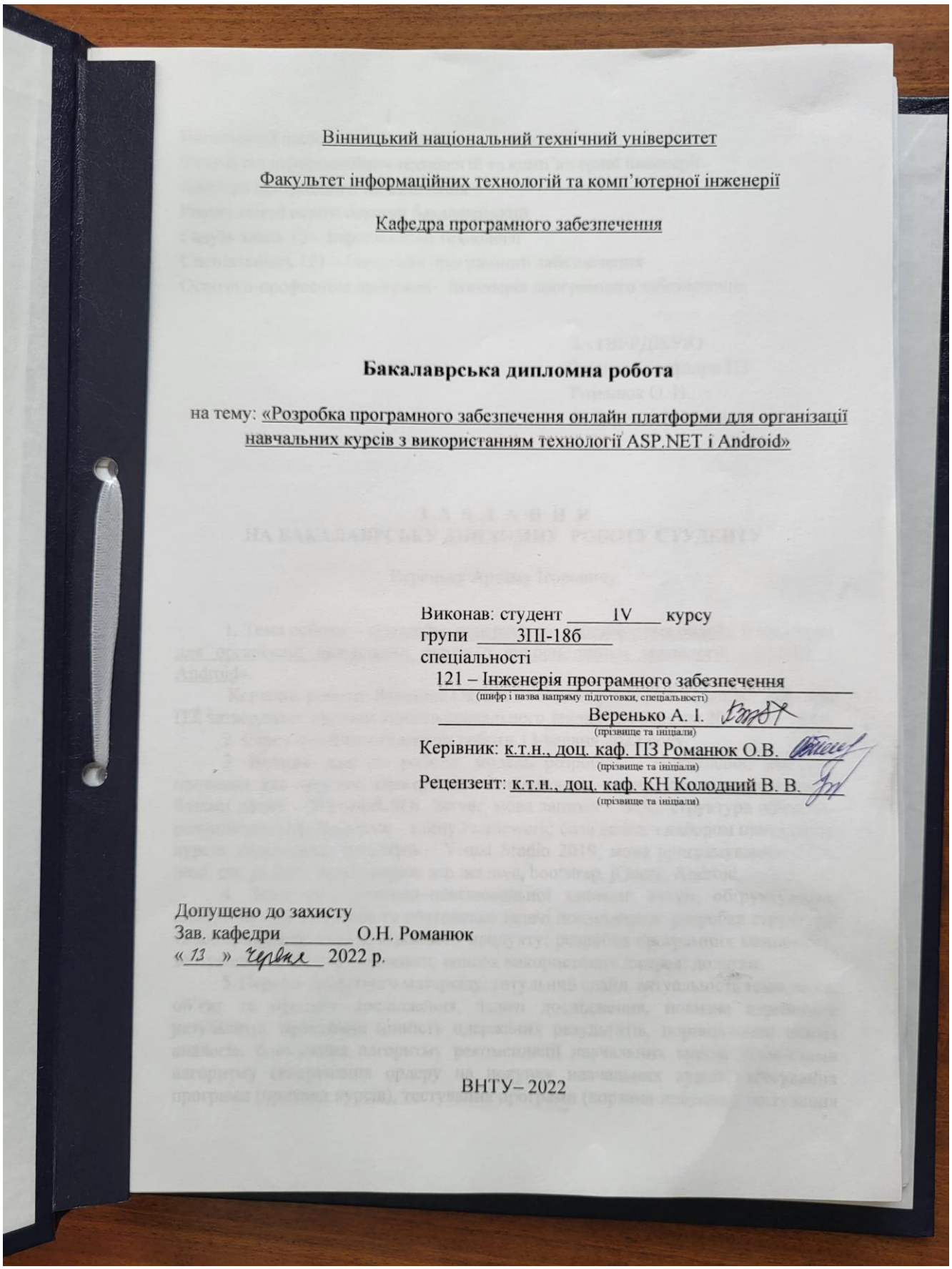

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення онлайн платформи для організації навчальних курсів з використанням технології ASP.NET і Android»

Виконав: студент IV курсу
групи ЗПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Веренько А. І.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Романюк О.В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН Колодний В. В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____ О.Н. Романюк
«13» червня 2022 р.

ВНТУ– 2022

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
25 березня 2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Вереньку Артему Ігоровичу

1. Тема роботи – «Розробка програмного забезпечення онлайн платформи для організації навчальних курсів з використанням технології ASP.NET і Android».

Керівник роботи: Романюк Оксана Володимирівна , к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу 24 березня 2022 року №66.

2. Строк подання студентом роботи 13 червня 2022 року.

3. Вихідні дані до роботи: модель розробки – водоспадна; інтернет протокол для передачі електронних повідомлень – smtp; система управління базами даних – Microsoft SQL Server; мова запитів – SQL; структура об'єктно-реляційного відображення – Entity Framework; база даних з набором навчальних курсів; середовище розробки – Visual Studio 2019; мова програмування – C#, html, css, js, Java; фреймворки: asp.net mvc, bootstrap, jQuery, Android.

4. Зміст розрахунково–пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структ та алгоритмів роботи програмного продукту; розробка програмних компон тестування програми; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд, актуальність теми, мета, об'єкт та предмет дослідження, задачі дослідження, новизна отриманих результатів, практична цінність одержаних результатів, порівняльний аналіз аналогів, блок-схема алгоритму рекомендації навчальних курсів, блок-схема алгоритму генерування ордеру на покупку навчальних курсів, тестування програми (приклад курсів), тестування програми (корзина покупок), тестування

програми (сторінки авторизації), тестування програми (інструменти управління контентом), тестування програми (Android додаток), апробація та публікації результатів роботи, фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О.В., к.т.н., доцент кафедри ПЗ	25.03.22	10.06.22

7. Дата видачі завдання 25 березня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження. Вибір архітектури програмного компоненту	26.03.2022 – 02.04.2022	Вик.
2	Розробка структури онлайн платформи	03.04.2022 – 09.04.2022	Вик.
3	Розробка алгоритмів персонального підбору курсів та генерування ордеру на покупку	10.04.2022 – 17.04.2022	Вик.
4	Вибір середовища та мови розробки	18.04.2022 – 28.04.2022	Вик.
5	Програмна реалізація онлайн додатку	29.04.2022 – 20.05.2022	Вик.
6	Тестування програми	21.05.2022 – 25.05.2022	Вик.
7	Оформлення матеріалів до захисту БДР	26.05.2022 – 10.06.2022	Вик.

Студент _____ (підпис) **Веренько А. І.** (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) **Романюк О. В.** (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 129 сторінок формату А4, на яких є 64 рисунки та 5 таблиць, список використаних джерел містить 29 найменувань.

У бакалаврській дипломній роботі проведено детальний аналіз стану існуючих освітніх онлайн платформ. Встановлено об’єкт, предмет, завдання та методи дослідження. Сформульовано мету досліджень – підвищення ефективності організації навчальних курсів на освітній онлайн платформі шляхом удосконалення алгоритмів персонального підбору курсів та генерування ордеру на покупку курсу.

Запропоновано алгоритм генерування ордеру на покупку шляхом впровадження в його роботу користувацьких сесій, які хешують дії користувача, що дозволило підвищити ймовірність покупки курсу. Удосконалено алгоритм персонального підбору навчальних курсів на основі розширеної інформації про користувача, що дозволяє підвищити рівень релевантності запропонованих курсів та в кінцевому рахунку підвищити рівень їх монетизації.

Програмне забезпечення онлайн платформи для організації навчальних курсів розроблено з використанням мови програмування C#, фреймворку ASP.NET MVC та середовища розробки Microsoft Visual Studio 2019. Android додаток розрблено з викорристанням мови Java. Для розробки бази даних використана мова програмування SQL та система управління базами даних Microsoft SQL server. В якості ORM використано Entity framework. В результаті виконання бакалаврської дипломної роботи розроблено програмні засоби, працездатність і правильність роботи яких перевірено, підготовлена інструкція користувача.

Отримані в бакалаврській дипломній роботі результати можна використати для побудови онлайн платформи для організації навчальних курсів.

Ключові слова: освітня платформа, навчальний курс, онлайн магазин, ASP.NET, Android.

ABSTRACT

The bachelor's thesis consists of 129 pages of A4 format, which has 64 figures, 5 tables, the of references contains 29 titles.

A detailed analysis of the state of existing online educational platforms was conducted in the bachelor's thesis. The object, subject, tasks, and research methods are established. The purpose of the research is formulated - to increase the efficiency of the organization of training courses on the educational online platform by improving the algorithms of personal selection of courses and generating a warrant for the purchase of the course.

An algorithm for generating a purchase order by introducing user sessions that hash the user's actions into its work has been proposed, which has increased the probability of buying a course. The algorithm of personal selection of training courses on the basis of the expanded information on the user is improved allows to increase a level of relevance of the offered courses and ultimately increases the level of their monetization.

The software of the online platform for organizing training courses is developed using the C # programming language, ASP.NET MVC framework, and Microsoft Visual Studio development environment. The Android application is developed using the Java language. SQL programming language and Microsoft SQL Server database management system were used to develop the database. Entity Framework is used as ORM. As a result of the bachelor's thesis, software tools have been developed, the efficiency and correctness of which have been tested, and user instructions have been prepared.

The results obtained in the bachelor's thesis can be used to build an online platform for training courses.

Keywords: educational platform, training course, online store, ASP.NET, Android.

ЗМІСТ

ВСТУП 8
1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ 12
1.1 Аналіз стану існуючих освітніх онлайн платформ 12
1.2 Порівняльний аналіз аналогів 14
1.3 Аналіз методів розв’язання поставленої задачі 17
1.4 Постановка задач розробки онлайн платформи для організації навчальних курсів 19
1.5 Висновки 19
2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ 20
2.1 Вибір архітектури програмного компоненту онлайн платформи 20
2.2 Застосування інтернет протоколів для управління електронною поштою . 23
2.3 Розробка структури онлайн платформи 25
2.4 Особливості та підходи до розробки android-додатків освітнього спрямування 29
2.5 Розробка алгоритмів роботи програмного компоненту 31
2.6 Висновки 35
3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ 36
3.1 Варіантний аналіз і обґрунтування вибору мови програмування 36
3.2 Порівняльний аналіз середовищ розробки 39
3.3 Порівняльний аналіз СУБД 41
3.4 Програмна реалізація основних компонентів онлайн платформи 43
3.5 Програмна реалізація алгоритмів роботи онлайн платформи для організації навчальних курсів 50
3.6 Розробка Android платформи для навчальних курсів 56
3.7 Висновки 59
4 ТЕСТУВАННЯ ПРОГРАМИ 60

4.1 Аналіз методів тестування програмного забезпечення 60
4.2 Тестування розробленого програмного продукту 61
4.3 Розробка інструкції користувача 67
4.4 Висновки 75
ВИСНОВКИ 76
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 77
ДОДАТКИ 80
Додаток А. Технічне завдання ..... **Ошибка! Закладка не определена.**
Додаток Б. Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень 85
Додаток В. Лістинг програми 87
Додаток Г. Графічна частина 121

ВСТУП

Обґрунтування вибору теми дослідження. В сучасному світі можна спостерігати щорічне зростання кількості запитів на перехід освітнього процесу в онлайн простір [1]. Найрізноманітніші онлайн курси, тренінги, інтенсиви, вебінари, спільні навчальні проекти стають дедалі популярніші не лише в Україні, а й у світі загалом. Формат онлайн навчання має безліч вагомих переваг у порівнянні з класичною освітою [2]. Освіта онлайн дозволяє залучити до навчального процесу одразу будь-яку кількість людей з будь-якого куточку планети за умови наявності необхідного технічного забезпечення та доступу до інтернету.

Насправді ринок онлайн освіти займає одне з ключових місць в освітній галузі. Оскільки ринковий попит на онлайн освіту перевищує пропозицію, це створює великий потік інвестицій в сферу освітніх технологій, що дає можливість розвивати все більш інноваційні та привабливіші онлайн навчальні платформи. В свою чергу, впровадження нових технологій в освітній процес дозволяє стверджувати, що попит на якісні освітні сервіси щорічно зростатиме через підвищення ефективності онлайн навчання.

За даними сайту statista.com очікувалося, що світовий ринок онлайн навчання до 2022 року перевищить 243 мільярди доларів [3]. Наприклад, у Сполучених Штатах загальний рівень вступу до вищих навчальних закладів дещо знижувався за останнє десятиліття, а онлайн-набір студентів зростає 14-й рік поспіль. У зв’язку з безпрецедентним поширенням пандемії COVID-19, системи освіти в усіх куточках світу були змушені перетворитися на якийсь тип дистанційного навчання [4].

Онлайн навчальна платформа – це веб-простір або портал для освітнього вмісту та ресурсів, який пропонує студентам усе необхідне для навчання в одному місці: курси, лекції, ресурси, можливості зустрічатися та спілкуватися з іншими студентами тощо. Це також чудовий спосіб для учня та вчителя стежити за прогресом учнів, вести журнали оцінок, розміщувати розклад [4].

Навчальні онлайн сайти – зазвичай це великі складні платформи з найрізноманітнішим функціоналом. В даній роботі буде розроблено сайт, що працюватиме в трьох режимах: адміністратора, авторизованого та не авторизованого користувача. Адміністратор матиме окрему адмін-панель з допомогою якої зможе розміщувати курси на сайті. Користувач може зайти на сайт, пройти реєстрацію та придбати доступ до начального курсу. Сам курс для проходження також буде доступний користувачеві в Android додатку, що скачується після оплати курсу.

Провівши огляд існуючих навчальних платформ можна зазначити, що на більшості з них неможливо розмістити курси під мало відомим брендом, а в разі продажу власного курсу з його власника часто стягується комісія платформи, що може бути дуже значною. Часто платформа має власний формат курсів, що не дає можливості адаптувати матеріали під потреби користувача. Більшість освітніх платформ не мають інтеграції з Android додатками. Тому проектування, розробка дизайну та програмування освітньої веб-платформи з урахуванням описаних недоліків, а також розробка Android додатку є досить актуальними задачами.

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

Мета та завдання дослідження. Метою дослідження є підвищення ефективності організації навчальних курсів на освітній онлайн платформі шляхом удосконалення алгоритмів персонального підбору курсів та генерування ордеру на покупку курсу.

Відповідно до поставленої мети в бакалаврській дипломній роботі потрібно вирішити такі **завдання**:

– провести аналіз найбільш ефективних підходів до розробки освітніх платформ;

– визначити принципи створення структури та архітектури онлайн освітньої платформи;

– розглянути можливості застосування існуючих інтернет протоколів для передачі електронних повідомлень;

– розробити алгоритм генерації ордерів на покупу;

– розробити алгоритм персонального підбору курсів зареєстрованим користувачам;

– розробити програмне забезпечення онлайн платформи з компонентами для розміщення та монетизації навчальних курсів;

– розробити Android платформу для розміщення навчальних курсів;

– провести тестування та розробити інструкцію користувача розробленої онлайн платформи для організації навчальних курсів.

Об’єкт дослідження – процес розробки онлайн платформ для організації навчальних курсів.

Предмет дослідження – методи та засоби розробки та керування онлайн платформою для організації навчальних курсів.

Методи дослідження. У процесі дослідження використовувались: комплексний аналіз для дослідження переваг та недоліків наявних технічних рішень задачі та формування нових функціональних характеристик і вимог; теорія алгоритмів для розробки та вдосконалення алгоритмів ПЗ; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень; методи тестування для підтвердження коректної роботи платформи та її відповідності поставленим вимогам.

Новизна отриманих результатів.

1. Удосконалено алгоритм генерування ордеру на покупку, у якому, на відміну від інших, впроваджено користувацькі сесії, які хешують дії користувача, що дозволило підвищити ймовірність покупки курсу.

2. Удосконалено алгоритм персонального підбору навчальних курсів, який, на відміну від інших алгоритмів, враховує розширену інформацію про користувача, що дозволило підвищити рівень релевантності запропонованих курсів та в кінцевому рахунку підвищити рівень їх монетизації.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмне забезпечення онлайн платформи для організації освітніх курсів.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У наукових працях, опублікованих у співавторстві, автору належать зокрема такі результати: вибір архітектури програмної компоненти платформи для монетизації навчальних курсів [5]; особливості використання інтернет протоколів для управління електронною поштою [6]; особливості та підходи до розробки android-додатків освітнього спрямування [7]; використання графічного фреймворку LIBGDX для розробки кросплатформних ігор [8]; оптимізація запитів до баз даних [9].

Апробація результатів роботи. Результати роботи доповідалися на:

– XXII Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій» (2022 р., м. Одеса);

– Міжнародній науково-практичній інтернет-конференції. Пам`яті Олексія Петровича Стахова «Електронні інформаційні ресурси: створення, використання, доступ» (2021 р., м. Суми/Вінниця).

– XXI Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій» (2021 р., м. Одеса);

– XLIX, LI Науково-технічні конференції факультету інформаційних технологій та комп'ютерної інженерії (2020, 2022 р., м. Вінниця);

Публікації. Основні результати досліджень опубліковано в 5 наукових працях у збірниках матеріалів конференцій.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану існуючих освітніх онлайн платформ

Комп’ютерні навчальні платформи не є чимось новим на ринку. Вони почали з’являтися синхронно з розвитком комп’ютерних технологій та тенденцією до оцифрування інформації. Спочатку це були примітивні комп’ютерні програми схожі на електронні довідники. Після появи інтернету освітньою платформою можна було вважати, наприклад, вікіпедією.

Зараз освітня платформа – це часто корпоративний продукт, що виконує потреби бізнесу. Колосальні інвестиції, що залучали до таких продуктів, сприяли розвитку їх зручності та функціональності. Тому освіта онлайн з кожним роком набуває все більшого поширення та стає однаково бажаною в порівнянні з традиційною освітою.

Подібні онлайн навчальні платформи бувають найрізноманітніші. Їх можна класифікувати за цільовим призначенням наступним чином:

– Learning Destination Sites (далі LDS) – це переважно невеликі веб-сайти, де викладачі можуть розмітити свої навчальні матеріли у вигляді структурованих відповідно до обмежень сайту курсів [4]. Дуже часто схожі сайти є інструментом для монетизації таких курсів. При цьому власники сайту можуть брати плату як за розміщення курсу так і певний відсоток від його продажу. Популярні LDS дозволяють користувачеві обирати напрямки курсів, теми, викладачів, тривалість програми та її складність. Деякі сайти можуть пропанувати підписку одразу на певний пакет курсів або давати користувачеві можливість купувати той курс, що здається йому привабливим. Курси, що пропонуються на LDS, як правило, розраховані на самостійне проходження.

– Learning Management Systems (далі LMS) – це більш спеціалізоване корпоративне програмне забезпечення, що пропонується в мережевому чи локальному форматах використання [4]. Їх часто використовують як навчальну

систему для шкіл, коледжів чи університетів або інших навчальних закладів. У ній реалізовані механізми для взаємодії учитель-учень.

– Learning Management Ecosystems (далі LME) – часто це потужна онлайн навчальна платформа, що включає в себе одразу декілька інформаційно-технічних рішень, таких як підтримка адаптивних механізмів навчання, функції керування навчальним вмістом, програмне забезпечення для створення курсів, інструменти для оцінювання тощо [4]. LME поширені у великих організаціях, школах або університетах, які пропонують багато онлайн-програм і навчальних ресурсів.

У даній роботі буде реалізовано компоненти LDS платформи. Не зважаючи на існування великої кількості подібних навчальних платформ розробка власного аналогу дозволить уникнути суттєві недоліки, що присутні в існуючих рішеннях.

По-перше, такі навчальні платформи часто створенні під потреби конкретного бізнесу без вільного доступу для розміщення курсів. Більшість платформ можуть брати плату за розміщення курсу а також відсоток від його продажу. Також щоб зробити ваш курс «більш видимим» на сайті, власники платформи можуть брати додаткову плату за активну його рекламу. Освітня платформа може встановлювати політику середньої ціни на розміщені на ній курси, що може обмежити потенційний заробіток викладача.

По-друге, існуючі LDS освітні платформи накладають ряд обмежень до того, як має бути структурований і представлений курс. Такі обмеження можуть не дозволяти реалізувати вимоги замовника. Часто платформи вимагають представити курс у форматі відеороликів, що може не підходити викладачу. Платформа може не мати функціоналу керування потоками студентів.

Отже, в даному підрозділі було розглянуто класифікацію онлайн навчальних платформ за цільовим призначенням та визначено доцільність розробки власного Learning Destination Sites. Розробивши власну платформу, буде отримано контроль над розміщенням та монетизацією навчальних курсів, що є достатньою перевагою для підтвердження актуальності обраної теми.

1.2 Порівняльний аналіз аналогів

Різноманітні онлайн навчальні платформи виникали і розвивалися уже досить довгий час. Ця сфера зараз є достатньо інноваційною і має велику кількістю комерційних продуктів на ринку. Далі розглянемо найпопулярніші представлені програмні рішення визначивши їхні основні переваги та недоліки:

– Coursera;
– Udemy;
– Prometheus.

Coursera – це одна з найбільших популярних платформ для розміщення навчальних курсів. Дана платформа була створена у 2012 році та орієнтована на англомовного споживача [10]. Її засновниками вважають професорів Стенфордського університету Ендрю Нґ і Дафні Коллер. Основна мета платформи Coursera дати можливість студентам з різних куточків світу отримати професійну освіту онлайн. Для цього Coursera співпрацює з найвідомішими університетами та багатьма іншими освітніми організаціями, що визнані передовими дистриб'юторами освіти. Сертифікати та навчальні ступені, що студент може отримати на Coursera визнаються у всьому світі.

Основними перевагами платформи Coursera є наявність великої кількості професійних курсів та зручний функціонал для їх перегляду. Основним недоліком є неможливість розмістити на платформі курс під маловідомим брендом. Головна сторінка освітньої платформи зображена на ринку 1.1.

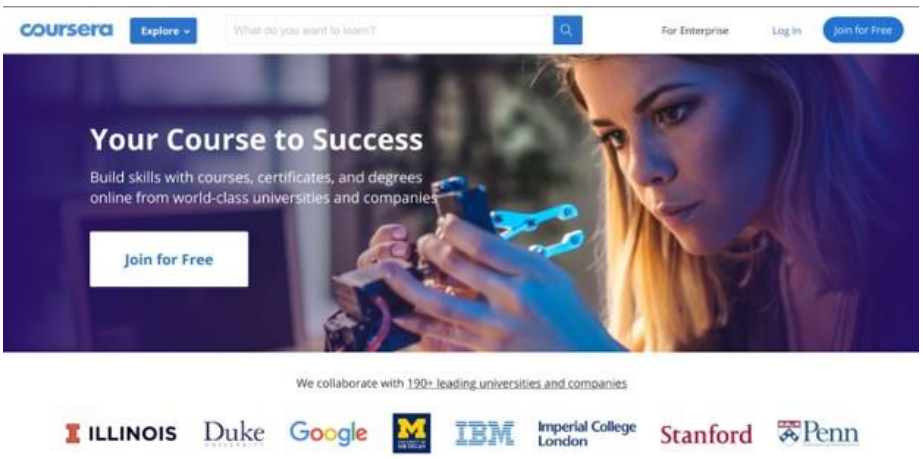

Рисунок 1.1 – Головна сторінка платформи Coursera

Udemy – можливо найпопулярніша онлайн освітня платформа, з понад 185 000 курсів різних рівнів та напрямків [11]. Сьогодні вона налічує близько 49 мільйонів студентів. Дана платформа була заснована 2010 року і з тих пір стрімко розвивалася набираючи популярність. Контент на платформі спрямований на підвищення рівня кваліфікації працівників а також на навчання студентів. На платформі пропонуються курси з програмування, маркетингу, дизайну, наук про дані, фондових ринків, криптоіндустрії, медицини тощо.

Основними перевагами платформи Udemy є наявність великої кількості професійних курсів, зручний функціонал для роботи з онлайн курсами, наявність особистого кабінету користувача. Основним недоліком є наявність великої комісії з продажу курсів та відсутність безкоштовного контенту. Головна сторінка освітньої платформи Udemy зображена на ринку 1.2.

Рисунок 1.2 – Головна сторінка сайту Udemy

Prometheus – одна з найбільших платформ онлайн-освіти в Україні [12]. Вона є хостом для багатьох навчальних курсів різних напрямків. Основною перевагою даної платформи є її українське походження а отже наявність курсів на українській мові. Також перевагами є можливість розмістити курси під маловідомим брендом та частково безкоштовний контент. Основним недоліком

є наявність комісії з продажу навчальних курсів, не можливість адаптувати формат курсу під потреби викладача. Головна сторінка платформи Prometheus зображена на рисунку 1.3.

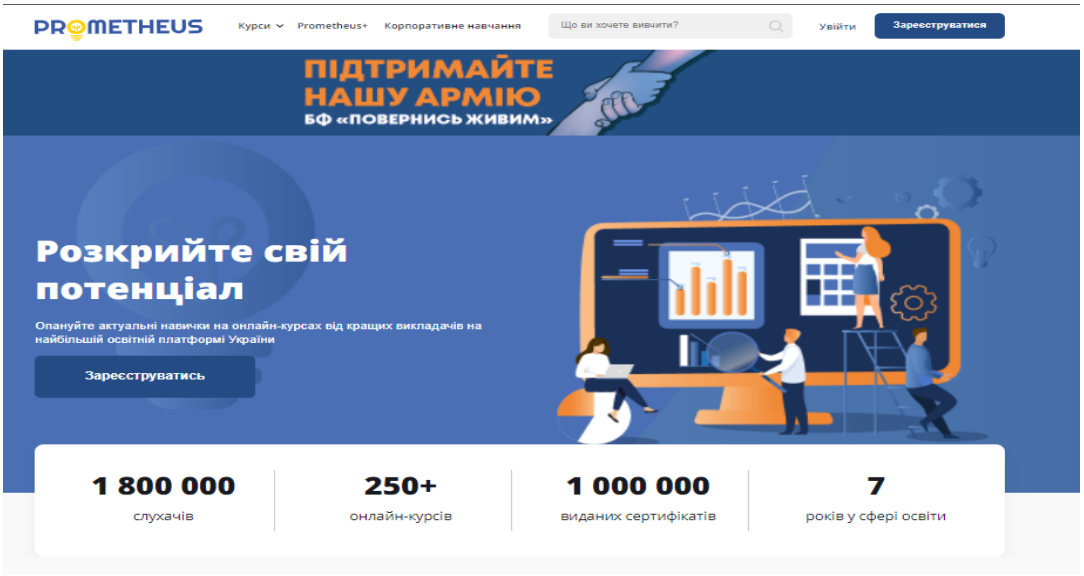

Рисунок 1.3 – Головна сторінка сайту Prometheus

Порівняльну характеристику освітніх платформ наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	Coursera	Udemy	Prometheus	AVDevU
Містить велику кількість навчальних курсів	1	1	0	1
Можливість розмістити курси під маловідомим брендом	0	1	1	1
Наявність комісії з продажів курсів на платформі	0	0	1	1
Можливість продавати офлайн навчальні курси	0	0	0	1
Наявнісь Android платформи	1	1	0	1
Підсумковий результат	2	3	2	5

В результаті порівняння існуючих аналогів було зроблено висновок, що розробка власної онлайн платформи для організації навчальних курсів є доцільною. В результаті розробки отримаємо продукт, який об’єднає переваги та не міститиме приведених недоліків існуючих аналогів.

1.3 Аналіз методів розв’язання поставленої задачі

Сьогодні онлайн освітні платформи бувають найрізноманітніші. Такими можна назвати: платформи для онлайн курсів, платформи для онлайн конференцій, електронні класи, електронні підручники, додатки для розвитку пам’яті, додатки для тестування, додатки для підготовки до екзаменів, додатки для розвитку дітей, додатки для навчання людей з обмеженими можливостями тощо [6]. При тому великі навчальні платформи можуть об’єднувати в собі одразу декілька видів з наведеного вище переліку. Така різноманітність сприяє появі безліч способів розробляти освітні платформи.

Одним із доволі популярних способів реалізувати навчальну платформу є розробка її у вигляді комп’ютерної гри. Переваги даного способу реалізації: представлення інформації у графічному вигляді та поєднування процесу навчання з процесом гри. Таким чином можна добре утримувати залучену аудиторію мотивуючи її ігровими досягненнями, змаганнями між собою та проходженням компанії гри. Гарним рішенням, що дозволяє реалізувати кросплатформену гру є графічний фреймворк LIBGDX [8]. Недоліком можна визначити розсіювання уваги користувача на проміжні дії, що не відносяться до навчання.

Найбільших представленою на ринку є освітня платформа у вигляді веб-додатку. Такий підхід забезпечує ряд суттєвих переваг в порівняні, наприклад, з платформою, що встановлюється на комп’ютер, андроїд додатком, графічною грою чи взагалі ботом, що інтегрований в іншу платформу (наприклад, освітнім телеграм ботом).

По-перше, веб-додатки дозволяють реалізувати весь функціонал потрібний сучасному користувачеві. Це означає, що будь-який вид освітніх платформ,

нехай це платформа для онлайн курсів чи онлайн клас, будуть здаватися органічними саме у веб-середовищі.

По-друге, будуючи веб-додаток програміст зосереджується на його майбутній роботі в мережі. Під цим розуміємо, що така платформа буде розмішена на спеціальному сервері до якого бажаючі будуть мати доступ через інтернет.

Розглянемо технології, які будуть використані для побудови освітньої веб-платформи. Оскільки це в першу чергу сайт, виділимо окремо технології для фронтенду і для бекенду.

Для написання фронтенду використаємо HTML, CSS, JS. Це стандартний набір технологій без яких не обходиться жодна сучасна веб-сторінка. Також до проекту включимо CSS фреймворк Bootstrap та бібліотеку JQuery.

Для розробки бекенд частини звернемося до популярного в даний момент рішення ASP.NET MVS фреймворку. Це фреймворк для розробки веб-додатків, випущений компанією Microsoft, що реалізує шаблон модель–представлення–контролер [13][14]. Даний фреймворк містить велику кількість готових архітектурних рішень, які легко можна впровадити розроблячи власний веб-додаток.

В якості бази даних проекту використаємо Microsoft SQL server. Це одна з найпопулярніших СУБД, що розроблена Microsoft [15][16]. Вона поширюється по ліцензії. Вона має широкий функціонал якого більше чим достатньо для розробки платформи.

Важливим компонентом розробленої онлайн платформи є ORM. Це інструмент, що використовується для перетворення моделей програми в формат, з яким може працювати баз даних. Це дозволяє програмісту описати логіку роботи з базою даних без написання SQL скриптів. В даній роботі в якості ORM буде впроваджено Entity Framework [17].

Отже, в даному підрозділі були розглянуті методи та технології, що можуть бути застосовані при розробці онлайн платформи для організації курсів.

1.4 Постановка задач розробки онлайн платформи для організації навчальних курсів

Після проведення аналізу питання розробки онлайн платформи для організації навчальних курсів, було визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

– провести аналіз найбільш ефективних підходів до розробки освітніх платформ;

– визначити принципи створення структури та архітектури онлайн освітньої платформи;

– розглянути можливості застосування існуючих інтернет протоколів для передачі електронних повідомлень;

– розробити алгоритм генерації ордерів на покупу;

– розробити алгоритм персонального підбору курсів зареєстрованим користувачам;

– розробити програмне забезпечення онлайн платформи з компонентами для розміщення та монетизації навчальних курсів;

– розробити Android платформу для розміщення навчальних курсів;

– провести тестування та розробити інструкцію користувача розробленої онлайн платформи для організації навчальних курсів.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто стан питання розвитку існуючих освітніх онлайн платформ. Також було проведено аналіз присутніх на ринку аналогів та проведено їх порівняння між собою та розроблюваним продуктом. У результаті доведено доцільність розробки бакалаврської дипломної роботи, а також проведено аналіз існуючих підходів до вирішення поставленої задачі. Було встановлено основні завдання, які необхідно виконати для розробки програмного продукту.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Вибір архітектури програмного компоненту онлайн платформи

Архітектура платформи для організації навчальних курсів має передбачати велику кількість користувачів, що одночасно хочуть отримати доступ до ресурсів розміщених на цій платформі. Такий вид платформи правильно називати розподіленим мережевим застосунком. Найбільш поширеною концепцією при побудові розподілених мережевих застосунків виділяють архітектуру сервер-клієнт, де сервер виконує усю логіку програми, що пов’язана з взаємодією з базою даних, обробкою запитів, генеруванням відповідей на запити тощо, а клієнт – це оболонка, що доступна кінцевому користувачеві для взаємодії з сайтом [18]. Схему такої архітектури можна побачити на рисунку 2.1.

