offers Selection

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

@{
    ViewBag.Title = "Selection";
}

<div class="row panel">
    <div id="parameters" class="col-sm-2">
        <h3>Параметри</h3>
        @Html.Partial("_SelectionForm")
        <h3>
            @Html.ActionLink("Назад", "Index")
        </h3>
    </div>
    <div class="col-sm-10">
        <h2>Конкурсні пропозиції</h2>
        <table class="table">
            <thead>
                @Html.Partial("_TableHead", @Model)
            </thead>
            <tbody id="data">
                @Html.Partial("_TableBody", @Model)
            </tbody>
        </table>
    </div>
</div>
```

Програмний код скрипта OffersCrudCreate

```
@model UniversitiesInfo.Web.Models.OfferEditingModel

@{
    ViewBag.Title = "Create";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <h4>Додати конкурсну пропозицію</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-
md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
            @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.SpecialityCode, htmlAttributes: new { @class = "control-label col-md-
2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.SpecialityCode, new { htmlAttributes = new { @class = "form-
control" } })
            @Html.ValidationMessageFor(model => model.SpecialityCode, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.NumberofPlaces, htmlAttributes: new { @class = "control-label col-
md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.NumberofPlaces, new { htmlAttributes = new { @class = "form-
control" } })
            @Html.ValidationMessageFor(model => model.NumberofPlaces, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.EducationLevel, htmlAttributes: new { @class = "control-label col-
md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.EducationLevel, new { htmlAttributes = new { @class = "form-
control" } })
            @Html.ValidationMessageFor(model => model.EducationLevel, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Додати" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
```

```
    @Html.ActionLink("Відмінити введення і повернутись до списку", "Index")
</div>
```

Програмний код скрипта OffersCrudDelete

```
@model UniversitiesInfo.Web.Models.OfferEditingModel

@{
    ViewBag.Title = "Delete";
}

<h3>Ви дійсно хочете видалити цю пропозицію?</h3>
<div>
    <h4>Видалення даних</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.SpecialityCode)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumberofPlaces)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EducationLevel)
        </dd>

    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Видалити" class="btn btn-default" /> |
            @Html.ActionLink("Відмінити зміни і повернутись до списку", "Index")
        </div>
    }
</div>
```

Програмний код скрипта OffersCrudDetails

```
@model UniversitiesInfo.Web.Models.OfferEditingModel

@{
```

```
    ViewBag.Title = "Details";
}

<div>
    <h4>Конкурсна пропозиція</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.SpecialityCode)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumberofPlaces)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EducationLevel)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Редагувати", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Повернутись до списку", "Index")
</p>
```

Програмний код скрипта OffersCrudEdit

```
@model UniversitiesInfo.Web.Models.OfferEditingModel

@{
    ViewBag.Title = "Edit";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <h4>Редагування даних</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    @Html.HiddenFor(model => model.Id)
    <p>
```

```
            @Html.ActionLink("Повернутись до списку", "Index")
        </p>
        <div class="form-group">
            @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SpecialityCode, htmlAttributes: new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.SpecialityCode, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.SpecialityCode, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.NumberofPlaces, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NumberofPlaces, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.NumberofPlaces, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.EducationLevel, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EducationLevel, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.EducationLevel, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Зберегти" class="btn btn-default" />
            </div>
        </div>
    </div>
    }
```

Програмний код скрипта OffersCrudIndex

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

@{
    ViewBag.Title = "Index";
}

<h3>Редагування даних</h3>
<h3>
    @Html.ActionLink("Назад", "Index", "Admin")
</h3>

<p>
```

```
    @Html.ActionLink("Додати конкурсну пропозицію", "Create",
            null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UniversityName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.ActionLink(item.UniversityName, "Details", new { id = item.Id })
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.SpecialityCode)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.NumberofPlaces)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.EducationLevel)
    </td>
    <td>
        @Html.ActionLink("Редагувати", "Edit", new { id = item.Id }) |
        @Html.ActionLink("Видалити", "Delete", new { id = item.Id })
    </td>
</tr>
    }
</table>
```

Програмний код скрипта Shared _CategoryLayout

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
    <link href="~/Content/Styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div class="navbar navbar-inverse" role="navigation">
        <a class="navbar-brand hidden-xs" href="/">
            Інформаційна система ЗВО України
        </a>
        <a class="navbar-brand visible-xs" href="#">Інформаційна система ЗВО України</a>
    </div>
    <div class="row panel">
```

```
        <div id="categories" class="col-sm-5 col-md-4 col-lg-3">
            @RenderSection("NavMenu", false)
        </div>
        <div class="col-sm-7 col-md-8 col-lg-9">
            @RenderBody()
        </div>
    </div>

    <footer>@RenderSection("Footer", false)</footer>
</body>
</html>
```

Програмний код скрипта Shared _Layout

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
    <link href="~/Content/Styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div class="navbar navbar-inverse" role="navigation">
        <a class="navbar-brand hidden-xs" href="/">
            Інформаційна система ЗВО України
        </a>
        <a class="navbar-brand visible-xs" href="#">Інформаційна система ЗВО України</a>
    </div>
    <div class="row panel">
        <div id="categories" class="col-sm-5 col-md-4 col-lg-3">
            @RenderSection("NavMenu", false)
        </div>
        <div class="col-sm-7 col-md-8 col-lg-9">
            @RenderBody()
        </div>
    </div>

    <footer>@RenderSection("Footer", false)</footer>
</body>
</html>
```

Програмний код скрипта Shared _NoteBlock

```
@model string

@if (string.IsNullOrWhiteSpace(@Model))
{
    @:(дані відсутні)
}
else
{
    <blockquote>
        <p class="lead">
            @Model
        </p>
    </blockquote>
}
```

Програмний код скрипта Shared _NoteInfo

```
@model IEnumerable<string>

@if (@Model != null)
```

```
{
   <hr />
   foreach (var m in @Model)
   {
      <p class="lead">@m</p>
   }
   <hr />
}
```

Програмний код скрипта Universities _BrowseData

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@foreach (var m in @Model)
{
   <div class="row panel">
      <div class="col-xs-12">
         <h3>
            <strong>@m.UniversityName</strong>
         </h3>
         <strong class="lead">Ідентифікаційний код: @m.Code</strong><br />
         <strong class="lead">Тип закладу: @m.Type</strong><br />
         <strong class="lead">Область: @m.Area</strong><br />
         <strong class="lead">Населений пункт: @m.Settlement</strong><br />
         <strong class="lead">Адреса: @m.Address</strong><br />
         <strong class="lead">Телефон: @m.Phone</strong><br />
         <strong class="lead">Електронна адреса: @m.Email</strong><br />
         <strong class="lead">Веб-сайт: @m.Website</strong><br />
      </div>
   </div>
}
```

Програмний код скрипта Universities _Details

```
@model UniversitiesInfo.Web.Models.UniversityViewModel

@{
   ViewBag.Title = "Details";
}

<h2>Детальна інформація</h2>

<div>
   <hr />
   <dl class="dl-horizontal">
      <dt>
         @Html.DisplayNameFor(model => model.UniversityName)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.UniversityName)
      </dd>

