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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки»

Виконала: студентка 2-го курсу групи 1ПІ-20м спеціальності 121 «Інженерія програмного забезпечення»
Рекута Ю. С.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ
Войтко В. В.
(прізвище та ініціали)
Опонент: к.т.н., ст. викл. каф. КН
Озеранський В. С.

«____» _____2021 р.

Допущено до захисту

Завідувач кафедри ПЗ

проф. О.Н. Романюк

«____» _____ 2021 р.

Вінниця ВНТУ – 2021

Вінницький національний технічний університет

Факультет Інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Рівень вищої освіти ІІ-й (магістерський)

Галузь знань12 – інформаційні технології

Спеціальність 121 - інженерія програмного забезпечення

Освітньо-професійна програма – інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

_____О.Н. Романюк

"_13_" _вересня_2021 р.__

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Рекуті Юлії Сергіївні

1. Тема роботи: «Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки»
керівник роботи: к.т.н., доц. кафедри ПЗ Войтко Вікторія Володимирівна
затверджені наказом вищого навчального закладу від "24" вересня 2021 р. №_277
2. Строк подання студентом роботи__1 грудня 2021 р._____
3. Вихідні дані до роботи: середовище для створення та перегляду графічного контенту, багатокритеріальне оцінювання робіт; середовище розробки – Visual Studio Code, мова розробки – TypeScript; операційна система – Windows, розширення файлів jpg, png, psd, ai, gif, zip, rar.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз стану розробки автоматизованих систем для проведення конкурсів та олімпіад і постановка задач дослідження, розробка структури та сервісів роботи системи, розробка програми оцінювання, тестування програми, висновки, додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
мета, об'єкт та предмет дослідження, основні задачі, порівняння з аналогами, розробка методу оцінювння робіт, модель системи, розробка інтерфейсу,

розробка програмної реалізації, тестування системи, результати робо публікації, наукова новизна.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	к.т.н., доц. каф. ПЗ Войтко В. В.		
5	к.т.н. доц. каф ЕПВМ Ратушняк Ольга Георгіївна		

7. Дата видачі завдання__14 вересня 2021 р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз, вибір та актуальність розробки	15.09.21 – 30.10.21	Вик.
2	Аналіз вибраних аналогів	01.10.21 – 12.10.21	Вик.
3	Аналіз інформаційного забезпечення	12.10.21 – 17.10.21	Вик.
4	Розробка методу та моделей автоматизаціїроботи системи	17.10.21 – 28.10.21	Вик.
5	Розробка макетів графічного інтерфейсу	28.10.21 – 04.11.21	Вик.
6	Програмна реалізація модулів додатку	04.11.21 – 15.11.21	Вик.
7	Тестування роботи додатку	15.11.21 – 25.11.21	Вик.
8	Економічна частина	25.11.21 – 30.11.21	Вик.

Студент _____ **Рекута Ю. С.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Войтко В. В..**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК. 004.4:004.92

Рекута Ю. С. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 128 с.

У магістерській кваліфікаційній роботі розроблено клієнтську частину веб-додатку «Majestics», призначеного для розміщення та оцінювання робіт з міжнародних конкурсів з комп'ютерної графіки та веб-дизайну. Під час виконання магістерської кваліфікаційної роботи було проаналізовано предметну область, проаналізовано основні аналоги, виявлено їх головні недоліки та переваги, на основі яких було обрано основні функції для додатку «Majestics». Також створено гнучку архітектуру для подальшої інтеграції нового функціоналу.

Дістав подальшого розвитку метод багатокритеріального оцінювання конкурсних робіт, який дозволяє врахувати як кількісні, так і якісні критерії оцінювання, а також, дозволяє підвищити якість оцінювання робіт в процесі формування узагальненої оцінки.

За допомогою розроблених програмних засобів можна прискорити процес проведення конкурсу, зробити його надійнішим, забезпечити можливість зберігати дані про учасників та оцінити роботи інших користувачів. Розроблено структуру програмного інтерфейсу, створено макети дизайну та реалізовано графічну складову веб-додатку.

Обрано мову програмування з TypeSctipt платформою, для виконання якої фреймворк Node.js, а фреймворком для створення графічної і складової і реалізації її логіки було обрано Angular 8.

Ключові слова: веб-додаток, багатокритеріальне оцінювання, комп'ютерна графіка, веб-дизайн.

ABSTRACT

Rekuta Y. S. Master's degree in specialty 121 – software engineering, educational program – software engineering. Vinnytsia: VNTU, 2021

In the master's qualification work, the client part of the web application "Majestics" is designed to place and evaluate works from international competitions in computer graphics and web design. During the master's qualification work the subject area was analyzed, the main analogues were analyzed, their main disadvantages and advantages were revealed, on the basis of which the main functions for the application "Majestics" were selected. A flexible architect has also been created to further integrate the new functionality.

The method of multicriteria evaluation of competitive works was further developed, which allows to take into account both quantitative and qualitative evaluation criteria, and also allows to improve the quality of evaluation of works in the process of forming a generalized evaluation.

With the help of developed software you can speed up the process of the competition, make it more reliable, provide the ability to store data about participants and evaluate the work of other users. The structure of the software interface has been developed, design layouts have been created and the graphic component of the web application has been implemented.

The programming language with the TypeSctipt platform was chosen, for which the Node.js framework was executed, and Angular 8 was chosen as the framework for creating the graphic and component and implementing its logic.

Keywords: web application, multicriteria evaluation, computer graphics, web design.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Використання веб-додатків помітно зростає у всіх сферах людського життя, зокрема це стосується сфери навчання та участі в різноманітних конкурсах [1]. Це є величезною можливістю і для багатьох творчих людей, які прагнуть показати свої роботи на конкурсах величезній аудиторії. При цьому вони не використовують багато коштів на квитки, щоб поїхати в інше місто (або навіть країну), не мають сумнівів у тому, що їх робота загубиться у дорозі, не бояться запізнитися через різні непередбачувані ситуації. Але навіть участь в різних конкурсах онлайн не дає гарантії того, все проходитиме чесно. Нажаль, існує багато людей, які переконані в тому, що їх робота буде кимось завантажена, потрапить в інші організації, буде використовуватись без їх відома, особливо якщо це стосується графічного та веб-дизайну.

Більшість існуючих аналогів мають проблему з доступністю для користувачів, і аби стати учасником конкурсу, потрібно оплатити участь в певному етапі [2-5]. Також аналоги розроблені для олімпіад лише одного типу, що робить користування сайтом незручним через недостачунеобхідних елементів управління. Проблемою й є те, що практично всі існуючі додатки схожого типу використовують сторонні сервіси для передачі даних, системи не дають можливість прозоро бачити, на базі якої інформації і як саме була сформована оцінка.

Тому актуальною є розробка автоматизованої веб-системи проведення конкурсів з графічного дизайну для того, аби досягнути зручності в

користуванні, простоти, надійності та економії часу. Така розробка матиме низку переваг:

- надання можливості участі в конкурсах з графічного дизайну будь-якій людині з будь-якої точки світу;
- одночасне проведення багатьох різних конкурсів для учасників різної вікової категорії (від школярів до студентів вищих навчальних закладів);
- впевненість у тому, що голосування проходитиме чесно, ніхто з учасників чи сторонніх осіб не зможе підробити результати роботи, навіть журі не матимуть змоги переглянути та відредагувати оцінку, поставлену іншими учасниками;
- результати та роботи учасників є надійно збережені в базі даних та на жорстких дисках серверів системи;
- дані є надійно захищеними, ніякі сторонні особи та організації не матимуть можливості використати роботи учасників у незаконних цілях, адже вони напряму потрапляють у базу даних конкурсів за допомогою захищених каналів передачі.

Більшість існуючих наразі аналогів використовують застарілі та ненадійні сервіси та технології автоматизації, а також розроблені для олімпіад різного типу, що робить користування сайтом незручним через надлишковість інтерфейсу чи недостачу необхідних елементів управління.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та задачі дослідження. Метою роботи розширення можливостей оцінювання графічних робіт з використанням кількісних і якісних критеріїв оцінювання, що дозволить більш глибоко оцінити характеристики роботи.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

- провести аналіз типів веб-додатків та методів їх створення;
- розробити інтерфейс програмного продукту;

- розробити модель автоматизованої системи оцінювання з використанням різнотипної системи оцінки;
- розробити багатокритеріальний метод формування загальної оцінки роботи з урахуванням експертних оцінок кількох членів журі за системою кількісних і якісних критеріїв оцінювання за 9-ти бальною шкалою Сааті;
- створити взаємодію між веб-додатком та серверною частиною;
- розробити інформаційне наповнення системи;
- провести тестування програмного продукту.

Об’єктом дослідження є процес багатокритеріального оцінювання варіантів та процес створення клієнтської частини веб системи.

Предметом дослідження є методи і засоби програмування та верстки веб-ресурсів з використанням мов програмування та розмітки TypeScript, HTML, CSS.

Методи дослідження. У процесі досліджень використовувались:

- методи багатокритеріального оцінювання варіантів для розробки модуля оцінювання робіт;
- методи обробки графічної інформації для розміщення графічних зображень у зручному форматі на сайті;
- методи побудови клієнт-серверної архітектури для забезпечення зв’язку з сервером.

Наукова новизна отриманих результатів.

1. Дістав подальшого розвитку метод багатокритеріального оцінювання конкурсних робіт, який, на відміну від існуючих, дозволяє врахувати як кількісні, так і якісні критерії оцінювання, з використанням 9-ти бальної шкали Сааті, що дозволяє підвищити якість оцінювання робіт в процесі формування узагальненої оцінки.

2. Дістала подальшого розвитку модель автоматизованої системи оцінювання конкурсних графічних робіт, яка, на відміну від існуючих, орієнтована на формування комплексного критерію оцінювання з урахуванням

якісних та кількісних часткових критеріїв, що дозволяє розширити діапазон показників, які впливають на результат експертної оцінки.

Практичне значення одержаних результатів. Практична цінність одержаних результатів полягає в наступному:

- розроблено та наведено сервіси та інструменти для взаємодії клієнтської та серверної частини;
- розроблено алгоритми й засоби реалізації клієнтської частини веб-системи для участі в міжнародних конкурсах і олімпіадах та оцінювання графічних робіт учасників.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У наукових роботах, опублікованих у співавторстві, автору належать: модель роботи розробленої веб-системи у XLIX Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії [6], структура програмного інтерфейсу у «Молодь у світі сучасних технологій» (МССТ-2020) [7], метод багатокритеріального оцінювання конкурсних робіт у міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ. Пам’яті Олексія Петровича Стахова» [8].

Апробація матеріалів магістерської кваліфікаційної роботи. Результати роботи доповідалися на:

XLIX науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії Вінницького національного технічного університету (2020), Міжнародній науково-практичній конференції молодих вчених та студентів «Молодь у світі сучасних технологій» (МССТ-2020), Міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ. Пам’яті Олексія Петровича Стахова» (2021).

Публікації. Результати роботи опубліковані в трьох наукових працях – тезах-доповідях на XLIX науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (Вінниця, 2020), IX Міжнародній науково-практичній конференції молодих вчених та студентів

«Молодь у світі сучасних технологій» (Херсон, 2020), Міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ. Пам’яті Олексія Петровича Стахова» (Вінниця, 2021) [6-8].

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 26 найменувань, 5 додатки. Робота містить 48 ілюстрацій, 7 таблиць.

1 АНАЛІЗ СТАНУ РОЗРОБКИ АВТОМАТИЗОВАНИХ СИСТЕМ ДЛЯ ПРОВЕДЕННЯ КОНКУРСІВ ТА ОЛІМПІАД І ПОСТАНОВКА ЗАДАЧ РОБОТИ

1.1 Аналіз стану розробки систем для проведення конкурсів та олімпіад

На сьогоднішній день питання автоматизації проведення міжнародних конкурсів досить актуальне. Це тому, що існує безліч людей, які хочуть брати участь в конкурсах міжнародного масштабу, проте не мають змоги через низку причин, таких як місце проживання, нестача матеріальних ресурсів або ж відсутність часу.

Автоматизація організаційних процесів проведення конкурсів має низку переваг. Вона полегшить і прискорить сам процес проведення конкурсу та голосування, зробить його надійнішим, забезпечить можливість зберігати дані про учасників, а також це є чудовою можливістю оцінити роботи інших користувачів за поданими критеріями.