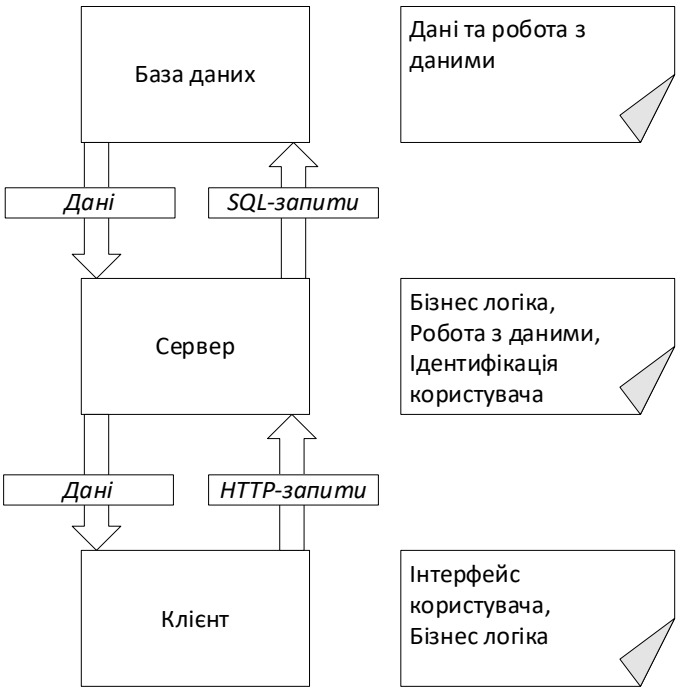

Рисунок 2.1 – Клієнт-серверна архітектура веб-додатку

Розглянемо компоненти представленої схеми більш детально. Основною програмною частиною веб-додатку є сервер. Він включає в себе всю складну бізнес-логіку та логіку взаємодії з клієнтом та базою даних. Сервер містить

механізм ідентифікації користувача. Такий механізм дозволяє забезпечити індивідуальний доступ клієнта до сервера через спеціальне API. Взаємодія з базою даних представленого веб-додатку доступна лише зі сторони сервера, що унеможливлює небажаний доступ до збережених даних зі сторони клієнта [5].

Широко розповсюдженим підходом до розробки архітектури веб-додатку є його базування на шаблоні Model-View-Controller (далі MVC). Це шаблон у розробці програмного забезпечення, що зазвичай використовується для реалізації інтерфейсів користувача, даних і логіки керування. Він виділяє поділ між бізнес-логікою програмного забезпечення та її представленням [14].

Власний підхід до реалізації шаблону MVC представив Microsoft у своєму фреймворку ASP.NET. До його екосистеми було додано велику кількість різних програмних рішень, що спрощують програмісту створення, розробку і підтримку проектів. Особливо зручними є механізми генерації програмного коду, що вбудовані в Visual Studio, після чого програмісту достатньо лише налаштувати проект для його коректної роботи. Розглянемо особливості даної шаблону ASP.NET MVC.

MVC ділить програму на три базові компоненти – модель, представлення та контролер:

– модель є нічим іншим, як формою даних. У C# аналогом моделі служить клас. Об’єкти моделі зберігають дані, отримані з бази даних;

– представлення у MVC – це інтерфейс користувача з допомогою якого можна переглядати дані моделі та змінювати їх. Представлення в ASP.NET MVC – це HTML, CSS і деякий спеціальний синтаксис (синтаксис Razor), який полегшує взаємодію з моделлю та контролером;

– контролер в ASP.NET MVC - це клас, що обробляє запити користувача. Такі запити можуть бути ініційовані користувачем з рівня представлення. Контролер обробивши запит повертає нове представлення користувачу як відповідь [14].

Наступний малюнок наглядно ілюструє взаємодію між моделлю, представленням і контролером (рисунок 2.2). Представлення, що бачить

користувач платформи є згенерованим відображенням певної моделі. Обробка запитів, рендер представлення та керування відбувається через контролер.

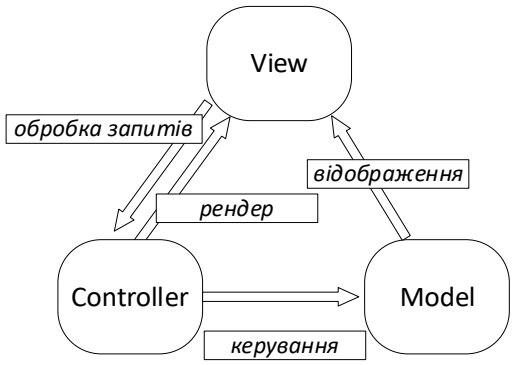

Рисунок 2.2 – Архітектура MVC веб-додатку

Перебіг запиту користувача в ASP.NET MVC зображено на рисунку 2.3.

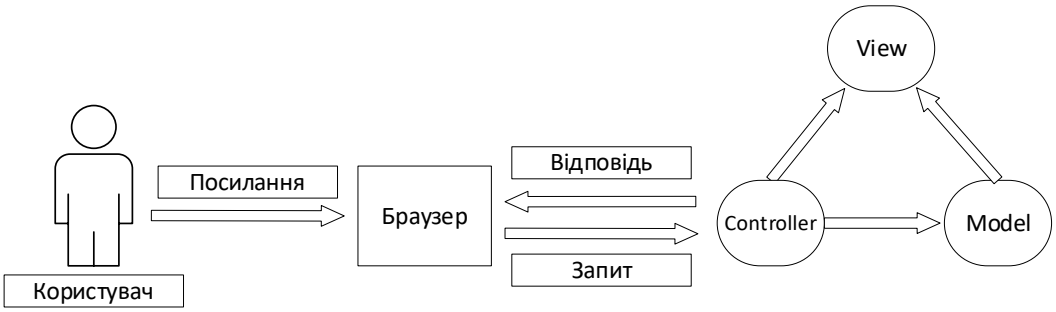

Рисунок 2.3 – Потік запитів в архітектурі MVC

Після вводу користувачем URL-адреси в пошукову стрічку браузера, вона передається на веб-сервер де направляється на контролер. Контролер взаємодіє з пов’язаними представленнями та моделями для цього запиту, створює відповідь і надсилає її назад до браузера [14].

Для розробки онлайн платформи для організації навчальних курсів було обрано ASP.NET MVC архітектуру програмного компоненту. Вона найкраще підходить для реалізації поставлених завдань, дозволяючи створити необхідний для платформи функціонал.

2.2 Застосування інтернет протоколів для управління електронною поштою

За довгий час розвитку мережевих технологій вміння застосовувати інтернет протоколи для управління електронними повідомленнями є необхідністю в індустрії інформаційних технологій. Користуючись поштовими сервісами можна уникнути роботи з графічними інтерфейсами, що представляють нам їх дистриб'ютори, а замість того працювати з повідомленнями безпосередньо в коді власних програм.

Використовуючи дані протоколи можна вирішувати широкий спектр задач, пов'язаних з накопиченням, передачею та збереженням інформації. До прикладу, їх можна застосувати при налаштуванні таргетованої реклами, автоматизованого листування з клієнтами, збереження електронних результатів, роботи зі звітами автоматизованого тестування тощо.

Розглянемо наступні протоколи: SMTP, IMAP, POP. На рисунку 2.4 зображена найпоширеніша схема їх використання. Так Simple Mail Transfer Protocol здебільшого використовується для надсилання повідомлень, а Internet Access Message Protocol та Post Office Protocol для отримання повідомлень.

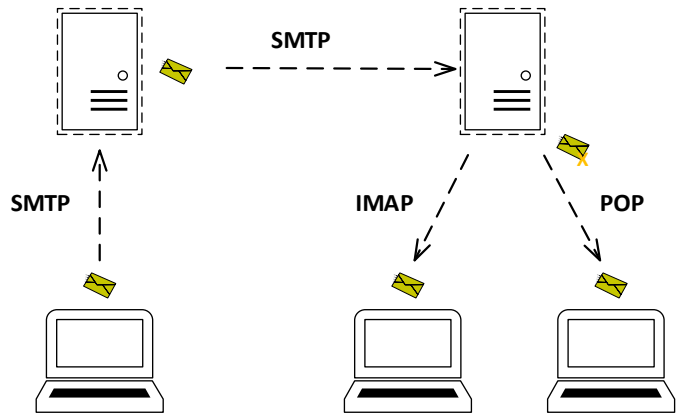

Рисунок 2.4 – Схема використання протоколів передачі повідомлень

Для зв'язку між поштовими серверами SMTP використовує стандартний порт TCP 25. Хоча поштові служби зазвичай приймають надсилання електронної

пошти від клієнтів на 587 або 465 порти. IMAP використовує порт 143 або 993. POP протокол – 110 чи 993 [19].

Розглянемо роботу SMTP детально. Щоб з'єднатися з одержувачем, відправнику пошти доступні команди MAIL, RCPT, DATA. Ці команди слугують для встановлення адреси повернення, встановлення одержувача та вказування початку повідомлення відповідно. Після встановлення з'єднання сервер може давати проміжні відповіді при виконанні команд. Ці відповіді міститимуть статуси виконання: 2xx – успіх, 4xx або 5xx – відмова.

На даний момент існує багато відкритих програмних рішень, що дозволяють уникнути роботи зі згаданими протоколами на низькому рівні. Такі рішення представляють зручне високорівневе API до цих протоколів. Розглянемо бібліотеку AE.Net.Mail [20]. Це POP/IMAP клієнтська бібліотека.

Щоб отримати повідомлення з почтового сервісу потрібно створити поштовий IMAP клієнт:

```
1. AE.Net.Mail.ImapClient imap = new AE.Net.Mail.ImapClient
2. (host, username, password, port, AuthMethods.Login, isSSL);
```

Конструктор даного класу приймає шість аргументів, серед яких: host – у Gmail це Imap. Gmail.com, username – зазвичай це адреса електронної пошти, password – спеціально згенерований токен доступу до поштової скриньки, port – 143 чи 993 залежно від налаштувань, метод Login та isSSL – встановлення шифрування. Ініціалізувавши ImapClient можна отримати доступ до широкого функціоналу для маніпуляції з повідомленнями на поштовому сервісі. Наприклад, роздрукувати усі повідомлення з сервера можна наступним чином:

```
1. var msg = imap.GetMessages(0, IC.GetMessageCount(), false);
2. foreach (var item in msg) {
3.   Console.WriteLine(item.Subject);
4.   Console.WriteLine(item.Body); }
```

Отже, описані технології буде застосовано при налаштуванні розсилки повідомлень на платформі для організації навчальних курсів.

2.3 Розробка структури онлайн платформи

У сучасному веб-просторі існує непомірна кількість різноманітних веб-сайтів здебільшого з унікальним зовнішнім виглядом. Але порівнюючи їх структуру доволі легко можна виділити спільні схожі за функціональністю структурні одиниці. Причиною такої подібності є те, що наявні зараз інтерфейси є звичними для користувачів і перегляд існуючих шаблонів несе ризики невигідного ускладнення взаємодії користувача з сайтом. Навпаки, частіше за все одними з основних завдань розробника є прогнозування очікування користувача про розташування тих чи інших сторінок сайту або їх блоків, спрощення розуміння користувачем логіки веб-сайту та зв'язку між різними його сторінками, що має забезпечити зручну та просту логіку навігації по веб-сайту.

Кожен сайт складається з сторінок та навігації між ними. Навігація на сторінках може бути представлена у вигляді різного виду меню, перехресних посилань чи карти сайту. Проста, легкозрозуміла структура веб-сайту допомагає користувачеві швидко знаходити потрібну йому інформацію. При розробці веб-магазину правильно розроблена його структура напряму впливає на кількість успішних продажів бізнесу. Тому розглянемо декілька основних різновидів структури сайту, визначивши найбільш вдалу для реалізації онлайн навчальної платформи для продажу курсів.

Лінійна структура сайту також відома як каскад – це структура, що нагадує ланцюжок, який складається із пов'язаних між собою компонентів [21]. На сайті не присутнє класичне меню використовуючи яке можна було б вільно переходити між різними сторінками. Натомість розробник пропонує користувачеві попередньо побудований цілісний маршрут переходів між сторінками, по якому користувач може рухатися вперед або назад. Такий вид структури сайту використовують тоді, коли власнику сайту потрібно провести користувача через попередньо визначені цілісні сценарії, наприклад, багатосторінкова реєстрація чи формування купівельного ордеру онлайн магазину. Приклад лінійної структури сайту зображено на рисунку 2.5.

Рисунок 2.5 – Схема каскадної моделі

Ієрархічна чи деревоподібна структура – характерна наявністю головної сторінки, що містить посилання на інші сторінки вниз по ієрархії [21]. Для навігації зазвичай використовується посилання на головну сторінку та перелік інших суттєвих для сайту посилань. Схема такої моделі зображена на рисунку 2.6.

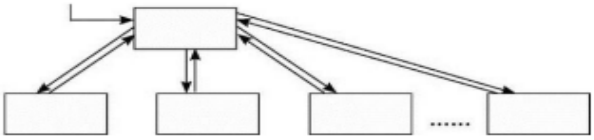

Рисунок 2.6 – Схема ієрархічної моделі

Сітка чи павутина – це одна з найхаотичніших систем побудови структури веб-сайту (Рисунок 2.7). Вона характеризується наявністю на кожній окремій сторінці посилання на будь-яку іншу сторінку веб-сайту. Посилання на сторінках створюють так званий ефект веб-павутини. Це дає можливість користувачеві легко переміщатися по сайту. Також особливістю даної схеми є можливість автоматизованого пошуку необхідної інформації на сайті.

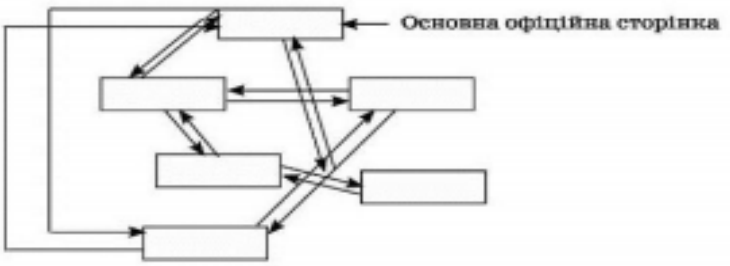

Рисунок 2.7 – Схема моделі сітка

Останньою розглянемо довільну структуру веб-сайту. Через гнучкість цієї системи вона є найпоширенішою серед запропонованих. Дана система накладає доволі мало обмежень на шаблон проектування структури сайту. Це дозволяє будувати сайт опираючись на логіку процесів, що на ньому відбуваються, що надає можливість прогнозувати та керувати поведінкою користувача. Тому користувач має можливість переходити на різні сторінки сайту в довільному порядку без відвідування проміжних сторінок.

Окремо слід виділити можливість створення посилання на певний контент всередині сторінки. Така практика широко застосовується і служить для того, щоб краще структурувати інформацію.

Для розробки онлайн платформи для організації освітніх курсів було обрано довільну структуру веб-сайту. Така структура дозволить найкраще реалізувати поставлені перед розробкою завдання сприяючи ефективності керування увагою клієнта під час перебування на сайті.

Схема розробленої онлайн платформи зображена на рисунку 2.8.

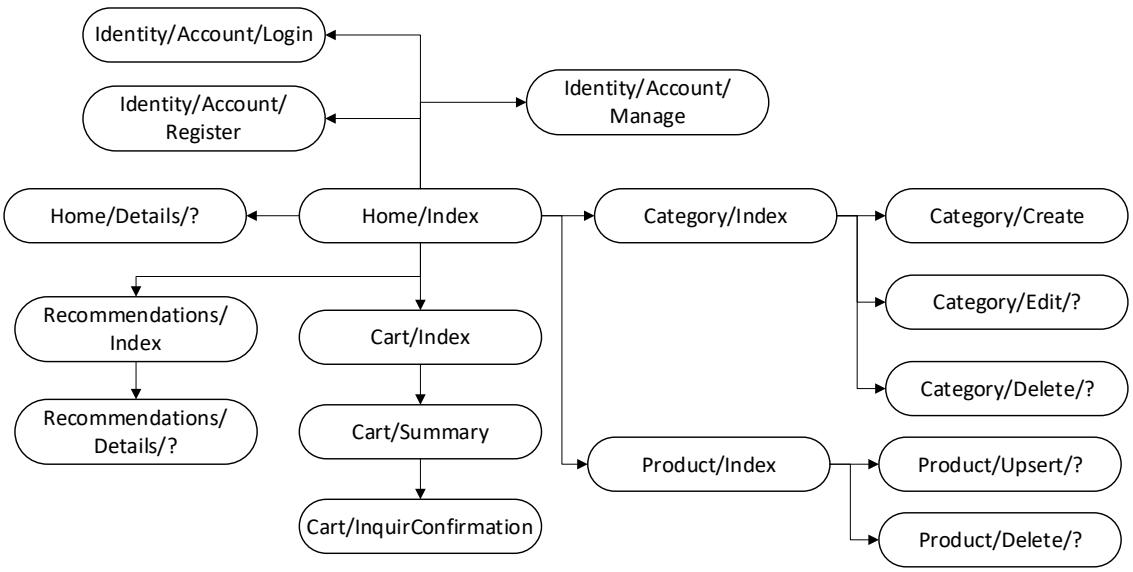

Рисунок 2.8 – Структурна схема онлайн платформи для організації курсів

Онлайн платформа включає такі сторінки:

– Home/Index – головна сторінка сайту, що містить меню сайту, панель для фільтрації навчальних курсів по категоріям та список навчальних курсів;

– Identity/Account/Login – сторінка для авторизації користувача;

– Identity/Account/Register – сторінка для реєстрації користувача;

– Identity/Account/Manage – сторінка керування акаунтом користувача;

– Home/Details/? – сторінка відображення детальної інформації про курс;

– Recommendations/Index – сторінка з рекомендованими курсами;

– Recommendations/Details/? – сторінка деталей про окремий курс;

– Cart/Index – сторінка для відображення корзини покупок;

– Cart/Summary – сторінка для відображення заявки на покупку;

– Cart/InquirConfirmation – сторінка з результатом транзакції;

– Category/Index – сторінка для управління категоріями курсів;

– Category/Create – сторінка для створення нової категорії курсів;

– Category/Edit/? – сторінка для редагування категорії курсів;

– Category/Delete /? – сторінка для видалення категорії;

– Product/Index– сторінка для управління курсами на сайті;

– Product/Upsert/? – сторінка для створення-редагування курсу;

– Product/Delete/? – сторінка для видалення курсу.

Керування основними сторінками веб-платформи відбувається з допомогою навігаційної панелі сайту. Розглянемо існуючі види навігаційних панелей, та виберемо одну, що найкраще підходитиме для поставлених завдань.

Поширеним видом розташування головного меню сайту є навігаційна панель у верхній частині екрану. Цей тип навігації складається з горизонтального списку розділів сайту, що зазвичай названі одним або двома словами.

Вертикальна навігаційна панель також досить поширена. Вона корисна для сайтів, що мають більш повний список розділів або довші заголовки. Вертикальна навігація найбільш часто зустрічається уздовж лівого боку веб-сторінки. Бічна панель навігації іноді потрібна саме як додаткова панель кнопок, наприклад, для підрозділів основного розділу.

Існує також варіант прихованої навігаційної панелі. Таке меню допомагає заощадити місце на веб-сайті. У більшості випадків її можна розкрити при натисканні на значок «гамбургера» в кутку. Багато користувачів виступають

проти прихованого меню, так як це тягне за собою зайві дії при переході на потрібну сторінку сайту. Але використання смартфонів змушує все більше використовувати приховану навігацію, а значок «гамбургера» все частіше з'являється на головних сторінках сайтів.

Для переходу між сторінками сайту розроблено звичайну горизонтальну навігаційну панель (Рисунок 2.9).

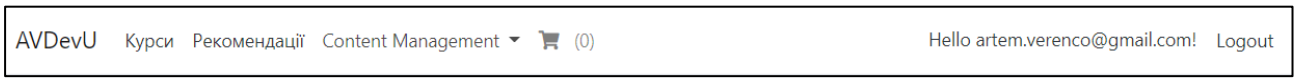

Рисунок 2.9 – Навігаційна панель сайту

На навігаційній панелі розміщено посилання на наступні розділи сайту: головна сторінка або сторінка курсів платформи, сторінка рекомендацій, розділ менеджменту контенту, який доступний тільки адміністратору сайту, розділ редагування особистої сторінки користувача, службові кнопки: вхід, вихід і реєстрація.

Отже, в розділі було розглянуто існуючі структури веб-сайту та обрано довільну структуру. Наведена структурна схема сторінок онлайн платформи. Також розроблена навігаційна панель сайту.

2.4 Особливості та підходи до розробки android-додатків освітнього спрямування

Android-додатки освітнього спрямування – це інтерактивні електронні інструменти, що розширюють можливості здобуття знань і навичок у найрізноманітніших напрямках. Вони пропонують рішення для реалізації доповненої реальності, штучного інтелекту, віддаленого навчання, гейміфікації освітнього процесу [7].

Існуючі освітні Android-додатки можна поділити на такі типи: платформи для онлайн курсів, платформи для онлайн конференцій, електронні класи, електронні підручники, додатки для розвитку пам’яті, додатки для тестів,

додатки для підготовки до іспиту, додатки для розвитку дитини, додатки для навчання людей з обмеженими можливостями (Рисунок 2.10).

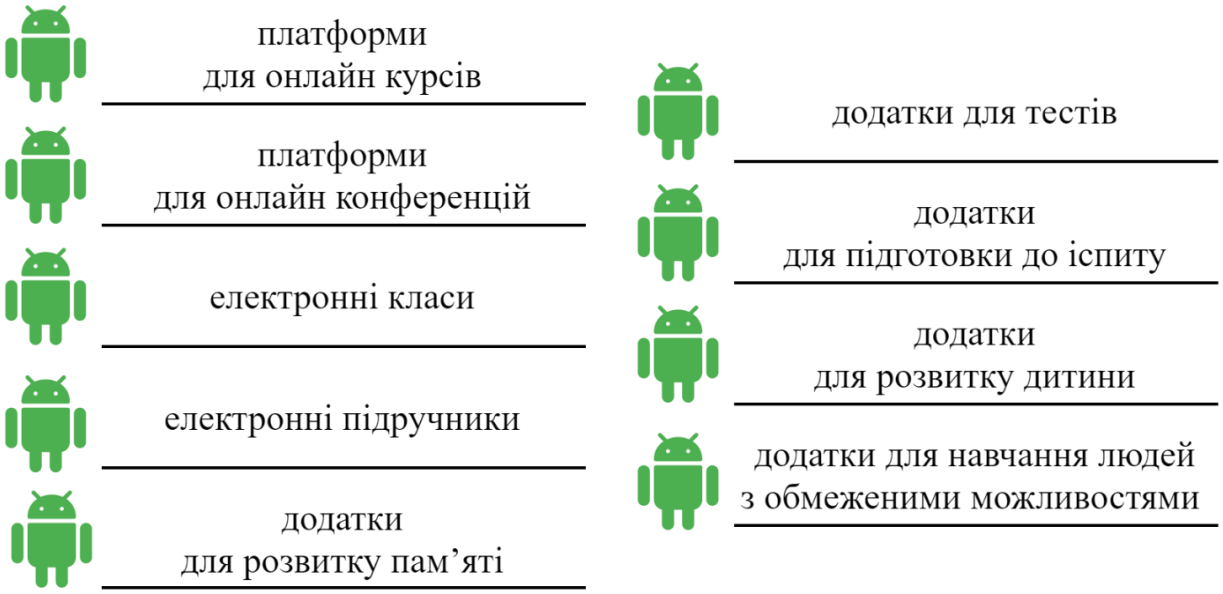

Рисунок 2.10 – Типи існуючих освітніх Android-додатків

Розробка будь-якого Android-додатку освітнього спрямування починається зі створення його мінімально життєздатної версії [7]. Потрібно розуміти, що компоненти, які необхідно реалізувати, разом повинні вирішувати проблему користувача на якого орієнтується проект. Розглянемо можливі компоненти освітніх Android-додатків (Таблиця 2.1).

Таблиця 2.1 – Компоненти освітніх Android-додатків

Форма реєстрації / авторизації	вікно з полями для вводу імені користувача та паролю, або вхід за допомогою соціальних мереж
Профіль користувача	вікно з полями для заповнення додаткової інформації про користувача; вікно для перегляду курсів, планів, оцінок тощо...
Платежі	правильно налаштовані грошові транзакції
Меню	зручна навігація між вікнами

Продовження таблиці 2.1

Налаштування	вікно налаштування програми
Панель приладів	вікно для створення курсів, тестів
Коментарі / форуми / чати	інструменти для взаємодії користувачів
Потокове відео	інструменти для проведення онлайн конференцій
Голосові команди	навігація в програмі для користувачів з вадами зору
AR/VR	інструменти для доповненої реальності

Існує багато сервісів, які надають готові рішення для реалізації згаданих вище компонентів. Наприклад, деякі рішення AWS як Amazon SES та Amazon SNS надають можливості для налаштування реєстрації/авторизації в додатку. Amazon S3 – забезпечує зберігання об’єктів через інтерфейс веб-служби. Платформа Firebase, що розроблена Google спеціально для мобільних додатків, також може розглядатися як комплексне рішення для реалізації реєстрації, авторизації, зберігання даних, аналітики тощо. Braintree і PayPal надають API для налаштування платежів в додатку [7].

В даній роботі буде розроблено Android-додаток, що відноситься до типу платформи для навчальних курсів.

2.5 Розробка алгоритмів роботи програмного компоненту

Одним із алгоритмів, що розглянемо у даній роботі є процес генерування ордеру на покупку вибраних користувачем курсів. Це доволі популярний алгоритм, що застосовується в багатьох інтернет магазинах. Особливістю даного алгоритму є робота з сесіями. Налаштувавши їх на сайті можна бути впевненим, що онлайн платформа збереже найнеобхідніші дані в пам’яті браузера. В даному випадку буде збережено інформацію про авторизованого користувача а також товари, які користувач додав до корзини. Після повторного входу на сайт, якщо сесія ще не завершилася користувач може продовжити роботу з сайтом з

останнього синхронізованого місця. Блок-схему алгоритму генерування ордеру на покупку вибраних користувачем курсів зображено на рисунку 2.11.

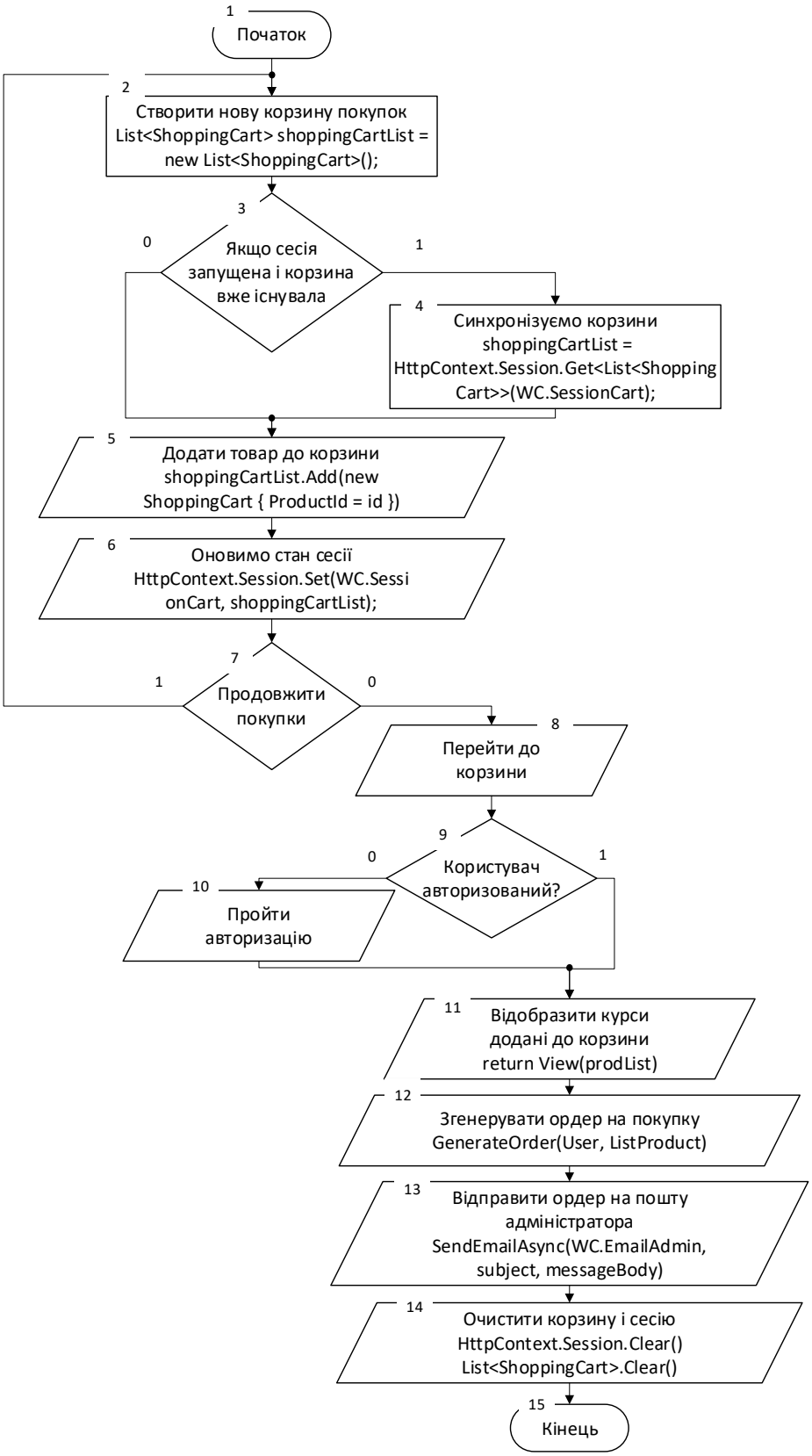

Рисунок 2.11 – Блок-схема алгоритму генерування ордеру на покупку курсів

Опишемо кроки, що містить приведений алгоритм:

Крок 1. Початок;

Крок 2. Створення нової корзини;

Крок 3. Перевірка чи існує сесія та корзина;

Крок 4. Синхронізація корзини;

Крок 5. Додати вибраний товар до корзини;

Крок 6. Оновлення стану сесії;

Крок 7. Перевірка бажання користувача продовжити покупки;

Крок 8. Перехід до корзини;

Крок 9. Перевірка чи користувач авторизований;

Крок 10. Авторизація користувача;

Крок 11. Відобразити курси додані до корзини;

Крок 12. Генерація ордеру на покупку;

Крок 13. Відправка ордеру на покупку адміністратору;

Крок 14. Очистити корзину і сесію;

Крок 15. Кінець.

Важливою частиною розробки онлайн навчальної платформи для організації курсів є створення алгоритму персонального підбору курсів для окремого користувача. Даний алгоритм опираючись на інтереси користувача генерує сторінку з курсами, що найбільше підходять зареєстрованому клієнту. Розглядаючи алгоритм детальніше можна виділити наступні кроки:

Крок 1. Початок;

Крок 2. Запит на відкриття сторінки рекомендацій;

Крок 3. Перевірити чи користувач авторизований на сайті;

Крок 4. Пройти авторизацію;

Крок 5. Виконати запит до бази даних для отримання інформації про поточного користувача;

Крок 6. Виконати запит до бази даних з метою вибору курсів спеціальність яких відповідає спеціальності користувача;

Кроки 7, 8, 9. Перевірити рівень поточного користувача;

Кроки 10, 11, 12. Відобразити курси, що підходять користувачу відповідно до його рівня;

Крок 13. Відобразити помилку, якщо інформація про користувача не знайдена:

Крок 14. Кінець.

Блок-схема даного алгоритму зображена на рисунку 2.12.