      <dt>
         @Html.DisplayNameFor(model => model.Code)
      </dt>

      <dd>
         @Html.DisplayFor(model => model.Code)
      </dd>

      <dt>
         @Html.DisplayNameFor(model => model.Type)
      </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.Type)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Area)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Area)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Settlement)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Settlement)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Address)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Address)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Phone)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Phone)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Website)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Website)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Назад", "List")
</p>
```

Програмний код скрипта Universities _SelectionForm

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
```

```
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))
{
    <p>
        Назва закладу<br />
        <input type="text" name="selUniversityName" />
    </p>
    <p>
        Ідентифікаційний код<br />
        <input type="text" name="selCode" />
    </p>
    <p>
        Тип закладу<br />
        <input type="text" name="selType" />
    </p>
    <p>
        Область<br />
        @Html.DropDownList("selArea", ViewBag.selArea as SelectList)
    </p>
    <p>
        Населений пункт<br />
        <input type="text" name="selSettlement" />
    </p>
    <p>
        <input type="submit" value="Відібрати" />
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Програмний код скрипта Universities _TableBody

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>


@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.UniversityName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Code)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Type)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Area)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Settlement)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Address)
        </td>
    </tr>
}
```

Програмний код скрипта Universities _TableHead

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

<tr>
   <th>
      @Html.DisplayNameFor(model => model.UniversityName)
   </th>
   <th>
      @Html.DisplayNameFor(model => model.Code)
   </th>
   <th>
      @Html.DisplayNameFor(model => model.Type)
   </th>
   <th>
      @Html.DisplayNameFor(model => model.Area)
   </th>
   <th>
      @Html.DisplayNameFor(model => model.Settlement)
   </th>
   <th>
      @Html.DisplayNameFor(model => model.Address)
   </th>
   <th></th>
</tr>
```

Програмний код скрипта UniversitiesBrowseByAreas

```
@using UniversitiesInfo.Web.Controllers
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
   ViewBag.Title = "BrowseByAreas";
}

<h3>
   @Html.ActionLink("Назад", "Index")
</h3>

<div class="ajaxLink">
   @foreach (string area in @ViewBag.Areas as IEnumerable<string>)
   {
      @Ajax.ActionLink(
            area,
            "_GetDataByArea",
            new { selArea = area },
            new AjaxOptions
            {
               UpdateTargetId = "data",
               LoadingElementId = "loading",
               LoadingElementDuration = 1000
            },
            new { @class = "btn btn-default" }
         )
   }
</div>
<div id="loading" class="load" style="display:none">
   <p>Завантаження даних...</p>
</div>

<div id="data">
   @Html.Action("_GetDataByArea",
         new { selNumber = @UniversitiesController.ALL_PAGE_LINK_NAME })
</div>
```

Програмний код скрипта UniversitiesIndex

```
@{
   ViewBag.Title = "Index";
}

<h2>Інформація про заклади освіти</h2>


<div class="row panel">
   @Html.ActionLink("Список всіх закладів",
            "List", "Universities", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
   @Html.ActionLink("Заклади освіти за областями",
            "UniversitiesByAreasInfo", "Universities", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
   @Html.ActionLink("Відбір даних про заклади освіти",
            "Selection", "Universities", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
   @Html.ActionLink("Назад",
            "EntitiesList", "Home", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
```

Програмний код скрипта UniversitiesList

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
   ViewBag.Title = "List";
}

<h2>Заклади освіти</h2>

<h3>
   @Html.ActionLink("Назад", "Index")
</h3>

<table class="table">
   <tr>
      <th>
         @Html.DisplayNameFor(model => model.UniversityName)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Code)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Type)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Area)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Settlement)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Address)
```

```
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.UniversityName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Code)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Type)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Area)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Settlement)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Address)
            </td>
            <td>
                @Html.ActionLink("Детально", "_Details", new { id = item.Id })
            </td>
        </tr>
    }
</table>
```

Програмний код скрипта UniversitiesRecommendation

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))

{
    <h2>Введіть оцінки з ЗНО для рекомендації спеціальностей</h2>
    <p>
        Математика<br />
        <input type="text" name="math" />
    </p>
    <p>
        Фізика<br />
        <input type="text" name="physics" />
    </p>
    <p>
        Українська мова<br />
        <input type="text" name="ukrainian" />
    </p>
    <p>
        Українська література<br />
        <input type="text" name="ukrliterature" />
    </p>
    <p>
```

```
        Історія України<br />
        <input type="text" name="historyofUkraine" />
    </p>
    <p>
        Біологія<br />
        <input type="text" name="biology" />
    </p>
    <p>
        Географія<br />
        <input type="text" name="geography" />
    </p>
    <p>
        Хімія<br />
        <input type="text" name="chemistry" />
    </p>
    <p>
        Англійська мова<br />
        <input type="text" name="english" />
    </p>
    <p>
        Німецька мова<br />
        <input type="text" name="deutsch" />
    </p>
    <p>
        Іспанська мова<br />
        <input type="text" name="spanish" />
    </p>
    <p>
        Французька мова<br />
        <input type="text" name="french" />
    </p>
    <p>
        <input type="submit" value="Аналізувати" />
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Програмний код скрипта UniversitiesSelection

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
    ViewBag.Title = "Selection";
}

<div class="row panel">
    <div id="parameters" class="col-sm-2">
        <h3>Параметри</h3>
        @Html.Partial("_SelectionForm")
        <h3>
            @Html.ActionLink("Назад", "Index")
        </h3>
    </div>
    <div class="col-sm-10">
        <table class="table">
            <thead>
                @Html.Partial("_TableHead", @Model)
            </thead>
            <tbody id="data">
                @Html.Partial("_TableBody", @Model)
            </tbody>
```

```
      </table>
   </div>
</div>
```

Програмний код скрипта UniversitiesUniversitiesByAreasInfo

```
@model IEnumerable<UniversitiesInfo.Domain.University>

@{
   ViewBag.Title = "UniversitiesByAreasInfo";
   Layout = "~/Views/Shared/_CategoryLayout.cshtml";
}

<h2>Заклади освіти по областях</h2>
<h3>
   @Html.ActionLink("Назад", "Index")
</h3>


<table class="table">
   <tr>
      <th>
         @Html.DisplayNameFor(model => model.UniversityName)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Code)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Type)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Area)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Settlement)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Address)
      </th>
      <th></th>
   </tr>

   @foreach (var item in Model)
   {
      <tr>
         <td>
            @Html.DisplayFor(modelItem => item.UniversityName)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Code)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Type)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Area)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Settlement)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Address)
         </td>
```

```
            <td>
                @Html.ActionLink("Детально", "_Details", new { id = item.Id })
            </td>
        </tr>
    }
</table>

@section NavMenu {
    @Html.Action("UniversitiesByAreasMenu", "Navigation",
          new { categoryName = ViewBag.SelectedCategoryName })
}
```

Програмний код скрипта UniversitiesCrud Create

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
    ViewBag.Title = "Create";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <h4>Додати заклад освіти</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-
md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
            @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Code, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Code, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Code, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Type, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Type, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Type, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Area, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.DropDownList("Area", ViewBag.Area as SelectList)
            @Html.ValidationMessageFor(model => model.Area, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
```

```
            @Html.LabelFor(model => model.Settlement, htmlAttributes: new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
              @Html.EditorFor(model => model.Settlement, new { htmlAttributes = new { @class = "form-control"
} })
              @Html.ValidationMessageFor(model => model.Settlement, "", new { @class = "text-danger" })
            </div>
          </div>

          <div class="form-group">
            @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
              @Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class = "form-control" }
})
              @Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-danger" })
            </div>
          </div>

          <div class="form-group">
            @Html.LabelFor(model => model.Phone, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
              @Html.EditorFor(model => model.Phone, new { htmlAttributes = new { @class = "form-control" } })
              @Html.ValidationMessageFor(model => model.Phone, "", new { @class = "text-danger" })
            </div>
          </div>

          <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
              @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
              @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
          </div>

          <div class="form-group">
            @Html.LabelFor(model => model.Website, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
              @Html.EditorFor(model => model.Website, new { htmlAttributes = new { @class = "form-control" }
})
              @Html.ValidationMessageFor(model => model.Website, "", new { @class = "text-danger" })
            </div>
          </div>

          <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
              <input type="submit" value="Додати" class="btn btn-default" />
            </div>
          </div>
        </div>
        }

        <div>
          @Html.ActionLink("Відмінити введення і повернутись до списку", "Index")
        </div>
```