Зазвичай передача даних про учасників та їх роботи здійснюється за допомогою приватних SMTP серверів або шляхом використання сторонніх сервісів для зберігання даних і файлів [9]. Це є не надійним шляхом передачі даних, адже по-перше, існує можливість, що лист просто не дійде до адресата, по-друге, проблема полягає в тому, що приватна інформація та конкурсні роботи передаються третім лицям, що є порушенням законів в випадку не затвердженого користувачем погодження, але мало наважиться передавати свої дані в руки третім лицям.

Є декілька аналогів такої системи, проте вони мають деякі недоліки, які обмежують точність об’єктивної оцінки роботи учасника конкурсу:

- відсутність можливості введення додаткових критеріїв оцінювання;
- неможливість коригування відсоткового впливу оцінок користувачів різних типів на загальні оцінку;
- неможливість визначення дублікатів для оцінок анонімними користувачами.

Багато систем такого типу використовують застарілі технології під час розробки. Це небезпечно тим, що дані не перебувають у цілковитій конфіденційності, адже існує безліч способів взлому.

Ще однією з проблем є використання звичайних жорстких дисків для зберігання робіт та даних користувачів, що є не надійним захистом, адже коли дані знаходяться фізично тільки в одному місці, вони перебувають під ризиком перманентної втрати.

Звісно існують різні види додатків і різні типи передачі даних та їх зберігання. Багато з них використовують VPS або сервіси, які надають сервіси хостингу, що знову ж таки не дає стільки можливостей скільки дають хмарні сервіси хостингу. Вони не прив'язані до локального місця знаходження серверів та дають значні можливості для масштабування додатку для будь-яких країн з часом роботи в 99.99%.

1.2 Порівняльний аналіз аналогів

Сьогодні існує декілька аналогів зі схожим функціоналом, проте вони мають принципові відмінності. Розглянемо найпопулярніші з них.

«Awwwards» [2] (рис. 1.1) – конкурс сайтів та мобільних додатків, який проводиться щорічно, починаючи з 2017-го року. Спочатку триває народне голосування, в якому може взяти участь кожен, після чого експертна комісія та члени журі оцінюють роботи.

Призові місця присуджуються щодня як веб-дизайнерам за зручність використання сайту, креативний підхід, так і розробникам за шикарні коди. Ті, хто переміг у щоденному конкурсі (відбираються 8 учасників із найвищими балами), беруть участь у щомісячному, а переможці щомісячного конкурсу змагаються за приз «Кращий сайт року».

За умовами конкурсу, на розгляд приймаються лише реалізовані проекти. Оцінка враховує 4 напрямки: дизайн - 40%; креативність - 30%; юзабіліті - 20%; контент – 10%.

Однак існує низка недоліків, які обмежують користувачів:

– участь можуть брати лише розробники сайтів або мобільних додатків (графічні роботи не приймаються);

– аби стати учасником, потрібно оплатити участь на даному етапі (вартість участі – від 40 до 150 євро.)

– до участі допускаються роботи осіб, старше 16 років.

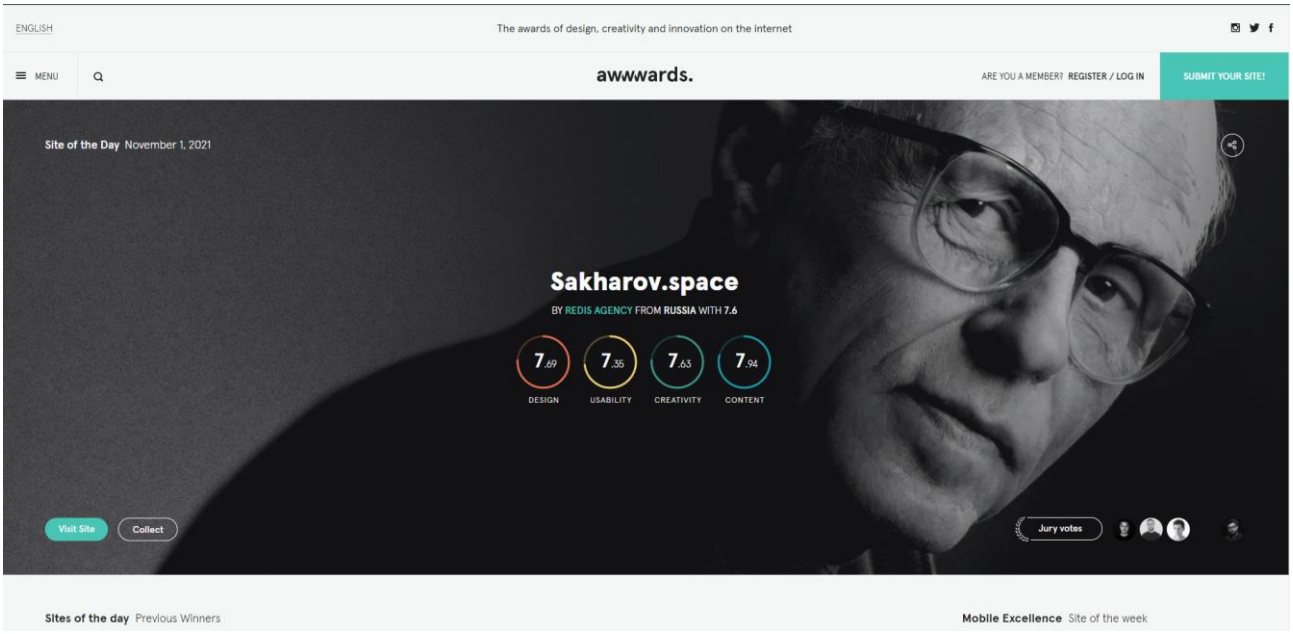


Рисунок 1.1 – Система «Awwwards»

«Webby Awards» [3] (рис. 1.2) – це міжнародна інтернет-премія для дизайнерів та розробників сайтів, мобільних програм, а також фахівців з реклами. Неформальна назва цього конкурсу – Інтернет-Оскар.

Критерії оцінювання: опрацьована структура та навігація сайту; сучасний дизайн; функціональність сайту; успішність проекту.

Вартість участі залежить від обраної категорії в середньому 425 доларів. На деякі категорії передбачено знижену ставку, наприклад: сайти-блоги про культуру; інтернет-мистецтво; особистий сайт-блог; захист навколишнього середовища. За цими напрямками ціна участі складає 150 доларів. Для студентів вартість подання заявки – 45 доларів.

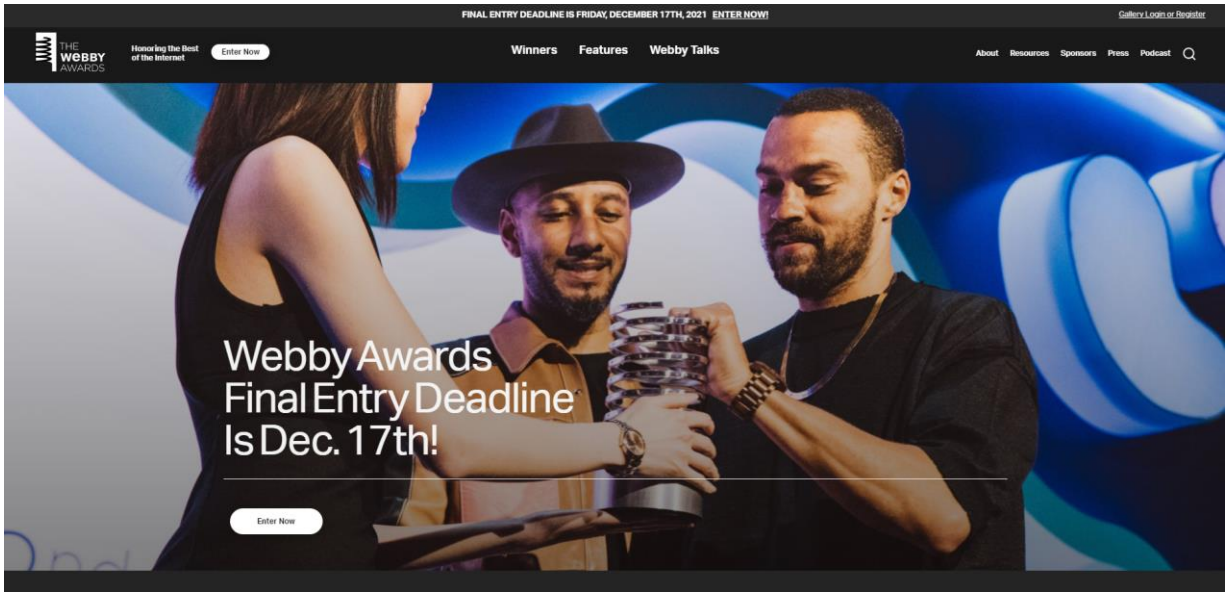


Рисунок 1.2 – Система «Webby Awards»

«Favourite Website Awards — FWA» [4] (рис. 1.3) – це конкурс найкреативніших та передових робіт з усього світу. Подати заявку на участь можна щодня, результат буде відомий приблизно через 14 днів. На конкурс приймаються: сайти, створені на будь-яких платформах; онлайн ігри; мобільні додатки; відеоролики та анімація; інші креативні роботи.

Вартість участі – 70,5 фунтів стерлінгів.

За підсумками конкурсу обирають сайт дня (SOTD), сайт місяця (SOTM) та сайт року (SOTY). З учасників, номінованих на сайт дня, обирають сайт місяця, а найкращий із 12 сайтів місяця стає сайтом року.

Сайт оцінюється не за особистими відчуттями суддів, а за чек-листом:

- 40 балів - непрацюючий сайт, бите посилання;
- 60 балів – звичайний середньостатистичний сайт;
- До 80 балів – круто, але до FWA не дотягує;
- Від 85 балів - топовий сайт;
- Від 95 - FAW.

«European Design Awards» [5] (рис. 1.4) – конкурс проводиться за 8 категоріями: брендинг; упакування; внутрішнє виробництво; публікації; діджитал (тут 2 номінації — найкращий сайт та найкращий мобільний додаток); просування; ілюстрації; різне.

Вартість участі – 140 євро. Є знижка за виставлення на конкурс кількох робіт — якщо у вас подано понад 5 заявок, вартість кожної становитиме 112 євро. Участь для студентів – 40 євро.

Проект оцінюють за такими критеріями: якість виконання; креативність; юзабіліті; технічне виконання.

Місто проведення заходу щороку змінюється. Переможця запрошують до однієї з європейських країн для вручення призу.

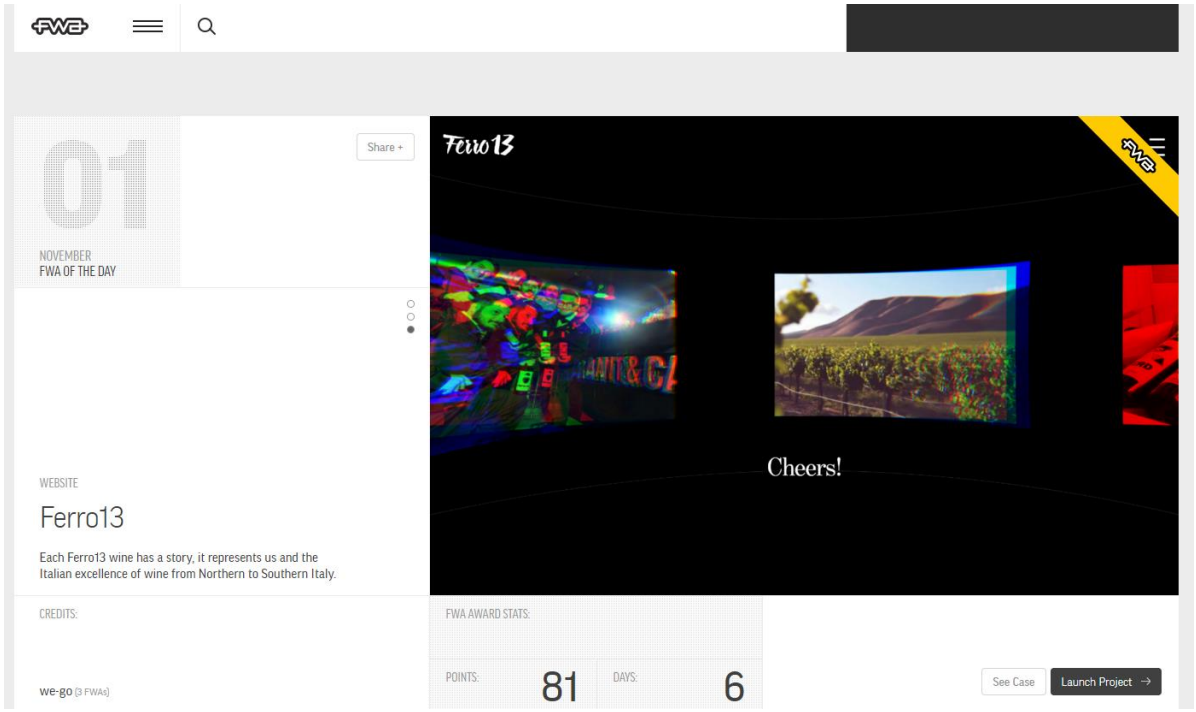


Рисунок 1.3 – Система «FWA»

Рисунок 1.4 – Система «European Design Awards»

Аналізуючи вищеописані додатки, можна виокремити для кожного з них низку недоліків. Порівняння їх з розроблюваною системою описано в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	«awwwards»	«Webby Awards»	« FWA»	«ED Awards»	«Majestics»
Умова	реальний робочий проєкт	реальний робочий проєкт	креативний проєкт	реальний робочий проект, гарне юзабіліті, креативний дизайн	креативний проєкт
Доступність для усіх користувачів	+	-	-	-	+
Надійність	-	+	+	+	+
Народне голосування	-	-	-	-	+
Критерії оцінювання робіт	+	+	-	+	+

1.3 Аналіз методів і засобів реалізації поставленої задачі

Існує безліч способів вирішити поставлені задачі. Для реалізації проведення конкурсів було прийнято рішення розробити його десктопну версію.