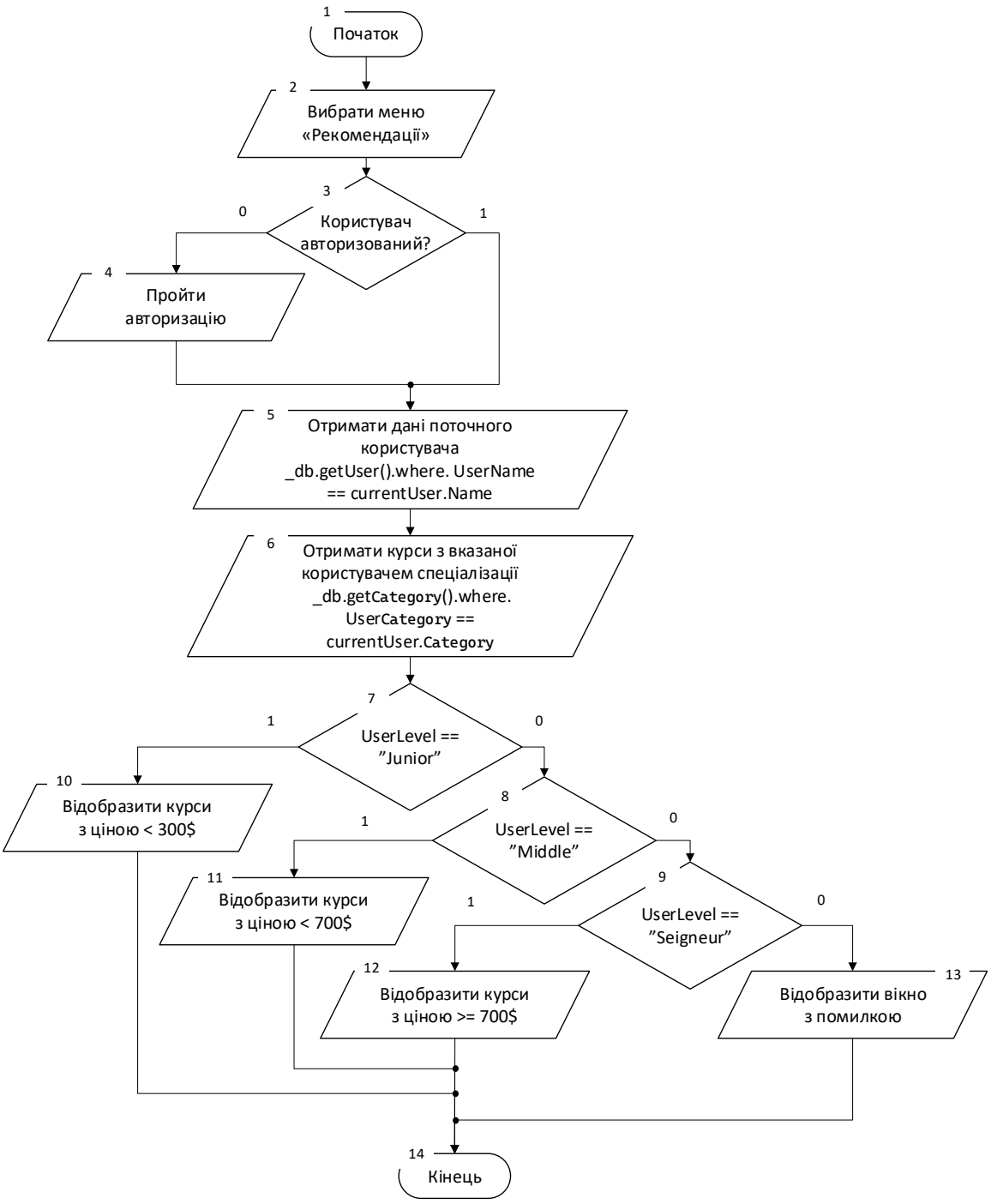

Рисунок 2.12 – Блок-схема алгоритму персонального підбору навчальних курсів

Отже, в даному підрозділі було розглянуто розроблені ключові алгоритми роботи онлайн платформи для організації навчальних курсів. Представлені блок-схеми для персонального підбору навчальних курсів користувачу, на основі його вподобань, та алгоритму генерування ордеру на покупку курсів.

2.6 Висновки

У другому розділі було розглянуто та обрано MVC архітектуру програмного компоненту для розробки онлайн платформи для організації навчальних курсів. Така архітектура найкраще підходить для реалізації поставлених завдань. Також була проведена робота над аналізом та проектуванням структури веб-платформи, приведена схема структури веб-сайту та розроблена його навігаційна панель. Розглянуто особливості та підходи до розробки android-додатків. Також у розділі приведені основні алгоритми програми, такі як: алгоритм генерування ордеру на покупку, алгоритм персонального підбору навчальних курсів окремому користувачу. Розроблені блок-схеми зазначених алгоритмів.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ

3.1 Варіантний аналіз і обґрунтування вибору мови програмування

При розробці онлайн платформи для організації навчальних курсів пріоритетними критеріями є її функціональність, стабільність і швидкодія, безпечність, привабливість та інтуїтивність дизайну. Вибір мови програмування відіграє в забезпеченні цих критеріїв не останню роль. Мова яку виберемо, повинна бути добре документованою і представляти необхідний нам функціонал і бібліотеки для реалізації запланованих функцій. Розглянемо найбільш відомі мови програмування, що можуть використовуватися для створення веб-додатків: C++, Python, Java та C#.22

C++ – це високорівнева мова програмування загального призначення [22]. На C++ можна програмувати як в традиційному декларативному стилі програмування, так і в ООП або ще більш складніших. Код написаний на C++ компілюється в машинний код. Мова C++ використовується для розробки прикладних програм, операційних систем, драйверів пристроїв, відеоігор тощо.

Для розробки веб-сайтів мову C++ використовують досить не часто, переважно в дуже нішових областях де вимагається максимальна оптимізація і контроль процесів. Переваги, що пропонує дана мова нівелюються її складністю.

Python – універсальна високорівнева мова програмування загального призначення [23]. Дана мова підтримує принципи ООП. Вона спроектована таким чином, що максимально дотримує філософію дизайну спрощення читабельності коду. Мова має динамічну типізацією. Написаний на Python код інтерпретується в байт-код, який переводиться віртуальною машиною в набір команд, що виконуються операційною системою.

Велика кількість розробників на Python у всьому світі щоденно працюють над впровадженням інновацій розробляючи удосконалені фреймворки та бібліотеки для вирішення технічних та бізнес-задач. Наприклад, для розробки веб-сайтів на Python існує популярна бібліотека Django. У цій бібліотеці підтримується велика кількість корисних функцій для кожного етапу розробки

веб-сайту. Django допомагає налаштувати реєстрацію, авторизацію та автентифікацію користувача, створити й налаштувати карти сайту, адмініструвати вміст, налаштувати RSS та інші складові. Django підтримує основні механізми захисту веб-сайту від можливих SQL ін'єкцій, крос-сайт підробки, клікджекінгу та багатьох інших небезпек. Важливою особливістю Django фреймворку є те, що він наслідує MVC архітектуру проекту, що спрощує проектування веб-сайту.

Java – це мова програмування, що підтримує принципи ООП [24]. Програми написані на Java обробляються компілятором в проміжний код або іншими словами байт-код. Після цього віртуальна машина Java відповідно до переданих її інструкцій знову перетворює отриманий код, що владний для вказаної операційної системи. Хоча синтаксис Java схожий на більш класичні мови програмування C чи C++, вбудовані інструменти дозволяють забрати відповідальність безпосереднім керуванням пам'яттю з користувача. Ці дії бере на себе віртуальна машина. Java розроблялась як платформо-незалежна мова, тому написана один раз програма може бути скомпільована для різних платформ.

Java має широкі можливості для розробки веб-сайтів. Розглянемо надзвичайно популярний серед Java розробників Spring фреймворк. Насправді, він являє собою звичайний контейнер залежностей з багатьма компонентами. Це, наприклад, доступ до бази даних, проксі, аспектно-орієнтоване програмування, RPC, веб-інфраструктура MVC. Усі ці компоненти в поєднанні дають можливість побудувати програмний додаток будь якої складності.

C# – об'єктно орієнтована мова програмування створена компанією Microsoft [25]. Її розробку очолював Андерс Хиджисберг. Основними концептом закладеним в розробку цієї мови було об'єднання в ній продуктивності C++ та простоти Java. Саме тому синтаксис C# близький до C++ і Java, а сама мова зберегла переваги обох своїх ідейних попередників. C# не підтримує динамічне присвоєння типів. Вона реалізовує поліморфізм та дає можливість перевантажувати функції. Також у C# реалізований широкий функціонал для інтеграції програм з XML файлами.

C# в даний час займає велику частину ринку програмних технологій. На цій мові розробляють як костюмерні програми та ігри, додатки на андроїд так і сучасні інноваційні веб-сайти. Для розробки веб-сайтів Microsoft створила та розвиває окремий фреймворк ASP.NET, що схожий на Spring у Java. Однією з переваг даного фреймворку є те, що програми написані на ньому можуть виконувати як клієнтські, так і серверні сценарії.

ASP.NET дотримується архітектури MVC, що спрощує проектування веб-сайту. Різноманітні функції генерації коду дозволяють скоротити час його написання і водночас забезпечують певний стандарт, якого має дотримуватися розробник, що призводить до підвищеної продуктивності і масштабованості. Також фреймворк має засоби контролю безпеки програмного продукту.

Результати порівняння розглянутих мов програмування за обраними критеріями наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій	C++	Python	Java	C#
Об’єктно-орієнтованість	0,5	1	1	1
Простота синтаксису	0,5	1	1	1
Повнота та доступність документації	1	1	1	1
Інструменти генерації коду програмних модулів	0	0,5	0,5	1
Легкість роботи із базами даних	0,5	1	1	1
Різноманітність наявних фреймворків для розробки веб сайтів	0,5	1	1	1
Підсумковий результат	3	5,5	5,5	6

Згідно таблиці 3.1, можна зробити висновок, що мова програмування C# найкраще підходить серед усіх розглянутих мов, так як задовольняє усі потреби, які можуть виникнути в процесі розробки онлайн платформи для організації навчальних курсів. Її фреймворк ASP.NET дозволить максимально спростити проектування та розробку програмного продукту.

3.2 Порівняльний аналіз середовищ розробки

Ще одним важливим етапом, який потрібно виконати перед початком розробки програмного компоненту онлайн платформи для продажів навчальних курсів є вибір інтегрованого середовища розробки (далі IDE). IDE – це платформа для керування програмним кодом проектів, що використовується під час їх написання. Подібні середовища містять інструменти для редагування, автоматизації, збирання та налагодження коду. Тому далі розглянемо найбільш популярні IDE, які використовуються для реалізації програмних продуктів мовою програмування C#, а саме Visual Studio та JetBrains Rider.

Visual Studio – це середовище розроблене компанією Microsoft. Воно слугує для розробки комп'ютерних програм, веб-сервісів та мобільних додатків. Visual Studio підтримує роботу з Windows API, Windows Forms, Windows Store та Microsoft Silverlight. Також дана IDE підтримує безліч плагінів, які розширюють функціональність [26]. Наразі підтримувана версія Visual Studio – 2022, інтерфейс якої зображено на рисунку 3.1.

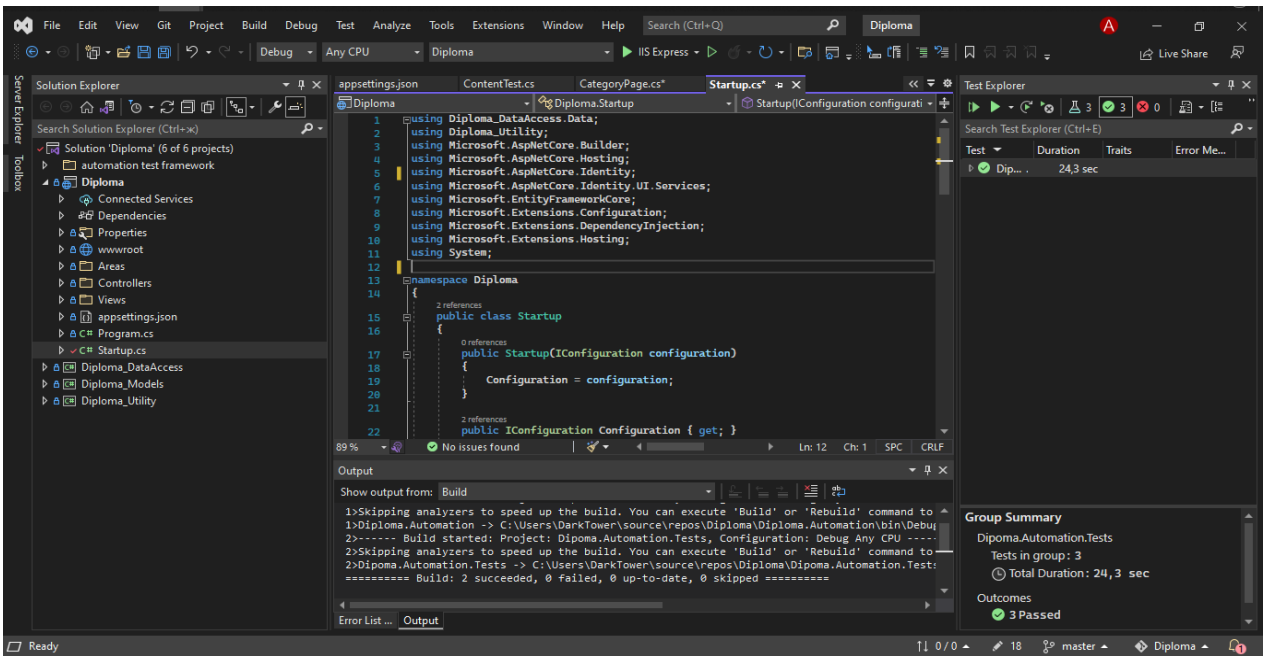

Рисунок 3.1 – Інтерфейс Visual Studio – 2022

Основною перевагою Visual Studio є те, що оскільки її розробляє компанія Microsoft, у ній підтримуються усі запропоновані компанією функції.

Середовище має доступний зручний менеджер пакетів NuGet. Це дозволяє легко керувати залежностями в проекті через консольний або графічний інтерфейси. Visual Studio має широкий функціонал для автоматичного тестування додатків. Також користувачеві представлені зручні способи компіляції і збирання проектів.

Окремо слід виділити інструменти для забезпечення роботи фреймворку ASP.NET. З допомогою них користувачеві доступна функція генерації коду програмних компонентів, що значно спрощує розробку веб-додатку.

Розглянемо також ще одну потужну IDE від всесвітньо відомої компанії JetBrains – Rider [27]. Це IDE за виключенням деяких винятків також дозволяє відкривати, редагувати, створювати, запускати та налагоджувати більшість видів програм на C# включаючи настільні програми, веб-програми, бібліотеки та служби. Інтерфейс робочої області даної програми зображено на рисунку 3.2.

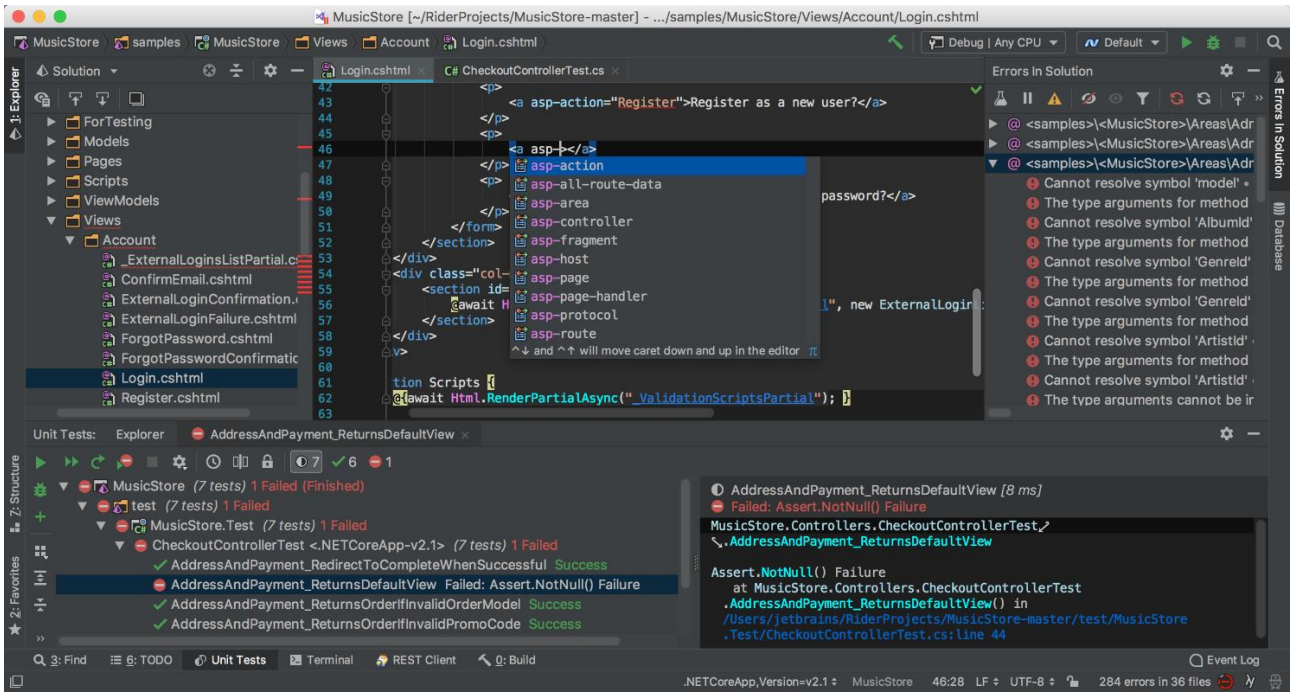

Рисунок 3.2 – Інтерфейс JetBrains Rider

Хоча дана IDE намагається підтримувати увесь необхідний функціонал для роботи з C#, вона не може в повній мірі конкурувати з продуктом Visual Studio, що уже є стандартом на ринку.

Результати порівняння розглянутих інтегрованих середовищ розробки за обраними критеріями наведено в таблиці 3.2.

Таблиця 3.2 – Порівняння інтегрованих середовищ розробки

Критерій	Rider	Visual Studio
Безкоштовність	0	0,5
Універсальність	0	1
Кросплатформність	1	1
Швидкість роботи	0,5	1
Невимогливість до продуктивності системи	0,5	0,5
Магазин розширень	0,5	1
Автодоповнення коду	1	1
Підсумковий результат	3,5	6

За результатами порівняння популярних інтегрованих середовищ розробки для мови програмування C#, наведеного у таблиці 3.2, можна зробити висновок, що серед усіх розглянутих аналогів Visual Studio найкраще підходить для реалізації онлайн платформи для організації навчальних курсів.

3.3 Порівняльний аналіз СУБД

Значна частина функціоналу розроблюваної онлайн платформи пов’язана з роботою з базою даних. Хоча в даній роботі ми будемо використовувати Entity фреймворк – програмне рішення, що суттєво спростить роботу з базою даних, вибір СУБД залишається відкритим питанням [9].

Реляційною називається база даних, у якій усі дані, доступні користувачу, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями [28]. Entity фреймворк підтримує роботу з багатьма реляційними СУБД, наприклад: SQLite, MySQL, PostgreSQL, Microsoft SQL Server. Щоб вибрати найкращий варіант розглянемо переваги та недоліки кожного з представлених варіантів.

SQLite – легка добре оптимізована СУБД, що дозволяє порівняно просто інтегрувати базу даних з додатком. Вона представлена набором файлів до яких при роботі безпосередньо звертається користувач. Це є альтернативним принципом, ніж користувач звертався би до даних через порти та гнізда, як це відбувається при використанні більш професійних мережевих СУБД. Після встановлення користувачу одразу доступний широкий набір інструментів для керування потоками і даними.

MySQL – це ще одна популярна система управління базами даних. Її перевагою є відкритий код програмного продукту. Дану СУБД можна вважати альтернативою корпоративним високопродуктивним аналогам. Користувачам, що її використовують є доступна велика кількість програмних графічних інтерфейсів. Вона володіє величезними функціональними можливостями та підходить для вирішення різноманітних задач.

PostgreSQL – напевно найбільш функціональна СУБД, що розповсюджується з відкритим кодом та без необхідності купівлі ліцензії. Вона позиціонується як система, що максимально відповідає стандартам SQL. Ключовою відмінністю даної системи від її подібних є те, що вона може підтримувати як об’єктно-орієнтований так і реляційний підходи до баз даних. PostgreSQL включає в себе багато сучасних технологічних рішень, що дозволяє позиціонувати дану СУБД як одну з найпродуктивніших.

Microsoft SQL Server – це корпоративна СУБД розроблена компанією Microsoft. Виступаючи сервером, представлений програмний продукт забезпечує зберігання, отримання та надсилання даних за запитом інших програмних додатків, що можуть працювати як на тому ж комп’ютері, так і на інших комп’ютерах в мережі. З усіх представлених СУБД ця єдина розроблена і підтримується компанією Microsoft, а отже найкраще і найпростіше інтегрується з розробленими додатками на мові C#. Також з даною СУБД легко працювати використовуючи ОРМ систему Entity фреймворк. За функціоналом Microsoft SQL Server є лідером на ринку по впровадженню різноманітних інновацій.

Результати порівняння розглянутих систем управління базами даних за обраними критеріями наведено в таблиці 3.3.

Таблиця 3.3 – Порівняння систем управління базами даних

Критерій	PostgreSQL	MySQL	SQLite	Microsoft SQL Server
Наявність функціоналу	1	0,5	0,5	1
Широка інтеграція з продуктами Microsoft	0	0	0,5	1
Швидкість читання даних	0	0,5	1	1
Легкість використання	0	1	1	1
Підсумковий результат	1	2	3	4

Аналізуючи таблицю 3.3, приходимо до висновку, що Microsoft SQL Server найкраще підходить серед усіх розглянутих СУБД, так як має широкий функціонал, досить просто інтегрується з продуктами Microsoft, має досить високу швидкість обробки даних та є легка у використанні, може інтегруватися з сервісом Firebase для Android.

3.4 Програмна реалізація основних компонентів онлайн платформи

Розроблений програмний код платформи для організації освітніх курсів є комплексним поєднанням багатьох технологій. Список та відсоток використання мов програмування, застосованих у проекті зображений на рисунку 3.3.

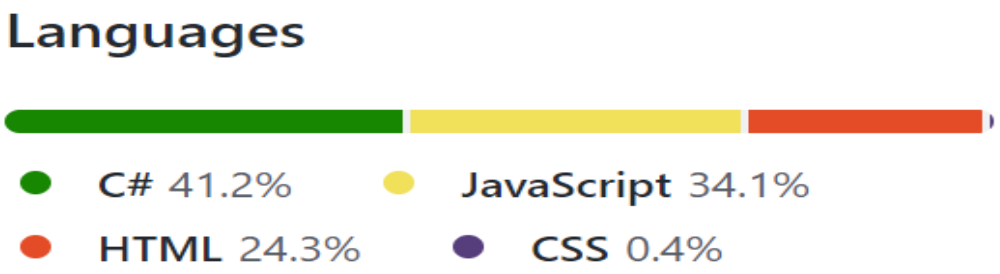

Рисунок 3.3 – Інтерфейс JetBrains Rider

Оскільки основою проекту є фреймворк ASP.NET MVC структура програмного коду додатку повністю відповідає накладеним фреймворком обмеженням та його логіці. Старт програми класично для додатків написаних на C# починається з функції Main() де відбувається збирання та запуск IHostBuilder. У функції запускається окремий процес, що надалі буде керувати веб-додатком. Для IHostBuilder використовуються налаштування за замовчуванням та деякі перевизначені властивості у класі Startup.cs. Лістинг описаного функціоналу зображено на рисунку 3.4.

```
public static void Main(string[] args)
{
    CreateHostBuilder(args).Build().Run();
}

1 reference
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
```

Рисунок 3.4 – Лістинг функції Main() та створення IHostBuilder

Наступним розглянемо клас Startup.cs. Даний клас містить конструктор де відбувається ініціалізація IConfiguration та два методи ConfigureServices() та Configure(). Ці методи відіграють важливу роль у забезпеченні роботи додатку.

У методі ConfigureServices() відбувається включення необхідних сервісів:

– AddDbContext – сервіс для управління контекстом з допомогою якого відбуваються з’єднання бази даних з проектом через Entity фреймворк;

– UseSqlServer – сервіс, що з’єднує проект з базою даних;

– AddIdentity – сервіс ASP.NET для налаштування ідентифікації користувача на сайті;

– AddDefaultTokenProviders – сервіс ASP.NET для налаштування ідентифікації на сайті, допомагає уніфікувати користувача;

– AddDefaultUI – сервіс ASP.NET, що додає та керує сторінками ідентифікації в проекті;

– AddEntityFrameworkStores – сервіс ASP.NET для налаштування бази даних для роботи з ідентифікацією користувачів в проекті;

– AddTransient – сервіс для налаштування керування поштовими сервісами в проекті;

– AddDistributedMemoryCache – сервіс для керування кеш даними додатку;

– AddHttpContextAccessor – сервіс для налаштування роботи з протоколом HTTP в додатку;

– AddSession – сервіс для керування сесіями в додатку;

– AddControllersWithViews – сервіс для налаштування роботи з патерном MVC в додатку.

Лістинг методу ConfigureServices() зображено на рисунку 3.5.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(
         Configuration.GetConnectionString("DefaultConnection")));
    services.AddIdentity<IdentityUser, IdentityRole>()
        .AddDefaultTokenProviders().AddDefaultUI()
        .AddEntityFrameworkStores<ApplicationDbContext>();
    services.AddTransient<IEmailSender, EmailSender>();
    services.AddDistributedMemoryCache();
    services.AddHttpContextAccessor();
    services.AddSession(Options =>
    {
        Options.IdleTimeout = TimeSpan.FromMinutes(10);
        Options.Cookie.HttpOnly = true;
        Options.Cookie.IsEssential = true;
    });
    services.AddControllersWithViews();
}
```

Рисунок 3.5 – Лістинг методу ConfigureServices()

В свою чергу Метод Configure() використовується для визначення того, як програма відповідає на запити HTTP. Конвеєр запитів налаштовується шляхом додавання компонентів проміжного програмного забезпечення до екземпляра IApplicationBuilder. В проекті додані наступні компоненти:

– UseDeveloperExceptionPage – компонент, що відповідає за захоплення синхронних та асинхронних екземплярів винятків із конвеєра та генерування відповіді на ці помилки;

– UseExceptionHandler – використовується щоб налаштувати спеціальну сторінку обробки помилок;

– UseHsts – додає компонент для роботи з HSTS протоколом безпеки транзакцій;

– UseHttpsRedirection – додає компонент для керування HTTP-запитами;

– UseStaticFiles – вмикає статичне обслуговування файлів для поточного шляху запиту;

– UseRouting – підтримує механізм маршруту до конвеєра;

– UseAuthentication – додає компонент для аунтифікації;

– UseAuthorization – додає компонент для авторизації;

– UseSession – додає компонент для роботи з сесіями;

– UseEndpoints – компонент для визначення кінцевих точок додатку;

Лістинг методу Configure() зображено на рисунку 3.6.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment()){
        app.UseDeveloperExceptionPage();
    } else {
        app.UseExceptionHandler("/Home/Error");
        app.UseHsts();
    }
    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseRouting();
    app.UseAuthentication();
    app.UseAuthorization();
    app.UseSession();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Рисунок 3.6 – Лістинг методу Configure()

Розглянемо реалізацію MVC патерну на прикладі окремих компонентів платформи для організації навчальних курсів. Центральним її компонентом можна вважати продукт, чим по суті є навчальний курс, що пропонується користувачу. Лістинг моделі продукту зображено на рисунку 3.7.

```
public class Product
{
    [Key]
    - references
    public int Id { get; set; }
    [Required]
    - references
    public string Name { get; set; }
    - references
    public string ShortDesc { get; set; }
    - references
    public string Description { get; set; }
    [Range(1, int.MaxValue)]
    - references
    public double Price { get; set; }
    - references
    public string Image { get; set; }
    [Display(Name = "Category Type")]
    - references
    public int CategoryId { get; set; }
    [ForeignKey("CategoryId")]
    - references
    public virtual Category Category { get; set; }
}
```

Рисунок 3.7 – Лістинг моделі продукту

Головна логіка обробки моделі описана в ProductController. Це клас унаслідуваний від базового класу MVC Controller. У ньому описані HTTP запити для взаємодії. Клас має конструктор де ініціалізуються контекст бази даних та хост веб середовища. На рисунку 3.8 зображено HTTP get запит Index().

```
public IActionResult Index()
{
    IEnumerable<Product> objList = _db.Product.Include(u => u.Category);
    return View(objList);
}
```

Рисунок 3.8 – Лістинг HTTP get запиту Product.Index()

Даний запит повертає інформацію про наявні продукти з бази даних та, звернувшись до представлення, відображає ці продукти на сайті.

Розглянемо представлення у середовищі ASP.NET MVS. Синтаксис Razor дозволяє розширити функціонал звичайного HTML та CSS шляхом інтегрування в код HTML коду C#. А використання бібліотеки Bootstrap дозволяє легко стилізувати сторінку. Лістинг представлення списку наявних продуктів зображено на рисунку 3.9.

```
@model IEnumerable<Diploma_Models.Product>
<div class="container p-3">
    <div class="row pt-4">
        <div class="col-6">
            <h2 class="text-primary">Product List</h2>
        </div>
        <div class="col-6 text-right">
            <a asp-action="Upsert" class="btn btn-primary">
                <i class="fas fa-plus"></i>   Create New Product
            </a>
        </div>
    </div>
    <br /><br />
    @if (Model.Count() > 0)
    {
        <table class="table table-bordered table-striped" style="width:100%">
            <thead>
                <tr>
                    <th>
                        Product Name
                    </th>
                    <th>
                        Price
                    </th>
                    <th>
                        Category
                    </th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var obj in Model)
                {
                <tr>
                    <td width="30%">@obj.Name</td>
                    <td width="10%">@obj.Price</td>
                    <td width="20%">@obj.Category.Name</td>
                    <td class="text-center">
                        <div class="w-75 btn-group" role="group">
                            <a asp-route-Id="@obj.Id" asp-action="Upsert" class="btn
btn-primary mx-2">
                                <i class="fas fa-edit"></i>
                            </a>
                            <a asp-route-Id="@obj.Id" asp-action="Delete" class="btn
btn-danger mx-2">
                                <i class="far fa-trash-alt"></i>
                            </a>
                        </div>
                    </td>
                </tr>
                }
            </tbody>
        </table>
    }
    else
    {
        <p> No product exists.</p>
    }
    </div>
```

Рисунок 3.9 – Лістинг представлення Product.Index()

Розглянемо інші запити, що знаходяться в ProductController. Запит HTTP GET – UPSERT зображено на рисунку 3.10.

```
//GET - UPSERT
0 references
public IActionResult Upsert(int? id)
{

    ProductVM productVM = new ProductVM()
    {
        Product = new Product(),
        CategorySelectList = _db.Category.Select(i => new SelectListItem
        {
            Text = i.Name,
            Value = i.Id.ToString()
        })
    };

    if (id == null)
    {
        //this is for create
        return View(productVM);
    }
    else
    {
        productVM.Product = _db.Product.Find(id);
        if (productVM.Product == null)
        {
            return NotFound();
        }
        return View(productVM);
    }
}
```

Рисунок 3.10 – Лістинг HTTP запиту Product GET UPSERT

Даний метод слугує одночасно для створення та редагування продуктів. Тобто якщо задане Id продукту не передане користувачу повертається представлення для створення нового продукту, а якщо Id наявне відбувається пошук товару у базі даних та повернення користувачу представлення для редагування.

Метод запиту Product POST UPSERT дещо складніший. Розглянемо частину цього запиту, яка відповідає за створення нового продукту. Його важливим елементом є обробка медіа файлу продукту. Лістинг Product POST UPSERT зображений на рисунку 3.11.