Програмний код скрипта UniversitiesCrud Delete

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
  ViewBag.Title = "Delete";
}
```

```
<h3>Ви дійсно бажаєте видалити цей запис?</h3>
<div>
    <h4>@Model.UniversityName: видалення даних</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Code)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Code)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Type)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Type)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Area)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Area)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Settlement)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Settlement)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Address)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Address)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Phone)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Phone)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
```

```
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Website)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Website)
        </dd>
    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Видалити" class="btn btn-default" /> |
            @Html.ActionLink("Відмінити зміни і повернутись до списку", "Index")
        </div>
    }
</div>
```

Програмний код скрипта UniversitiesCrud Details

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
    ViewBag.Title = "Details";
}

<div>
    <h4>@Model.UniversityName: перегляд даних</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Code)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Code)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Type)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Type)
        </dd>

        <dt>
```

```
            @Html.DisplayNameFor(model => model.Area)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Area)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Settlement)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Settlement)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Address)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Address)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Phone)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Phone)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Website)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Website)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Редагувати", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Повернутись до списку", "Index")
</p>
```

Програмний код скрипта UniversitiesCrud Edit

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
    ViewBag.Title = "Edit";
}

@using (Html.BeginForm())
```

```
        {
            @Html.AntiForgeryToken()

        <div class="form-horizontal">
            <h4>@Model.UniversityName: редагування даних</h4>
            <hr />
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            @Html.HiddenFor(model => model.Id)

            <div class="form-group">
                @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-
md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
                    @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Code, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Code, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Code, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Type, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Type, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Type, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Area, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownList("Area", ViewBag.Area as SelectList)
                    @Html.ValidationMessageFor(model => model.Area, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Settlement, htmlAttributes: new { @class = "control-label col-md-2"
})
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Settlement, new { htmlAttributes = new { @class = "form-control"
} })
                    @Html.ValidationMessageFor(model => model.Settlement, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class = "form-control" }
})
                    @Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
```

```
            @Html.LabelFor(model => model.Phone, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Phone, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Phone, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Website, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Website, new { htmlAttributes = new { @class = "form-control" }
})
                @Html.ValidationMessageFor(model => model.Website, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Зберегти" class="btn btn-default" />
            </div>
        </div>
    </div>
    }

    <div>
        @Html.ActionLink("Відмінити зміни і повернутись до списку", "Index")
    </div>
```

Програмний код скрипта UniversitiesCrud Index

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
    ViewBag.Title = "Index";
}

<h3>Редагування даних про заклади освіти</h3>
<h3>
    @Html.ActionLink("Назад", "Index", "Admin")
</h3>

<p>
    @Html.ActionLink("Додати новий заклад", "Create",
        null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UniversityName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Code)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Type)
```

```
            </th>
            <th>
              @Html.DisplayNameFor(model => model.Area)
            </th>
            <th>
              @Html.DisplayNameFor(model => model.Settlement)
            </th>
            <th>
              @Html.DisplayNameFor(model => model.Address)
            </th>
            <th></th>
          </tr>

          @foreach (var item in Model)
          {
        <tr>
          <td>
            @Html.ActionLink(item.UniversityName, "Details", new { id = item.Id })
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.Code)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.Type)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.Area)
          </td>
          <td>
            @Html.DisplayFor(model => item.Settlement)
          </td>
          <td>
            @Html.DisplayFor(model => item.Address)
          </td>
          <td>
            @Html.ActionLink("Редагувати", "Edit", new { id = item.Id }) |
            @Html.ActionLink("Видалити", "Delete", new { id = item.Id })
          </td>
        </tr>
          }
        </table>
```