Веб додаток – це рішення, в основі якого лежить взаємодія браузера і веб-сервера. Такі додатки є кроссплатформенними сервісами, доступними з будь-якого сучасного пристрою, і не прив'язані до архітектури мережі: ви можете

отримати доступ до них з локального комп'ютера або зі смартфона на іншому кінці світу за зручним для вас протоколу, наприклад, найбільш швидкому або зашифрованого.

На відміну від веб-сайту, веб-додаток – це повноцінна програма, доступ до якої користувач отримує через інтернет, тобто вона не вимагає установки на пристрій. Веб-додаток інтерактивний і дозволяє користувачам взаємодіяти з різними елементами: наприклад, залишити заявку на покупку товару, оформити покупку авіаквитка або прокоментувати пост друга.

Веб-додатки можна класифікувати по-різному: залежно від їхнього функціоналу та призначення. Є три основні шаблони побудови сайтів:

- MPA (multi-page application): багатосторінковий додаток, який надсилає запит на сервер і повністю оновлює сторінку, коли з нею відбувається дія;
- SPA (single-page application): односторінковий додаток, що містить HTML-сторінку, яка динамічно оновлюється залежно від дій користувача без повного перезавантаження;
- PWA (progressive web application): програма, яку користувач встановлює та може використовувати у режимі офлайн.

Переваги веб-додатків [10]:

Економія – у ході розробки не доведеться створювати окремі програми для різних операційних систем – вони працюють однаково в будь-яких браузерах: Internet Explorer, Opera, Safari, Google Chrome і т.д.

Безпека – веб-система має єдину точку входу, тому можна централізовано налаштувати її захист. Крім того, дані користувачів зберігаються у хмарі, тому при пошкодженні жорсткого диска інформація вціліє.

Доступ з різних пристроїв – користувач може взаємодіяти з веб-програмою через комп'ютер, смартфон, планшет і т. д. Головне - доступ до інтернету.

Відсутність клієнтського ПЗ – користувачам не потрібно нічого завантажувати і, що важливіше, оновлювати. Можна міняти інтерфейс клієнта,

а оновлення до останньої версії відбудеться при черговому завантаженні сторінки.

Масштабованість – навіть якщо навантаження на систему збільшиться, все рівно не доведеться збільшувати потужність клієнтських місць. Зазвичай веб-програми можуть обробляти більше даних лише силами апаратних ресурсів, тому доведеться переписувати код і змінювати архітектуру.

Розробка веб-додатків включає кілька кроків і може бути досить довгим і трудомістким процесом. Було виокремлено основні етапи веб розробки [11]:

1. Постановка цілей та завдань програми. Визначити, навіщо та який продукт потрібен, вимоги, яким має задовольняти, розробляється загальна концепція. Це включає як основні функції, а й глобальні цілі.

2. Опрацювання технічного завдання. Максимально точно описати вимоги до фінального продукту.

3. Розробка структури сайту. Включає все, що стосується його змісту та інформаційної стратегії, що визначає, як має бути організована подача інформації, щоб майбутні відвідувачі сайту могли швидко та легко її знайти. Першочерговим завданням на даному етапі є створення карти сайту, що відображатиме взаємозв'язки типових сторінок та їх найбільш значущі функціональні можливості.

4. Прототипування. Створення сайту та прикладів майбутніх веб-сторінок.

5. Створення макету дизайну веб-додатку. На цій стадії створюються всі елементи веб-дизайну відповідно до стилю подання інформації та загальної концепції. Головним при дизайні сайту є вміння розробити графічні об'єкти, які швидко завантажувалися і добре виглядали.

6. Розробка та верстка. Процес поділяється на дві частини: backend та frontend. Backend-частина включає внутрішні процеси сайту, такі як синхронізація пристроїв або авторизація користувачів. Frontend-частина - це зовнішній вигляд продукту, тобто те, як кнопки реагують на натискання і як з'являються вікна, що спливають. Після цього етапу програма вже практично готова до використання.

7. Тестування. Перевірка на коректність дій і чи всі об'єкти відображаються належним чином.

Отже, розробка веб-додатків – це комплексний багатокроковий процес, що вимагає знання безлічі різних технологій та мов програмування, вміння працювати з базами даних, використовувати безліч інструментальних засобів та програмних пакетів.

1.4 Вибір моделі життєвого циклу для розробки додатку

Життєвий цикл ПЗ - це стадії, які проходить програмний продукт від появи ідеї до її реалізації в коді, імплементації в бізнес та подальшої підтримки. Моделі життєвого циклу багато в чому визначають і методологію розробки ПЗ [12].

1. Waterfall (каскадна модель). Основна суть у тому, що етапи залежать один від одного і наступний починається, коли закінчено попередній, утворюючи таким чином поступальний рух уперед (рисунок 1.4). Паралелізм етапів у каскадній моделі, хоч і обмежений, але можливий для абсолютно незалежних робіт. При цьому інтеграція паралельних шматків все одно відбувається на наступному етапі, а не в рамках одного. Команди різних етапів між собою не спілкуються, кожна команда відповідає чітко за свій етап. Недоліками цієї моделі є отримання результату проходження всіх етапів і складність виявлення помилок. Повертатись назад важко. Не зрозуміло, що повертати: якщо стався збій на якомусь етапі, його наслідки видно лише наприкінці.

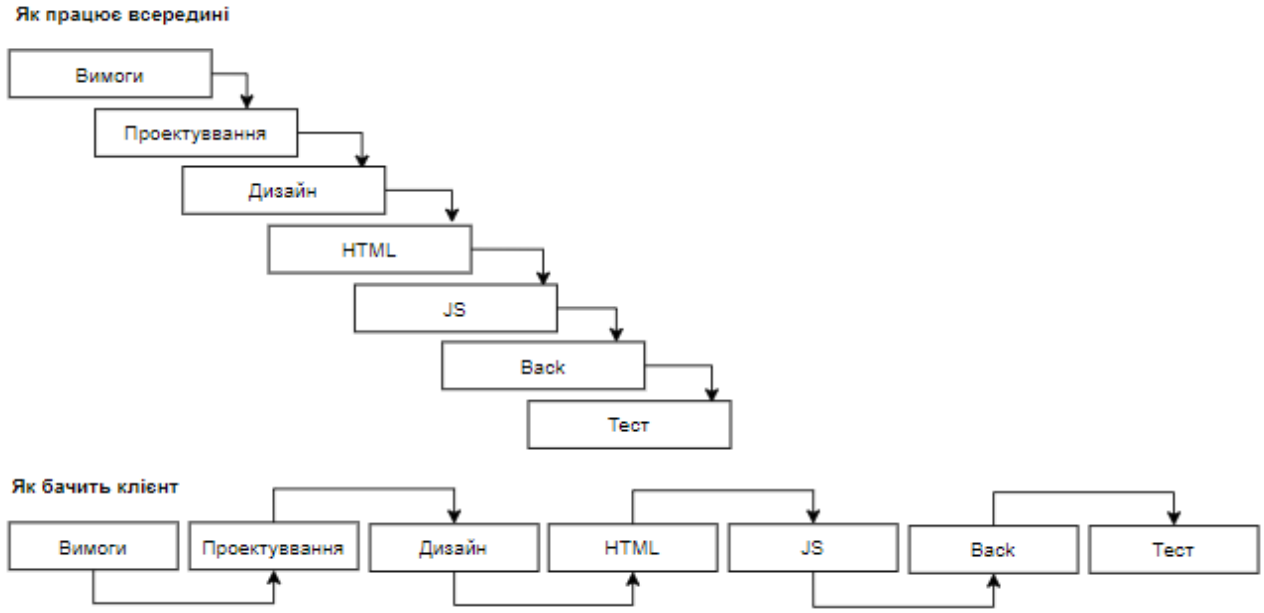


Рисунок 1.4 – Каскадна модель

Однак уявлення про простоту каскадної моделі є ілюзорним. Воно з'являється через обмежене бачення клієнтом всього процесу, адже дана модель не передбачає залучення замовника до деталей процесів розробки, і демонструє зрозумілий та кінцевий результат роботи тільки на контрольних точках та наприкінці проекту.

Насправді каскадну модель не можна назвати простою, практично нею складно управляти. Внесення замовником значних змін у процесі розробки по waterfall або спрацювання серйозних не передбачених проектом ризиків несуть руйнівний характер для всього процесу - модель доводиться перебудовувати, перепланувати графіки.

2. Ітераційна модель передбачає розбиття проекту на частини (етапи, ітерації) та проходження етапів життєвого циклу на кожному з них. Кожен етап є закінченим сам собою, сукупність етапів формує кінцевий результат. З кожним етапом розробка наближається до кінцевого бажаного результату або уточнюються вимоги до результату в процесі розробки, і відповідно у будь-який момент поточна ітерація може бути останньою або черговою на шляху до завершення (рисунок 1.5). Цей підхід дозволяє боротися з невизначеністю, знімаючи її етап за етапом, і перевіряти правильність технічного, маркетингового чи іншого рішення на ранніх стадіях.

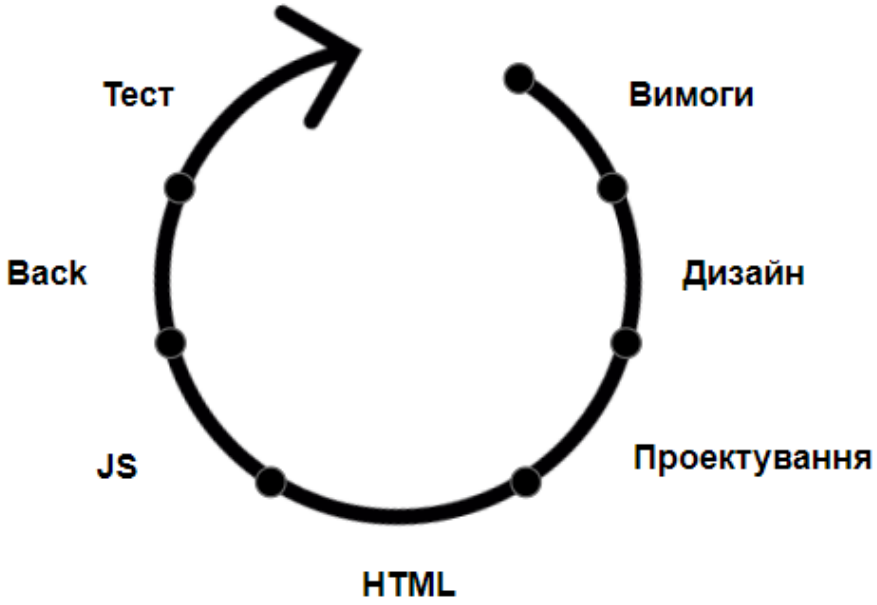


Рисунок 1.5 – Ітераційна модель

Використання ітераційної моделі знижує ризики глобального провалу та розтрати всього бюджету, отримання несинхронізованих очікувань та помилкового розуміння процесів як клієнтом, так і кожним учасником команди розробки. Воно також дає можливість завершення розробки в кінці будь-якої ітерації (у каскадній моделі ви повинні спочатку завершити всі етапи).

3. Усі етапи життєвого циклу при спіральній моделі йдуть витками, кожному з яких відбуваються проектування, кодування, дизайн, тестування тощо. буд. Такий процес відображає суть назви: піднімаючись, проходить один виток (цикл) спіралі задля досягнення кінцевого результату (рисунок 1.6). Причому не обов'язково, що той самий набір процесів буде повторюватися від витка до витка. Але результати кожного із витків ведуть до головної мети.