```
//POST - UPSERT
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public IActionResult Upsert(ProductVM productVM)
{
    if (ModelState.IsValid)
    {
        var files = HttpContext.Request.Form.Files;
        string webRootPath = _webHostEnvironment.WebRootPath;

        if (productVM.Product.Id == 0)
        {
            //Creating
            string upload = webRootPath + WC.ImagePath;
            string fileName = Guid.NewGuid().ToString();
            string extension = Path.GetExtension(files[0].FileName);

            using (var fileStream = new FileStream(Path.Combine(upload, fileName + extension), FileMode.Create))
            {
                files[0].CopyTo(fileStream);
            }

            productVM.Product.Image = fileName + extension;

            _db.Product.Add(productVM.Product);
        }
```

Рисунок 3.11 – Лістинг HTTP запиту Product POST UPSERT

Інші запити, що наявні в ProductController а також компоненти MVC, такі як: Category, Recommendations, Cart, Home будуть представленні в додатках даної роботи.

Отже, в даному підрозділі було розглянуто програмну реалізацію основних компонентів платформи для організації навчальних курсів. Повний лістинг наведено у додатку В.

3.5 Програмна реалізація алгоритмів роботи онлайн платформи для організації навчальних курсів

Під час написання програмної компоненти було розроблено велику кількість алгоритмів основними серед них є наступні:

– алгоритм підбору навчальних курсів для окремого користувача;

– алгоритму генерування ордеру на покупку.

Розглянемо ключові елементи програмної реалізації цих алгоритмів. Працюючи з елементами меню у шапці профілю, ми взаємодіємо з частковим представленням _Layout. Це представлення являє собою Razor сторінку умовно розділену на три частини: шапку сайту, шаблон для тіла сторінки, та нижній

заголовок сайту. Також у даному представлені підключаються CSS та JS компоненти до проекту. Лістинг меню сайту зображено на рисунку 3.12. Маємо наступні розділи: Home, Recommendations, Cart, Login, Registration чи Logout, а також елемент Content Management, що доступний лише адміністратору сайту.

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
</li>
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="Recommendations" asp-action="Index">Recommendations</a>
</li>
@if (User.IsInRole(WC.AdminRole))
{
<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        Content Management
    </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="nav-link text-dark" asp-area="" asp-controller="Category" asp-action="Index">Category</a>
        <a class="nav-link text-dark" asp-area="" asp-controller="Product" asp-action="Index">Product</a>
        <div class="dropdown-divider"></div>
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Create Admin User</a>
    </div>
</li>
```

Рисунок 3.12 – Лістинг меню сайту

Дослідимо поведінку програмного додатку після натискання кнопки Recommendations. Оскільки платформа пропонує рекомендовані курси користувачам на основі інформації, що користувач заповнює при реєстрації, алгоритм перевіряє чи авторизований користувач, і основуючись на результаті даної перевірки користувачу пропонується авторизуватися або відображаються рекомендовані курси. Забезпечує даний функціонал анртація Authorize бібліотеки Microsoft.AspNetCore.Authorization, що прописується над контролером (Рисунок 3.13).

```
[Authorize]
3 references
public class RecommendationsController: Controller
```

Рисунок 3.13 – Використання анотації Authorize

Наступним кроком, коли користувач авторизований, у HTTP GET запиті Index() через контекст бази даних отримуємо інформацію про поточного

користувача і з поміж усіх доступних курсів, використовуючи фільтри, відображаємо ті курси, що найбільше підходять користувачу. Описаний алгоритм зображено на рисунку 3.14.

```
2 references
public IActionResult Index()
{
    ApplicationUser CurrentApplicationUser = (ApplicationUser)_db.Users
        .Where(n => n.UserName == User.Identity.Name).FirstOrDefault();

    int rank = 0;
    if (CurrentApplicationUser.Level == "Junior")
    {
        rank = 300;
    }
    else if (CurrentApplicationUser.Level == "Middle")
    {
        rank = 700;
    }
    else
    {
        rank = int.MaxValue;
    }
    RecommendationsVM recommendationsVM = new RecommendationsVM()
    {
        Products = _db.Product.Where(p => p.Price <= rank)
        .Where(p => p.Category.Name == CurrentApplicationUser.Specialization).Include(u => u.Category),
        Categories = _db.Category
        .Where(c => c.Name == CurrentApplicationUser.Specialization)
    };
    return View(recommendationsVM);
}
```

Рисунок 3.14 – HTTP GET запиті Index() сторінки рекомендацій

Важливою частиною алгоритму генерування ордеру на покупку є робота корзини покупок на сайті. Як зазначено в алгоритмі після додання товару до корзини створюється сесія, що кешує дії користувача. Сесії в ASP.NET MVC це окремий сервіс, що підключений в класі Startup. Сесія в даному проекті має наступні налаштування (Рисунок 3.15).

```
services.AddSession(Options =>
{
    Options.IdleTimeout = TimeSpan.FromMinutes(10);
    Options.Cookie.HttpOnly = true;
    Options.Cookie.IsEssential = true;
});
```

Рисунок 3.15 – Налаштування сесії на платформі

Розглянемо HTTP запити, що забезпечують функціонування корзини на сайті. HTTP GET запиті Index() відповідає за відображення доданих до корзини курсів (Рисунок 3.16).

```
public IActionResult Index()
{

    List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
    if (HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
        && HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
    {
        //session exsits
        shoppingCartList = HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
    }

    List<int> prodInCart = shoppingCartList.Select(i => i.ProductId).ToList();
    IEnumerable<Product> prodList = _db.Product.Where(u => prodInCart.Contains(u.Id));

    return View(prodList);
}
```

Рисунок 3.16 – HTTP GET запиті Index() корзини покупок

Користувачу доступний функціонал видалення товарів з корзини. За це відповідає запит Remove (Рисунок 3.17).

```
public IActionResult Remove(int id)
{
    List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
    if (HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
        && HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
    {
        //session exsits
        shoppingCartList = HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
    }

    shoppingCartList.Remove(shoppingCartList.FirstOrDefault(u => u.ProductId == id));
    HttpContext.Session.Set(WC.SessionCart, shoppingCartList);
    return RedirectToAction(nameof(Index));
}
```

Рисунок 3.17 – HTTP запиті Remove() корзини покупок

Якщо користувач все ж хоче купити товар, йому доступна функція генерування ордеру на покупку. За даний фунціонал відповідають GET і POST запити Summary() (Рисунки 3.18, 3.19).

```
public IActionResult Summary()
{
    var claimsIdentity = (ClaimsIdentity)User.Identity;
    var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);
    //var userId = User.FindFirstValue(ClaimTypes.Name);

    List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
    if (HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
        && HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
    {
        //session exsits
        shoppingCartList = HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
    }

    List<int> prodInCart = shoppingCartList.Select(i => i.ProductId).ToList();
    IEnumerable<Product> prodList = _db.Product.Where(u => prodInCart.Contains(u.Id));

    ProductUserVM = new ProductUserVM()
    {
        ApplicationUser = _db.ApplicationUser.FirstOrDefault(u => u.Id == claim.Value),
        ProductList = prodList.ToList()
    };

    return View(ProductUserVM);
}
```

Рисунок 3.18 – HTTP GET запиті Summary() корзини покупок

```
[HttpPost]
[ValidateAntiForgeryToken]
[ActionName("Summary")]
0 references
public async Task<IActionResult> SummaryPost(ProductUserVM ProductUserVM)
{
    var PathToTemplate = _webHostEnvironment.WebRootPath + Path.DirectorySeparatorChar.ToString()
        + "templates" + Path.DirectorySeparatorChar.ToString() +
        "Inquiry.html";

    var subject = "New Inquiry";
    string HtmlBody = "";
    using (StreamReader sr = System.IO.File.OpenText(PathToTemplate))
    {
        HtmlBody = sr.ReadToEnd();
    }

    StringBuilder productListSB = new StringBuilder();
    foreach (var prod in ProductUserVM.ProductList)
    {
        productListSB.Append($" - Name: { prod.Name} <span style='font-size:14px;'> (ID: {prod.Id})</span><br />");
    }

    string messageBody = string.Format(HtmlBody,
        ProductUserVM.ApplicationUser.FullName,
        ProductUserVM.ApplicationUser.Email,
        ProductUserVM.ApplicationUser.PhoneNumber,
        productListSB.ToString());

    await _emailSender.SendEmailAsync(WC.EmailAdmin, subject, messageBody);
    return RedirectToAction(nameof(InquiryConfirmation));
}
```

Рисунок 3.19 – HTTP POST запиті Summary() корзини покупок

Запит Summary() GET відображає користувачу представлення, де можна редагувати інформацію про користувача, що робить покупку а також список товарів, що потраплять до ордеру.

Запит Summary() POST дещо складніший. Його задачею є сформувати ордер на покупку та відправити його адміністратору сайту. Наявна інформація про покупця а також список товарів використовуються для заповнення шаблону повідомлень. Далі використовуючи сервіс для відправки повідомлень заповнений шаблон надсилається адміністратору у вигляді HTML листа.

Розглянемо код поштового сервісу (Рисунок 3.20).

```
2 references
public class EmailSender : IEmailSender
{
    private readonly IConfiguration _configuration;
    1 reference
    private GMailSetings _GMailSetings { get; set; }

    0 references
    public EmailSender(IConfiguration configuration)
    {
        _configuration = configuration;
    }
    0 references
    public Task SendEmailAsync(string email, string subject, string htmlMessage)
    {
        return Execute(email, subject, htmlMessage);
    }
    1 reference
    public async Task Execute(string email, string subject, string htmlMessage)
    {
        _GMailSetings = _configuration.GetSection("GMailSetings").Get<GMailSetings>();
        using (SmtpClient client = new SmtpClient("smtp.gmail.com", 587))
        {
            client.EnableSsl = true;
            client.DeliveryMethod = SmtpDeliveryMethod.Network;
            client.UseDefaultCredentials = false;
            client.Credentials = new NetworkCredential("artemverenkotest@gmail.com", "ungvzrlxsohseilf");
            MailMessage msgObj = new MailMessage();
            msgObj.To.Add(email);
            msgObj.From = new MailAddress("artemverenkotest@gmail.com");
            msgObj.Subject = subject;
            msgObj.Body = htmlMessage;
            msgObj.IsBodyHtml = true;
            client.Send(msgObj);
        }
    }
}
```

Рисунок 3.20 – Код поштового сервісу

Це клас, що наслідує інтерфейс IEmailSender та реалізує метод даного інтерфейсу EmailSender(). Даний метод створює асинхронні запити на відправку

повідомлення. Для передачі повідомлень використовується інтернет протокол SMTP.

Отже, в даному підрозділі було розглянуто програмний код елементів основних алгоритмів платформи для організації навчальних курсів. Повний лістинг наведено у додатку В.

3.6 Розробка Android платформи для навчальних курсів

Android додаток це окремий проект, в якому користувач може отримати доступ до купленого на сайті курсу. Він розроблений на мові програмування Java. Розглядаючи структуру програмного додатку можна виділити декілька компонентів: компонент авторизації та компонент курсу. Розглянемо програмну реалізацію кожного компоненту окремо.

За можливість авторизації користувача в додатку відповідає клас Login.java. Він є контролером, що керує представленням activity_login.xml. В самому представлені наявні поля для вводу логіну і паролю користувача, прогрес бар та кнопка вхід. На рисунку 3.21 зображено лістинг коду обробки кнопки логін на представлені.

```
mLoginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = mEmail.getText().toString().trim();
                String password = mPassword.getText().toString().trim();
                progressBar.setVisibility(View.VISIBLE);
                // authenticate the user

fAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if(task.isSuccessful()){
                            Toast.makeText(Login.this, "Logged in Successfully",
Toast.LENGTH_SHORT).show();
                            startActivity(new
Intent(getApplicationContext(),MainActivity.class));
                        }else {
                            Toast.makeText(Login.this, "Error ! " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                            progressBar.setVisibility(View.GONE);
                        }
                    }
                });
            }
        });
```

Рисунок 3.21 – Лістинг коду обробки кнопки логін на представлені

Розглянемо компонент для розміщення навчального курсу. Він складається двох представлень. Перше представлення містить список уроків курсу. Натиснувши на один із уроків списку відкривається представлення вибраного уроку. Розглянемо компонент списку. Лістинг шаблонного класу даного представлення зображено на рисунку 3.22.

```
package com.darktower.myfavoriteseries;

public class RecyclerViewItem {
    private int imageResource;
    private String text1;
    private String text2;
    private String text3;
    public RecyclerViewItem(int imageResource, String text1, String text2,
String text3) {
        this.imageResource = imageResource;
        this.text1 = text1;
        this.text2 = text2;
        this.text3 = text3;
    }
    public int getImageResource() {
        return imageResource;
    }
    public String getText1() {
        return text1;
    }
    public String getText2() {
        return text2;
    }
    public String getText3() {
        return text3;
    }
}
```

Рисунок 3.22 – Лістинг шаблонного класу представлення зі списком уроків

Керування представленням зі списком уроків відбувається у класі RecyclerViewAdapter.java. На рисунку 3.23 зображено оголошення та контролер даного класу. Даний клас наслідує RecyclerView.Adapter. В контролері ініціалізується масив уроків платформи.

```
public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.RecyclerViewViewHolder> {
    private ArrayList<RecyclerViewItem> arrayList;
    Context context;

    public RecyclerViewAdapter(ArrayList<RecyclerViewItem> arrayList, Context
context){
        this.arrayList = arrayList;
        this.context = context;
    }
```

Рисунок 3.23 – Лістинг контролера класу RecyclerViewAdapter.java

На рисунку 3.24 бачимо перевизначені методи для керування списком.

```
@NonNull
    @Override
    public RecyclerViewViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.recycler_view_item,
parent, false);
        RecyclerViewViewHolder recyclerViewViewHolder = new
RecyclerViewViewHolder(view);
        return recyclerViewViewHolder;
    }
    @Override
    public void onBindViewHolder(@NonNull RecyclerViewViewHolder holder, int
position) {
        RecyclerViewItem recyclerViewItem = arrayList.get(position);
        holder.imageView.setImageResource(recyclerViewItem.getImageResource());
        holder.textView1.setText(recyclerViewItem.getText1());
        holder.textView2.setText(recyclerViewItem.getText2());
    }
    @Override
    public int getItemCount() {
        return arrayList.size();
    }
```

Рисунок 3.24 – Лістинг перевизначених методів класу RecyclerViewAdapter.java

Розглянемо клас RecyclerViewViewHolder.java яким безпосередньо керує клас RecyclerViewAdapter.java. Лістинг даного класу зображено на рисунку 3.25. Він також служить для передачі даних між представленням і контролером і об’єднує пункт списку безпосередньо з самим заняттям курсу.

```
public class RecyclerViewViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
        public ImageView imageView;
        public TextView textView1;
        public TextView textView2;
        public RecyclerViewViewHolder(@NonNull View itemView) {
            super(itemView);
            itemView.setOnClickListener(this);
            imageView = itemView.findViewById(R.id.imageView);
            textView1 = itemView.findViewById(R.id.textView1);
            textView2 = itemView.findViewById(R.id.textView2);
        }
        @Override
        public void onClick(View v) {
            int position = getAdapterPosition();
            RecyclerViewItem recyclerViewItem = arrayList.get(position);
            Intent intent = new Intent(context, MViewItemLayout.class);
            intent.putExtra("imageV", recyclerViewItem.getImageResource());
            intent.putExtra("titleTV", recyclerViewItem.getText1());
            intent.putExtra("textTV", recyclerViewItem.getText3());
            context.startActivity(intent);
        }
    }
```

Рисунок 3.25 – Лістинг класу RecyclerViewViewHolder.java

Останнім розглянемо контролер що відповідає за представлення окремого уроку. Лістинг даного контролера зображено на рисунку 3.26. Він отримує компонент, що переданий адаптером після вибору уроку в меню.

```
public class MViewItemLayout extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_m_view_item_layout);
        TextView title = findViewById(R.id.titleTV);
        TextView text = findViewById(R.id.textTV);
        ImageView imageView = findViewById(R.id.imageV);
        Intent intent = getIntent();
        if(intent != null){
            title.setText(intent.getStringExtra("titleTV"));
            text.setText(intent.getStringExtra("textTV"));
            imageView.setImageResource(intent.getIntExtra("imageV", 0));
        }
    }
```

Рисунок 3.26 – Лістинг контролера окремого уроку

Отже, в даному підрозділі розглянуто програмну реалізацію Android платформи для навчальних курсів. Повний лістинг наведено у додатку В.

3.7 Висновки

У даному розділі було проведено аналіз мов програмування C++, Python, Java та C#, середовищ розробки Visual Studio та Rider і систем управління базами даних SQLite, MySQL, PostgreSQL, Microsoft SQL Server. В результаті даного аналізу було прийнято рішення використовувати мову програмування C#, середовище розробки Visual Studio та СУБД Microsoft SQL Server для розробки онлайн платформи для організації навчальних курсів. Крім того, було наведено опис програмної реалізації алгоритмів роботи основних модулів онлайн платформи. Також описано програмний код Android платформи для навчальних курсів.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Процес тестування розробленої програмної компоненти – хід технічного аналізу, що служить для виявлення даних про якість програмного продукту відносно вимог, що надані замовником [29]. Існує доволі багато підходів до тестування. Розглянемо класифікацію тестування за знанням системи. До цієї класифікації відносять тестування «білої скриньки», тестування «чорної скриньки», тестування «сірої скриньки».

Про процес тестування відповідно методу «білої скриньки» говорять тоді, коли програміст хоче тестувати внутрішню будову програми тобто програмний код. За цим методом досліджуються внутрішні елементи програми і зв’язки між ними. При використанні даного методу, очевидними стають обмеження, що кількість незалежних програмних маршрутів, що потрібно тестувати може бути дуже великою. Тому даний вид не може гарантувати відповідність програми вимогам замовника. Перевагою донного методу є можливість врахувати особливості програмних помилок, та запобігти їх існуванню.

Процес тестування відповідно методу «чорної скриньки» – це ще один вид динамічного тестування, що є противагою методу «білої скриньки». В даному випадку тестувальник має безпосередній доступ до розробленого програмного продукту але нічого не знає про його код. Тому тестування відбувається з боку звичайного користувача, а тестуються відповідно кінцеві точки програми. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. При виконанні тестів за даним методом можна виявити дані, які призводять до аномалій у поведінці програми та результати, які демонструють її дефекти. Тестування «чорної скриньки» дозволяє провести аналіз на наявність помилок, як: некоректні чи відсутні функції, помилки в проектуванні архітектури даних, неправильно встановлені властивості, в повилковому встановлені та відновлювані значень. Даний принцип помилково вважати повноцінною заміною «білої скриньки». Краще використовувати їх паралельно.

Метод сірої скриньки є універсальним поєднанням попередніх двох згаданих підходів до тестування. Його застосовують коли компоненти програми дуже складні, наприклад, програмно змінюючи поведінку окремих сервісів програми чи жорстко встановлюючи проміжні результати, що пришвидшує тестування.

У результаті аналізу для тестування онлайн платформи для організації навчальних курсів було обрано метод «чорної скриньки», оскільки використання даного методу тестування дозволить виявити помилки в логікі функціонування програмного додатку та можливі неполадки у роботі його основних алгоритмів, чого не завжди можна досягти при використанні інших методів динамічного тестування.

4.2 Тестування розробленого програмного продукту

Тестування платформи для організації навчальних курсів за методикою «чорної скриньки» передбачає перевірку правильності функціонування додатку при виконанні основних алгоритмів його використання та порівняння фактичного результату виконання із очікуваним. Такі алгоритми використання програмного додатку називають тест-кейсами. Для тестування розробленого додатку було розроблено наступні тест-кейси:

Тест-кейс №1 – Увійти в систему як адміністратор:

1.Завантажити платформу;

2. Натиснути кнопку увійти на головному меню сайту;

3. Ввести існуючий логін адміністратора;

4. Ввести існуючий пароль адміністратора;

5. Натиснути кнопку увійти, що підтверджує вхід на сторінку;

6. Перевірити чи доступні на сайті інструменти адміністратора.

Очікуваним результатом даного тест-кейсу є успішна авторизація адміністратора на сайті. Це дає доступ до панелі керування контентом на сайті. Кроки та результат виконання тест-кейсу наведено на рисунках 4.1, 4.2, 4.3.

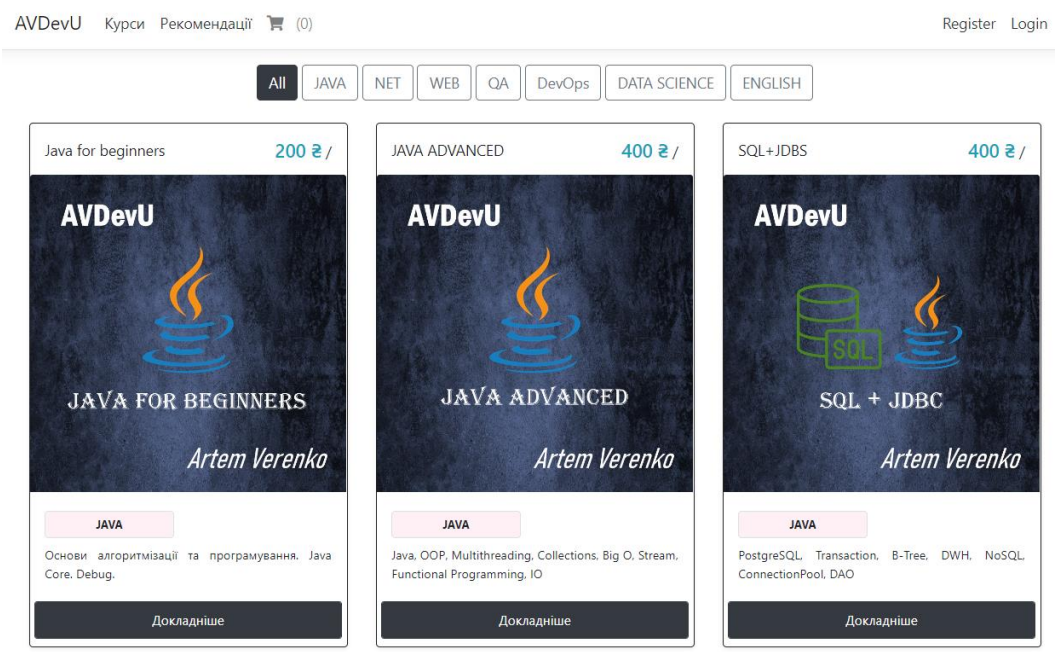

Рисунок 4.1 – Запуск платформи для організації курсів

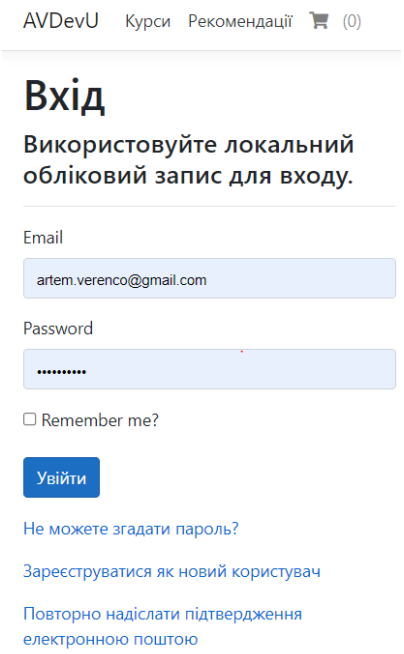

Рисунок 4.2 – Сторінка авторизиції

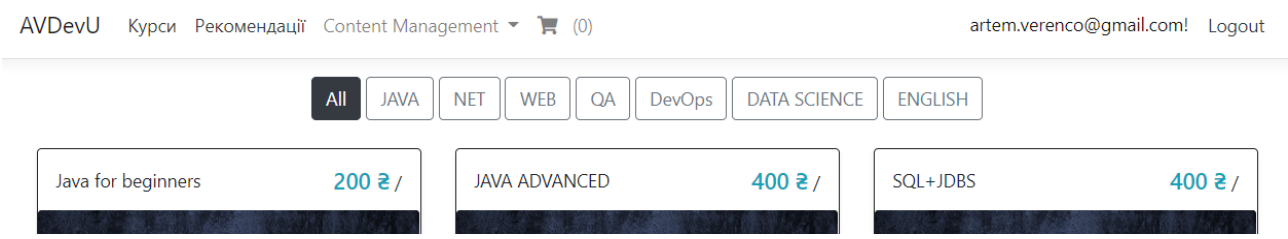

Рисунок 4.3 – Результат виконання тест-кейсу авторизації користувача

Тест-кейс №2 – Увійти в систему як звичайний користувач:

1. Завантажити платформу;
2. Натиснути кнопку вхід на головному меню сайту;
3. Ввести існуючий логін користувача сайту;
4. Ввести існуючий пароль користувача сайту;
5. Натиснути кнопку вхід, що підтверджує вхід на сторінку;
6. Перевірити чи вдалося авторизуватися.

Цей тест-кейс відрізняється від попереднього тим, що авторизація відбувається на акаунт звичайного користувача сайту. Результат тест-кейсу зображено на рисунку 4.4.

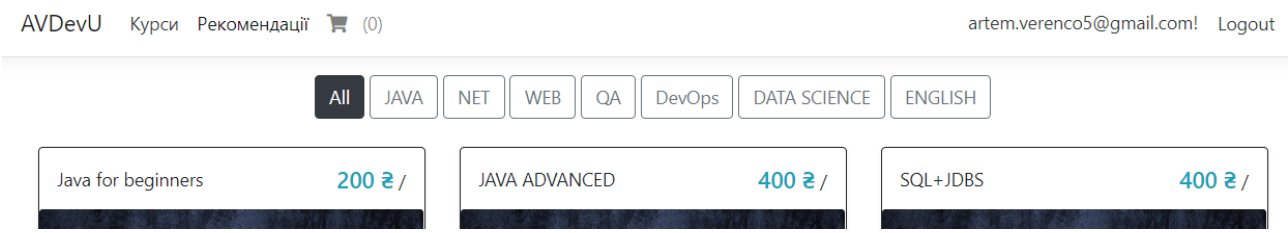

Рисунок 4.4 – Результат виконання тест-кейсу авторизації адміністратора

Тест-кейс №3 – Додати нову категорію товарів:

1. Виконати дії тест-кейсу №1 (ввійти в систему як адміністратор);
2. Натиснути кнопку Category, що доступна на головному меню сайту;
3. Натиснути на кнопку Create New Category;
4. Ввести ім'я нової категорії;
5. Ввести порядковий номер відображення категорії;
6. Натиснути на кнопку Create.
7. Перевірити чи створена категорія

Даний тест кейс перевіряє можливість додавання контенту на сайт з допомогою панелі адміністратора. В даному випадку спробуємо додати нову категорію товарів. Натиснувши на пункт меню Category ми перейдемо на сторінку списку вже існуючих категорій. Тут можна переглянути категорії, редагувати або видаляти їх, а також створювати нові. Натиснувши на кнопку

Create New Category ми переходимо на сторінку створення категорії. Тут потрібно ввести назву категорії і порядковий номер за яким вона буде відображатися в списку. Після заповнення даних натискаємо на кнопку Create. Останнім кроком потрібно перевірити, чи категорія успішно створилася. На рисунку 3.5 бачимо список доступних категорій на сайті. На рисунку 3.6 відображена сторінка для створення нової категорії. На рисунку 3.8 ми бачимо нову категорію на сайті, що є підтвердженням успішного виконання тест-кейсу.

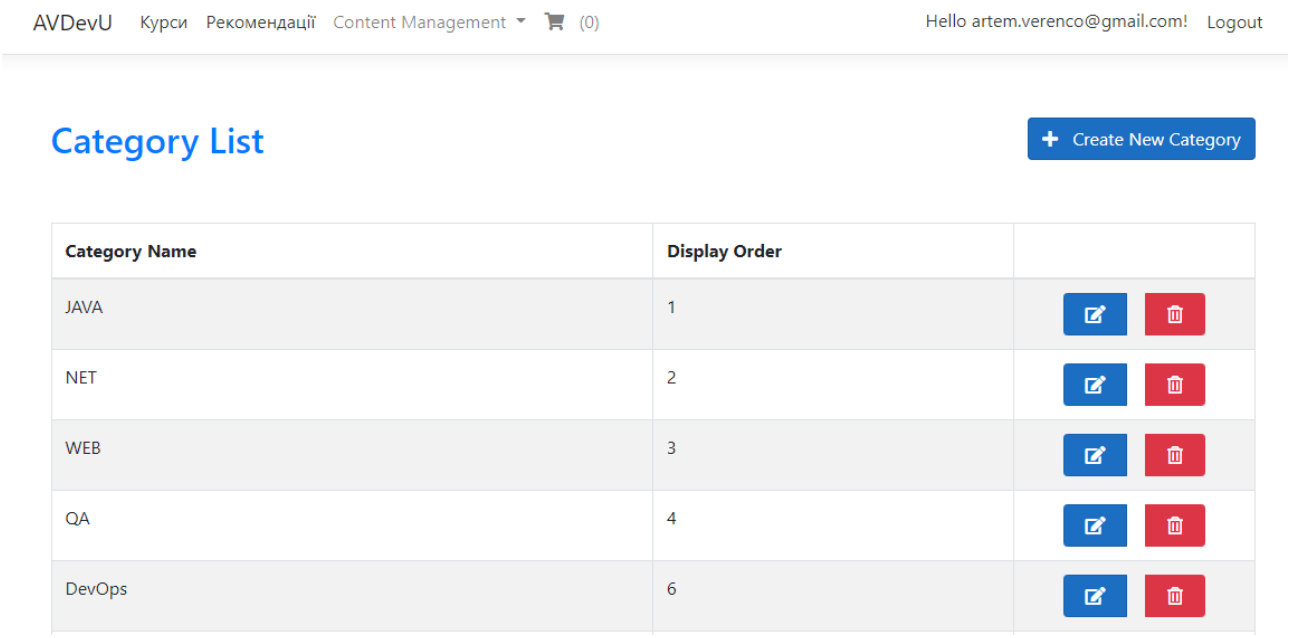

Рисунок 4.5 – Список наявних категорій на сайті

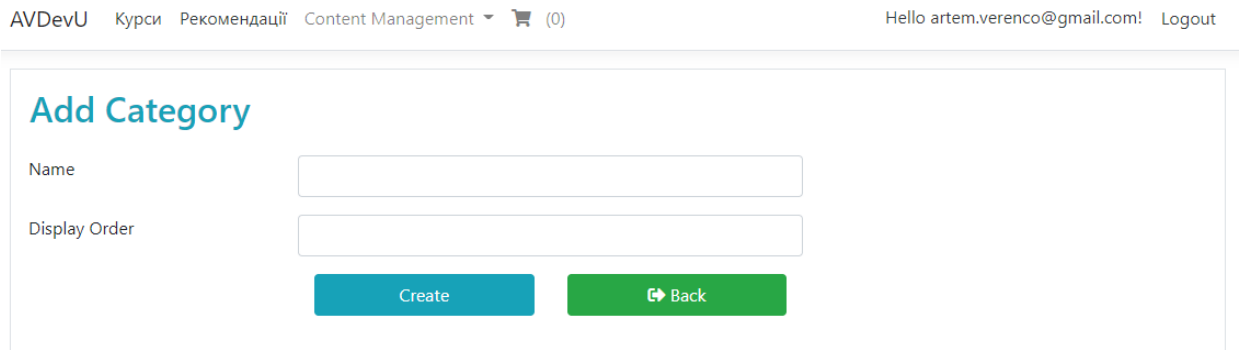

Рисунок 4.6 – Сторінка створення нової категорії

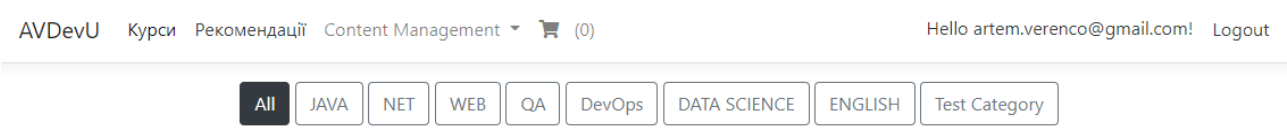

Рисунок 4.7 – наявність нової категорії на головній сторінці.

Тест-кейс №4 – Формування ордеру на покупку:

1. Виконати дії тест-кейсу №2;
2. Обрати товар на сторінці та натиснути на кнопку Докладніше;
3. Натиснути на кнопку Додати в кошик;
4. Натиснути на іконку корзини, що розміщена на головному меню сайту;
5. Натиснути на кнопку Продовжити, щоб сформувати ордер.
6. Натиснути на кнопку Подати запит.

На рисунку 4.8 зображено корзину покупок, до якої додано один товар.

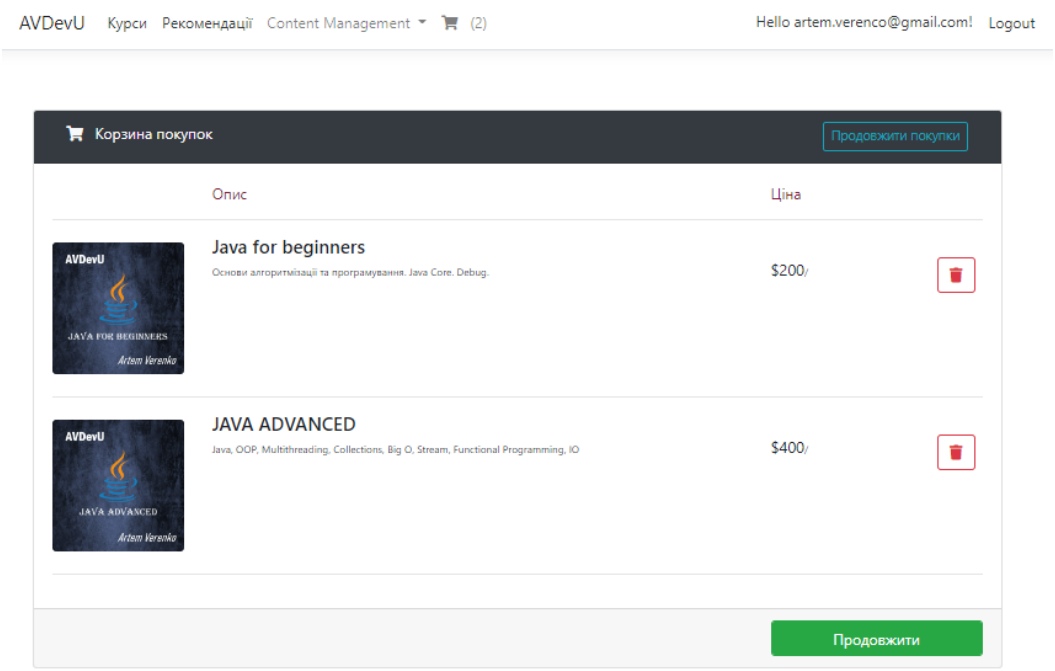

Рисунок 4.8 – Корзина покупок з доданими курсами

Натиснувши на кнопку Продовжити платформа підтягне додані до корзини товари та інформацію про поточного користувача з чого буде сформований ордер на покупку. На даному етапі користувачу дозволено вносити зміни в замовлення. Тобто можна вручну змінити ім’я замовника, його номер телефону

чи поштову адресу. Коли ордер готовий, користувач натискає на кнопку Подати запит після чого ордер надсилається на електронну пошту адміністратора сайту. Сторінка редагування ордеру на покупку зображена на рисунку 4.9. На рисунку 4.10 зображено інформаційне вікно про успішне надсилання ордеру.