Програмний код скрипта Web.config

```
<?xml version="1.0"?>

<configuration>
  <configSections>
    <sectionGroup name="system.web.webPages.razor" type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">
      <section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
      <section name="pages" type="System.Web.WebPages.Razor.Configuration.RazorPagesSection, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
    </sectionGroup>
  </configSections>

  <system.web.webPages.razor>
    <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=5.2.4.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
```

```
        <pages pageBaseType="System.Web.Mvc.WebViewPage">
          <namespaces>
            <add namespace="System.Web.Mvc" />
            <add namespace="System.Web.Mvc.Ajax" />
            <add namespace="System.Web.Mvc.Html" />
            <add namespace="System.Web.Routing" />
            <add namespace="UniversitiesInfo.Web" />
          </namespaces>
        </pages>
      </system.web.webPages.razor>

      <appSettings>
        <add key="webpages:Enabled" value="false" />
      </appSettings>

      <system.webServer>
        <handlers>
          <remove name="BlockViewHandler"/>
          <add name="BlockViewHandler" path="*" verb="*" preCondition="integratedMode"
type="System.Web.HttpNotFoundHandler" />
        </handlers>
      </system.webServer>

      <system.web>
        <compilation>
          <assemblies>
            <add assembly="System.Web.Mvc, Version=5.2.4.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" />
          </assemblies>
        </compilation>
      </system.web>
    </configuration>
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ЕКСПЕРТНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИБОРУ СПЕЦІАЛЬНОСТЕЙ В ЗАКЛАДАХ ВИЩОЇ ОСВІТИ УКРАЇНИ