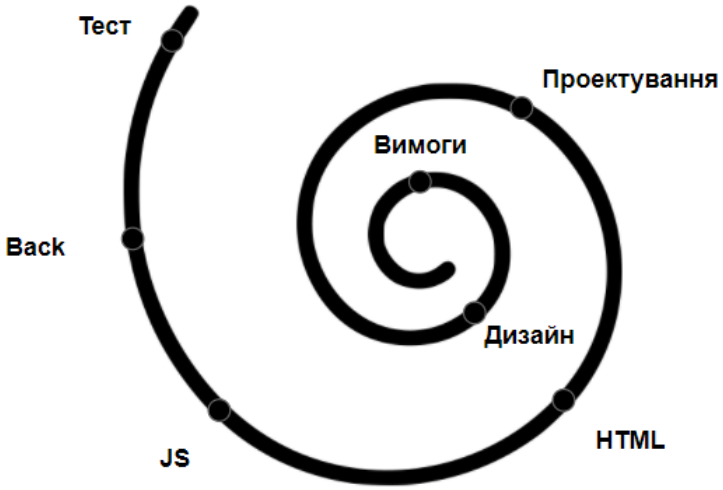


Рисунок 1.6 – Спіральна модель

Аналізуючи вищенаведені характеристики моделей життєвого циклу, було прийнято рішення використовувати ітераційну модель життєвого циклу ПЗ. Перевагами ітеративного підходу є:

- зниження впливу серйозних ризиків на ранніх стадіях проекту;
- організація ефективного зворотного зв'язку проектної команди зі споживачем та створення продукту, що реально відповідає його потребам;
- акцент зусиль на найважливіші та критичні напрями проекту;
- безперервне ітеративне тестування, що дозволяє оцінити успішність усього проекту загалом;
- раннє виявлення конфліктів між вимогами, моделями та реалізацією проекту;
- більш рівномірне завантаження учасників проекту;
- ефективне використання накопиченого досвіду;
- реальна оцінка поточного стану проекту та, як наслідок, велика впевненість замовників та безпосередніх учасників у його успішному завершенні.

1.5 Аналіз методів реалізації клієнтської частини веб-системи

Багато сайтів і порталів, на які ми заходимо щодня, насправді є веб-додатками. Це поштові ресурси, сторінки соцмереж, пошукові інструменти,

системи інтернет-банкінгу, інтернет-магазини, онлайнові редактори, ігри та багато іншого.

Як мовилося раніше, ключова відмінність веб-додатків від веб-сайтів у цьому, що вони є не статичні дані як документів в Інтернеті, а виконувані програми. Веб-програми – це клієнт-серверні рішення, які зберігаються на сервері, а виконуються і на сервері, і на комп'ютері або смартфоні користувачів.

Клієнтська частина веб-застосунку (Front-end) – інтерактивна частина програми, що виконується у веб-браузері на комп'ютері, смартфоні або планшеті користувача. Вона реалізує інтерфейс користувача веб-додатку і завантажується на пристрої у вигляді динамічних веб-сторінок. Веб-застосунки запускаються на будь-яких пристрої та операційних системах, де є інтернет-браузери. Для розробки Front-end елементів веб-застосунків залучають девелоперів, знайомих з технологіями React, Vue.js, Angular, JavaScript, HTML, CSS, Ajax та іншими. Дуже популярні останнім часом розробка web-додатків на Java.

Фреймворк — програмне забезпечення, яке дозволяє спростити процес розробки та складання найрізноманітніших модулів (частин) одного програмного проекту. Найпопулярнішими фреймворками на сьогодні є React та Angular. Варто окремо відзначити, що програми дуже відрізняються, тому неможливо сказати однозначно, що краще — React або Angular. Вони просто різні та використовуються для вирішення різних завдань. При цьому, як свідчить програма Google Trends, React та Angular у попередні роки були найбільш популярними у сфері розробки. Більше того, аналітики обіцяють їм шалений успіх і в 2019-2020 роках [13].

Основною відмінністю React та Angular вважається той момент, що перший використовує Virtual DOM. І цей факт відносять до переваг зазначеного фреймворку.

Переваги Angular 5:

- нові функції, такі як покращений RXJS, прискорена компіляція (менше 3 секунд) та новий лаунчер HttpClient;

- детальна документація, яка дозволяє кожному розробнику отримати всю необхідну інформацію без звернення по допомогу до колег;
- двостороння прив'язка даних, яка забезпечує чудову поведінку програми, що мінімізує ризики можливих помилок;
- MVVM (Model-View-ViewModel), яка дозволяє розробникам окремо працювати в одному розділі програми з використанням одного і того ж набору даних;
- впровадження залежностей функцій пов'язаних із компонентами з модулями та модульності в цілому.

Компанії, які використовують Angular 5: Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.

Переваги React Native:

- економічна ефективність;
- швидкий випуск проектів;
- використання JavaScript;
- вимагає невеликої команди;
- перевага відкритого вихідного коду;
- нативний зовнішній вигляд;
- модульний дизайн.

Обидва фреймворка працюють з нативними і веб-додатками:

React:

- нативні додатки розробляються з React Native;
- кросплатформені (Android, iOS) розробляються з React Native Renderer.

Angular:

- нативні додатки розробляються з NativeScript;
- розробка гібридних додатків відбувається з Ionic Framework.

Отже, для створення клієнтської частини потрібно обрати фреймворки, які потрібно використати аби точно визначитись з призначенням цього веб-додатку, та платформи, на яких він буде працювати.

1.6 Постановка задач розробки

Після аналізу можливостей створення надійної та швидкої системи для проведення міжнародних конкурсів і методів вирішення поставлених питань було сформульовано наступні задачі, які необхідно виконати для подальшої розробки програмного продукту:

- провести аналіз типів веб-додатків та методів їх створення;
- розробити інтерфейс програмного продукту;
- розробити модель автоматизованої системи оцінювання з використанням різнотипної системи оцінки;
- розробити багатокритеріальний метод формування загальної оцінки роботи з урахуванням експертних оцінок кількох членів журі за системою кількісних і якісних критеріїв оцінювання за 9-ти бальною шкалою Сааті;
- створити взаємодію між веб-додатком та серверною частиною;
- розробити інформаційне наповнення системи;
- провести тестування програмного продукту.

1.7 Висновки

1. Проаналізовано стан питання та предметну область. Розглянуто особливості створення систем для проведення різноманітних конкурсів з графіки та дизайну на сьогоднішній день.

2. Проведений аналіз методів розв’язання задачі. Прийнято рішення використовувати веб-систему для вирішення задачі. Відображено доцільність розробки веб-системи та проаналізовано можливі методи вирішення поставлених питань.

3. Розглянуто та проаналізовано низку аналогів, наведено їх переваги та недоліки. Основними конкурентами веб-системи є: Awwwards, Webby Awards, Favourite Website Awards (FWA), European Design Awards. Їх було порівняно між собою, а також з розробленим програмним продуктом.

4. Обрано ітераційну модель життєвого циклу та наведено основні його переваги для розроблюваної системи.

5. Було сформульовано завдання, які необхідні вирішити для розробки програмного продукту.

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ АВТОМАТИЗАЦІЇ РОБОТИ СИСТЕМИ

2.1 Розробка моделі автоматизованої системи оцінювання графічних робіт

Інформаційне забезпечення – це сукупність проектних рішень за обсягами, розміщенням та формами організації інформації, що циркулює в ІВ. Інформаційне забезпечення призначене для відображення сукупності інформації, що характеризує стан керованого об'єкта і є основою прийняття управлінських рішень. Інформаційне забезпечення (ІЗ) призначена для постачання користувачів інформацією, що характеризує стан керованого об'єкта і є основою для прийняття управлінських рішень.

Інформаційне забезпечення можна розділити на позамашинне та внутрішньомашинне.

Позамашинне ІЗ – це системи показників, класифікаторів, кодів та документації.

Внутрішньомашинне ІЗ – це різні файли на машинних носіях, автоматизовані банки даних (АБД).

У ході проектування інформаційного забезпечення виконуються такі роботи:

- визначення складу показників, необхідні вирішення економічних завдань, їх об'ємно-тимчасових характеристик та інформаційних зв'язків;
- дослідження можливостей використання загальнодержавних та галузевих класифікаторів, розробка локальних класифікаторів та кодів;
- проектування форм нових первинних документів та виявлення можливостей застосування уніфікованої системи документації;
- визначення типу організації автоматизованого банку даних (АБД);
- проектування форм виведення результатів.

Створено модель роботи системи «Majestics». Це необхідно для того, аби включити всі необхідні елементи навігації, форми, зрозуміти, як правильно повинен працювати додаток і що саме повинно бути на сторінках. Було

виключено всі ui-елементи, які б заважали перегляду, або ж відволікали користувача від необхідної інформації, чи були б недоречними на сайті. Нижче показана модель, для кращого уявлення про роботи веб-системи (рис. 2.1).

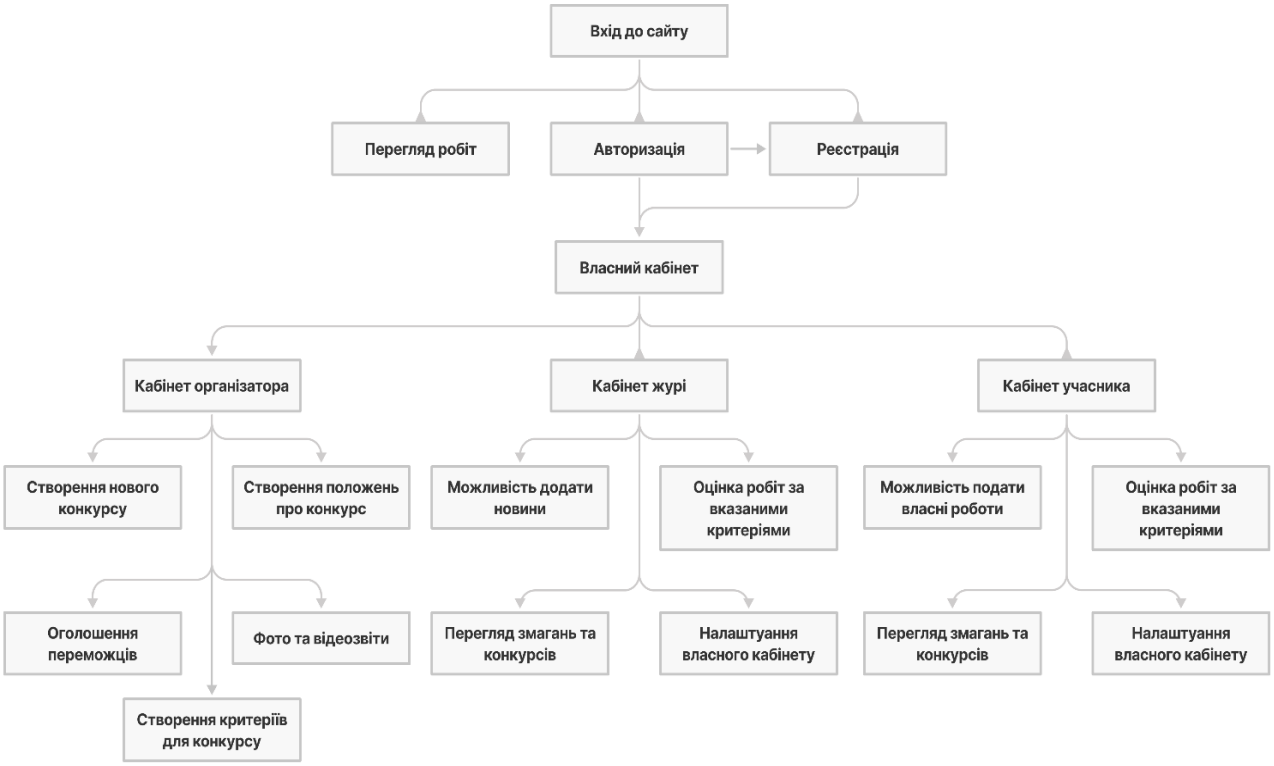


Рисунок 2.1 – Модель роботи системи «Majestics»

З вищенаведеного рисунку ми бачимо наступне:

- навігація по системі можлива в якості як зареєстрованого так і незареєстрованого користувача;
- зареєструватись в системі можна в якості журі, учасника або організаторів;
- у кабінеті організаторів можливо створити конкурс, додати відповдні критерії оцінювання, розмістити його, додати умови проведення і все що з ним пов'язано;
- у кабінеті журі можливе налаштування свого профілю, перегляд списку змагань, додання новин про конкурс або роботи, оцінка уже існуючих робіт за наведеними критеріями;

- кабінет учасника містить у собі такі функції як налаштування власного профілю, відкриття списку змагань, оцінка робіт інших учасників за наведеними критеріями та додання своїх робіт до активних конкурсів.