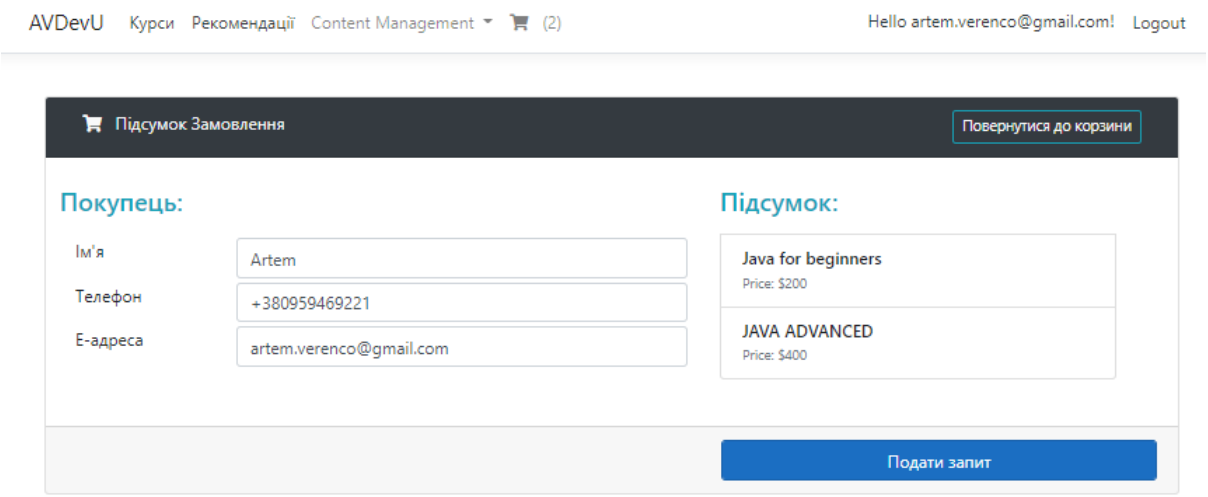

Рисунок 4.9 – Сторінка редагування ордеру на покупку

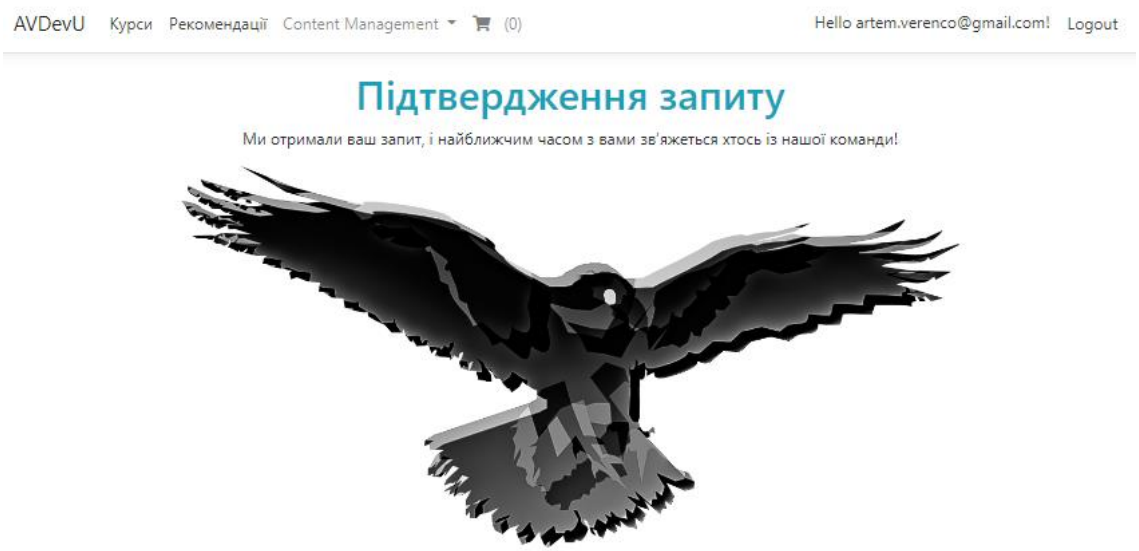

Рисунок 4.10 – Інформаційна сторінка про успішне надсилання ордеру

Фактичний результат виконання тест-кейсів повністю відповідає очікуваному, отже тестування пройдено успішно.

Отже, при проведенні тестування програмного додатку було отримано повну відповідність фактичних результатів очікуваним, доведено повну працездатність додатку та його відповідність технічному завданню.

4.3 Розробка інструкції користувача

Розроблена платформа для організації освітніх курсів передбачає користування нею за трьома сценаріями: сценарій для адміністратора сайту, сценарій для зареєстрованого користувача сайту, сценарій для не зареєстрованого користувача. Розглянемо функціональні можливості кожного з згаданих сценаріїв.

Сценарій для не зареєстрованого користувача надає можливість переглядати наявні курси на сайті (Рисунок 4.11), відкривати детальну інформацію про товар (Рисунок 4.12), додавати товар до корзини.

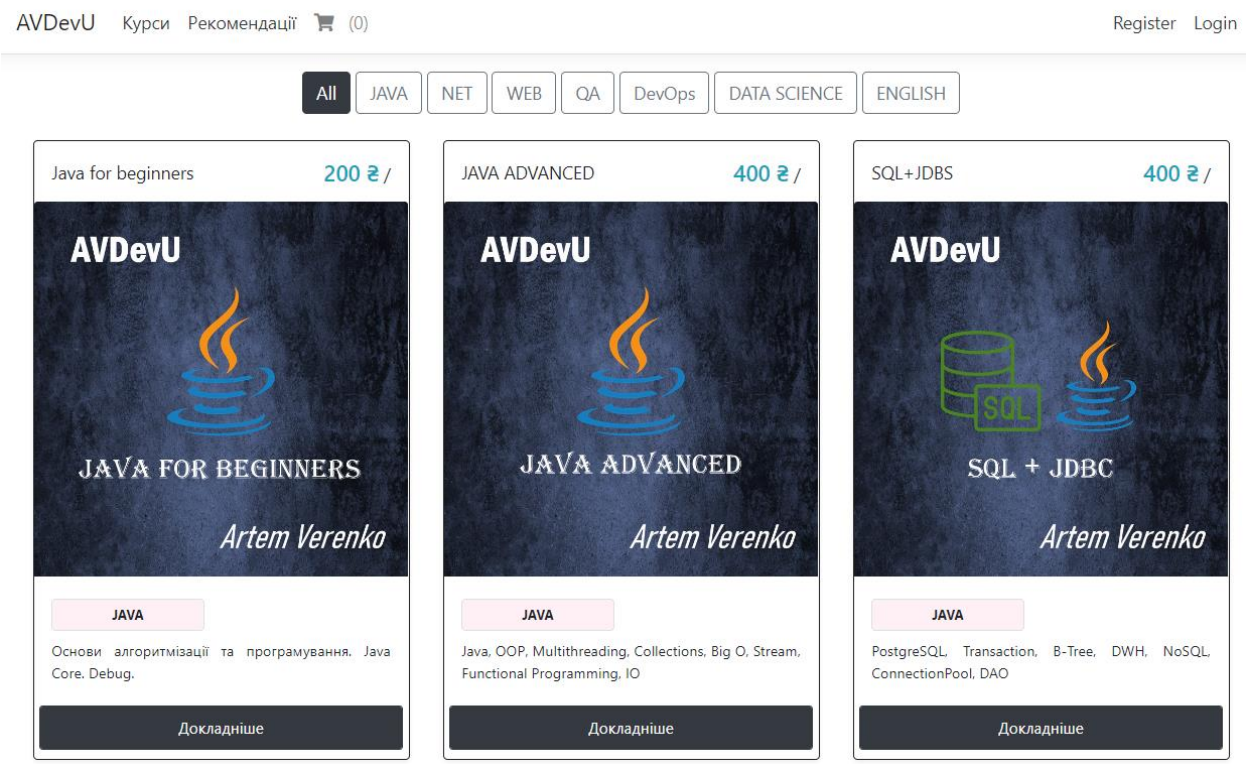

Рисунок 4.11 – Головна сторінка для не зареєстрованого користувача

Користувачу доступна функція сортування курсів за такими категоріями: Java, .NET, WEB, QA, DevOps, Data Science, Enough. Адміністратор сайту має можливість створювати, видаляти та керувати цими категоріями. На картці кожного курсу міститься його короткий опис. Програму курсу та час його проведення можна знайти натиснувши на кнопку Докладніше для кожного окремо.

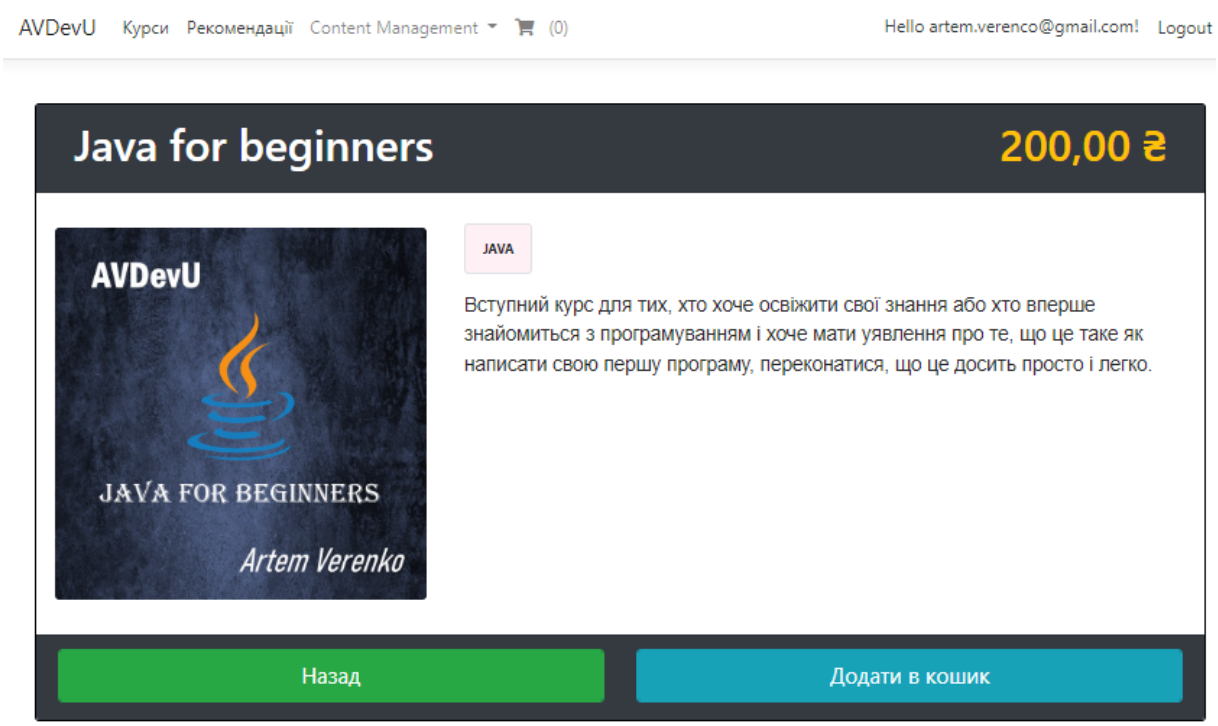

Рисунок 4.12 – Вікно перегляду товару

Для того, щоб відкрити корзину покупок, або перейти на сторінку рекомендацій користувачеві потрібно авторизуватися в системі. Користувач може увійти до системи, якщо його акаунт попередньо створений або зареєструватися. Форма для входу в систему зображена на рисунку 4.13.

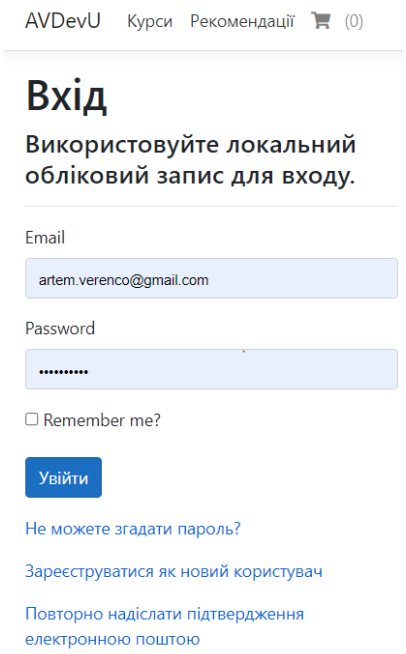

Рисунок 4.13 – Форма входу в систему

На наступному рисунку зображена форма для реэстрації користувача на сайті.

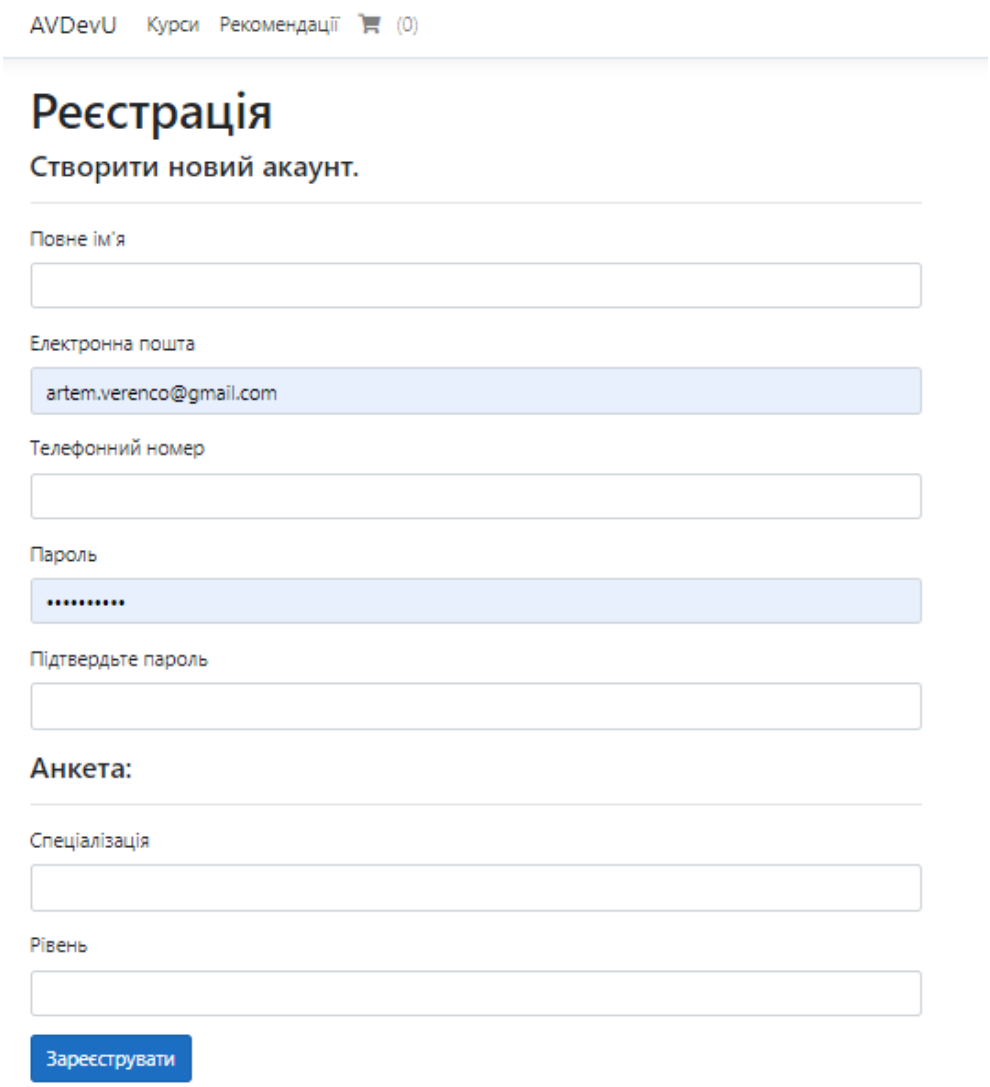

Рисунок 4.14 – Форма реєстрації на сайті

Після входу на сайт користувачеві стає доступними сторінка з рекомендованими користувачеві та операції з корзиною покупок. Рекомендації для окремого користувача надаються шляхом фільтрування усіх курсів відповідно до інтересів користувача. Під час реєстрації користувач вводить в окремі поля свій рівень та бажаний напрямок навчання. Це дає змогу згенерувати сторінку рекомендацій з курсами, що з великою вірогідністю будуть йому підходити. Щоб перейти на сторінку рекомендацій користувач має обрати її в головному меню зверху сторінки.

Розглянемо можливості користувача взаємодіяти з корзиною покупок. Сторінка корзини для покупок з доданими користуввачем курсаси зображена на рисунку 4.15.

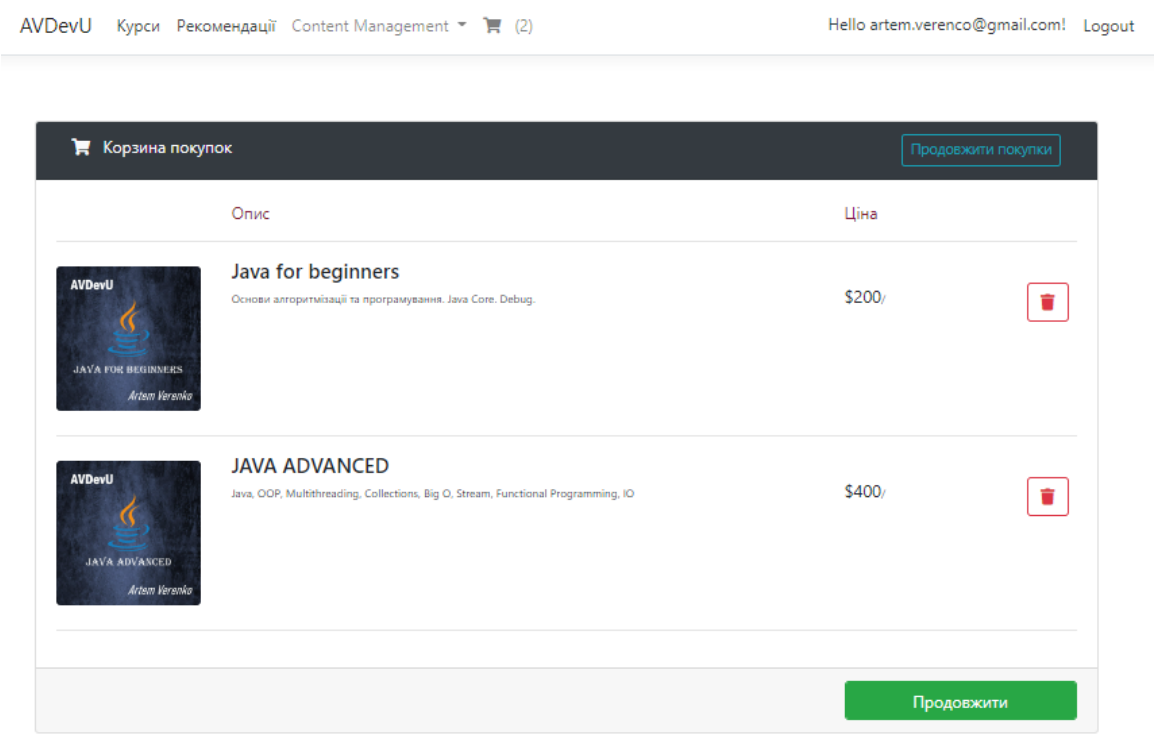

Рисунок 4.15 – Корзина покупок з доданими курсами

Користувач може видалити вибраний курс з корзини, повернутися на сторінку продовження покупок або згенерувати ордер на покупку. Згенерований ордер зображено на рисунку 4.16.

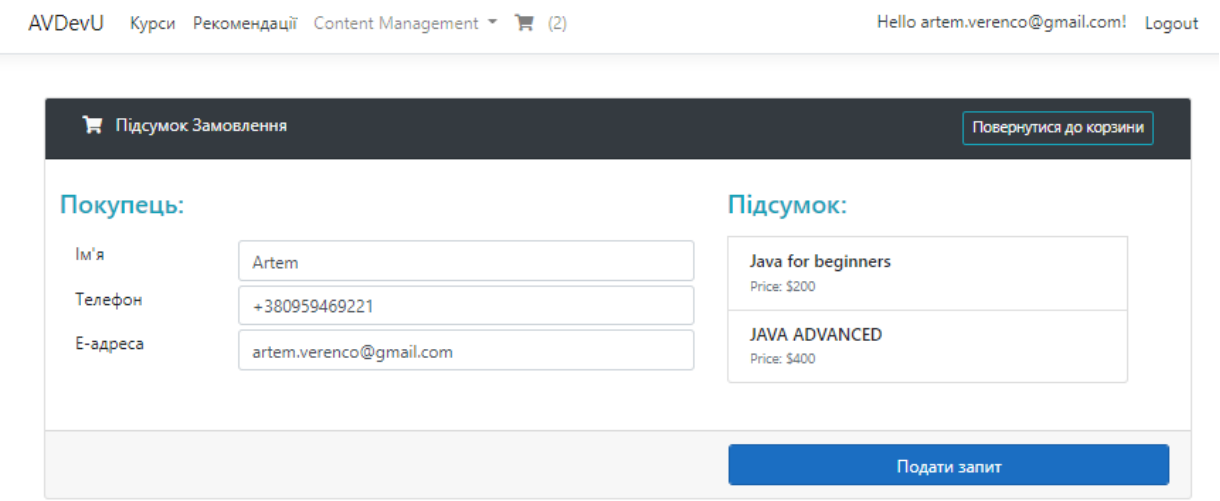

Рисунок 4.16 – Згенерований ордер на покупку курсів

На сторінці згенерованого ордеру користуввач має можливість змінити інформацію про себе та переглянути список курсів, що він купує.

Розглянемо останній можливий сценарій, що доступний адміністратору сайту. Адміністратор на відмінну від звичайного користувача має доступ до панелі інструментів для редагування контенту на сайті. Дана панель зображена на рисунку 4.17.

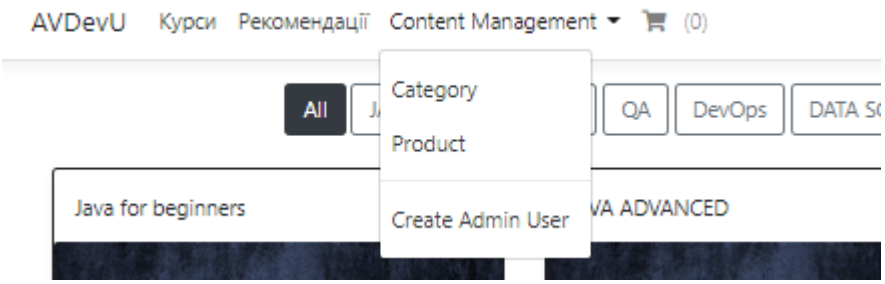

Рисунок 4.17 – Панель керування контентом на сайті

Адміністратор може створювати, редагувати та видаляти категорії товарів на сайті, додавати, редагувати чи видаляти курси до доступних категорій. Також адміністратор може зареєструвати новий акаунт адміністратора.

Сторінка для керування категоріями зображена на рисунку 4.18.

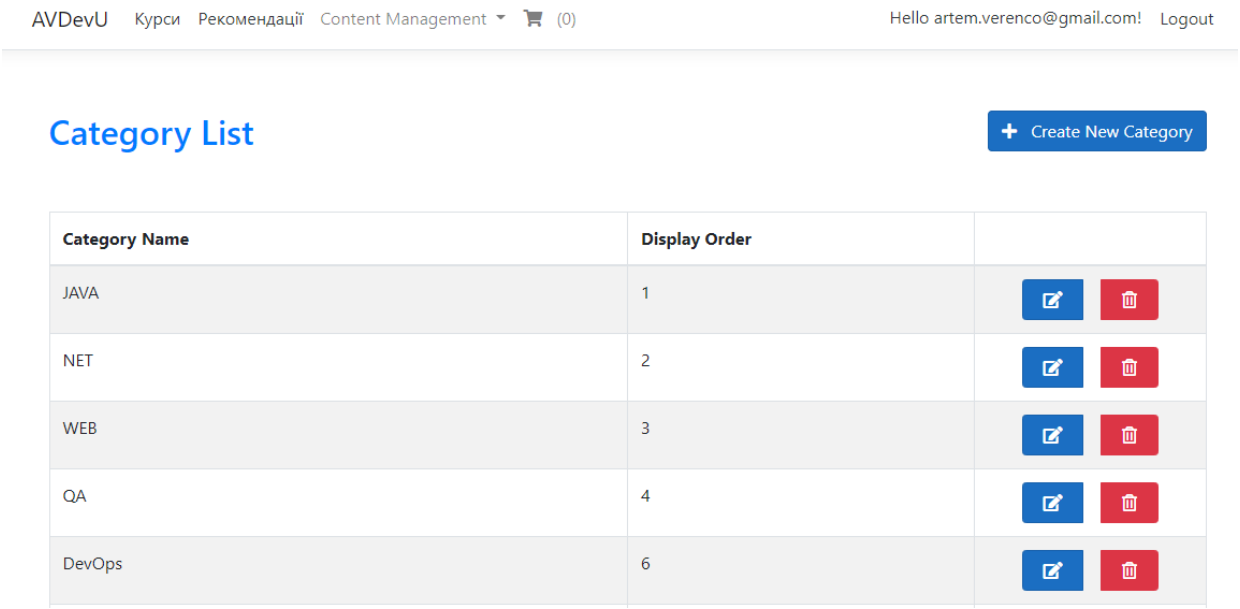

Рисунок 4.18 – Сторінка для керування категоріями курсів

Натискаючи на кнопку Створити нову категорію адміністратору відкривається наступна форма (рисунок 4.19). Для створення ногвої категорії повинні бути заповненні два поля: ім’я категорії, та порядковий номер відображення на сторінці.

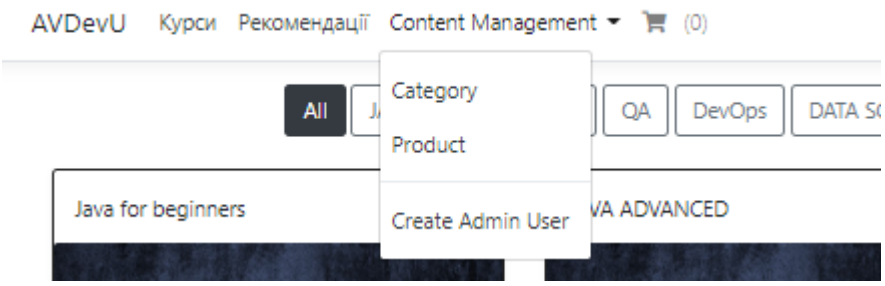

Рисунок 4.19 – Форма створення нової категорії курсів

Сторінка для керування курсами зображена на рисунку 4.20

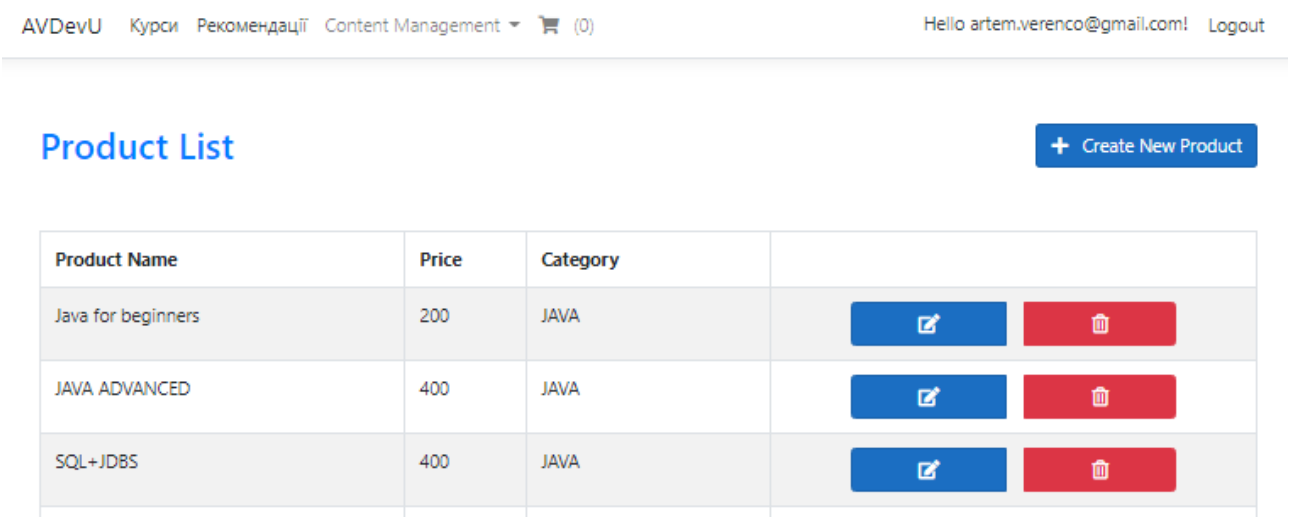

Рисунок 4.20 – Сторінка для керування курсами

Натискаючи на кнопку «Створити новий курс» адміністратору відкривається наступна форма (рисунок 4.21). Для створення нового курсу повинні бути заповненні наступні поля: ім’я курсу, ціна, короткий опис, опис, додана картинка та вибрана категорія.

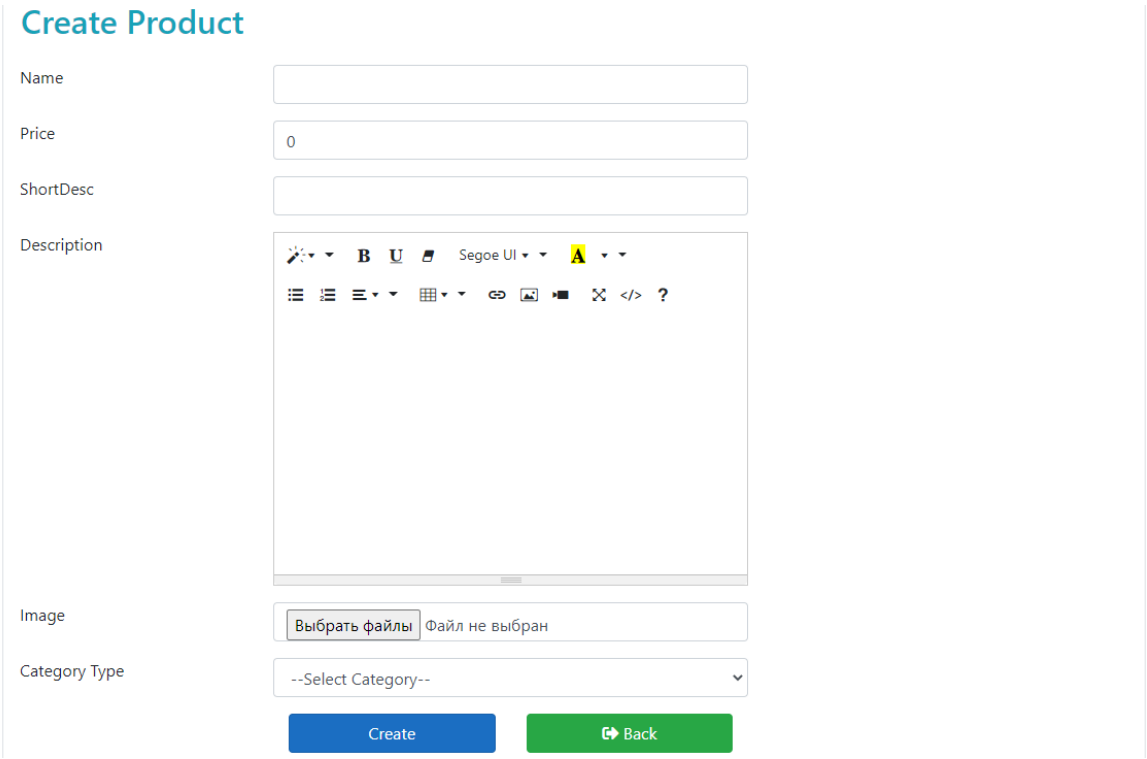

Рисунок 4.21 – Форма створення нового курсу

Розглянемо форму створення нового адміністратора (рисунок 4.22).