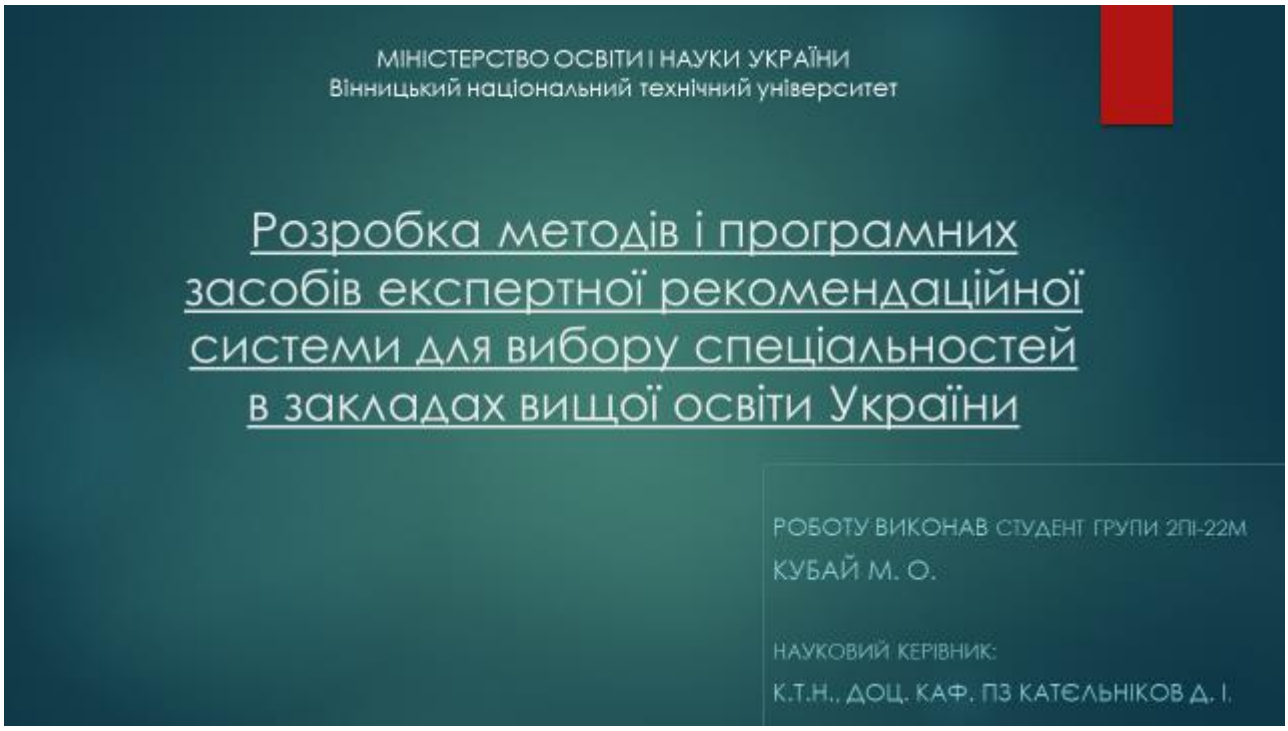

Рисунок Г.1 – Титульний аркуш

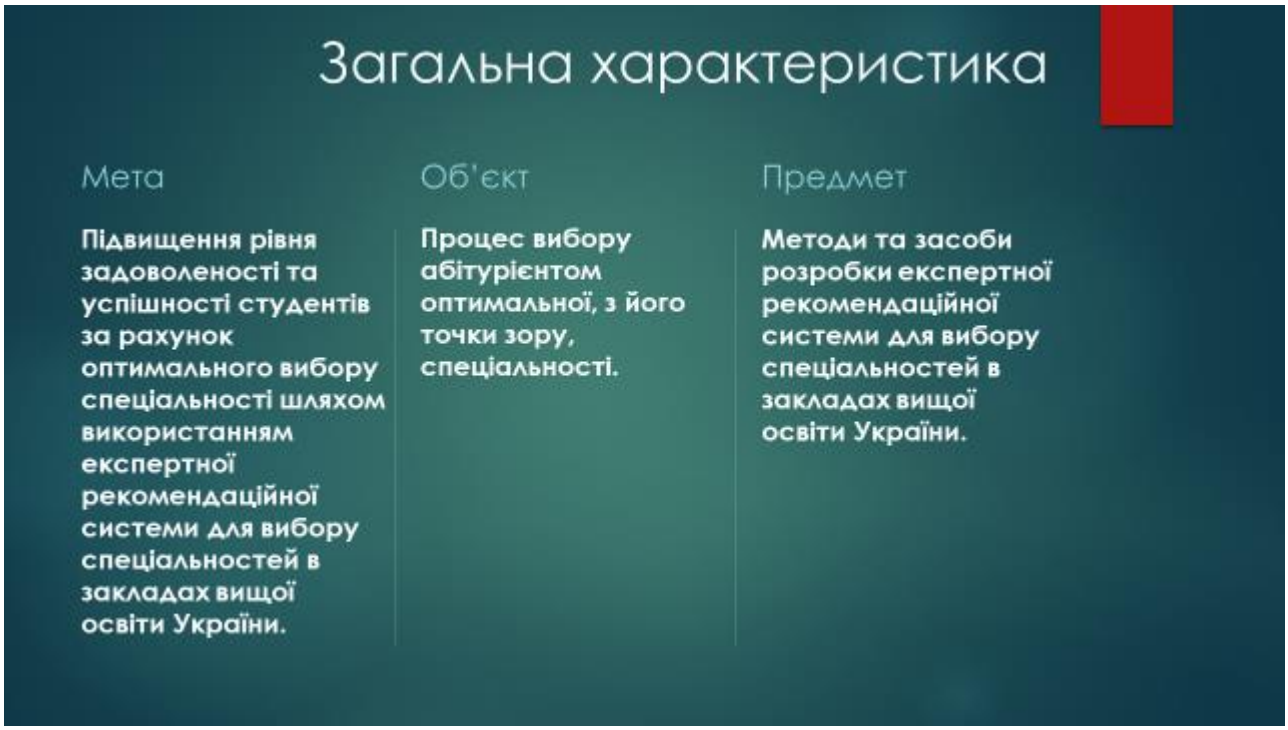

Рисунок Г.2 – Загальна характеристика

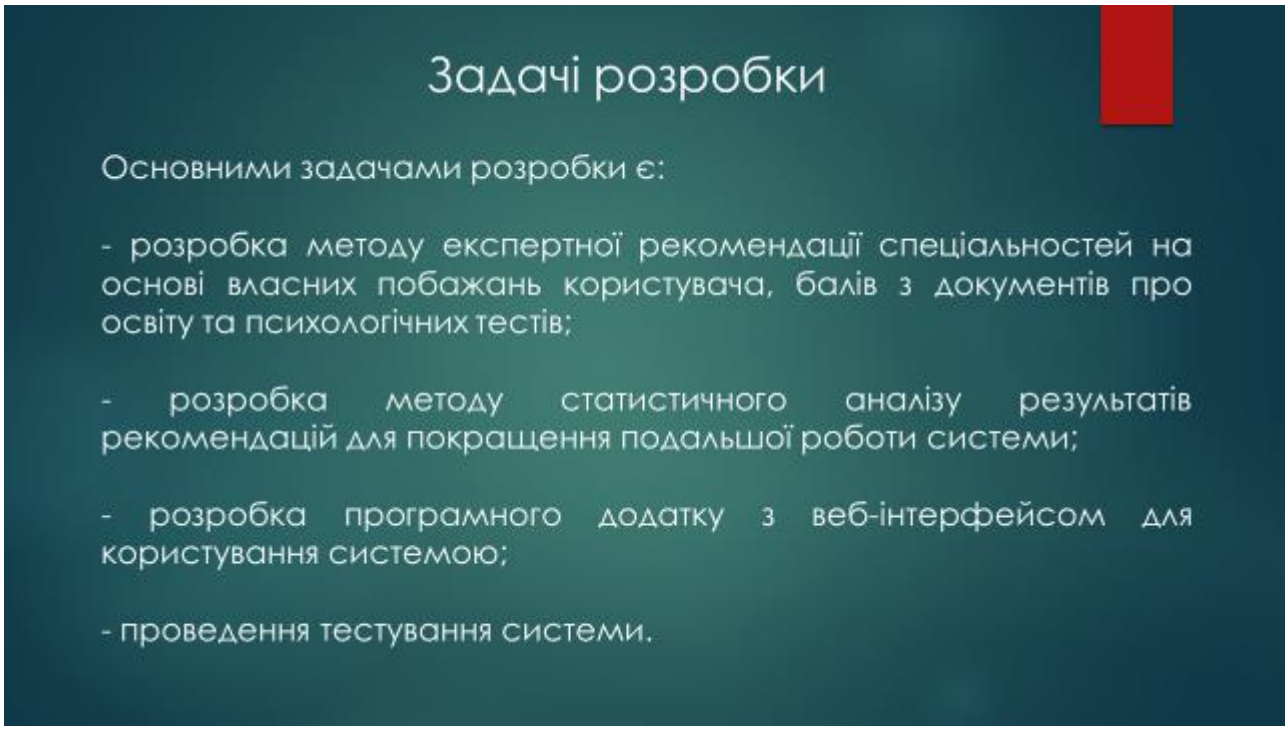

Рисунок Г.3 – Задачі розробки

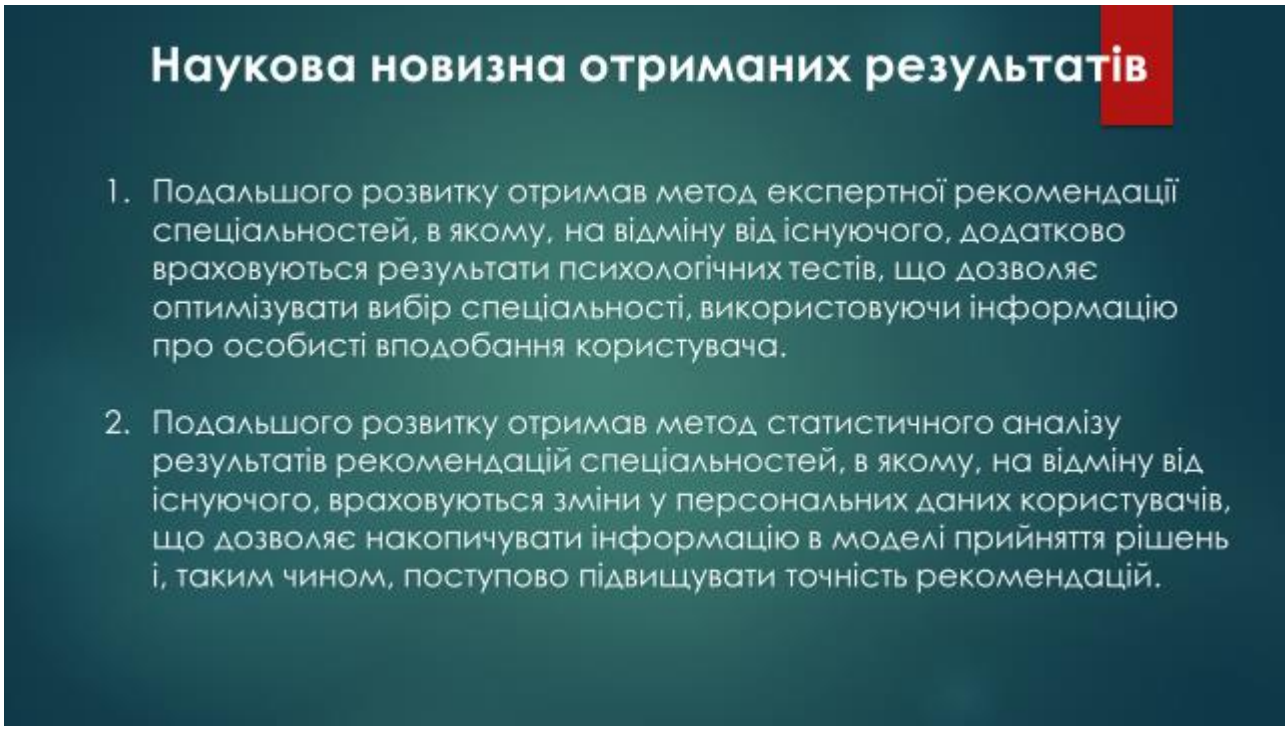

Рисунок Г.4 – Наукова новизна отриманих результатів

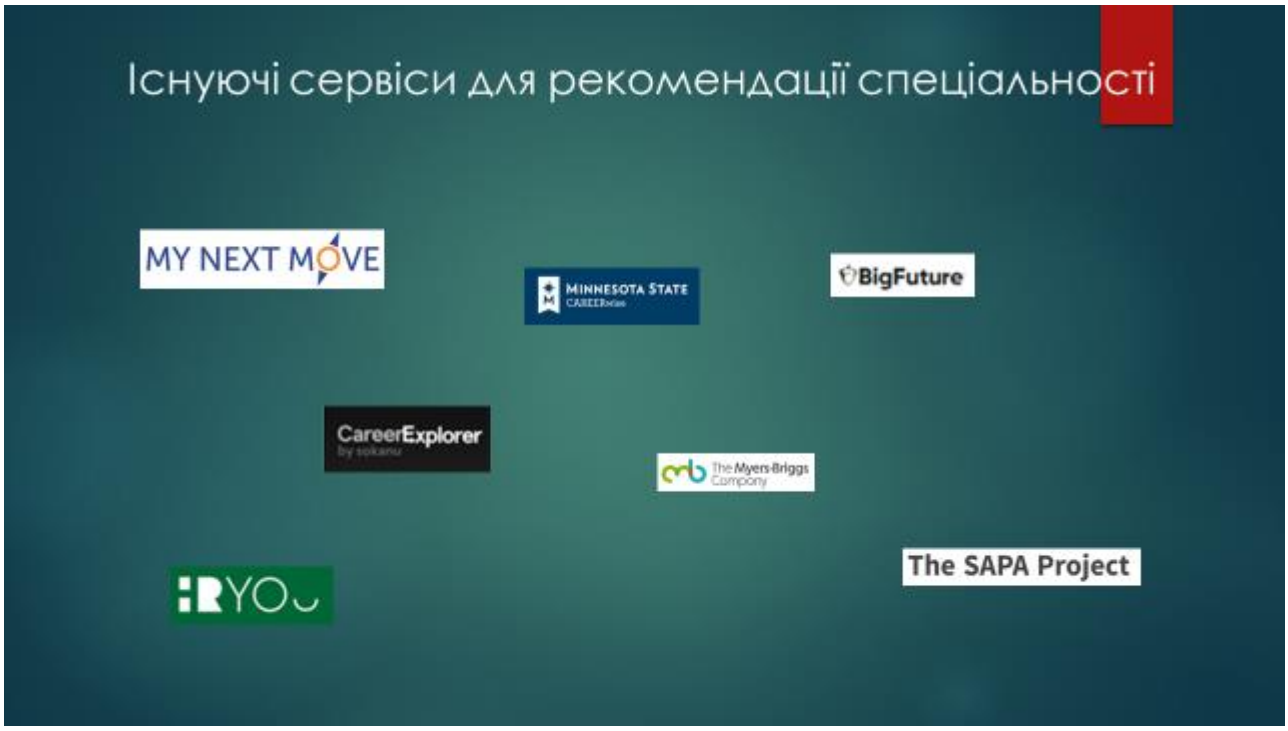

Рисунок Г.5 – Існуючі сервіси для рекомендації спеціальностей

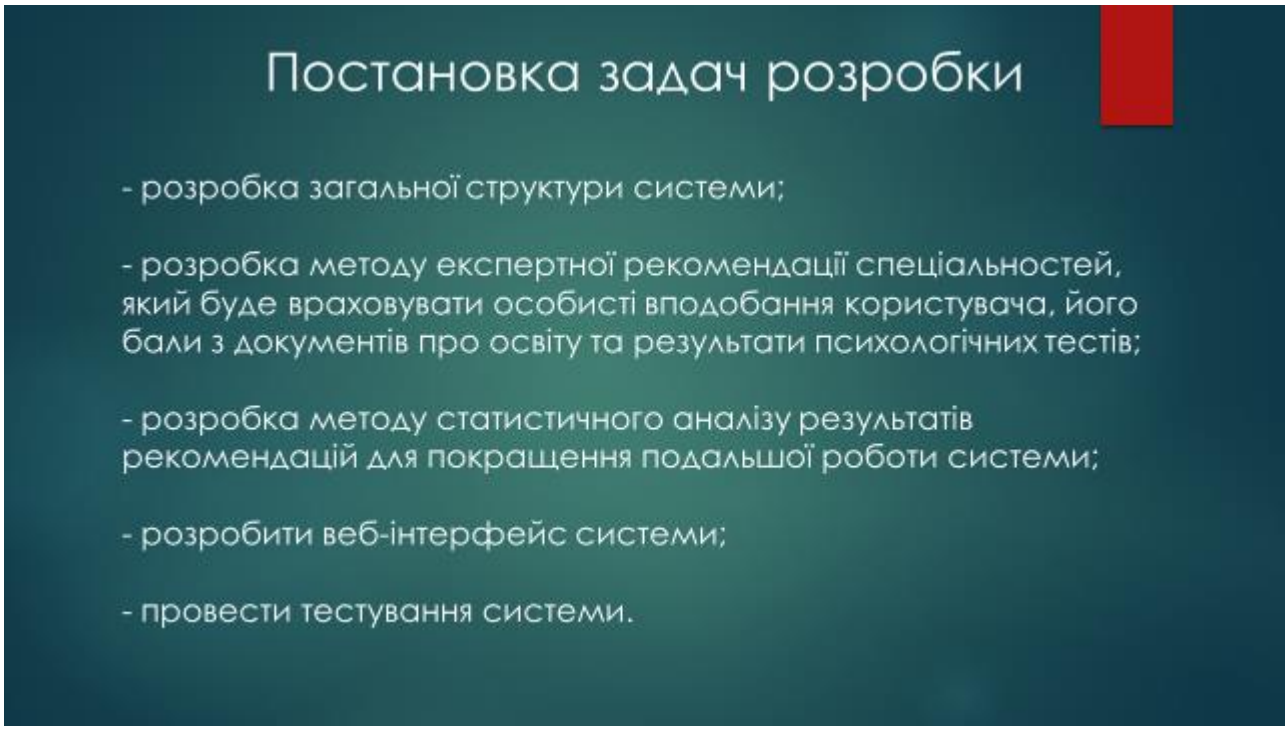

Рисунок Г.6 – Постановка задач розробки

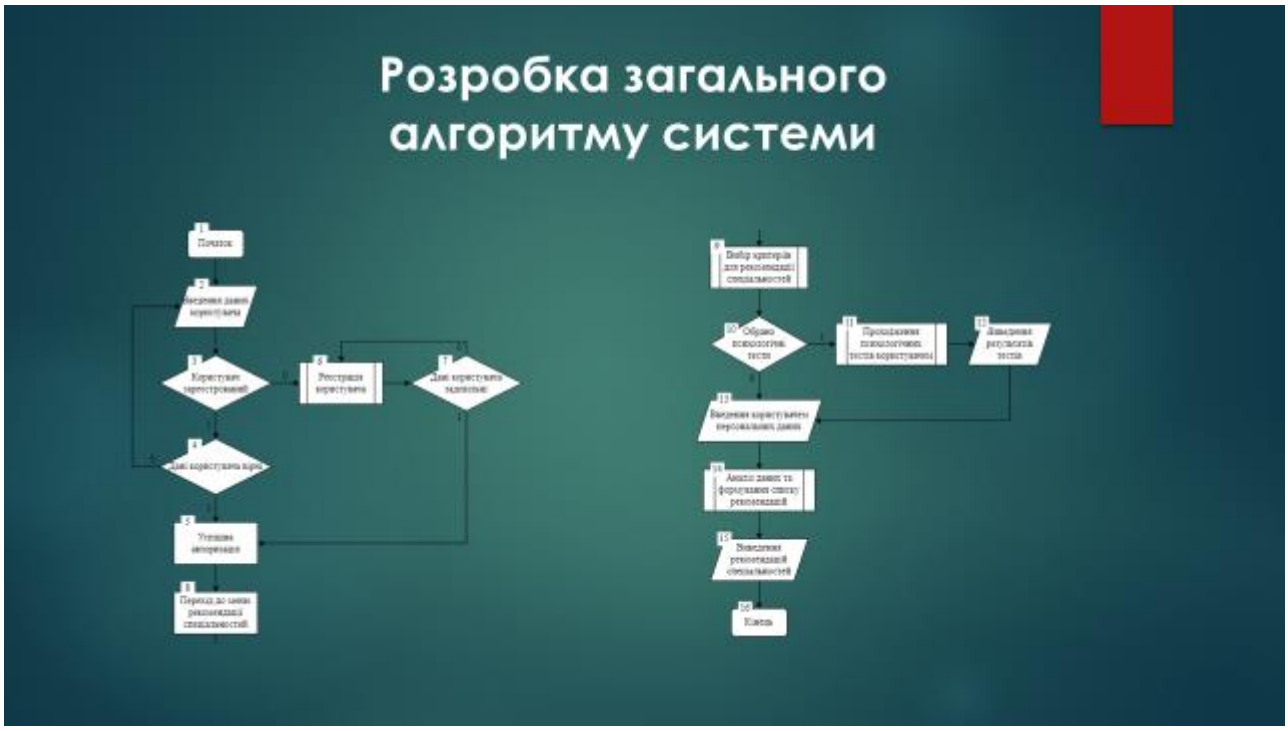

Рисунок Г.7 – Розробка загального алгоритму системи