Провідними критеріями якості надісланої роботи, за допомогою яких може здійснюватися її оцінка, виступають технічність, композиція, настрій, оригінальність та закінченість. Вони враховують:

- технічність – наскільки учасник володіє технічними аспектами графіки, враховується якість і складність кінцевої роботи;
- композиція – враховується колірна гама, шрифтові рішення, дотримання перспективи, відповідність світотіньових елементів, дотримання єдиного силю роботи;
- настрій – рівень емоцій (позитивних або ж негативних), які виникають під час перегляду роботи;
- оригінальність – враховується ідея та сюжет, наскільки вони є справжніми та автентичними, чи відрізняються вони від інших;
- закінченість – встановлюється міра повноти практичного втілення ідеї учасника в роботу.

Дані критерії, в залежності від виду номінації, можуть доповнюватись огранізаторами конкурсу.

Існує безліч фреймворків та технологій для створення динамічних та реактивних веб-додатків. Для реалізації даного веб-додатку використано фреймворк під назвою Angular версії 8. Це є дуже потужний сервіс, що дає змогу розробнику створювати багатофункціональні і гнучкі веб-додатки.

Ангуляр базується на MVC шаблоні і головною його задачею було створити фреймворк, який дасть змогу проводити всі операції на веб-сайті в мержах однієї сторінки, що в своїй реалізації виконує ті самі функції що і десктопні або мобільні додатки [14].

MVC є популярним, оскільки ізолює логіку додатка від призначеного для користувача інтерфейсу і підтримує поділ завдань. Контролер приймає всі запити для додатку, а потім працює з моделлю, щоб підготувати будь-які дані, необхідні для подання. Вид потім використовує дані, отримані за допомогою

контролера, щоб генерувати кінцеву відповідь. Абстракції MVC, можна графічно представити таким чином.

Модель. Модель надає знання: дані та методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, як ці знання можна візуалізувати.

Вид. Відповідає за відображення інформації (візуалізацію). Часто в якості уявлення виступає форма (вікно) з графічними елементами. Засновані системи шаблонів, такі як JSP, ASP, PHP і дуже легко інтегрувати за допомогою технології AJAX.

Контролер. Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель і уявлення для реалізації необхідної реакції.

Для реалізації графічної частини інтерфейсу було використано мову розмітки HTML (Hyper Text Language Markup) та каскадні таблиці стилів CSS (Cascade Styles Sheets).

HTML (HyperText Markup Language) - стандартна мова розмітки гіпертекстових сторінок в Інтернеті. Є й інші мови розмітки гіпертексту, але велика частина сторінок сайтів Інтернету розмічена саме на мові HTML. Такі сторінки успішно інтерпретуються браузерами, які відображають їх на екранах різних електронних пристроїв в зручному для людини вигляді.

HTML є теговою мовою розмітки гіпертексту: щоб перетворити текст в гіпертекст, використовують роздільники (дескриптори), для стислості названі тегами. Ось приклад тега: - цей відкриває тег забезпечує виведення тексту жирним шрифтом до тих пір, поки не зустрінеться закриває тег </ strong>.

CSS – це формальна мова, службовець для опису оформлення зовнішнього вигляду документа, створеного з використанням мови розмітки (HTML, XHTML, XML). Назва походить від англійського Cascading Style Sheets, що означає «каскадні таблиці стилів».

Така технологія:

- забезпечує відносно просту і швидку розробку, тому що одного разу створене оформлення можна застосовувати до багатьох сторінок;
- підвищує гнучкість і зручність редагування – досить внести правку в CSS, щоб оформлення змінилося всюди;
- робить код більш простим, знижуючи повторюваність елементів. Його простіше читати програмістам і пошуковим роботам;
- прискорює час завантаження, тому що CSS може кешуватися при першому відкритті, а в наступних зчитуються тільки структура і дані;
- збільшує кількість візуальних рішень для подання вмісту;
- забезпечує можливість легко застосовувати до одного документу. різні стилі (наприклад, створювати адаптовану версію для мобільних пристроїв або спеціальні стилі для людей з вадами зору).

Тобто каскадні таблиці служать не тільки для втілення дизайну, а й кардинально змінюють підхід до сайтобудування, спрощуючи працю розробників і забезпечуючи гнучкість реалізації. Ось для чого потрібен CSS.

Також для реалізації знайомої поведінки і вигляду елементів було використано безкоштовну бібліотеку написану розробниками Twitter – Bootstrap. Bootstrap - HTML / CSS / JS-фреймворк. Це набір CSS-стилів і JavaScript-скриптів для швидкого створення сучасних адаптивних сайтів.

При розробці користувацького інтерфейсу з використанням Angular в залежності від потреби і релевантності, потрібно ділити код на компоненти, для того щоб в подальшому у розрозбника була можливість легко повторно використати потрібні йому елементи. Також це знижує кількість даних, які потрібно завантажити для відображення сторінки, та значно підвищує швидкість роботи сайту. Компонент за визначенням – це блок програмного коду, який контролює вигляд моделі і представлення.

Angular дозволяє використовувати запрограмовані «аліаси», які дозволяють відображати дані шляхом прив'язки (binding) до даних моделі, причому можуть працювати, як тільки в одну сторону так і в дві (two-way binding) [15]. За допомогою аліасів також можна проводити калькуляції прямо в HTML коді сторінки.

При розробці було використане середовище розробки: Visual Studio Code. Нижче зображено вікно цієї програми (рис. 2.2).

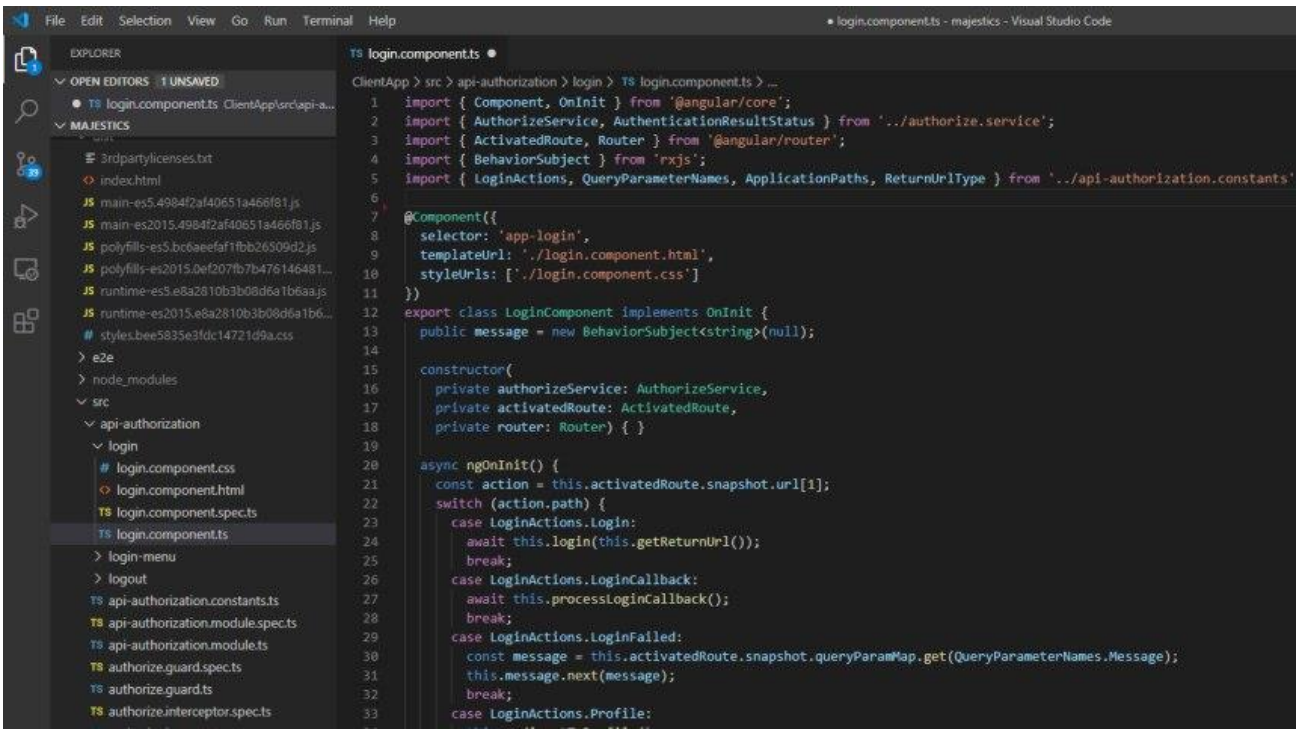


Рисунок 2.2 – Вікно програми Visual Studio Code

Особливості Visual Studio Code [16]:

- VS Code дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-програми, веб-служби як в рідному, так і в керованому кодах для всіх платформ;
- продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легким рішенням, яке дозволяє обійтися без повного інтегрованого середовища розробки.
- великим плюсом редактора є підтримка великої кількості мов, таких як C++, C#, Python, PHP, JavaScript та інші.

Можливості Visual Studio Code:

- вбудовані інструменти інтеграції з GitHub, GIT, а також Visual Studio Team Services для швидкого тестування, складання, пакування та розгортання різних типів програм;
- зручність роботи з Unity-проектами;

- робота з Mono та Node.js за допомогою вбудованого відладчика;
- підтримка TypeScript та JavaScript;
- публікація створених програм у Microsoft Azure через сервіс Visual Studio Team Services;
- велика бібліотека шаблонів, готових фрагментів коду та сніпетів з можливістю додавання своїх елементів;
- одночасна робота з кількома проектами (у кількох вікнах);
- інтерфейс можна поділити на дві панелі для порівняння коду.

Переваги Visual Studio Code:

- безліч налаштувань (як усієї програми, так і інтерфейсу);
- бібліотека доповнень і готових рішень, що розширюється;
- мультифункціональність (редактор підтримує майже всі мови, які використовуються для створення програм);
- простота та гнучкість.

2.2 Розробка методу багатокритеріального оцінювання та вибору найкращого варіанту

Основними методами, що застосовуються під час порівняння та відбору розроблених варіантів рішень клієнтських проблем, можуть бути якісні і кількісні оцінки. До якісних методів оцінки належить конкретне числове значення показника для варіанта, що оцінюється.

Кількісні критерії, дозволяють оцінювати результати прийнятих рішень, прийнято називати критеріями ефективності. Кожне рішення призводить до певного результату (виходу), наслідки якого оцінюються за критеріями (оцінними критеріями) [17].

Найбільш зручні для аналізу альтернативи, у яких мірилом ефективності є єдиний кількісний критерій. Єдиний критерій, який використовується для оцінки альтернатив, називають скалярним, а сукупність критеріїв, що характеризують альтернативи, називають векторним критерієм. Завдання оцінки ефективності рішень одночасно за декількома критеріями називають багатокритеріальними.

До якісних критеріїв можна віднести експертні оцінки. Це якісні оцінки, засновані на інформації кількісного (якісного) характеру, які можуть бути отримані лише за допомогою фахівців – експертів. Експерт – це висококваліфікований фахівець, що покладається на свої знання, інтуїцію, досвід, вміння оцінювати складні фактори і здатний створити власну обґрунтовану модель аналізованого явища, якщо він має необхідну для цього вихідну інформацію. Сутність методу експертних оцінок полягає у логіко-інтуїтивному аналізі, розробці альтернатив та кількісній оцінці їх якості. Узагальнена думка експертів є підставою для здійснення вибору найкращого серед поданих варіантів. Комплексне використання інтуїтивного та логічного мислення, формальної обробки кількісно виражених суджень експертів дозволяє отримати показники якості альтернатив при вирішенні поставленої задачі.

Методом експертного оцінювання вирішуються такі типові завдання:

- генерування альтернатив;
- визначення пріоритетності;
- визначення складу можливих подій у будь-якій системі у певному інтервалі часу;
- фільтрація безлічі альтернатив та оцінка їхньої переваги.

Експертні судження — змістовні висловлювання, кількісна чи якісна оцінка будь-якої сутності [18].

Альтернативи є ключовим компонентом для ефективного вирішення задачі. Ефективність рішення багато в чому визначається тим, із якої кількості альтернативних варіантів обраний даний варіант рішення.