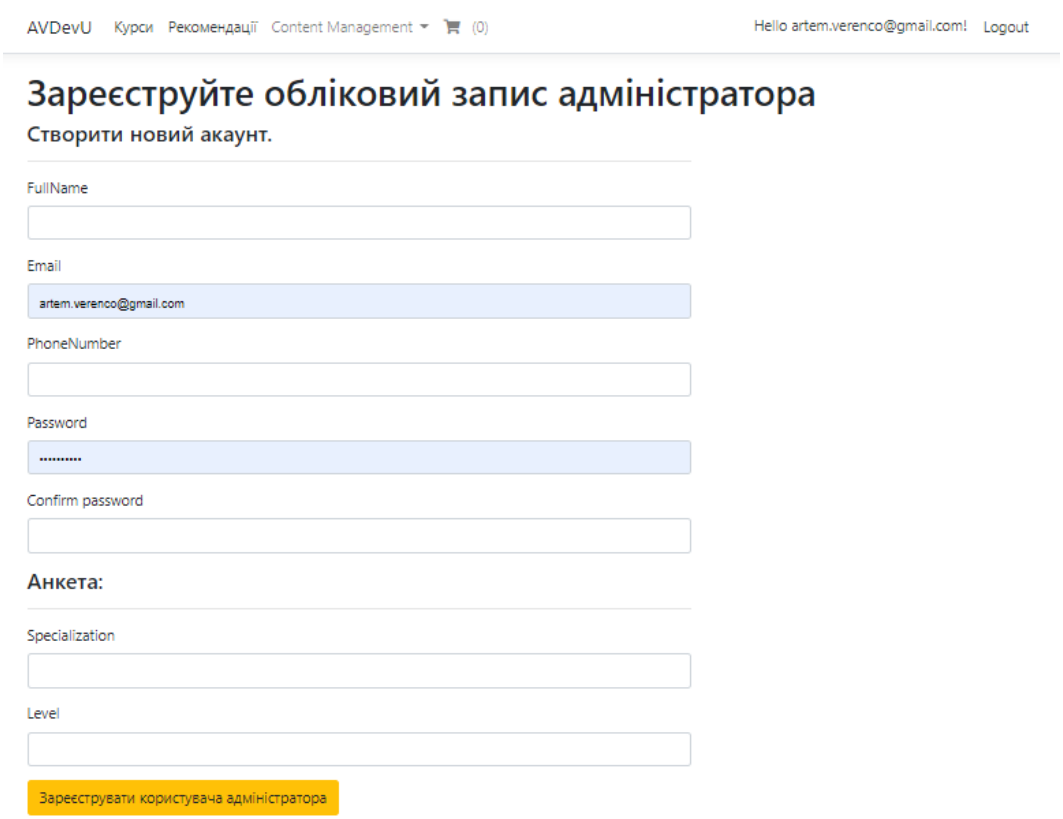

Рисунок 4.22 – Форма реєстрації нового адміністратора сайту

Особливісю розробленої платформи є те, що перший зареєстрований у системі користувач автоматично стає головним адміністратором сайту. Після цього, головний адміністратор має можливість реєструвати інших адміністраторів. Це забезпечує можливість розподіленого між адміністраторами доступу до керування контентом на сайті.

Розглянемо процес використання Android додатку. На рисунку 4.23 зображено його основні робочі області: вікно авторизації, вікно списку уроків та вікно ккремого уроку.

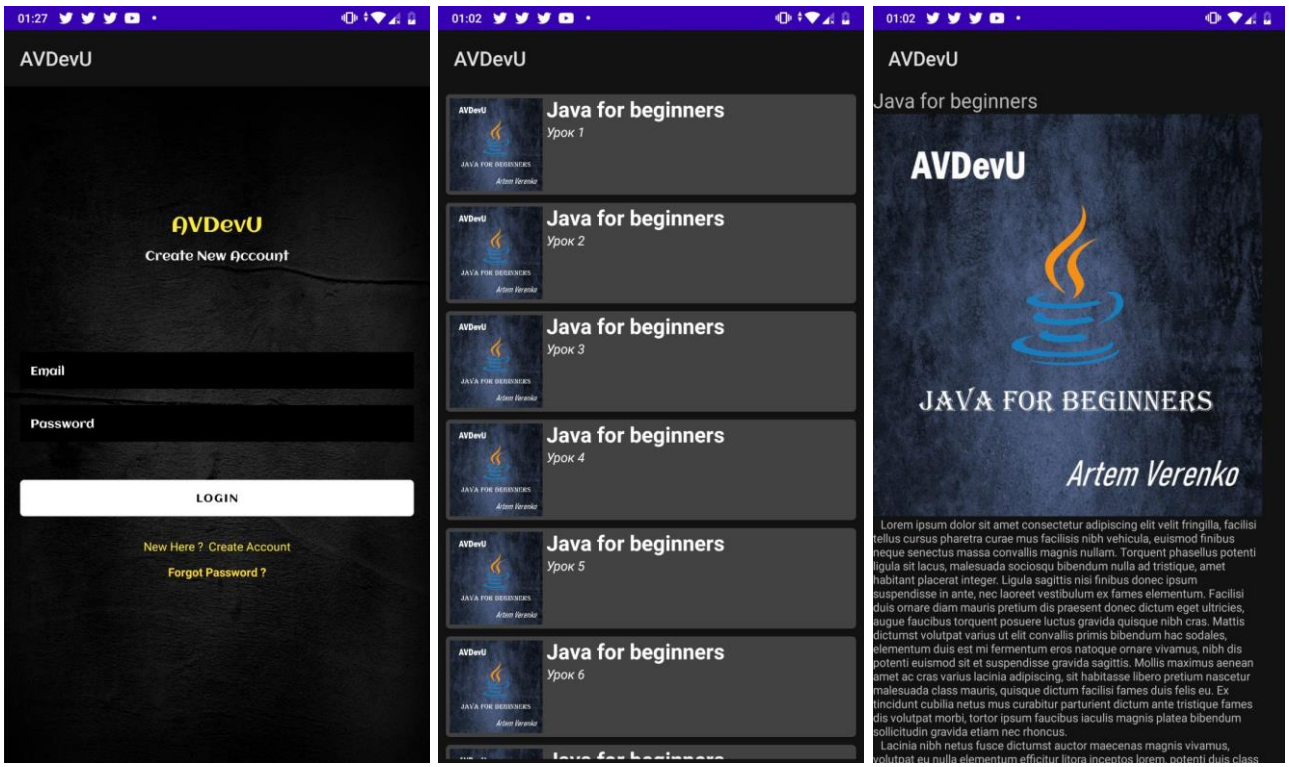

Рисунок 4.23 – Основні робочі області Android додатку

Отже, в даноу підрозділі було розроблено інструкцію користувача сайту відповідно трьом можливим сценаріям використання. Розглянуто процес використання Android додатку для проходження навчального курсу з можливістю авторизації користувача. Було наведено скріншоти інтерфейсу користувача онлайн платформи та Android додатку.

4.4 Висновки

Для проведення тестування онлайн платформи для організації навчальних курсів було обрано методику «чорної скриньки», яка передбачає перевірку функціональності додатку за допомогою тест-кейсів. Результат тестування програмного продукту довів його повну працездатність та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено онлайн платформу для організації навчальних курсів з використанням технології ASP.NET під назвою AVDevU. Розроблений додаток призначений для підвищення ефективності організації навчальних курсів на освітній онлайн платформі. Також було розроблено Android платформу для розміщення навчальних курсів.

Було проведено аналіз існуючих аналогів платформ для організації навчальних курсів. В результаті аналізу аналогів було виявлено основні їх недоліки. Проведено їхнє порівняння із власним програмним продуктом, в результаті було визначено доцільність розробки нової платформи для організації навчальних курсів. Виконано постановку задач розробки програмного продукту.

Розроблено структурну схему онлайн платформи. Запропоновано блок-схеми алгоритмів персонального підбору навчальних курсів та генерування ордеру на покупку навчального курсу.

Створену онлайн платформу розроблено з використанням мови програмування C#, фреймворку ASP.NET MVC та середовища розробки Microsoft Visual Studio 2019. Android платформу розроблено на мові Java. В якості СУБД використоно – Microsoft SQL Server. Як ORM застосовано Entity framework. В результаті виконання бакалаврської дипломної роботи розроблено програмний засіб, працездатність і правильність роботи якого перевірено, підготовлена інструкція користувача.

Було розглянуто основні види та методи тестування, обрано методику «чорного ящика». Проведено тестування за допомогою ряду тест-кейсів, яке довело повну працездатність програмних додатків та їх відповідність поставленому технічному завданню. Також було описано інструкцію користувача розробленої платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sasseen S. The Rising Popularity of Earning a Degree Online [Електронний ресурс] / Stephanie Sasseen. – 2021. – Режим доступу до ресурсу: https://www.onlineu.com/magazine/rising-popularity-of-online-degrees.

2. Miller K. The Benefits of Online Learning: 7 Advantages of Online Degrees [Електронний ресурс] / Kelsey Miller. – 2019. – Режим доступу до ресурсу: https://www.northeastern.edu/graduate/blog/benefits-of-online-learning/.

3. E-learning and digital education - Statistics & Facts Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/topics/3115/e-learning-and-digital-education/#dossierKeyfigures.

4. What are Online Learning Platforms? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mycomputercareer.edu/news/what-are-online-learning-platforms/.

5. Веренько А. І. Вибір архітектури програмної компоненти платформи для монетизації навчальних курсів [Електронний ресурс] / А. І. Веренько, О. В. Романюк // LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця. – 2022. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/16044

6. Веренько А. І. Особливості використання інтернет протоколів для управління електронною поштою [Електронний ресурс] / А. І. Веренько, О. В. Романюк // XXII Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій», Одеса. – 2022. – Режим доступу до ресурсу: https://www.onaft.edu.ua/download/konfi/2022/Conference_abstract-IT-21-22-04-22.pdf.

7. Веренько А. І. Особливості та підходи до розробки android-додатків освітнього спрямування [Електронний ресурс] / А. І. Веренько, О. В. Романюк // Міжнародна науково-практична інтернет-конференція «Електронні

інформаційні ресурси: створення, використання, доступ», Вінниця. – 2021. – Режим доступу до ресурсу: http://ir.lib.vntu.edu.ua/handle/123456789/34850.

8. Веренько А. І. Використання графічного фреймворку LIBGDX для розробки кросплатформних ігор [Електронний ресурс] / А. І. Веренько, О. Н. Романюк // XXI Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій», Одеса. 2021. – Режим доступу до ресурсу: https://ir.lib.vntu.edu.ua/handle/123456789/31898?show=full.

9. Веренько А. І. Оптимізація запитів до баз даних [Електронний ресурс] / А. І. Веренько, І. В. Кучерявий О. В. Романюк // XLIX Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця. – 2021. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9493.

10. Coursera. [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/.

11. Udemy. [Електронний ресурс] – Режим доступу до ресурсу: https://www.udemy.com/.

12. Prometheus. [Електронний ресурс] – Режим доступу до ресурсу: https://prometheus.org.ua/.

13. Freeman A. Pro ASP.NET MVC5 Platform / Adam Freeman., 2014. –411 с.

14. ASP.NET MVC. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/mvc/.

15. Petkovic D. Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition / Dušan Petkovic., 2020. – 896 с.

16. SQL Server 2022. [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/en-us/sql-server.

17. Hirani Z. Entity Framework 6 Recipes/ Zeeshan Hirani., 2013. –548с.

18. Berson A. Client/server architecture / Alex Berson. – New York: McGraw-Hill, 1992. – 452 с.

19. Hughes L. Internet E-mail: Protocols, Standards, and Implementation / Hughes Lawrence., 1998. –456 с..

20. Andy E. AE.Net.Mail [Електронний ресурс] / Andy Edinborough. – 2022. – Режим доступу до ресурсу: https://github.com/andyedinborough/aenetmail.

21. Ford N. Fundamentals of Software Architecture: A Comprehensive Guide to Patterns, Characteristics, and Best Practices / N. Ford, M. Richards., 2020. – 500 с.

22. Lippman S. C++ Primer, Fifth Edition / Stanley Lippman, Josée Lajoie., 2012. - 976 с.

23. Lutz M. Learning Python, 5th Edition / Mark Lutz., 2013. - 1645 с.

24. Urma R. Modern Java in Action: Lambdas, Streams, Functional and Reactive Programming / Raoul-Gabriel , Alan Mycroft., 2018. 838 с.

25. Skeet J. C# in Depth / Jon Skeet., 2008. 787 с.

26. Strauss D. Getting Started with Visual Studio 2019: Learning and Implementing New Features / Dirk Strauss., 2019. 278 с.

27. Rider. Fast & powerful cross-platform .NET IDE [Електронний ресурс] 2022. – Режим доступу до ресурсу: https://www.jetbrains.com/rider/.

28. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с. – ISBN 966-641-081-8.

29. Канер С. Тестування програмного забезпечення / Сем Канер., Діасофт., 2001. 544 с.

ДОДАТКИ

81

Додаток А. Технічне завдання

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
"31" березня 2022 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного забезпечення онлайн платформи для організації навчальних курсів з використанням технології ASP.NET і Android»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доцент О.В.Романюк
"31" березня 2021 р.

Виконав:
_____ студент гр. ЗПІ-18б А. І. Веренько
"31" березня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного забезпечення онлайн платформи для організації навчальних курсів з використанням технології ASP.NET і Android».

Галузь застосування – організація навчальних курсів за допомогою онлайн платформ.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №66 від 24 березня 2022 р. ректора по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою дослідження є підвищення ефективності організації навчальних курсів на освітній онлайн платформі шляхом удосконалення алгоритмів персонального підбору курсів та генерування ордеру на покупку курсу.

Призначення роботи – розробка та програмна реалізація онлайн платформи для організації та монетизації навчальних курсів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Freeman A. Pro ASP.NET MVC5 Platform / Adam Freeman., 2014. –411 с.

2. Petkovic D. Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition / Dušan Petkovic., 2020. – 896 с.

3. Hirani Z. Entity Framework 6 Recipes/ Zeeshan Hirani., 2013. –548с.

4. Ford N. Fundamentals of Software Architecture: A Comprehensive Guide to Patterns, Characteristics, and Best Practices / N. Ford, M. Richards., 2020. – 500 с.

5. Berson A. Client/server architecture / Alex Berson. – New York: McGraw-Hill, 1992. – 452 с.

5. Технічні вимоги

Модель розробки – водоспадна; інтернет протокол для передачі електронних повідомлень – smtp; система управління базами даних – Microsoft SQL Server; мова запитів – SQL; структура об'єктно-реляційного відображення – Entity Framework; база даних з набором навчальних курсів; середовище розробки – Visual Studio 2019; мова програмування – C#, html, css, js, Java; фреймворки: asp.net mvc, bootstrap, jQuery, Android.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації по закінченню робіт:

1. Пояснювальна записка до БДР;
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 бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження. Вибір архітектури програмного компоненту	26.03.2022 – 02.04.2022	Вик.
2	Розробка структури онлайн платформи	03.04.2022 – 09.04.2022	Вик.
3	Розробка алгоритмів персонального підбору курсів та генерування ордеру на покупку	10.04.2022 – 17.04.2022	Вик.
4	Вибір середовища та мови розробки	18.04.2022 – 28.04.2022	Вик.
5	Програмна реалізація онлайн додатку	29.04.2022 – 20.05.2022	Вик.

Продовження таблиці «Стадії та етапи розробки»

6	Тестування програми	21.05.2022 – 25.05.2022	Вик.
7	Оформлення матеріалів до захисту БДР	26.05.2022 – 10.06.2022	Вик.

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

86

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка програмного забезпечення онлайн платформи для організації навчальних курсів з використанням технології ASP.NET і Android»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Романюк О. В.

Оригінальність	**93,4%**
Схожість	6,6%

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 _____ Веренько А. І.

Керівник роботи _____ Романюк О. В.

Додаток В. Лістинг програми

Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Diploma
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Startup.cs

```
using Diploma_DataAccess.Data;
using Diploma_Utility;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System;

namespace Diploma
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<ApplicationDbContext>(options =>
            options.UseSqlServer(
```

```
                Configuration.GetConnectionString("DefaultConnection")));
            services.AddIdentity<IdentityUser, IdentityRole>()
                .AddDefaultTokenProviders().AddDefaultUI()
                .AddEntityFrameworkStores<ApplicationDbContext>();
            services.AddTransient<IEmailSender, EmailSender>();
            services.AddDistributedMemoryCache();
            services.AddHttpContextAccessor();
            services.AddSession(Options =>
            {
                Options.IdleTimeout = TimeSpan.FromMinutes(10);
                Options.Cookie.HttpOnly = true;
                Options.Cookie.IsEssential = true;
            });
            services.AddControllersWithViews();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment()){
                app.UseDeveloperExceptionPage();
            } else {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseSession();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapRazorPages();
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```

Category.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diploma_Models
{
    public class Category
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [DisplayName("Display Order")]
        [Required]
        [Range(1, int.MaxValue, ErrorMessage = "Display Order for category must
be greater than 0")]
        public int DisplayOrder { get; set; }
    }
}
```

CategoryController.cs

```
using Diploma_DataAccess.Data;
using Diploma_Models;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace Diploma.Controllers
{
    public class CategoryController : Controller
    {
        private readonly ApplicationDbContext _db;

        public CategoryController(ApplicationDbContext db)
        {
            _db = db;
        }

        public IActionResult Index()
        {
            IEnumerable<Category> objList = _db.Category;
            return View(objList);
        }

        //GET - CREATE
        public IActionResult Create()
        {
            return View();
        }

        //POST - CREATE
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(Category obj)
        {
            if (ModelState.IsValid)
            {
                _db.Category.Add(obj);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(obj);
        }

        //GET - EDIT
        public IActionResult Edit(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var obj = _db.Category.Find(id);
            if (obj == null)
            {
                return NotFound();
            }
            return View(obj);
        }

        //POST - EDIT
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(Category obj)
        {
```

```
            if (ModelState.IsValid)
            {
                _db.Category.Update(obj);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(obj);
        }

        //GET - DELETE
        public IActionResult Delete(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var obj = _db.Category.Find(id);
            if (obj == null)
            {
                return NotFound();
            }
            return View(obj);
        }

        //POST - DELETE
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult DeletePost(int? id)
        {
            var obj = _db.Category.Find(id);
            if (obj == null)
            {
                return NotFound();
            }
            _db.Category.Remove(obj);
            _db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
```

Category Index.cshtml

```
@model IEnumerable<Diploma_Models.Category>

<div class="container p-3">
    <div class="row pt-4">
        <div class="col-6">
            <h2 class="text-primary">Category List</h2>
        </div>
        <div class="col-6 text-right">
            <a asp-controller="Category" asp-action="Create" class="btn btn-
primary">
                <i class="fas fa-plus"></i>   Create New Category
            </a>
        </div>
    </div>

    <br /><br />

    @if (Model.Count() > 0)
    {
```

```
        <table class="table table-bordered table-striped" style="width:100%">
            <thead>
                <tr>
                    <th>
                        Category Name
                    </th>
                    <th>
                        Display Order
                    </th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach(var obj in Model)
                {
                <tr>
                    <td width="50%">@obj.Name</td>
                    <td width="30%">@obj.DisplayOrder</td>
                    <td class="text-center">
                        <div class="w-75 btn-group" role="group">
                            <a asp-controller="Category" asp-route-Id="@obj.Id"
asp-action="Edit" class="btn btn-primary mx-2">
                                <i class="fas fa-edit"></i>
                            </a>
                            <a asp-controller="Category" asp-route-Id="@obj.Id"
asp-action="Delete" class="btn btn-danger mx-2">
                                <i class="far fa-trash-alt"></i>
                            </a>
                        </div>
                    </td>
                </tr>
                }
            </tbody>
        </table>
    }
    else
    {
        <p> No category exists.</p>
    }

</div>
```

Category Edit.cshtml

```
@model Diploma_Models.Category

<form method="post">
    <input asp-for="Id" hidden />
    <div class="border p-3">

        <div class="form-group row">
            <h2 class="text-info pl-3">Edit Category</h2>
        </div>
        <div class="row">
            <div class="col-8">
                <div class="form-group row">
                    <div class="col-4">
                        <label asp-for="Name"></label>
                    </div>
                    <div class="col-8">
                        <input asp-for="Name" class="form-control" />
```

```
                        <span asp-validation-for="Name" class="text-
danger"></span>
                    </div>
                </div>
                <div class="form-group row">
                    <div class="col-4">
                        <label asp-for="DisplayOrder"></label>
                    </div>
                    <div class="col-8">
                        <input asp-for="DisplayOrder" class="form-control" />
                        <span asp-validation-for="DisplayOrder" class="text-
danger"></span>
                    </div>
                </div>
                <div class="form-group row">
                    <div class="col-8 offset-4 row">

                        <div class="col">
                            <input type="submit" class="btn btn-info w-100"
value="Update" />
                        </div>
                        <div class="col">
                            <a asp-action="Index" class="btn btn-success w-
100"><i class="fas fa-sign-out-alt"></i> Back</a>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-4">
                @* Keep this empty *@
            </div>
        </div>
    </div>
</form>

@section Scripts{
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}
```

Category Delete.cshtml

```
@model Diploma_Models.Category

<form method="post" asp-action="DeletePost">
    <input asp-for="Id" hidden />
    <div class="border p-3">
        <div class="form-group row">
            <h2 class="text-info pl-3">Delete Category</h2>
        </div>
        <div class="row">
            <div class="col-8">
                <div class="form-group row">
                    <div class="col-4">
                        <label asp-for="Name"></label>
                    </div>
                    <div class="col-8">
                        <input asp-for="Name" disabled class="form-control" />
                    </div>
                </div>
                <div class="form-group row">
```

```
                    <div class="col-4">
                        <label asp-for="DisplayOrder"></label>
                    </div>
                    <div class="col-8">
                        <input asp-for="DisplayOrder" disabled class="form-
control" />
                    </div>
                </div>
                <div class="form-group row">
                    <div class="col-8 offset-4 row">

                        <div class="col">
                            <input type="submit" class="btn btn-danger w-100"
value="Delete" />
                        </div>
                        <div class="col">
                            <a asp-action="Index" class="btn btn-success w-
100"><i class="fas fa-sign-out-alt"></i> Back</a>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-4">
                @* Keep this empty *@
            </div>
        </div>
    </div>
</form>

@section Scripts{
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}
```

Category Create.cshtml

```
@model Diploma_Models.Category

<form method="post" asp-action="Create">
    <div class="border p-3">
        @*<div asp-validation-summary="ModelOnly" class="text-danger"></div>*@
        <div class="form-group row">
            <h2 class="text-info pl-3">Add Category</h2>
        </div>
        <div class="row">
            <div class="col-8">
                <div class="form-group row">
                    <div class="col-4">
                        <label asp-for="Name"></label>
                    </div>
                    <div class="col-8">
                        <input asp-for="Name" class="form-control" />
                        <span asp-validation-for="Name" class="text-
danger"></span>
                    </div>
                </div>
                <div class="form-group row">
                    <div class="col-4">
                        <label asp-for="DisplayOrder"></label>
                    </div>
                    <div class="col-8">
```

```
                        <input asp-for="DisplayOrder" class="form-control" />
                        <span asp-validation-for="DisplayOrder" class="text-
danger"></span>
                    </div>
                </div>
                <div class="form-group row">
                    <div class="col-8 offset-4 row">

                        <div class="col">
                            <input type="submit" class="btn btn-info w-100"
value="Create"/>
                        </div>
                        <div class="col">
                            <a asp-action="Index" class="btn btn-success w-
100"><i class="fas fa-sign-out-alt"></i> Back</a>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-4">
                @* Keep this empty *@
            </div>
        </div>
    </div>
</form>

@section Scripts{
    @{ <partial name="_ValidationScriptsPartial" /> }
}
```

CartController.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Claims;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Diploma_DataAccess.Data;
using Diploma_Models;
using Diploma_Models.ViewModels;
using Diploma_Utility;
using Diploma;

namespace Rocky.Controllers
{
    [Authorize]
    public class CartController : Controller
    {
        private readonly ApplicationDbContext _db;
        private readonly IWebHostEnvironment _webHostEnvironment;
        private readonly IEmailSender _emailSender;

        [BindProperty]
        public ProductUserVM ProductUserVM { get; set; }
```

```
        public CartController(ApplicationDbContext db, IWebHostEnvironment
webHostEnvironment, IEmailSender emailSender)
        {
            _db = db;
            _webHostEnvironment = webHostEnvironment;
            _emailSender = emailSender;
        }
        public IActionResult Index()
        {
            List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
            if
(HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
                &&
HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
            {
                //session exsits
                shoppingCartList =
HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
            }
            List<int> prodInCart = shoppingCartList.Select(i =>
i.ProductId).ToList();
            IEnumerable<Product> prodList = _db.Product.Where(u =>
prodInCart.Contains(u.Id));

            return View(prodList);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        [ActionName("Index")]
        public IActionResult IndexPost()
        {
            return RedirectToAction(nameof(Summary));
        }
        public IActionResult Summary()
        {
            var claimsIdentity = (ClaimsIdentity)User.Identity;
            var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);
            //var userId = User.FindFirstValue(ClaimTypes.Name);
            List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
            if
(HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
                &&
HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
            {
                //session exsits
                shoppingCartList =
HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
            }
            List<int> prodInCart = shoppingCartList.Select(i =>
i.ProductId).ToList();
            IEnumerable<Product> prodList = _db.Product.Where(u =>
prodInCart.Contains(u.Id));

            ProductUserVM = new ProductUserVM()
            {
                ApplicationUser = _db.ApplicationUser.FirstOrDefault(u => u.Id
== claim.Value),
                ProductList = prodList.ToList()
            };
            return View(ProductUserVM);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        [ActionName("Summary")]
```

```
        public async Task<IActionResult> SummaryPost(ProductUserVM
ProductUserVM)
        {
            var PathToTemplate = _webHostEnvironment.WebRootPath +
Path.DirectorySeparatorChar.ToString()
                + "templates" + Path.DirectorySeparatorChar.ToString() +
                "Inquiry.html";

            var subject = "New Inquiry";
            string HtmlBody = "";
            using (StreamReader sr = System.IO.File.OpenText(PathToTemplate))
            {
                HtmlBody = sr.ReadToEnd();
            }
            StringBuilder productListSB = new StringBuilder();
            foreach (var prod in ProductUserVM.ProductList)
            {
                productListSB.Append($" - Name: { prod.Name} <span style='font-
size:14px;'> (ID: {prod.Id})</span><br />");
            }
            string messageBody = string.Format(HtmlBody,
                ProductUserVM.ApplicationUser.FullName,
                ProductUserVM.ApplicationUser.Email,
                ProductUserVM.ApplicationUser.PhoneNumber,
                productListSB.ToString());
            await _emailSender.SendEmailAsync(WC.EmailAdmin, subject,
messageBody);
            return RedirectToAction(nameof(InquiryConfirmation));
        }
        public IActionResult InquiryConfirmation()
        {
            HttpContext.Session.Clear();
            return View();
        }
        public IActionResult Remove(int id)
        {
            List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
            if
(HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
                &&
HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
            {
                shoppingCartList =
HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
            }
            shoppingCartList.Remove(shoppingCartList.FirstOrDefault(u =>
u.ProductId == id));
            HttpContext.Session.Set(WC.SessionCart, shoppingCartList);
            return RedirectToAction(nameof(Index));
        }
    }
}
```

ProductController.cs

```
using Diploma_DataAccess.Data;
using Diploma_Models;
using Diploma_Models.ViewModels;
using Diploma_Utility;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
```

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace Diploma.Controllers
{
    public class ProductController : Controller
    {
        private readonly ApplicationDbContext _db;
        private readonly IWebHostEnvironment _webHostEnvironment;
        public ProductController(ApplicationDbContext db, IWebHostEnvironment
webHostEnvironment)
        {
            _db = db;
            _webHostEnvironment = webHostEnvironment;
        }
        public IActionResult Index()
        {
            IEnumerable<Product> objList = _db.Product.Include(u => u.Category);
            return View(objList);
        }

        //GET - UPSERT
        public IActionResult Upsert(int? id)
        {
            ProductVM productVM = new ProductVM()
            {
                Product = new Product(),
                CategorySelectList = _db.Category.Select(i => new SelectListItem
                {
                    Text = i.Name,
                    Value = i.Id.ToString()
                })
            };

            if (id == null)
            {
                //this is for create
                return View(productVM);
            }
            else
            {
                productVM.Product = _db.Product.Find(id);
                if (productVM.Product == null)
                {
                    return NotFound();
                }
                return View(productVM);
            }
        }
        //POST - UPSERT
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Upsert(ProductVM productVM)
        {
            if (ModelState.IsValid)
            {
                var files = HttpContext.Request.Form.Files;
                string webRootPath = _webHostEnvironment.WebRootPath;

                if (productVM.Product.Id == 0)
                {
```

```
                    //Creating
                    string upload = webRootPath + WC.ImagePath;
                    string fileName = Guid.NewGuid().ToString();
                    string extension = Path.GetExtension(files[0].FileName);
                    using (var fileStream = new FileStream(Path.Combine(upload,
fileName + extension), FileMode.Create))
                    {
                        files[0].CopyTo(fileStream);
                    }
                    productVM.Product.Image = fileName + extension;
                    _db.Product.Add(productVM.Product);
                }
                else
                {
                    //updating
                    var objFromDb = _db.Product.AsNoTracking().FirstOrDefault(u
=> u.Id == productVM.Product.Id);

                    if (files.Count > 0)
                    {
                        string upload = webRootPath + WC.ImagePath;
                        string fileName = Guid.NewGuid().ToString();
                        string extension = Path.GetExtension(files[0].FileName);

                        var oldFile = Path.Combine(upload, objFromDb.Image);

                        if (System.IO.File.Exists(oldFile))
                        {
                            System.IO.File.Delete(oldFile);
                        }
                        using (var fileStream = new
FileStream(Path.Combine(upload, fileName + extension), FileMode.Create))
                        {
                            files[0].CopyTo(fileStream);
                        }
                        productVM.Product.Image = fileName + extension;
                    }
                    else
                    {
                        productVM.Product.Image = objFromDb.Image;
                    }
                    _db.Product.Update(productVM.Product);
                }
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            productVM.CategorySelectList = _db.Category.Select(i => new
SelectListItem
            {
                Text = i.Name,
                Value = i.Id.ToString()
            });
            return View(productVM);
        }
        //GET - DELETE
        public IActionResult Delete(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            Product product = _db.Product.Include(u =>
u.Category).FirstOrDefault(u => u.Id == id);
            //product.Category = _db.Category.Find(product.CategoryId);
```

```
            if (product == null)
            {
                return NotFound();
            }
            return View(product);
        }
        //POST - DELETE
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public IActionResult DeletePost(int? id)
        {
            var obj = _db.Product.Find(id);
            if (obj == null)
            {
                return NotFound();
            }
            string upload = _webHostEnvironment.WebRootPath + WC.ImagePath;
            var oldFile = Path.Combine(upload, obj.Image);

            if (System.IO.File.Exists(oldFile))
            {
                System.IO.File.Delete(oldFile);
            }

            _db.Product.Remove(obj);
            _db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
```