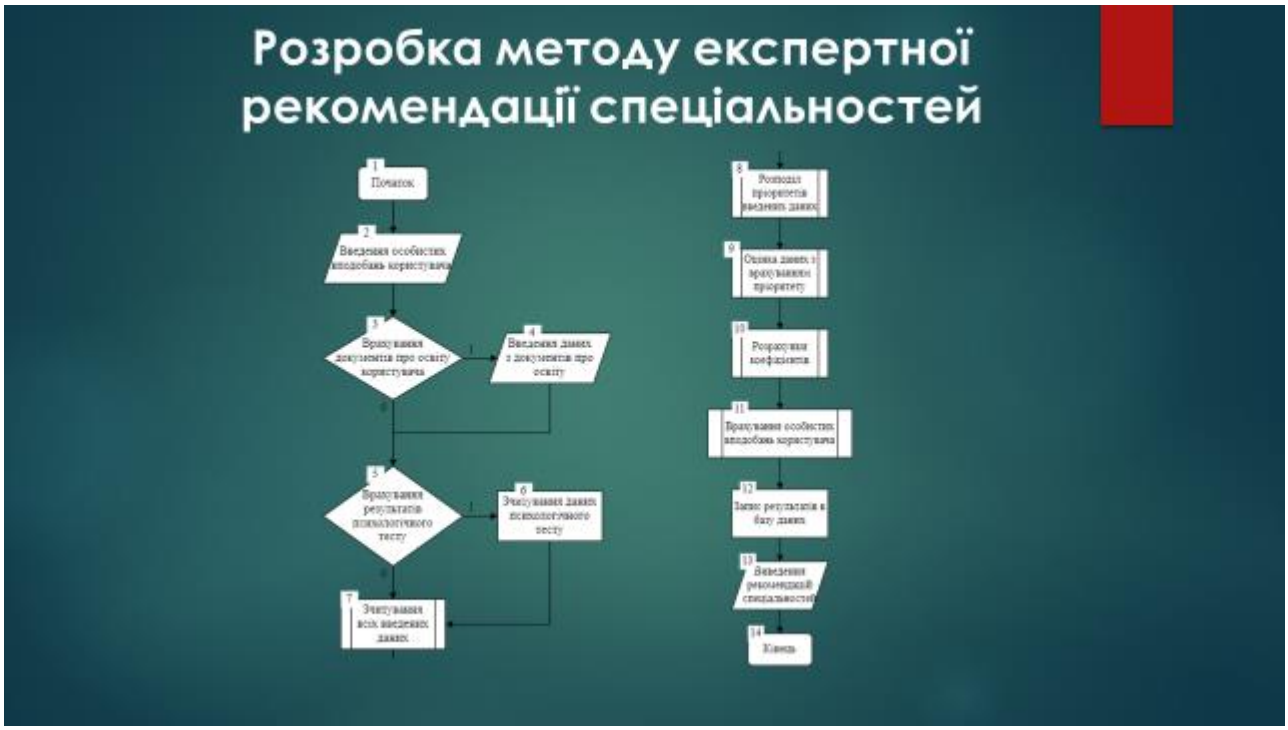

Рисунок Г.8 – Розробка методу експертної рекомендації спеціальностей

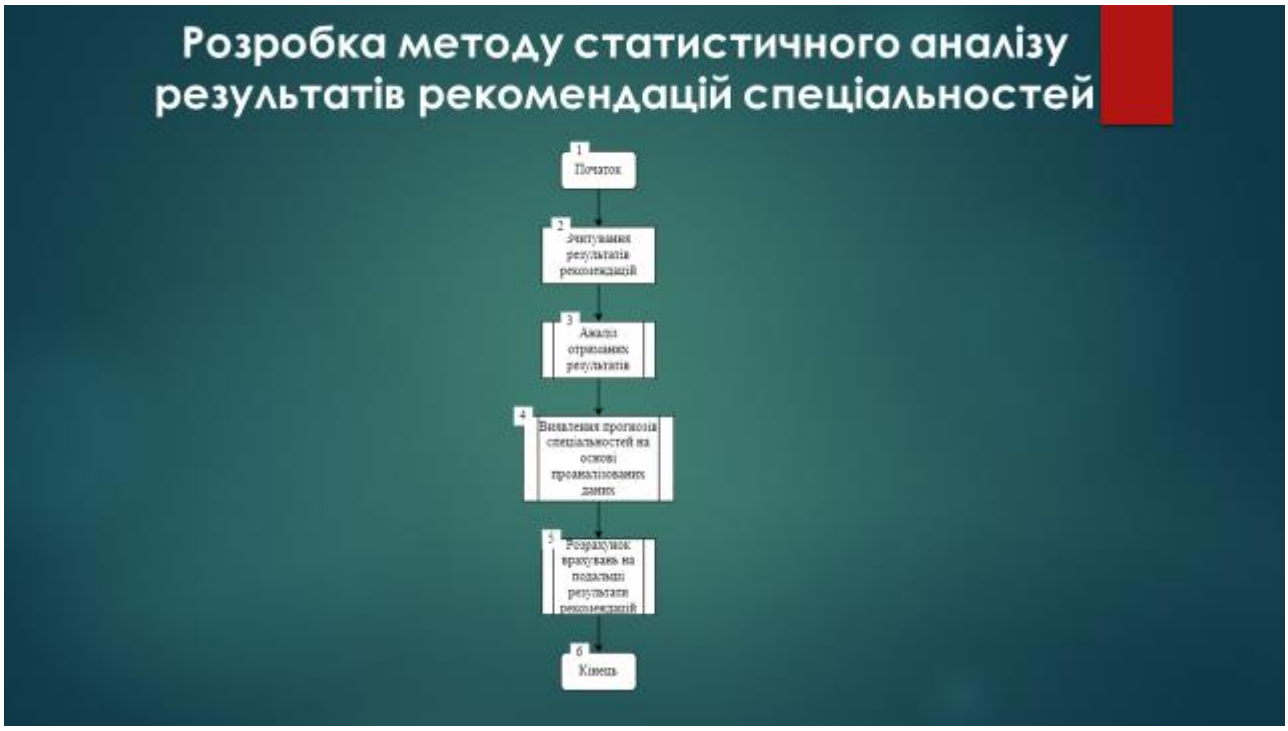

Рисунок Г.9 – Розробка методу статистичного аналізу результатів рекомендацій спеціальностей

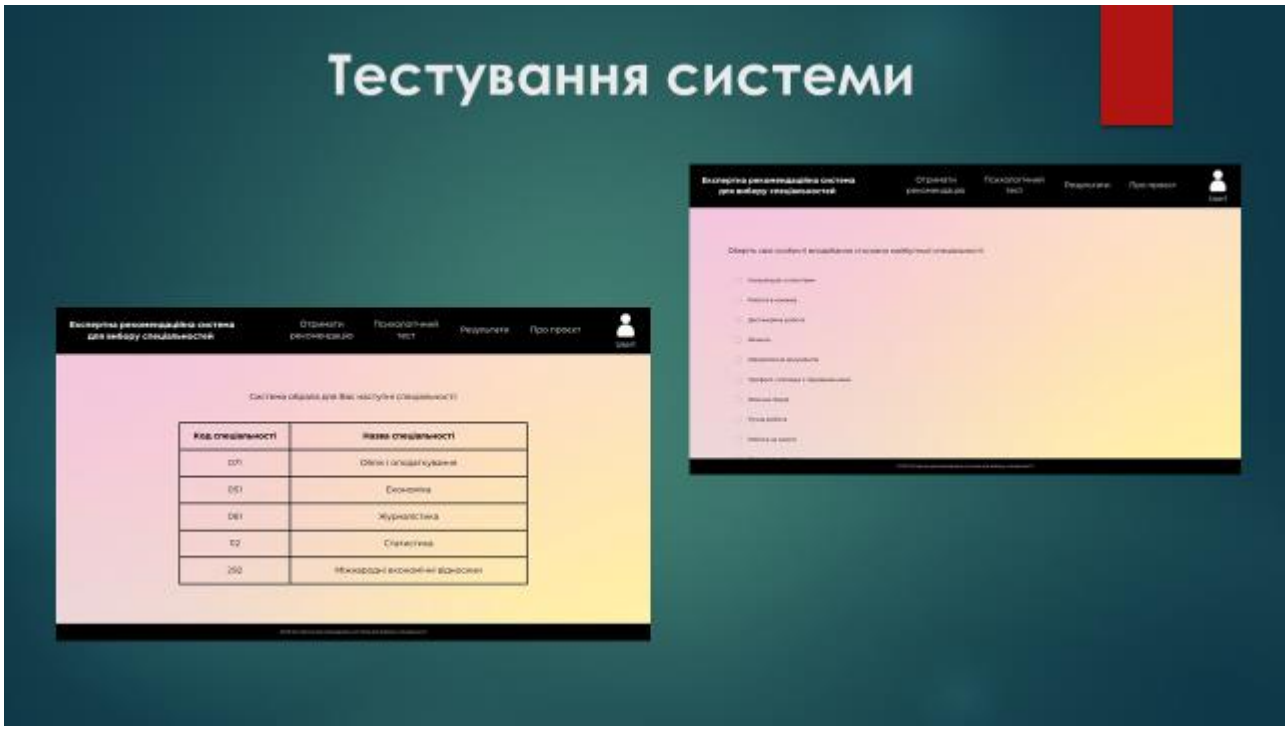

Рисунок Г.10 – Тестування системи (частина 1)

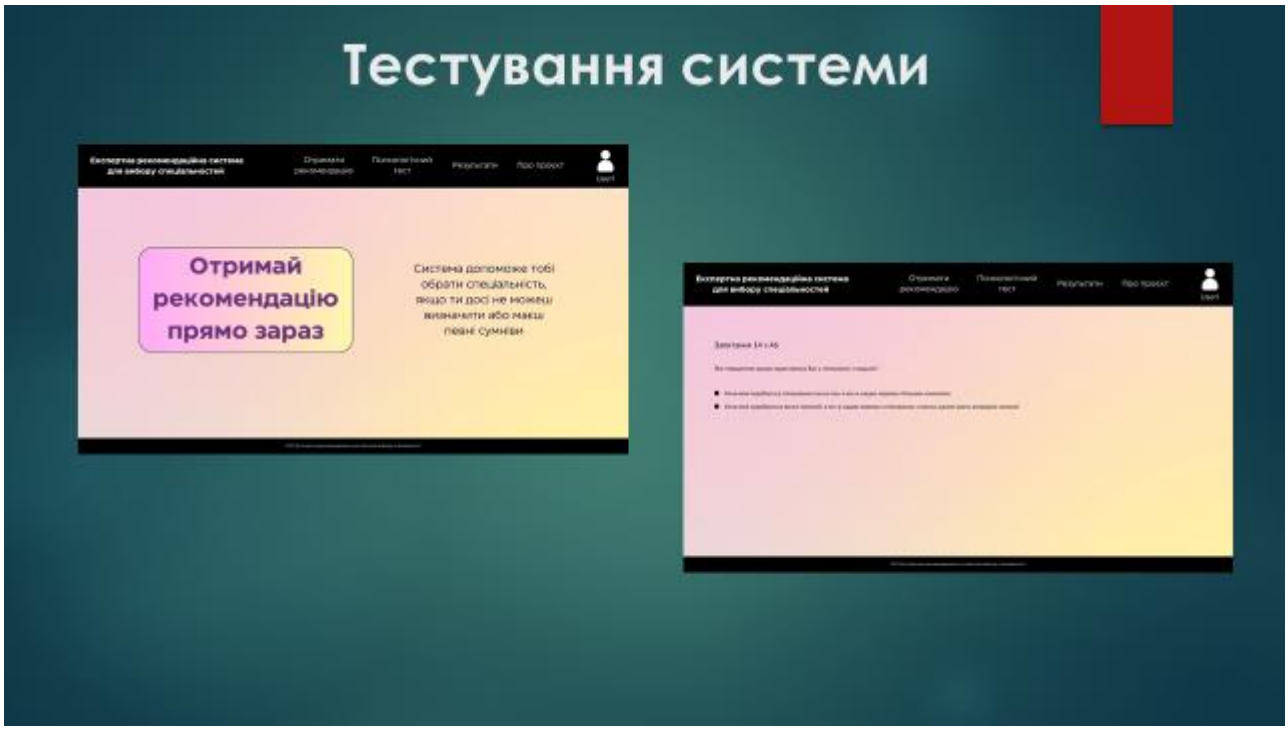

Рисунок Г.11 – Тестування системи (частина 2)

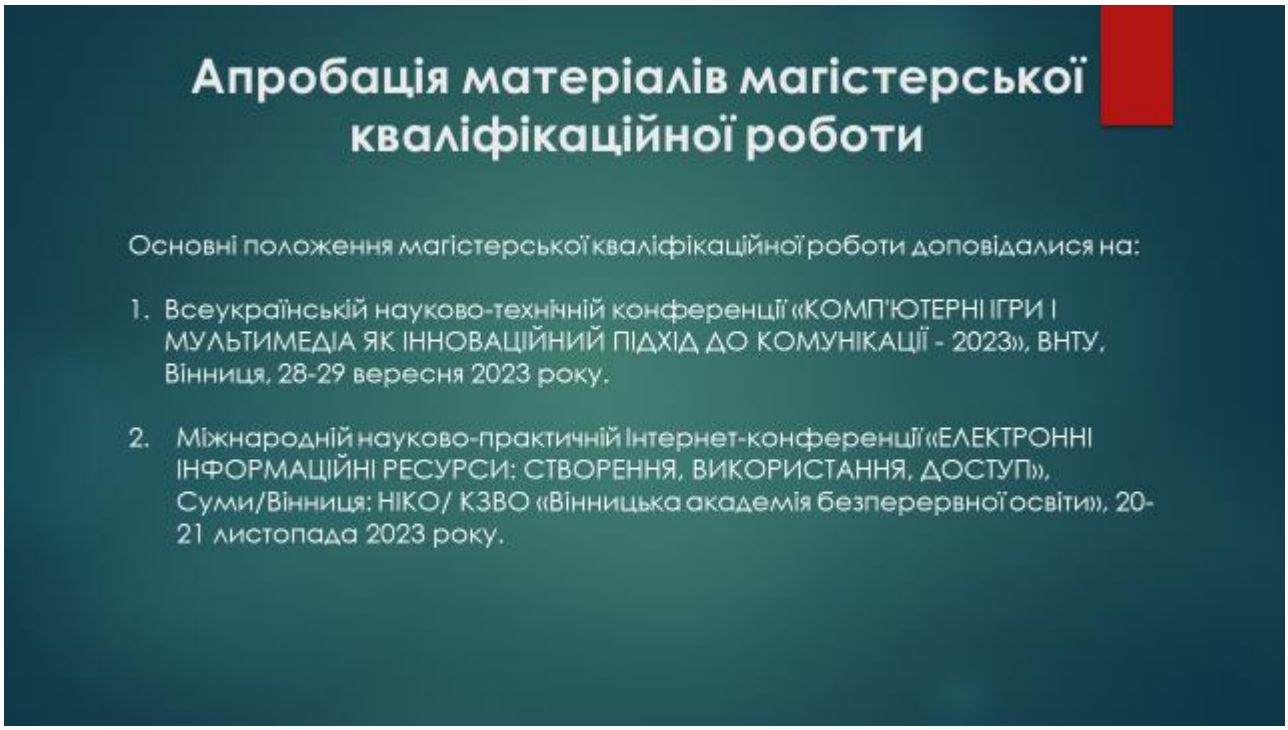

Рисунок Г.12 – Апробація матеріалів магістерської кваліфікаційної роботи

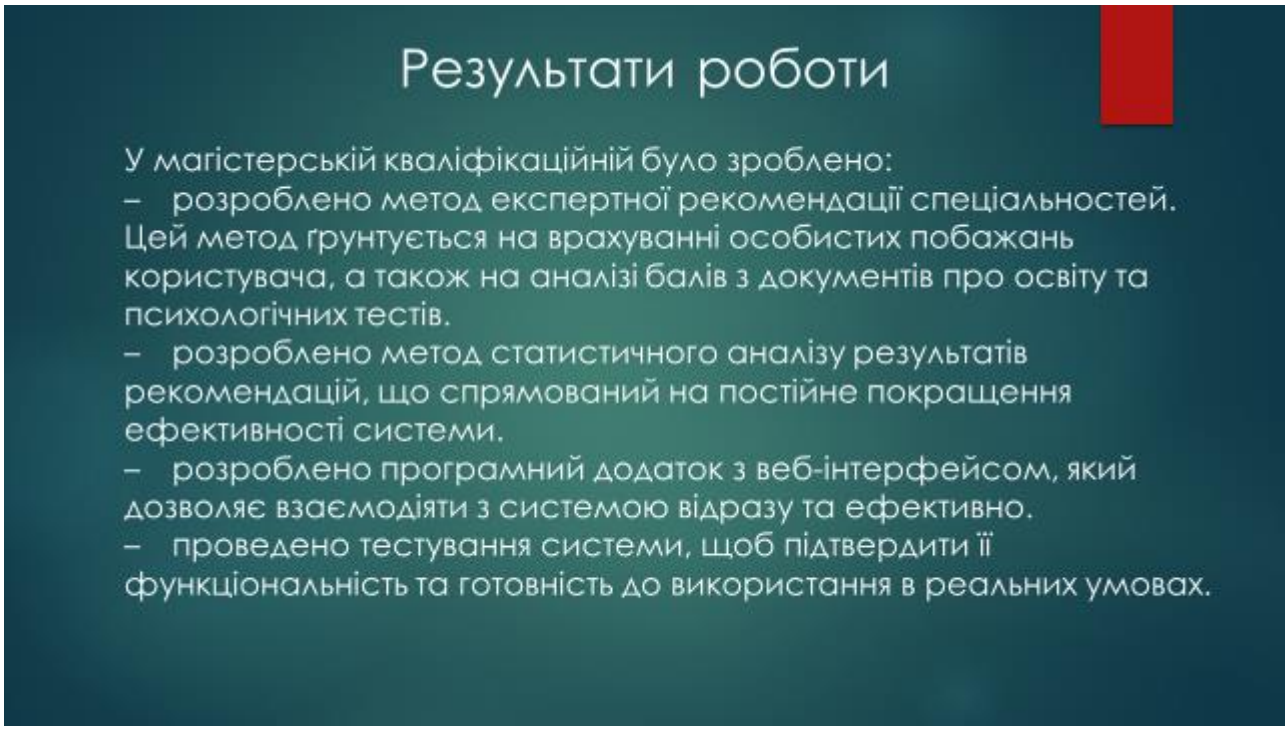

Рисунок Г.13 – Результати роботи