Якщо дані альтернативні варіанти відсутні, це свідчить або про недостатню поінформованість особи, яка приймає рішення, або про дефіцит часу, який відводиться на ретельну перевірку цього рішення. Це підвищує ступінь ймовірності помилковості у прийнятті рішення та значно ускладнює вибір оптимального варіанта.

З урахуванням вимог до критеріїв оптимізації розроблено комбінований метод багатокритеріального аналізу альтернатив, який створено на базі методу

парних порівнянь Т.Сааті. Такий метод дозволить врахувати особливості множини критеріїв оцінювання. При цьому слід передбачити механізм вагового ранжування критеріїв оцінювання за умов їх нерівноважності. Ранжування застосовується у випадках, коли неможлива чи недоцільна безпосередня оцінка. При цьому «ранжування об'єктів містить лише інформацію про те, який з них кращий, і не містить інформації про те, наскільки або у скільки разів один об'єкт краще іншого. Ранг — відзнака за якою-небудь ознакою, а ранжування — процес визначення рангів, відносних кількісних оцінок ступенів відмінностей за якісними ознаками. Використовуються такі методи ранжування:

- метод простого ранжування, який полягає у тому, що експерти мають у своєму розпорядженні об'єкти ранжування (критерії) в порядку зменшення їх значущості;
- метод безпосередньої оцінки, полягає у визначенні переваг елементів, розташованих у лівому стовпці, над елементами, розташованими у верхньому рядку;
- метод парних порівнянь.

Визначимо вимоги до критеріїв оцінки альтернатив. Алгоритмом прийняття рішень може бути така схема:

- охарактеризувати розглянуту задачу;
- сформувати поле допустимих альтернатив (виділити усі допустимі та відкинути ті, які є свідомо нездійсненні);
- визначити критерії оцінки;
- ранжувати критерії важливості;
- відкинути маловажливі критерії (якими можна знехтувати);
- призначити відповідні бали, що відповідають відносній важливості критеріїв;
- визначити функції корисності кожного з критеріїв.

В основу комбінованого методу багатокритеріального оцінювання покладемо алгоритм:

1. Сформуємо множину альтернатив представлених робіт: $A=\{a_1,a_2,...,a_n\}$.
2. Сформуємо множину критеріїв для оцінювання роботи: $K=\{k_1,k_2,...,k_m\}$.

3. Проводимо ранжування критеріїв оцінювання.

Нехай $v_1,v_2,...v_m$ — вагові коефіцієнти важливості критеріїв $k_1,k_2,...,k_m$ такі, що $v_1+v_2+...+v_m=1$. Щоб визначити коефіцієнти v_j, $j=\overline{1,m}$ необхідно сформувати матрицю для парних порівнянь важливості критеріїв $k_l \in K$ (2.1).

$$K^1 = \begin{array}{c} \\ k_1 \\ k_2 \\ : \\ k_m \end{array} \begin{array}{c} \begin{array}{cccc} \boldsymbol{k_1} & \boldsymbol{k_2} & \boldsymbol{...} & \boldsymbol{k_m} \end{array} \\ \begin{bmatrix} k_{11}^1 & k_{12}^1 & ... & k_{1m}^1 \\ k_{21}^1 & k_{22}^1 & ... & k_{2m}^1 \\ ... & ... & ... & ... \\ k_{m1}^1 & k_{m2}^1 & ... & k_{mm}^1 \end{bmatrix} \end{array}, \tag{2.1}$$

де елемент k_{ij}^l оцінюється за 9-ти бальною шкалою Сааті:

Ступені належності, які необхідні для формування множини оцінювання знаходяться за виразом (2.2):

$$k_i = \frac{1}{k_{i1}^1 + k_{i2}^1 + ... + k_{im}^1}. \tag{2.2}$$

Кінцеві значення критеріїв з урахуванням процесу нормалізації знаходяться за формулою (2.3):

$$v_j = \frac{\mu'(k_j)}{\sum_{j=1}^{m} \mu'(k_j)}, \tag{2.3}$$

де v_j – нормований ранг критерію, $\mu'(k_j)$ – число, що характеризує вагу критерію k_j.

Кількісні оцінки альтернатив нормуються за кожним із критеріїв для оцінювання за формулою (2.4):

$$a_{к_{ij}}^{нор} = \frac{a_{к_{ij}}}{\sum_{i=1}^{n} a_{к_{ij}}}. \tag{2.6}$$

Визначення інтегральних оцінок для альтернатив за кількісними критеріями проводиться з урахуванням ваги критеріїв за методом компромісу (2.6):

$$A_{к_i} = \prod_{j=1}^{m} (a_{к_{ij}}^{нор})^{v_j} . \quad (2.6)$$

Наступним кроком є нормування кожної інтегральної оцінки за кількісними показниками за формулою (2.7):

$$A_{к_i}^{нор} = \frac{A_{к_i}}{\sum_{i=1}^{n} A_{к_i}} . \quad (2.7)$$

Для альтернатив роботи за кожним з якісних критеріїв для оцінювання складаються матриці парних порівнянь (2.8) на основі 9-ти бальної шкали Сааті:

$$A^{l} = \begin{matrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{matrix} \begin{bmatrix} a_{11}^{l} & a_{12}^{l} & \dots & a_{1n}^{l} \\ a_{21}^{l} & a_{22}^{l} & \dots & a_{2n}^{l} \\ \dots & \dots & \dots & \dots \\ a_{n1}^{l} & a_{n2}^{l} & \dots & a_{nn}^{l} \end{bmatrix} . \quad (2.8)$$

Для кожної альтернативи за кожним якісним критерієм проводиться нормування результатів оцінювання за формулою (2.9):

$$a_{як_{ij}}^{нор} = \frac{1}{r_{i1}^{l} + r_{i2}^{l} + \dots + r_{in}^{l}} . \quad (2.9)$$

За формулою 2.9 визначаються цифрові значення кількісних і якісних оцінок кожної роботи за кожним критерієм. Інтегральна оцінка за кожним критерієм формується числовим значенням порівняльної оцінки кожної роботи за цим критерієм. Тобто формується масив значень оцінок усіх робіт окремо за кожним критерієм.Визначення кінцевого інтегрального значення відбувається шляхом усереднення результатів оцінювання, при цьому, для забезпечення об’єктивності оцінювання відкидаються межові (найбільше і найменше значення) оцінок роботи за кожним критерієм. Це дозволяє виключити

можливість зумисного підвищення або заниження оцінок експертами та дозволяє забезпечити високу об’єктивність оцінювання в процесі реалізації еспертної системи.

Найкращою визнається та робота, значення загального інтегрального критерію якої є найбільшим.

Якщо для оцінюваного варіанту не існує аналогу, тоді оцінку проводять у порівнянні з формальним еталоном, наділяючи еталон 100% перевагою за всіма обраними критеріями. Таким чином можна визначити, наскільки оцінюваний варіант наближений до еталону.

Відомі:

Ar={a_r1,a_r2,...,a_rn} — множина аналогів роботи, які підлягають багатокритеріальному аналізу;

$E_в$ — еталонний варіант;

К={к1,к2,...,km} — множина кількісних та якісних критеріїв, за якими оцінюються аналоги.

В даному випадку матриці парних порівнянь A′ будуть одномірними і матимуть вигляд:

$$A' = [1, a'_1, a'_2, \ldots, a'_n].$$

За формулою (2.14) ступені належності, необхідні для формування нечітких множин:

$$\mu^l(ar_i) = \frac{a_i^l}{1 + a_i^l}, \qquad (2.14)$$

де a'_1 визначається допомогою 9-тибальної шкали Сааті з використанням властивостей:

1 – якщо відсутня перевага варіанта ar_j над варіантом ar_i;

3 – якщо є слабка перевага ar_j над ar_i;

5 – якщо є суттєва перевага ar_j над ar_i;

7 – якщо є явна перевага ar_j над ar_i;

9 – якщо є абсолютна перевага ar_j над ar_i;

2,4,6,8 – проміжні порівняльні оцінки,

0 – об’єкти неможливо порівнювати.

Найближчим до еталону є той, що є одночасно кращим за всіма обраними критеріями.

Метод багатокритеріального оцінювання полягає у виконанні такої послідовності дій:

1. Визначення системи критеріїв оцінювання робіт, які можуть бути як кількісними, так і якісними; система є відкритою і може модифікуватися і доповнюватися за потреби.

2. Кожен експерт проводить оцінювання робіт за методом парних порінянь за 9-ти бальною шкалою Сааті та формує матриці парних порівнянь.

3. За матрицями парних порівнянь визначаються функції належності, що дозволяють отримати цифрові оцінки за кількісними критеріями оцінювання з використанням цифрових еквівалентів 9-ти бальної шкали Сааті.

4. Визначаються оцінки всіх робіт за кожним критерієм окремо як множина цифрових значень, отриманих у результаті визначених функцій належності.

5. Визначення інтегрального критерію здійснюється шляхом усереднення результату з відкиданням межових (найбільшої і найменшої) оцінок експертів з метою підвищення об'єктивності оцінювання шляхом виключення можливостей зумисного збільшення чи зменшення рейтингу конкретної роботи.

Використання методу парних порівнянь за 9-ти бальною шкалою Сааті дає змогу використовувати систему якісних і кількісних критеріїв оцінювання. Використання матриць парних порівняь за рахунок симетричного дзеркального відображення відносно головної діагоналі дозволяє експерту перевірити свої оцінки парного парівняння і у разі визначення несиметричностей переглянути конкретні оцінки визначених пар об'єктів. Це дозволяє підвищити якість процесу оцінювання робіт.

2.3 Розробка користувальницького інтерфейсу веб-додатку

Інтерфейс користувача, або UI (User Interface) – це зовнішній вигляд продукту, спосіб спілкування між користувачем і програмою. А ще інтерфейс

впливає на те, чи продукт приноситиме гроші і користуватиметься повагою та любов'ю аудиторії [19]. Завдання проектування інтерфейсу – оптимальне розподілення необхідного функціоналу по всьому простору сайту (по всіх екранах). Проектування це початковий етап розробки інтерфейсу користувача.

Проектування – розподіл функцій інтерфейсу на окремих екранах, логічна розмітка макетів екранів шляхом визначення контенту, елементів управління та їх поведінки.

На етапі проектування вирішуються такі завдання:

- вибудовування повної ієрархічної структури сайту;
- визначення розташування на сторінках сайту (екранах) контенту та елементів керування (кнопок, меню);
- пошук способів угруповання однотипних елементів;
- визначення поведінки керуючих елементів сайту.

Ключові персони – це характерні представники ЦА. Вони можуть бути різними за професією, рівнем життя, мотивацією користуватися додатком та іншими параметрами, але досвід, очікування та страхи кожної персони лягають в основу зовнішнього вигляду продукту та його функціональності.

Така особа стає центром user story, або користувальницької історії. Це коротка, у кілька рядків, розповідь про персону і те, як вона працює з функціональністю програми та якої мети досягає. User story будується за шаблоном:

«Як <роль користувача>, я <щось хочу отримати> <з такою метою>»

Наприклад, розглянемо 18-річного студента, який цікавиться розробкою 3D моделей ще зі шкільних років, та помістимо у цей шаблон:

«Як <18-річний студент, що цікавиться 3D розробкою>, я <шукаю різні сервіси для проведення конкурсів з комп’ютерної графіки> <щоб спробувати свої сили і отримати призове місце>»

Від User story переходимо до User scenario. Це маршрут взаємодії користувача з продуктом та досягнення мети. На рисунку 2.3 наведено схему бажання користувача завантажити роботу на конкурс до його втілення.

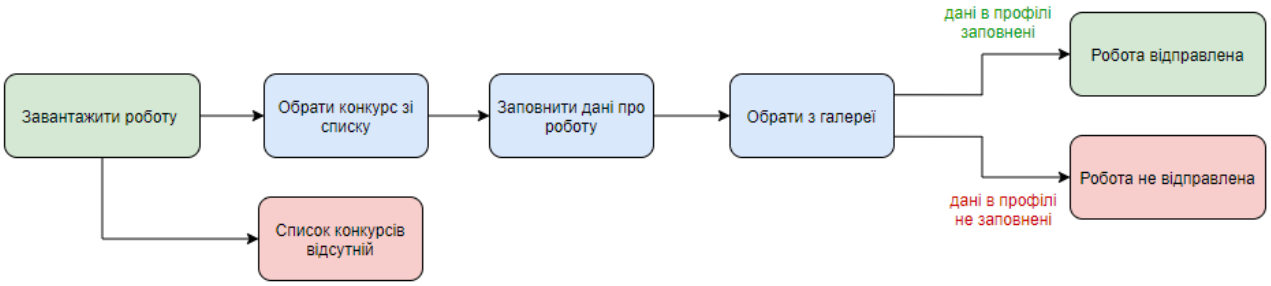


Рисунок 2.3 – Схема завантаження роботи

У гонитві за основною метою (участь у конкурсі, додавання та оцінка робіт) користувач може вирішувати допоміжні завдання (оновлення профілю, перегляд переможців) і досягати вторинних цілей (зручне отримання сертифікатів, фото учасників та інформація, що змагаються); ці додаткові маршрути дизайнеру також потрібно врахувати.