RecommendationsController.cs

```
using Diploma_DataAccess.Data;
using Diploma_Models;
using Diploma_Models.ViewModels;
using Diploma_Utility;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

namespace Diploma.Controllers
{
    [Authorize]
    public class RecommendationsController: Controller
    {
        private readonly ILogger<RecommendationsController> _logger;
        public readonly ApplicationDbContext _db;
        public RecommendationsController(ILogger<RecommendationsController>
logger, ApplicationDbContext db)
        {
            _logger = logger;
            _db = db;
        }
        public IActionResult Index()
        {
            ApplicationUser CurrentApplicationUser = (ApplicationUser)_db.Users
                .Where(n => n.UserName == User.Identity.Name).FirstOrDefault();
```

```
            int rank = 0;
            if (CurrentApplicationUser.Level == "Junior")
            {
                rank = 300;
            }
            else if (CurrentApplicationUser.Level == "Middle")
            {
                rank = 700;
            }
            else
            {
                rank = int.MaxValue;
            }
            RecommendationsVM recommendationsVM = new RecommendationsVM()
            {
                Products = _db.Product.Where(p => p.Price <= rank)
                .Where(p => p.Category.Name ==
CurrentApplicationUser.Specialization).Include(u => u.Category),
                Categories = _db.Category
                .Where(c => c.Name == CurrentApplicationUser.Specialization)
            };
            return View(recommendationsVM);
        }
        public IActionResult Details(int id)
        {
            List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
            if
(HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
                &&
HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
            {
                shoppingCartList =
HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
            }
            DetailsVM DetailsVM = new DetailsVM()
            {
                Product = _db.Product.Include(u => u.Category)
                .Where(u => u.Id == id).FirstOrDefault(),
                ExistsInCart = false
            };
            foreach (var item in shoppingCartList)
            {
                if (item.ProductId == id)
                {
                    DetailsVM.ExistsInCart = true;
                }
            }
            return View(DetailsVM);
        }
        [HttpPost, ActionName("Details")]
        public IActionResult DetailsPost(int id)
        {
            List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
            if
(HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
                &&
HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
            {
                shoppingCartList =
HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
            }
            shoppingCartList.Add(new ShoppingCart { ProductId = id });
            HttpContext.Session.Set(WC.SessionCart, shoppingCartList);
```

```
            return RedirectToAction(nameof(Index));
        }
        public IActionResult RemoveFromCart(int id)
        {
            List<ShoppingCart> shoppingCartList = new List<ShoppingCart>();
            if
(HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart) != null
                &&
HttpContext.Session.Get<IEnumerable<ShoppingCart>>(WC.SessionCart).Count() > 0)
            {
                shoppingCartList =
HttpContext.Session.Get<List<ShoppingCart>>(WC.SessionCart);
            }
            var itemToRemove = shoppingCartList.SingleOrDefault(r => r.ProductId
== id);
            if (itemToRemove != null)
            {
                shoppingCartList.Remove(itemToRemove);
            }
            HttpContext.Session.Set(WC.SessionCart, shoppingCartList);
            return RedirectToAction(nameof(Index));
        }
        public IActionResult Privacy()
        {
            return View();
        }
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

Product .cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Diploma_Models
{
    public class Product
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public string ShortDesc { get; set; }
        public string Description { get; set; }
        [Range(1, int.MaxValue)]
        public double Price { get; set; }
        public string Image { get; set; }
        [Display(Name = "Category Type")]
        public int CategoryId { get; set; }
        [ForeignKey("CategoryId")]
        public virtual Category Category { get; set; }
    }
}
```

ApplicationUser.cs

```
using Microsoft.AspNetCore.Identity;

namespace Diploma_Models
{
    public class ApplicationUser : IdentityUser
    {
        public string FullName { get; set; }
        public string Specialization { get; set; }
        public string Level { get; set; }
        public ApplicationUser()
        {
        }
        public ApplicationUser(string fullName, string specialization, string
level)
        {
            FullName = fullName;
            Specialization = specialization;
            Level = level;
        }
    }
}
```

ShoppingCart.cs

```
namespace Diploma_Models
{
    public class ShoppingCart
    {
        public int ProductId { get; set; }
    }
}
```

DetailsVM.cs

```
namespace Diploma_Models.ViewModels
{
    public class DetailsVM
    {
        public DetailsVM()
        {
            Product = new Product();
        }
        public Product Product { get; set; }
        public bool ExistsInCart { get; set; }
    }
}
```

ProductUserVM.cs

```
using System.Collections.Generic;

namespace Diploma_Models.ViewModels
{
    public class ProductUserVM
    {
        public ProductUserVM()
```

```
        {
            ProductList = new List<Product>();
        }
        public ApplicationUser ApplicationUser { get; set; }
        public IList<Product> ProductList { get; set; }
    }
}
```

ProductVM.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;
using System.Collections.Generic;

namespace Diploma_Models.ViewModels
{
    public class ProductVM
    {
        public Product Product { get; set; }
        public IEnumerable<SelectListItem> CategorySelectList { get; set; }
        public IEnumerable<SelectListItem> ApplicationTypeSelectList { get; set;
}
    }
}
```

RecommendationsVM.cs

```
using System.Collections.Generic;

namespace Diploma_Models.ViewModels
{
    public class RecommendationsVM
    {
        public ApplicationUser CurrentApplicationUser { get; set; }
        public IEnumerable<Product> Products { get; set; }
        public IEnumerable<Category> Categories { get; set; }
    }
}
```

ApplicationDbContext.cs

```
using Diploma_Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace Diploma_DataAccess.Data
{
    public class ApplicationDbContext: IdentityDbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options): base(options)
        {
        }
        public DbSet<Category> Category { get; set; }
        public DbSet<Product> Product { get; set; }
        public DbSet<ApplicationUser> ApplicationUser { get; set; }

    }
}
```

EmailSender.cs

```
using Microsoft.AspNetCore.Identity.UI.Services;

using Microsoft.Extensions.Configuration;
using System.Net;
using System.Net.Mail;
using System.Threading.Tasks;

namespace Diploma_Utility
{
    public class EmailSender : IEmailSender
    {
        private readonly IConfiguration _configuration;
        private GMailSetings _GMailSetings { get; set; }

        public EmailSender(IConfiguration configuration)
        {
            _configuration = configuration;
        }
        public Task SendEmailAsync(string email, string subject, string
htmlMessage)
        {
            return Execute(email, subject, htmlMessage);
        }
        public async Task Execute(string email, string subject, string
htmlMessage)
        {
            _GMailSetings =
_configuration.GetSection("GMailSetings").Get<GMailSetings>();
            using (SmtpClient client = new SmtpClient("smtp.gmail.com", 587))
            {
                client.EnableSsl = true;
                client.DeliveryMethod = SmtpDeliveryMethod.Network;
                client.UseDefaultCredentials = false;
                client.Credentials = new
NetworkCredential("artemverenkotest@gmail.com", "vzgolubwzggeoylq");
                MailMessage msgObj = new MailMessage();
                msgObj.To.Add(email);
                msgObj.From = new MailAddress("artemverenkotest@gmail.com");
                msgObj.Subject = subject;
                msgObj.Body = htmlMessage;
                msgObj.IsBodyHtml = true;
                client.Send(msgObj);
            }
        }
    }
}
```

GMailSetings.cs

```
namespace Diploma_Utility
{
    public class GMailSetings
    {
        public string Address { get; set; }
        public string SecretKey { get; set; }
    }
}
```

SessionExtensions.cs

```
using Microsoft.AspNetCore.Http;
using System.Text.Json;
```

```
namespace Diploma_Utility
{
    public static class SessionExtensions
    {
        public static void Set<T>(this ISession session, string key, T value)
        {
            session.SetString(key, JsonSerializer.Serialize(value));
        }
        public static T Get<T>(this ISession session, string key)
        {
            var value = session.GetString(key);
            return value == null ? default :
JsonSerializer.Deserialize<T>(value);
        }
    }
}
```

WC.cs

```
namespace Diploma_Utility
{
    public static class WC
    {
        public const string ImagePath = @"\images\product\";
        public const string SessionCart = "ShoppingCartSession";

        public const string AdminRole = "Admin";
        public const string CustomerRole = "Customer";

        public const string EmailAdmin = "artem.verenco@gmail.com";
    }
}
```

Login.cshtml

```
@page
@model LoginModel
@{
    ViewData["Title"] = "Вхід";
}
<h1>@ViewData["Title"]</h1>
<div class="row">
    <div class="col-md-4">
        <section>
            <form id="account" method="post">
                <h4>Використовуйте локальний обліковий запис для входу.</h4>
                <hr />
                <div asp-validation-summary="All" class="text-danger"></div>
                <div class="form-group">
                    <label asp-for="Input.Email"></label>
                    <input asp-for="Input.Email" class="form-control" />
                    <span asp-validation-for="Input.Email" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="Input.Password"></label>
                    <input asp-for="Input.Password" class="form-control" />
                    <span asp-validation-for="Input.Password" class="text-
danger"></span>
                </div>
                <div class="form-group">
                    <div class="checkbox">
```

```
                        <label asp-for="Input.RememberMe">
                            <input asp-for="Input.RememberMe" />
                            @Html.DisplayNameFor(m => m.Input.RememberMe)
                        </label>
                    </div>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-primary">Увійти</button>
                </div>
                <div class="form-group">
                    <p>
                        <a id="forgot-password" asp-page="./ForgotPassword">Не можете
згадати пароль?</a>
                    </p>
                    <p>
                        <a asp-page="./Register" asp-route-
returnUrl="@Model.ReturnUrl">Зареєструватися як новий користувач</a>
                    </p>
                    <p>
                        <a id="resend-confirmation" asp-
page="./ResendEmailConfirmation">Повторно надіслати підтвердження електронною
поштою</a>
                    </p>
                </div>
            </form>
        </section>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Login.cshtml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text.Encodings.Web;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Logging;

namespace Diploma.Areas.Identity.Pages.Account
{
    [AllowAnonymous]
    public class LoginModel : PageModel
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly SignInManager<IdentityUser> _signInManager;
        private readonly ILogger<LoginModel> _logger;

        public LoginModel(SignInManager<IdentityUser> signInManager,
            ILogger<LoginModel> logger,
            UserManager<IdentityUser> userManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
```

```
            _logger = logger;
        }

        [BindProperty]
        public InputModel Input { get; set; }

        public IList<AuthenticationScheme> ExternalLogins { get; set; }

        public string ReturnUrl { get; set; }

        [TempData]
        public string ErrorMessage { get; set; }

        public class InputModel
        {
            [Required]
            [EmailAddress]
            public string Email { get; set; }

            [Required]
            [DataType(DataType.Password)]
            public string Password { get; set; }

            [Display(Name = "Remember me?")]
            public bool RememberMe { get; set; }
        }

        public async Task OnGetAsync(string returnUrl = null)
        {
            if (!string.IsNullOrEmpty(ErrorMessage))
            {
                ModelState.AddModelError(string.Empty, ErrorMessage);
            }

            returnUrl ??= Url.Content("~/");

            // Clear the existing external cookie to ensure a clean login process
            await HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);

            ExternalLogins = (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();

            ReturnUrl = returnUrl;
        }

        public async Task<IActionResult> OnPostAsync(string returnUrl = null)
        {
            returnUrl ??= Url.Content("~/");

            ExternalLogins = (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();

            if (ModelState.IsValid)
            {
                // This doesn't count login failures towards account lockout
                // To enable password failures to trigger account lockout, set
lockoutOnFailure: true
                var result = await _signInManager.PasswordSignInAsync(Input.Email,
Input.Password, Input.RememberMe, lockoutOnFailure: false);
                if (result.Succeeded)
                {
                    _logger.LogInformation("User logged in.");
                    return LocalRedirect(returnUrl);
                }
                if (result.RequiresTwoFactor)
```

```
                {
                    return RedirectToPage("./LoginWith2fa", new { ReturnUrl =
returnUrl, RememberMe = Input.RememberMe });
                }
                if (result.IsLockedOut)
                {
                    _logger.LogWarning("User account locked out.");
                    return RedirectToPage("./Lockout");
                }
                else
                {
                    ModelState.AddModelError(string.Empty, "Invalid login attempt.");
                    return Page();
                }
            }

            // If we got this far, something failed, redisplay form
            return Page();
        }
    }
}
```

Register.cshtml

```
@page
@model RegisterModel

@if (User.IsInRole(Diploma_Utility.WC.AdminRole))
{
    <h1>Зареєструйте обліковий запис адміністратора</h1>
}
else
{
    <h1>Реєстрація</h1>
}

<div class="row">
    <div class="col-md-8">
        <form asp-route-returnUrl="@Model.ReturnUrl" method="post">
            <h4>Створити новий акаунт.</h4>
            <hr />
            <div asp-validation-summary="All" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Input.FullName"></label>
                <input asp-for="Input.FullName" class="form-control" />
                <span asp-validation-for="Input.FullName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.Email"></label>
                <input asp-for="Input.Email" class="form-control" />
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.PhoneNumber"></label>
                <input asp-for="Input.PhoneNumber" class="form-control" />
                <span asp-validation-for="Input.PhoneNumber" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.Password"></label>
                <input asp-for="Input.Password" class="form-control" />
                <span asp-validation-for="Input.Password" class="text-danger"></span>
            </div>
            <div class="form-group">
```

```
                <label asp-for="Input.ConfirmPassword"></label>
                <input asp-for="Input.ConfirmPassword" class="form-control" />
                <span asp-validation-for="Input.ConfirmPassword" class="text-
danger"></span>
            </div>
             <h4>Анкета:</h4>
            <hr />
            <div class="form-group">
                <label asp-for="Input.Specialization"></label>
                <input asp-for="Input.Specialization" class="form-control" />
                <span asp-validation-for="Input.Specialization" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.Level"></label>
                <input asp-for="Input.Level" class="form-control" />
                <span asp-validation-for="Input.Level" class="text-danger"></span>
            </div>

            @if (User.IsInRole(Diploma_Utility.WC.AdminRole))
            {
                <button type="submit" class="btn btn-warning">Зареєструвати
користувача адміністратора</button>
            }
            else
            {
                <button type="submit" class="btn btn-primary">Зареєструвати</button>
            }

        </form>
    </div>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Register.cshtml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Text.Encodings.Web;
using System.Threading.Tasks;
using Diploma_Models;
using Diploma_Utility;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.WebUtilities;
using Microsoft.Extensions.Logging;

namespace Diploma.Areas.Identity.Pages.Account
{
    [AllowAnonymous]
    public class RegisterModel : PageModel
    {
        private readonly SignInManager<IdentityUser> _signInManager;
        private readonly UserManager<IdentityUser> _userManager;
```

```
        private readonly ILogger<RegisterModel> _logger;
        private readonly IEmailSender _emailSender;
        private readonly RoleManager<IdentityRole> _roleManager;

        public RegisterModel(
            UserManager<IdentityUser> userManager,
            SignInManager<IdentityUser> signInManager,
            ILogger<RegisterModel> logger,
            IEmailSender emailSender,
            RoleManager<IdentityRole> roleManager)
        {
            _roleManager = roleManager;
            _userManager = userManager;
            _signInManager = signInManager;
            _logger = logger;
            _emailSender = emailSender;
        }

        [BindProperty]
        public InputModel Input { get; set; }

        public string ReturnUrl { get; set; }

        public IList<AuthenticationScheme> ExternalLogins { get; set; }

        public class InputModel
        {
            [Required]
            [EmailAddress]
            [Display(Name = "Email")]
            public string Email { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max
{1} characters long.", MinimumLength = 6)]
            [DataType(DataType.Password)]
            [Display(Name = "Password")]
            public string Password { get; set; }

            [DataType(DataType.Password)]
            [Display(Name = "Confirm password")]
            [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
            public string ConfirmPassword { get; set; }

            public string FullName { get; set; }
            public string PhoneNumber { get; set; }

            public string Specialization { get; set; }
            public string Level { get; set; }
        }

        public async Task OnGetAsync(string returnUrl = null)
        {
            if (!await _roleManager.RoleExistsAsync(WC.AdminRole))
            {
                await _roleManager.CreateAsync(new IdentityRole(WC.AdminRole));
                await _roleManager.CreateAsync(new IdentityRole(WC.CustomerRole));
            }

            ReturnUrl = returnUrl;
            ExternalLogins = (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
        }
```

```
        public async Task<IActionResult> OnPostAsync(string returnUrl = null)
        {
            returnUrl ??= Url.Content("~/");
            ExternalLogins = (await
_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
            if (ModelState.IsValid)
            {
                var user = new ApplicationUser { UserName = Input.Email, Email =
Input.Email, PhoneNumber = Input.PhoneNumber, FullName = Input.FullName,
Specialization = Input.Specialization, Level = Input.Level  };
                var result = await _userManager.CreateAsync(user, Input.Password);
                if (result.Succeeded)
                {
                    if (User.IsInRole(WC.AdminRole))
                    {
                        await _userManager.AddToRoleAsync(user, WC.AdminRole);
                    }
                    else
                    {
                        await _userManager.AddToRoleAsync(user, WC.CustomerRole);
                    }
;
                    _logger.LogInformation("User created a new account with
password.");
                    var code = await
_userManager.GenerateEmailConfirmationTokenAsync(user);
                    code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
                    var callbackUrl = Url.Page(
                        "/Account/ConfirmEmail",
                        pageHandler: null,
                        values: new { area = "Identity", userId = user.Id, code =
code, returnUrl = returnUrl },
                        protocol: Request.Scheme);

                    await _emailSender.SendEmailAsync(Input.Email, "Confirm your
email",
                        $"Please confirm your account by <a
href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

                    if (_userManager.Options.SignIn.RequireConfirmedAccount)
                    {
                        return RedirectToPage("RegisterConfirmation", new { email =
Input.Email, returnUrl = returnUrl });
                    }
                    else
                    {
                        await _signInManager.SignInAsync(user, isPersistent: false);
                        return LocalRedirect(returnUrl);
                    }
                }
                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
            }
            return Page();
        }
    }
}
```

Login.java

```
package com.artem_verenko.beyond_order.artwq;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class Login extends AppCompatActivity {
    EditText mEmail,mPassword;
    Button mLoginBtn;
    TextView mCreateBtn,forgotTextLink;
    ProgressBar progressBar;
    FirebaseAuth fAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        mEmail = findViewById(R.id.Email);
        mPassword = findViewById(R.id.password);
        progressBar = findViewById(R.id.progressBar);
        fAuth = FirebaseAuth.getInstance();
        mLoginBtn = findViewById(R.id.loginBtn);
        mCreateBtn = findViewById(R.id.createText);
        forgotTextLink = findViewById(R.id.forgotPassword);
        mLoginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = mEmail.getText().toString().trim();
                String password = mPassword.getText().toString().trim();
                if(TextUtils.isEmpty(email)){
                    mEmail.setError("Email is Required.");
                    return;
                }
                if(TextUtils.isEmpty(password)){
                    mPassword.setError("Password is Required.");
                    return;
                }
                if(password.length() < 6){
                    mPassword.setError("Password Must be >= 6 Characters");
                    return;
                }
                progressBar.setVisibility(View.VISIBLE);
                // authenticate the user

fAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if(task.isSuccessful()){
```

```
                            Toast.makeText(Login.this, "Logged in Successfully",
Toast.LENGTH_SHORT).show();
                            startActivity(new
Intent(getApplicationContext(),MainActivity.class));
                        }else {
                            Toast.makeText(Login.this, "Error ! " +
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                            progressBar.setVisibility(View.GONE);
                        }
                    }
                });
            }
        });
        mCreateBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(getApplicationContext(),Register.class));
            }
        });
        forgotTextLink.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final EditText resetMail = new EditText(v.getContext());
                final AlertDialog.Builder passwordResetDialog = new
AlertDialog.Builder(v.getContext());
                passwordResetDialog.setTitle("Reset Password ?");
                passwordResetDialog.setMessage("Enter Your Email To Received
Reset Link.");
                passwordResetDialog.setView(resetMail);
                passwordResetDialog.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // extract the email and send reset link
                        String mail = resetMail.getText().toString();

fAuth.sendPasswordResetEmail(mail).addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Toast.makeText(Login.this, "Reset Link Sent To
Your Email.", Toast.LENGTH_SHORT).show();
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Toast.makeText(Login.this, "Error ! Reset Link
is Not Sent" + e.getMessage(), Toast.LENGTH_SHORT).show();
                            }
                        });

                    }
                });
                passwordResetDialog.setNegativeButton("No", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // close the dialog
                    }
                });
                passwordResetDialog.create().show();
            }
        });
```

```
    }
}
```

activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".Login">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/aclonica"
        android:text="AVDevU"
        android:textColor="#FFEB3B"
        android:textSize="25sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.19" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:fontFamily="@font/aclonica"
        android:text="Create New Account"
        android:textColor="#F7F7F8"
        android:textSize="15sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <EditText
        android:id="@+id/Email"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:background="@android:color/black"
        android:ems="10"
        android:fontFamily="@font/aclonica"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:padding="10dp"
        android:textColor="#FFFFFF"
        android:textColorHint="#FFFFFF"
        android:textSize="14sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        app:layout_constraintVertical_bias="0.19" />
```

```
<EditText
    android:id="@+id/password"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:background="@android:color/black"
    android:ems="10"
    android:fontFamily="@font/aclonica"
    android:hint="Password"
    android:inputType="textPassword"
    android:padding="10dp"
    android:textColor="#FFFFFF"
    android:textColorHint="#FFFFFF"
    android:textSize="14sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Email" />

<Button
    android:id="@+id/loginBtn"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:fontFamily="@font/aclonica"
    android:text="Login"
    android:textSize="13sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/password" />

<TextView
    android:id="@+id/createText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="New Here ?  Create Account"
    android:textColor="#FFEB3B"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/loginBtn" />

<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/createText"
    app:layout_constraintVertical_bias="0.39" />

<TextView
    android:id="@+id/forgotPassword"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Forgot Password ?"
    android:textColor="#FDD835"
```

```
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/progressBar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/createText"
        app:layout_constraintVertical_bias="0.19999999" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

MViewItemLayout.java

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

public class MViewItemLayout extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_m_view_item_layout);
        TextView title = findViewById(R.id.titleTV);
        TextView text = findViewById(R.id.textTV);
        ImageView imageView = findViewById(R.id.imageV);
        Intent intent = getIntent();
        if(intent != null){
            title.setText(intent.getStringExtra("titleTV"));
            text.setText(intent.getStringExtra("textTV"));
            imageView.setImageResource(intent.getIntExtra("imageV", 0));
        }
    }
}
```

MainActivity.java

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private RecyclerView.Adapter adapter;

    private RecyclerView.LayoutManager layoutManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayList<RecyclerViewItem> recyclerViewItems = new ArrayList<>();
        for (int i = 1; i <= 10; i++) {
            recyclerViewItems.add( new RecyclerViewItem(R.drawable.java1, "Java
for beginners", "Урок " + i, MyConstants.Lesson1_L));
        }
        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
```

```
        adapter = new RecyclerViewAdapter(recyclerViewItems, this);
        layoutManager = new LinearLayoutManager(this);
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(layoutManager);
    }
}
```

RecyclerViewAdapter.java

```
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;

public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.RecyclerViewViewHolder> {
    private ArrayList<RecyclerViewItem> arrayList;
    Context context;

    public RecyclerViewAdapter(ArrayList<RecyclerViewItem> arrayList, Context
context){
        this.arrayList = arrayList;
        this.context = context;
    }

    @NonNull
    @Override
    public RecyclerViewViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.recycler_view_item,
parent, false);
        RecyclerViewViewHolder recyclerViewViewHolder = new
RecyclerViewViewHolder(view);
        return recyclerViewViewHolder;
    }
    @Override
    public void onBindViewHolder(@NonNull RecyclerViewViewHolder holder, int
position) {
        RecyclerViewItem recyclerViewItem = arrayList.get(position);
        holder.imageView.setImageResource(recyclerViewItem.getImageResource());
        holder.textView1.setText(recyclerViewItem.getText1());
        holder.textView2.setText(recyclerViewItem.getText2());
    }
    @Override
    public int getItemCount() {
        return arrayList.size();
    }

    public class RecyclerViewViewHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
        public ImageView imageView;
        public TextView textView1;
        public TextView textView2;
        public RecyclerViewViewHolder(@NonNull View itemView) {
            super(itemView);
```

```
            itemView.setOnClickListener(this);
            imageView = itemView.findViewById(R.id.imageView);
            textView1 = itemView.findViewById(R.id.textView1);
            textView2 = itemView.findViewById(R.id.textView2);
        }
        @Override
        public void onClick(View v) {
            int position = getAdapterPosition();
            RecyclerViewItem recyclerViewItem = arrayList.get(position);
            Intent intent = new Intent(context, MViewItemLayout.class);
            intent.putExtra("imageV", recyclerViewItem.getImageResource());
            intent.putExtra("titleTV", recyclerViewItem.getText1());
            intent.putExtra("textTV", recyclerViewItem.getText3());
            context.startActivity(intent);
        }
    }
}
```

RecyclerViewItem.java

```
public class RecyclerViewItem {
    private int imageResource;
    private String text1;
    private String text2;
    private String text3;
    public RecyclerViewItem(int imageResource, String text1, String text2,
String text3) {
        this.imageResource = imageResource;
        this.text1 = text1;
        this.text2 = text2;
        this.text3 = text3;
    }
    public int getImageResource() {
        return imageResource;
    }
    public String getText1() {
        return text1;
    }
    public String getText2() {
        return text2;
    }
    public String getText3() {
        return text3;
    }
}
```

activity_m_view_item_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MViewItemLayout"
    android:orientation="vertical">

    <ScrollView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">
    <TextView
        android:id="@+id/titleTV"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <ImageView
        android:id="@+id/imageV"
        android:layout_width="400dp"
        android:layout_height="400dp"
        android:layout_gravity="center"
        />
    <TextView
        android:id="@+id/textTV"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```

recycler_view_item.xml

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="wrap_content"
xmlns:app="http://schemas.android.com/apk/res-auto"
app:cardCornerRadius="4dp"
android:layout_margin="4dp">
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:padding="4dp"/>
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Text 1"
        android:textSize="25sp"
        android:textStyle="bold"
        android:textColor="@android:color/white"
        android:layout_alignParentTop="true"
        android:layout_toEndOf="@+id/imageView"
        />
    <TextView
        android:id="@+id/textView2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Text 2"
        android:textSize="15sp"
        android:textStyle="italic"
        android:textColor="@android:color/white"
        android:layout_below="@+id/textView1"
        android:layout_toEndOf="@+id/imageView"
        />
</RelativeLayout>
</androidx.cardview.widget.CardView>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="4dp"
        android:scrollbars="vertical" />
</RelativeLayout>
```

Додаток Г. Графічна частина

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН ПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ НАВЧАЛЬНИХ КУРСІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ ASP.NET І ANDROID