Для даної веб-системи було обрано flat дизайн в поєднанні з мінімалізмом. У більшості випадків такий дизайн застосовується для залучення уваги молодих користувачів. Flat дизайн відрізняє простий користувальницький інтерфейс. Він легко адаптується під будь-які пристрої [20].

Колір – потужний інструмент для вирішення більшості завдань в дизайні. Вірно підібрана колірна гама здатна привернути увагу до потрібної частини сайту. Один і той же макет в різних кольорах може сприйматися по різному. Грамотне використання кольорів в оформленні web сайтів дозволить створювати потрібні Вам враження і емоції, підкреслити стиль компанії. Розуміння значення кольору необхідно дизайнерам для найбільш ефективного донесення певної інформації від сайту до його користувачам і відвідувачам.

Для даної системи був обраний первинний колір – синій, адже він символізує довіру, консервативність, надійність, спокій та міцність. Синій - колір надійності, сталості, досконаліть і світу. Сприяє розслабленню і створює атмосферу безпеки і довіри. Об'єкти синього кольору зазвичай створюють відчуття холоду і спокою. Символізує вічні цінності, відданість, розсудливість, увагу, самозаглиблення, правосуддя. Колір королівської влади і благородного походження. Синій колір може відображати потужність сучасних технологій, чистоти повітря, води. Крім того, в поєднанні з червоним і жовтим

народжуються чіткі асоціації зі здоров'ям і довірою. Саме такі відчуття повинні виникати у людей, які заходитимуть на даний веб-додаток.

Прототип – це малюнок продукту, в якому укладені його зовнішній вигляд, логіка роботи та основна функціональність. Робота над ним починається із створення макета. Одним із варіантів макета є вайрфрейм (від англійського wireframe – «каркасний»). Зовні він виглядає як купа прямокутних блоків, підперезаних лініями та стрілочками. У цих блоках та стрілочках закладена структура продукту та порядок взаємодії користувача з ним. Вайрфрейм може бути грубим начерком, який зроблений ручкою на папері для принтера, чи створеним в графічному редакторі організованою карткою екранів.

Деталізований прототип – наступний крок вайрфрейму по сходах еволюції інтерфейсу користувача. Як і вайрфрейм, це макет, але трохи більш конкретний.

Розроблюваний веб-додаток складається з кількох вікон. Основними є вікно авторизації (рис.2.4), вікно власного профілю (рис. 2.5), вікно усіх конкурсних робіт (рис. 2.6) та вікно подачі заявки на конкурс (рис. 2.7).

Вікно рестрації є початковим вікном при створенні поїздки. Тут користувач повинен обрати варіант, який йому підходить (учасник або журі). Далі заповнюються дані для реєстрації, такі як ПІБ, телефон, пошта та пароль. У вікні власного профілю вказана основна інформація про учасника. Цю інформацію можна змінити у будь-який час. Також є можливість додати свої навички та інтереси. Вікно з усіма роботами користувачів призначене для перегляду та оцінки усіх проектів. Містить фільтр та список категорій, які можна прокручувати. Кожна робота містить назву та можливість переходу на її детальний опис. Також вікно містить кнопку для подання заявки на участь.