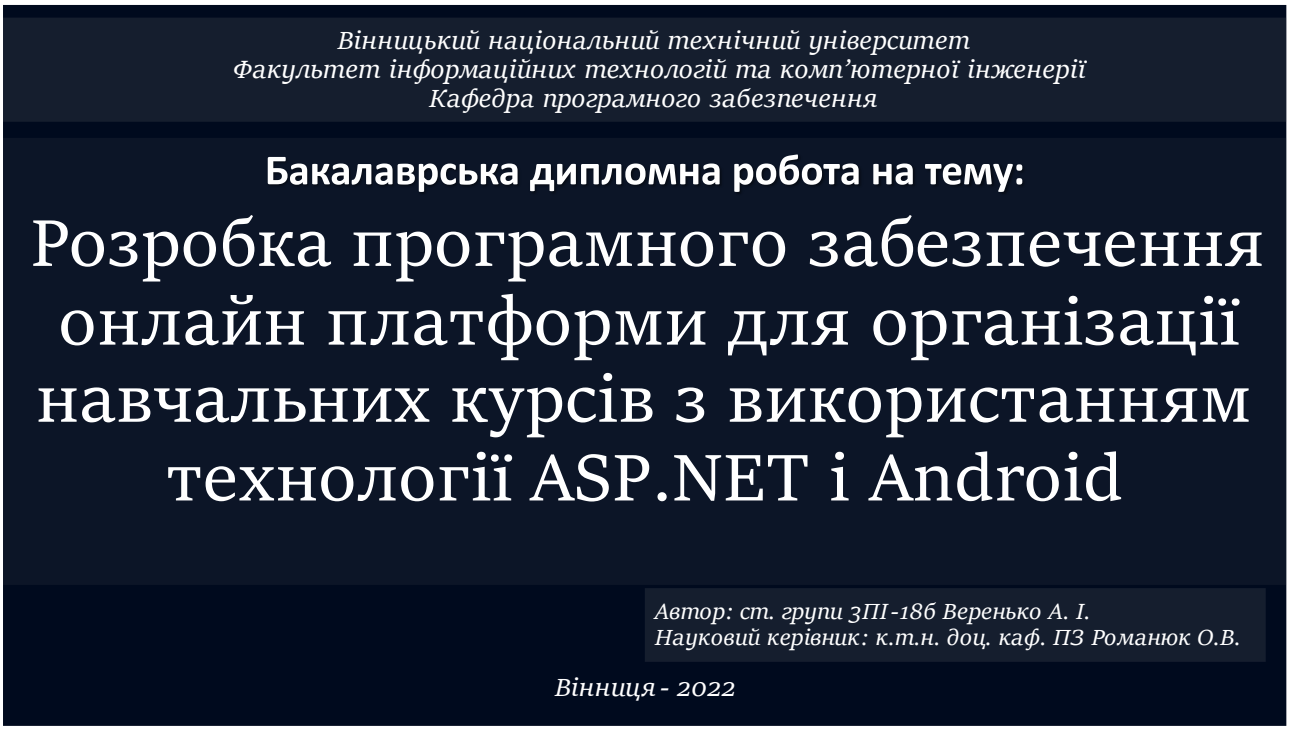

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

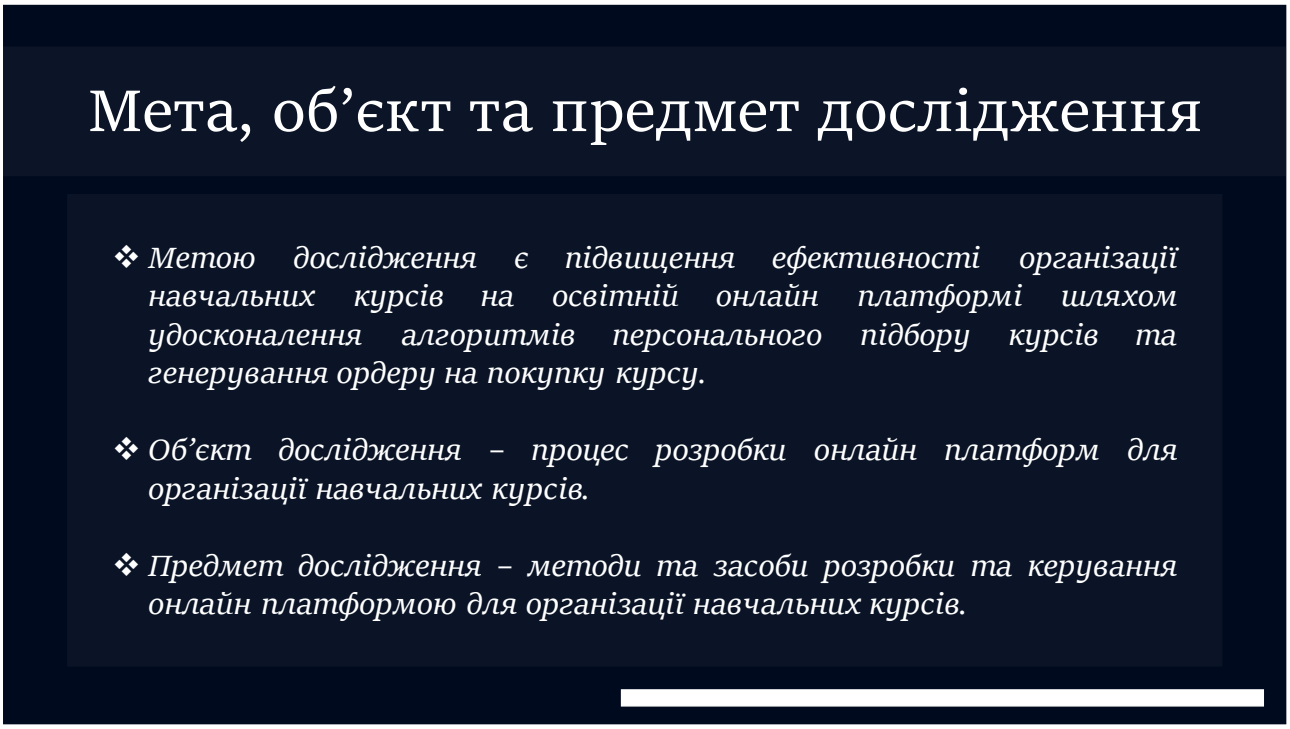

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

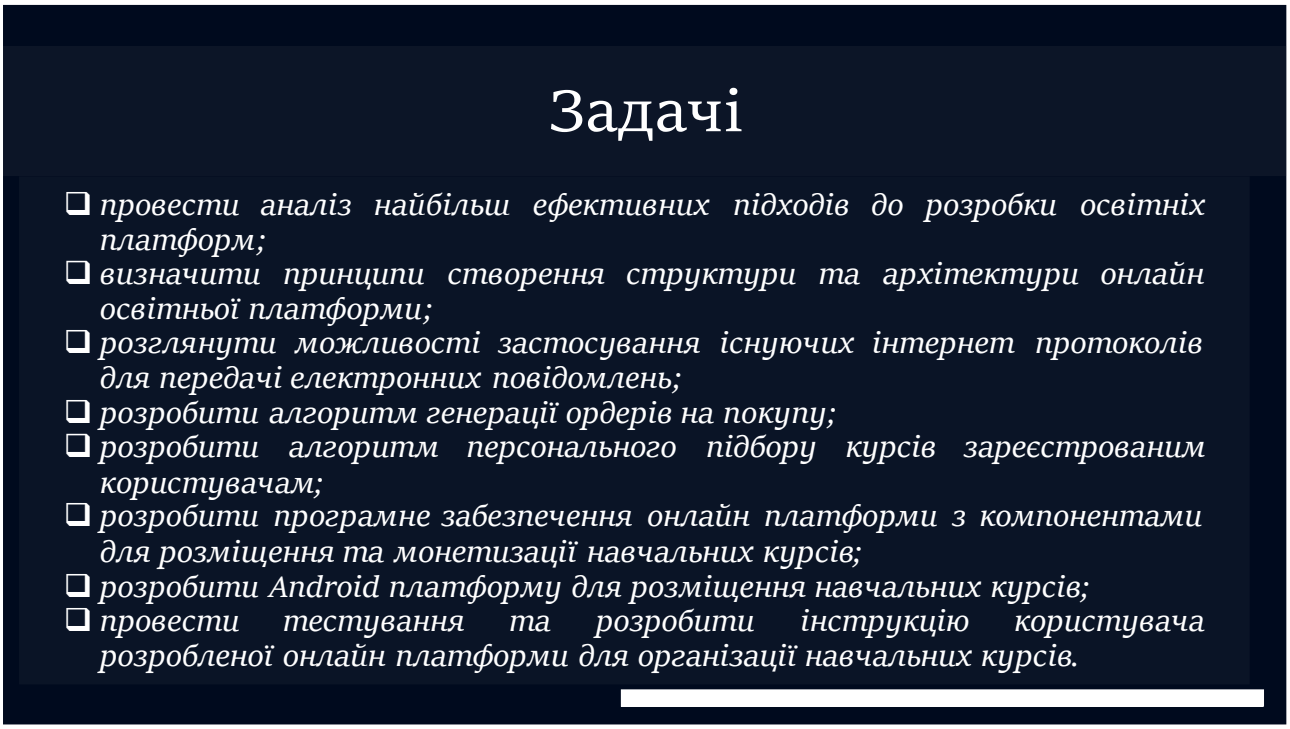

Рисунок Г.4 – Задачі дослідження

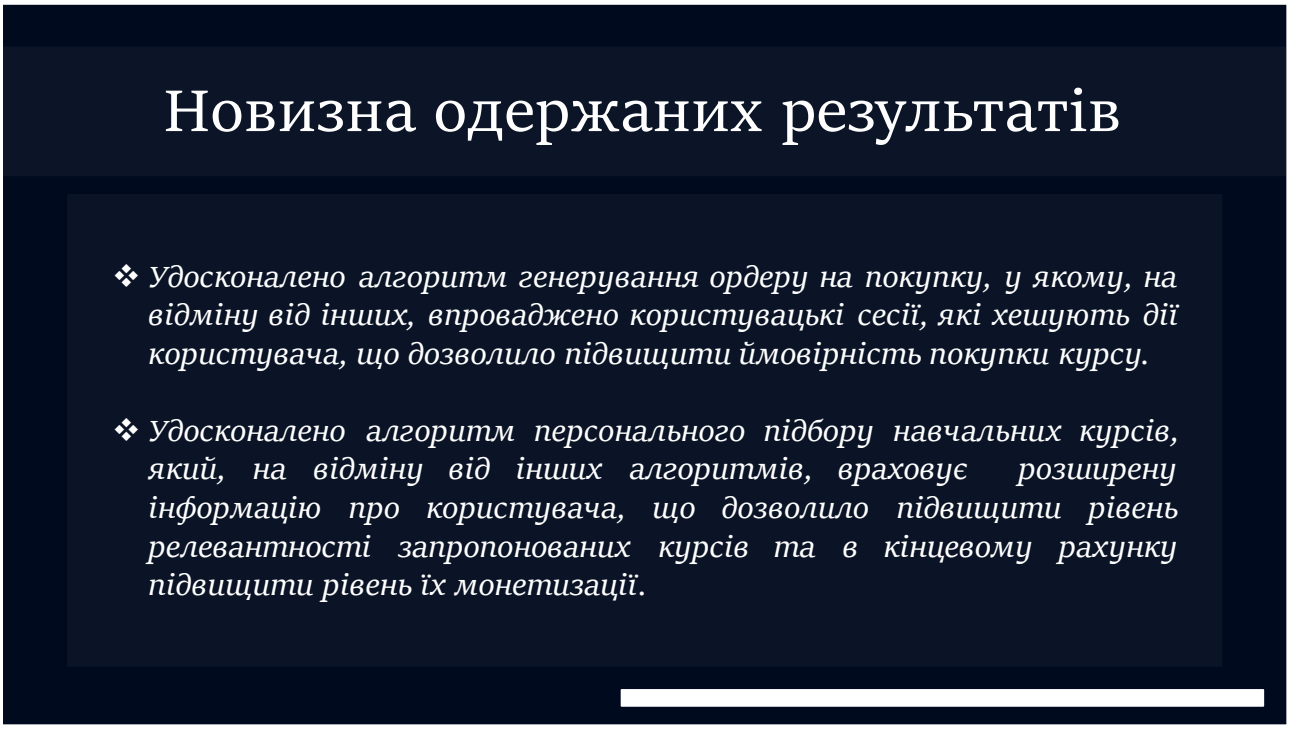

Рисунок Г.5 – Новизна одержаних результатів

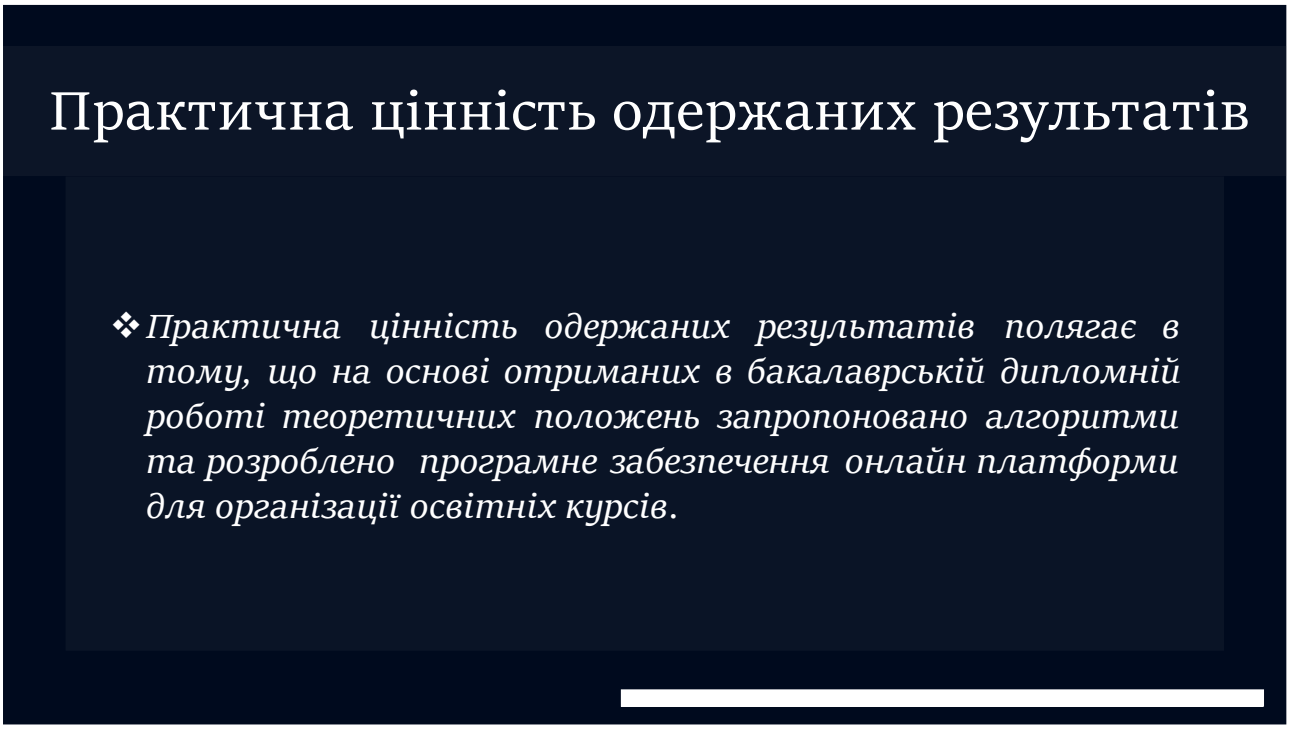

Рисунок Г.6 – Практична цінність одержаних результатів

Порівняльний аналіз аналогів

Критерій	Coursera	Udemy	Prometheus	AVDevU
Містить велику кількість навчальних курсів	1	1	1	1
Можливість розмістити курси під маловідомим брендом	0	1	1	1
Наявність комісії з продажів курсів на платформі	0	0	0	1
Можливість продавати офлайн навчальні курси	0	0	0	1
Наявнісь Android платформи	1	1	0	1
Підсумковий результат	2	3	2	5

Рисунок Г.7 – Порівняльний аналіз аналогів

Блок-схеми

Блок-схема алгоритму персонального підбору навчальних курсів

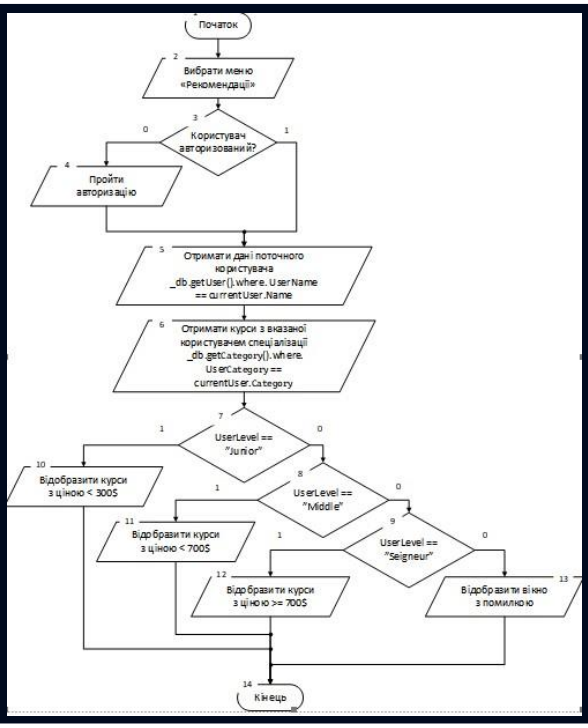

Рисунок Г.8 – Блок-схема алгоритму персонального підбору навчальних курсів

Рисунок Г.9 – Блок-схема алгоритму генерування ордеру на покупку навчальних курсів

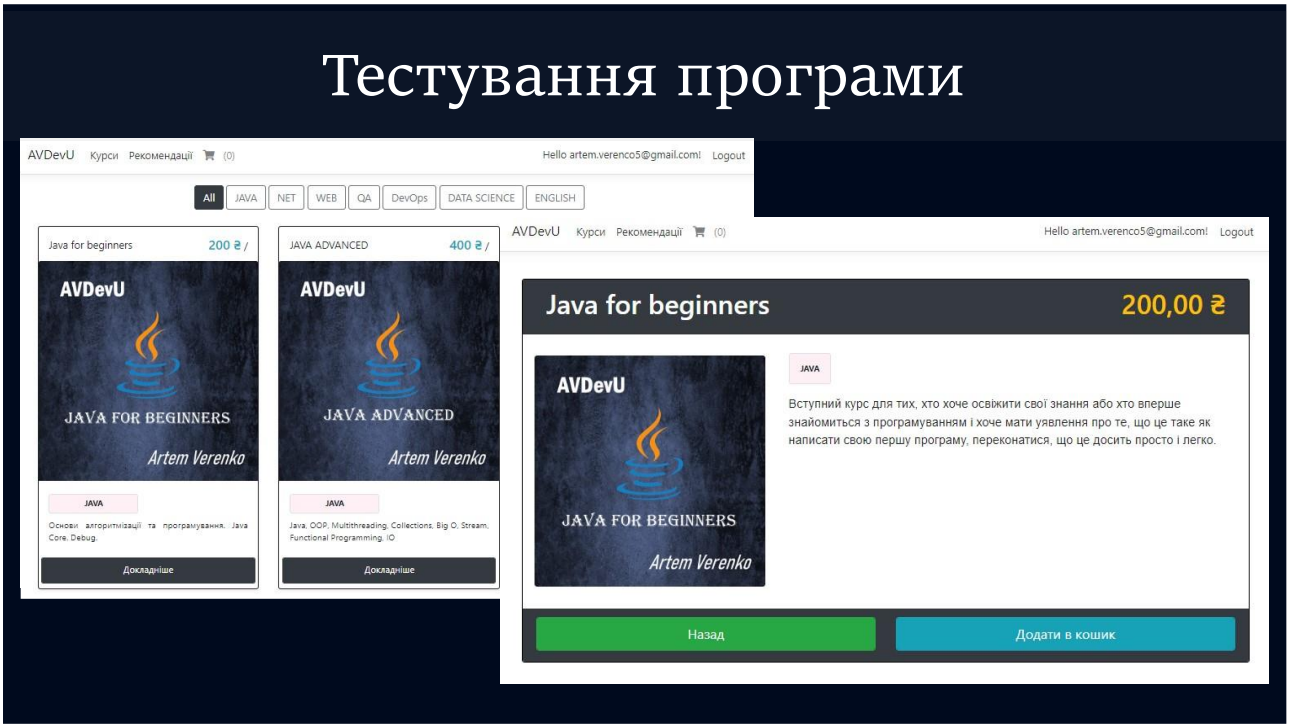

Рисунок Г.10 – Тестування програми (приклад курсів)

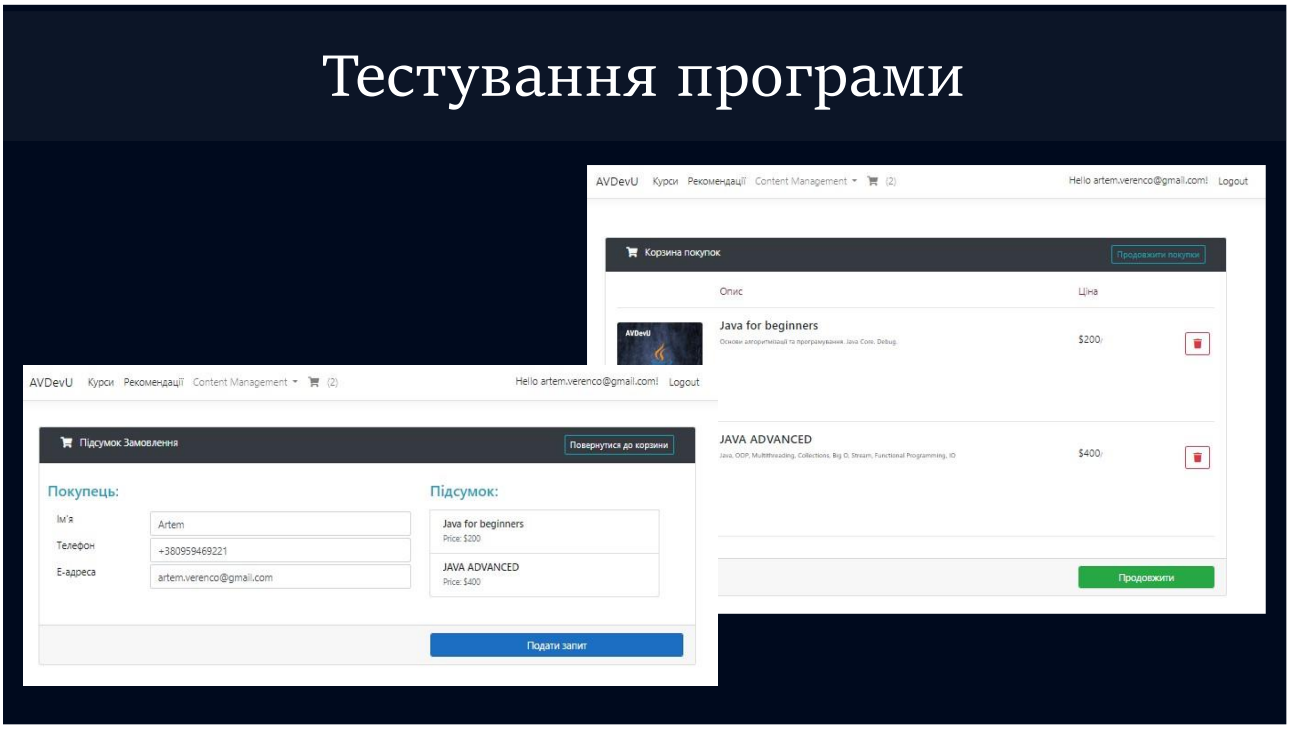

Рисунок Г.11 – Тестування програми (корзина покупок)

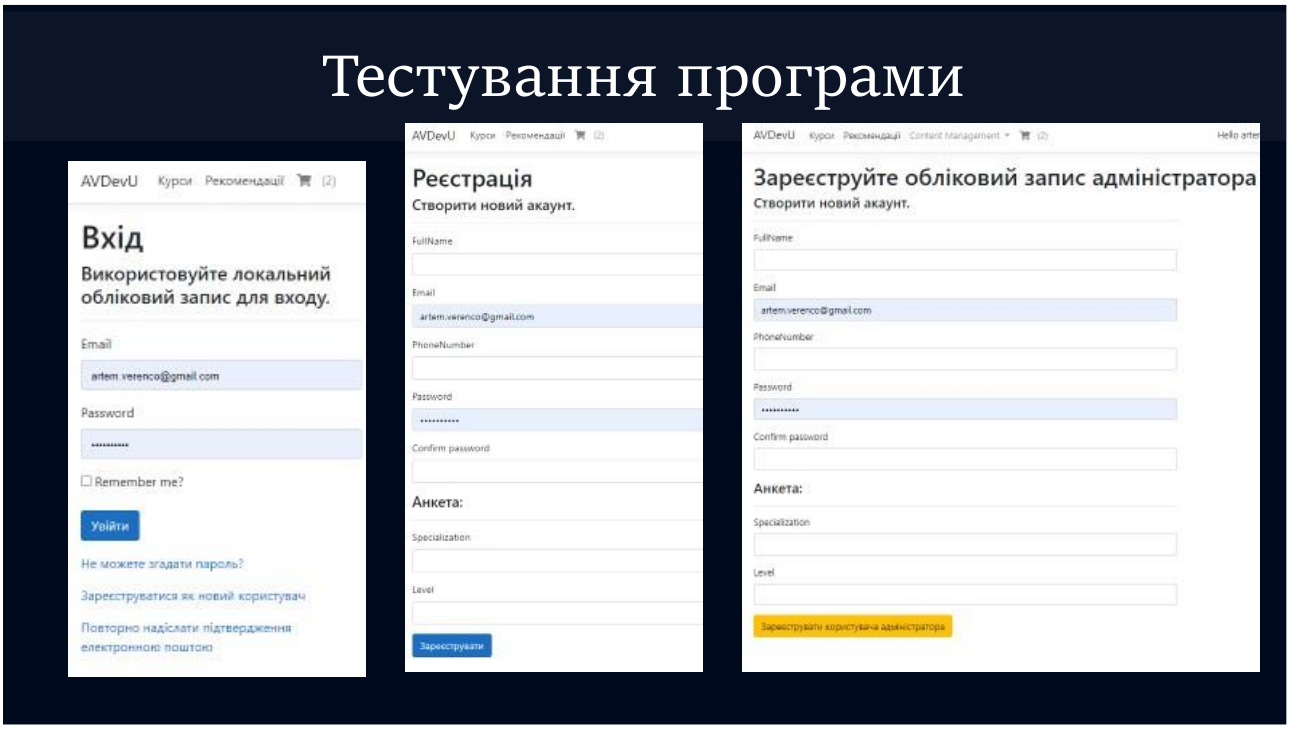

Рисунок Г.11 – Тестування програми (сторінки авторизації)

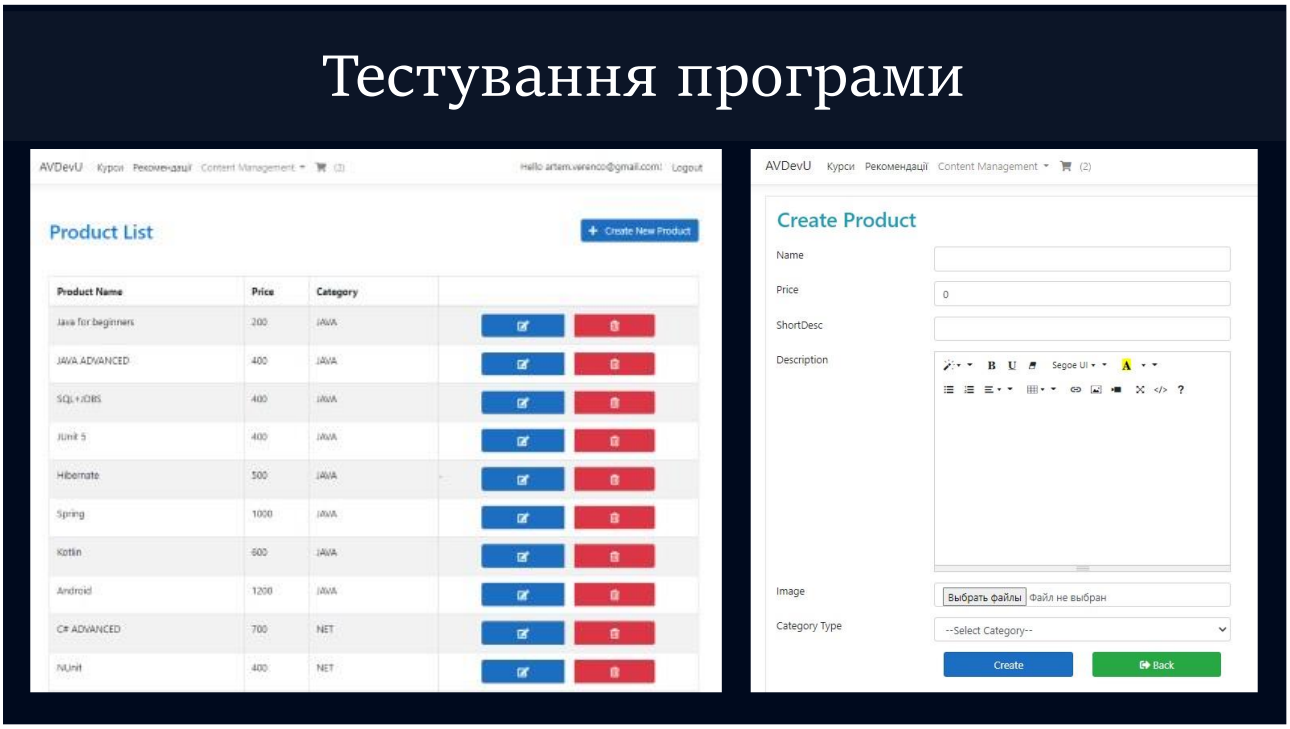

Рисунок Г.13 – Тестування програми (інструменти управління контентом)

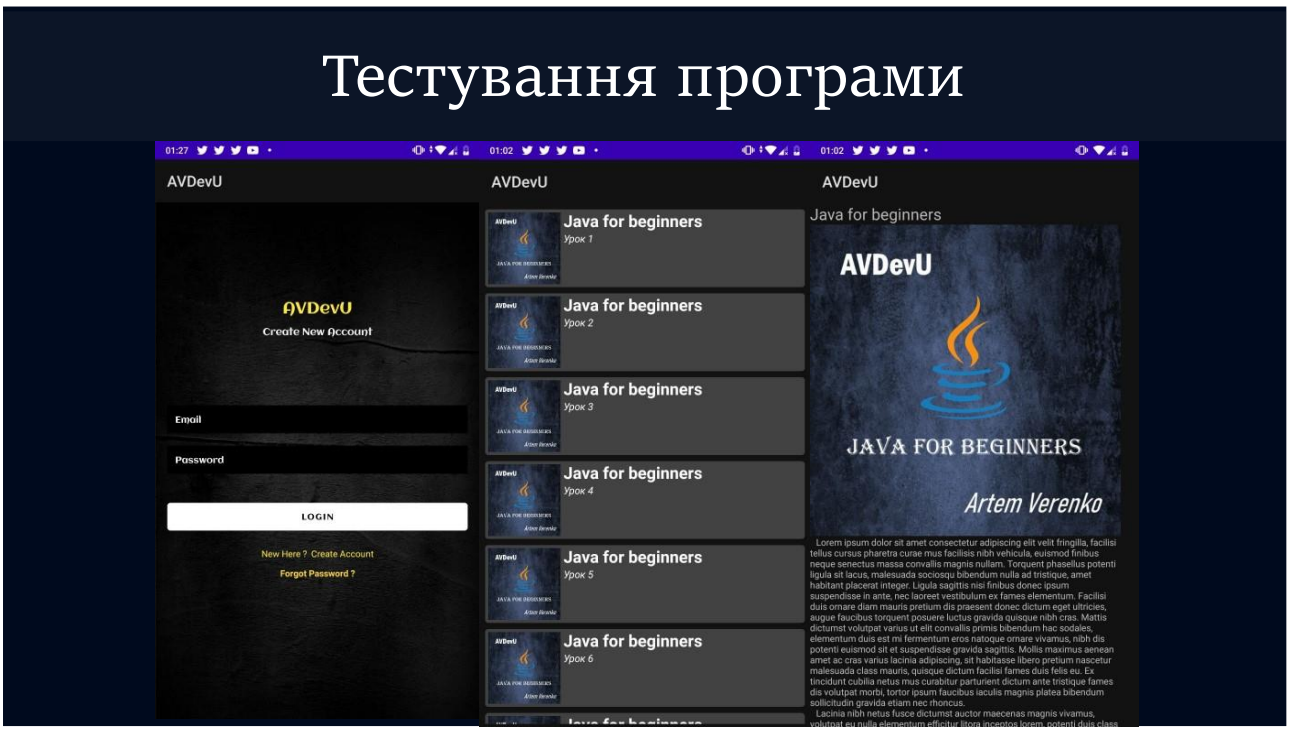

Рисунок Г.14 – Тестування програми (Android додаток)

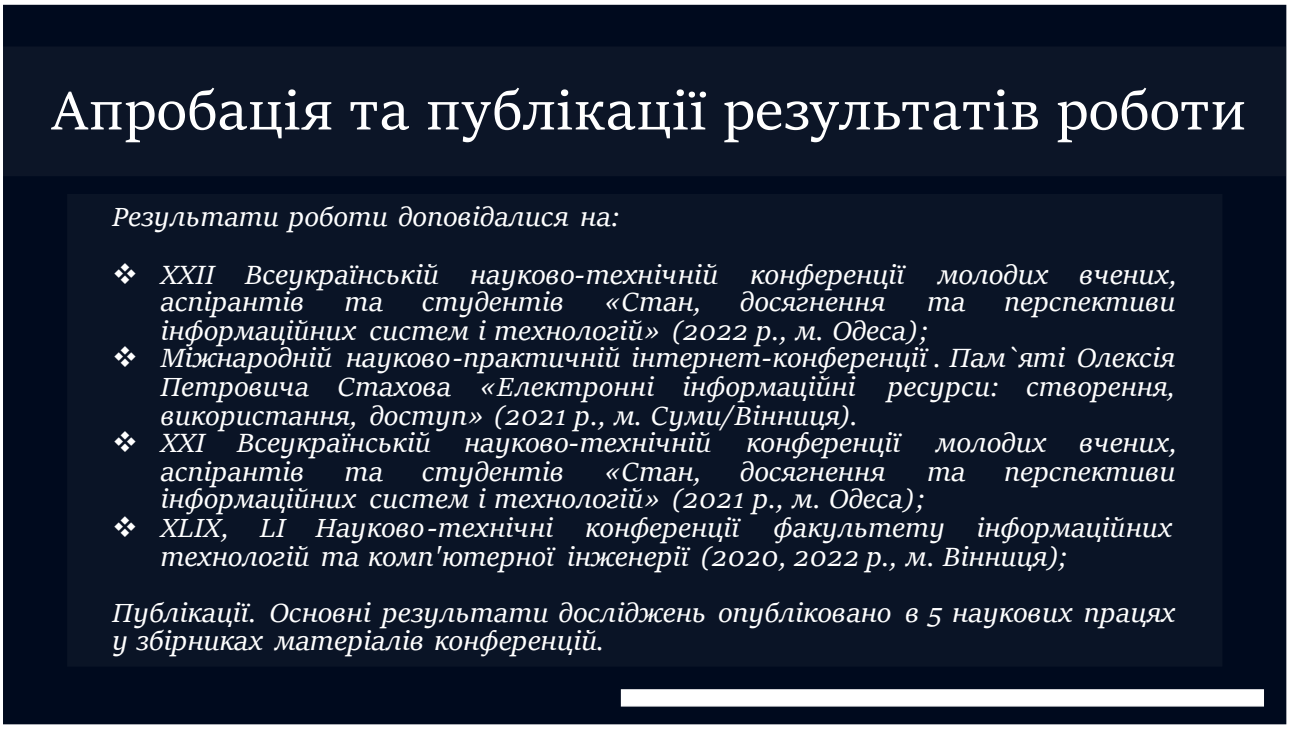

Рисунок Г.15 – Апробація та публікації результатів роботи

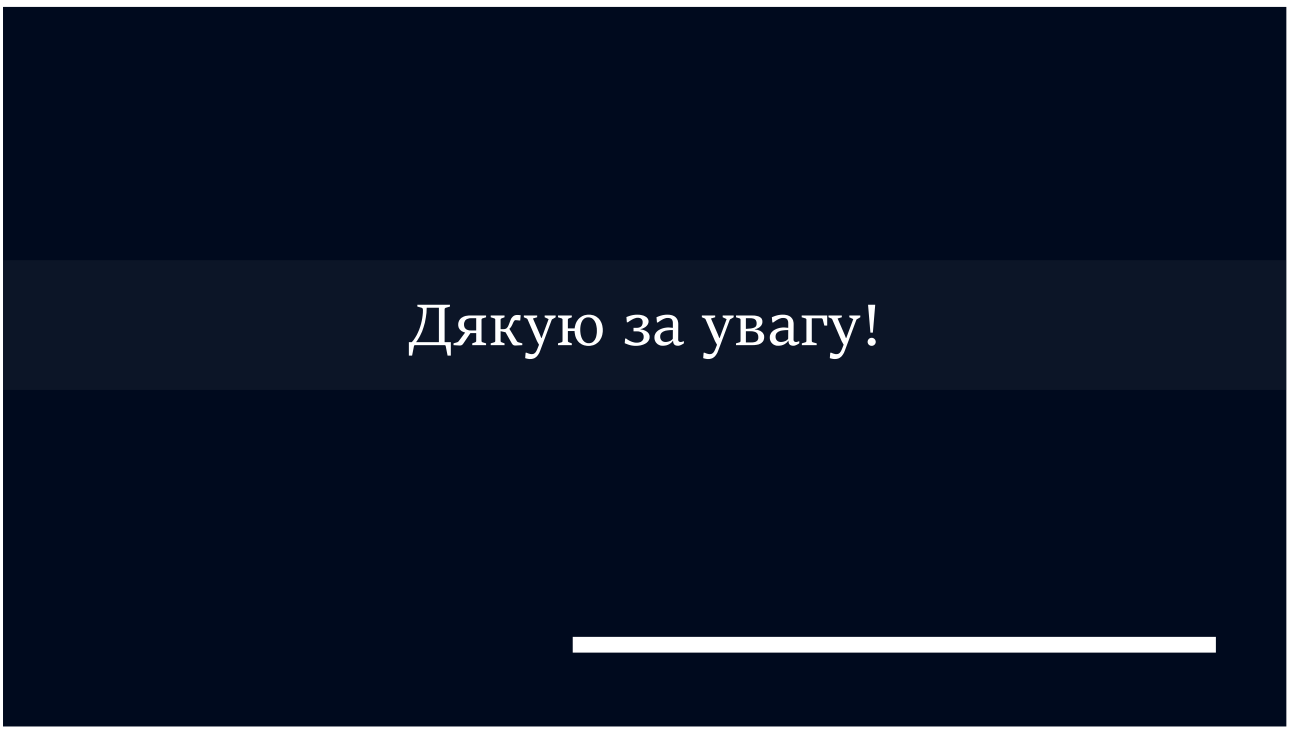

Рисунок Г.16 – Фінальний слайд