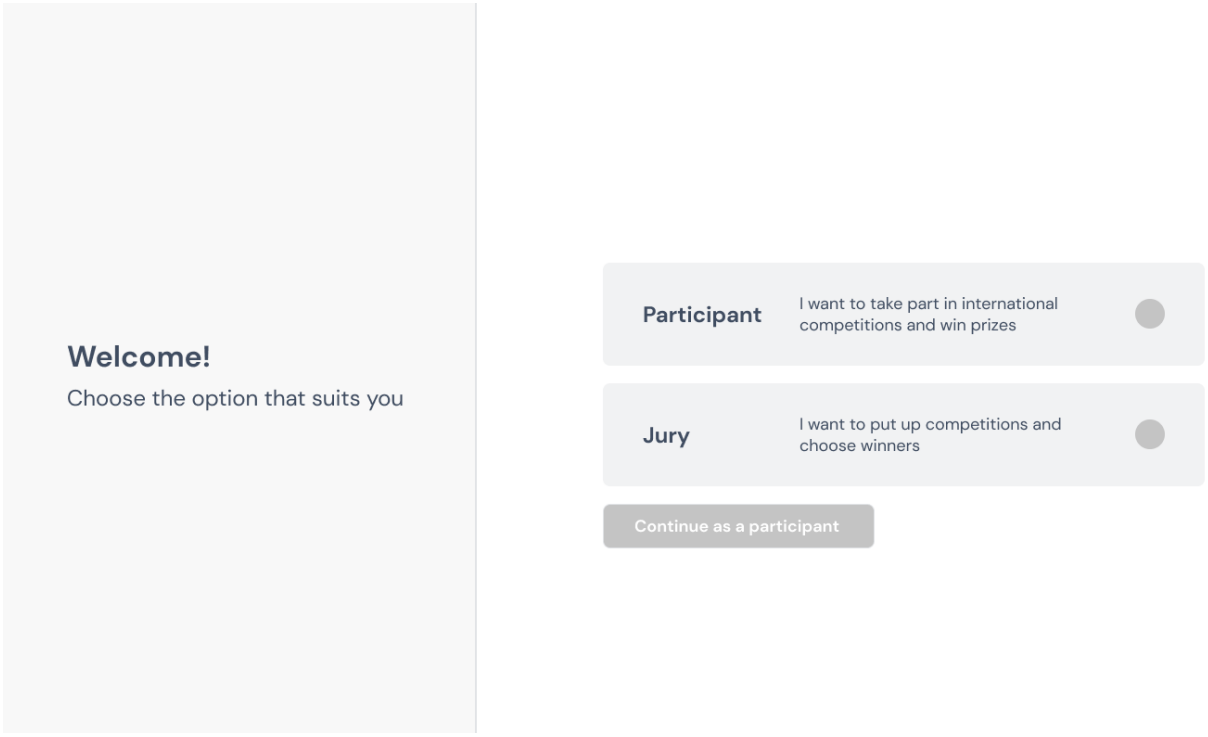


Рисунок 2.4 – Початкове вікно авторизації

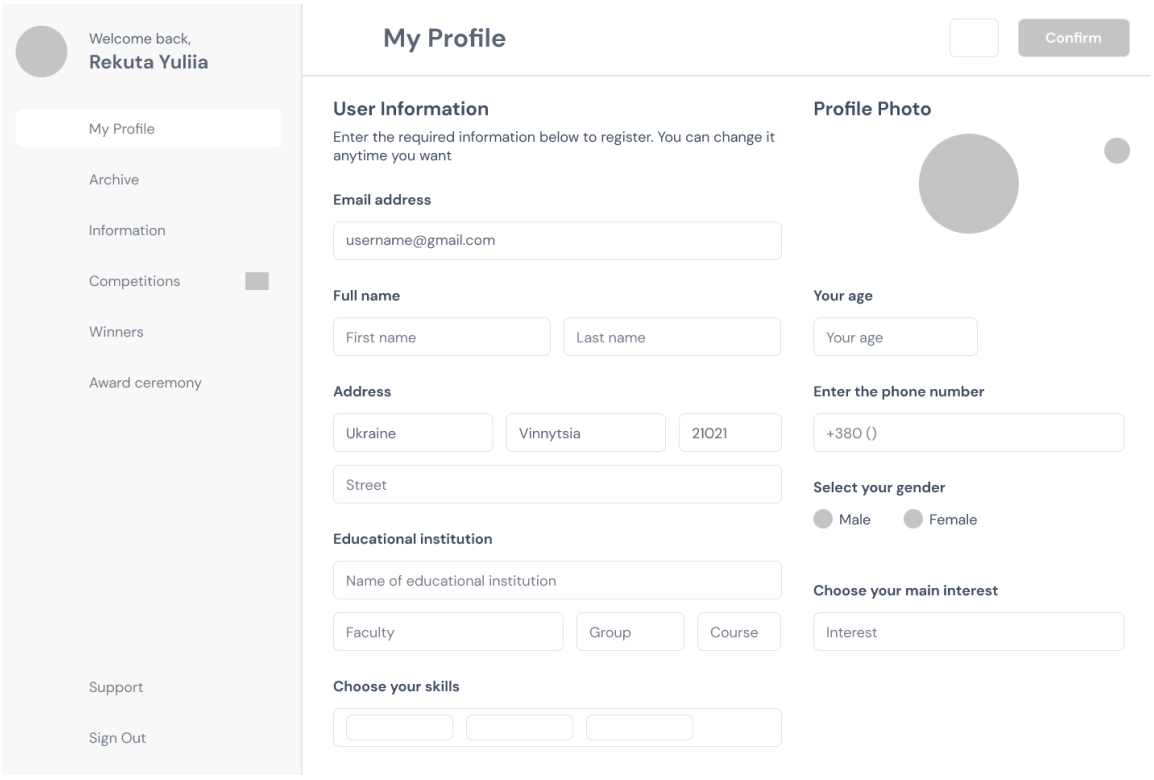


Рисунок 2.5 – Вікно власного профілю

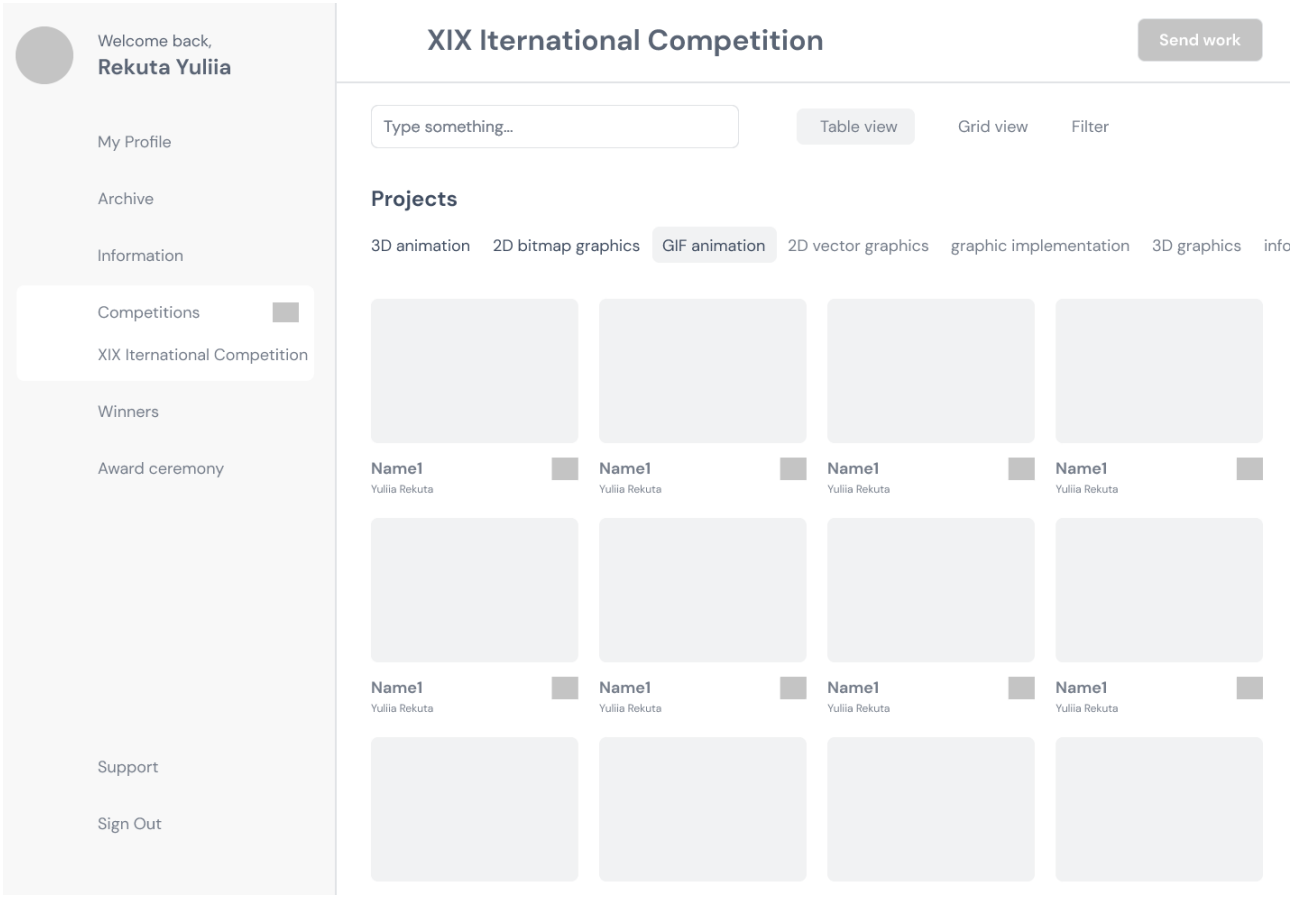


Рисунок 2.6 – Вікно усіх конкурсних робіт

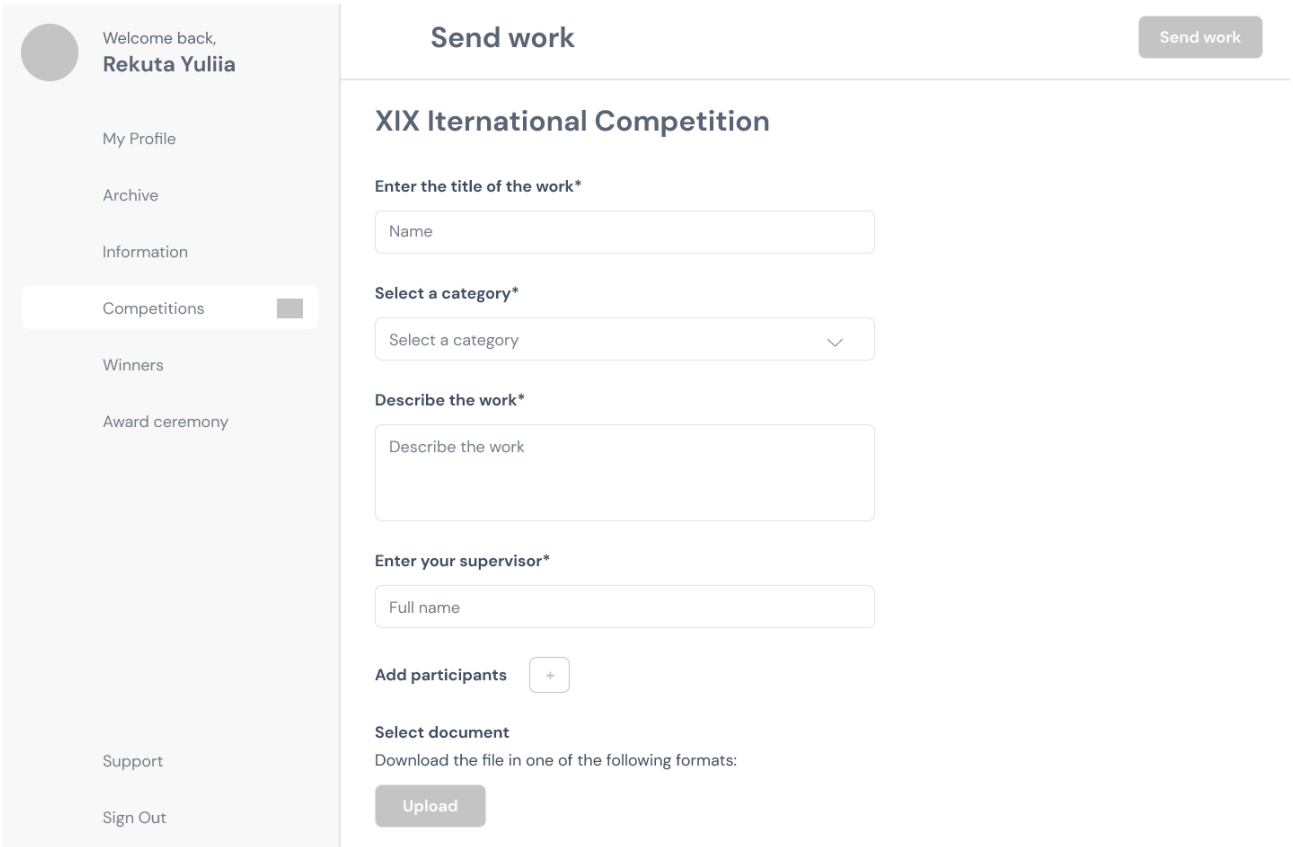


Рисунок 2.7 – Вікно подачі заявки на конкурс

Функціонал веб-системи «Majestics» надає користувачеві можливість реєстрації під різними типами аккаунтів, додавати новини, створювати конкурси, додавати роботи на конкурси, оцінювати роботи учасників, оцінювати роботи учасників як жюрі, налаштовувати свій профіль.

Розроблений графічний інтерфейс є досить простим для користування, та у поєднанні з гарним дизайном дозволить користувачеві швидко та зручно подавати свої заявки на участь.

2.4 Розробка сервісів та інструментів для взаємодії з серверною частиною

Для роботи з серверною частиною Angular дає змогу розробнику використовувати такі технології як TypeScript (типізована надбудова над мовою програмування JavaScript), що дозволяє розробнику бути впевненим в цілісності даних та безпосередньо структури будови моделі. За допомогою TypeScript розробник може користуватись всіма перевагами об'єктно-орієнтованого програмування в більшій степені аніж при розробці на чистому JavaScript [22].

Переваги TypeScript:

- виловлює помилки на етапі компіляції, а не під час виконання.
- покращена читаність коду. Структура, яка надається TS, значно спрощує аналіз нового коду, коли строго типізовані змінні, функції і об'єкти. Це усуває більшу частину припущень про те, яку форму приймуть дані.
- додаткова статична типізація. TypeScript не вимагає статичної типізації всього, тому ви можете поступово перетворювати проект.
- автодоповнення і підсвічування синтаксису. TypeScript підтримується більшістю основних IDE і текстових редакторів, включаючи VS Code і Atom. Вони пропонують потужні інтеграції TypeScript, які роблять автопідстановку коду, вказують аргументи функції, не звертаючи уваги на джерело, і забезпечують вбудоване розпізнавання помилок.

Це все є можливим через те, що в якості програми інтерпретатора використовується технологія Node.js що дозволяє трансляцію TypeScript коду в

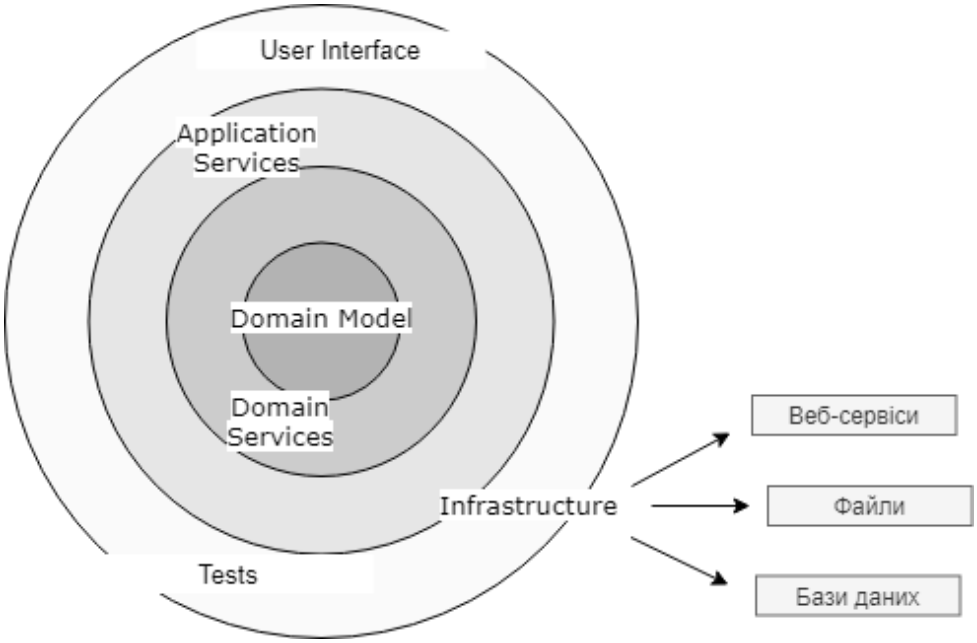
зрозумілий усім браузерам JavaScript сценарій в прямому часі, за рахунок чого підвищується не тільки швидкість роботи веб-додатку, а й кількість ресурсів, що браузер потребує.

Для взаємодії з серверною частиною було обрано ONION архітектуру, суть якої полягає в поділі всіх сервісів на шари (рис. 2.8). Onion-архітектура є поділом програми на рівні. При чому є один незалежний рівень, що знаходиться у центрі архітектури. Від цього рівня залежить другий рівень, від другого – третій тощо. Тобто виходить, що довкола першого незалежного рівня нашаровується другий-залежний. Навколо другого нашаровується третій, який може залежати і від першого. Образно це може бути виражене у вигляді цибулі, в якій також є серцевина, навколо якої нашаровуються всі інші шари, аж до лушпиння [21].

Кількість рівнів може відрізнятися, але в центрі завжди знаходиться модель домену (Domain Model), тобто класи моделей, які використовуються в додатку і об'єкти яких зберігаються в базі даних. Перший рівень довкола моделі домену утворюють інтерфейси, які управляють роботою з моделлю домену. Зазвичай це інтерфейси репозиторіїв, якими взаємодіємо з базою даних.

Зовнішній рівень представляє такі компоненти, які часто змінюються. Зазвичай зовнішній рівень утворюють інтерфейс користувача, тести, якісь допоміжні класи інфраструктури програми. До цього рівня також відносяться конкретні реалізації інтерфейсів, оголошених нижче рівнях. Наприклад, реалізація інтерфейсу репозиторію, який оголошено лише на рівні Domain Services. Взагалі всі внутрішні рівні, які можна об'єднати в Application Core, визначають тільки інтерфейси, а конкретна реалізація цих інтерфейсів розташовується на зовнішньому рівні.

Також варто відзначити, що всі зовнішні сховища, як бази даних, файли, зовнішні веб-сервіси, від яких ми можемо отримувати дані - все це є зовнішнім по відношенню до архітектури.



Рисунлк 2.8 – ONION архітектура

В підготовку моделі входить правильне форматування даних та виділення потрібних, для того щоб не перенавантажувати інтерфейс користувача. На рисунку 2.9 зображено модель взаємодії компонентів клієнтської частини між собою, та взаємодію її з серверною частиною.

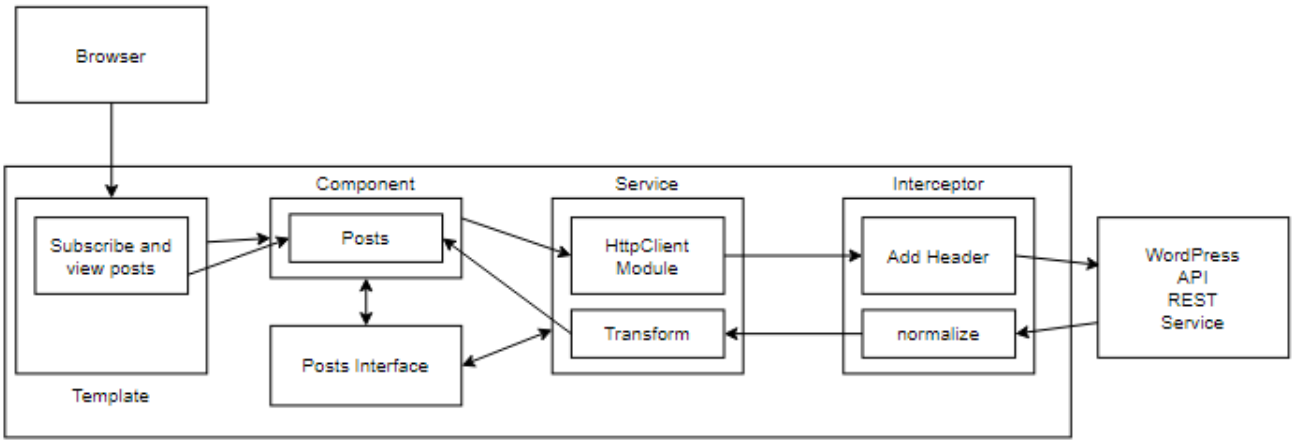


Рисунок 2.9 – Модель взаємодії клієнтської та серверної частини

Завдання клієнтської частини (програми-клієнта) полягає у взаємодії з користувачем, передачі запиту користувача серверу, отримання результату запиту від серверної частини (програми-сервера) і представлення його в

зручному для користувача вигляді. Програма-сервер обробляє запити клієнта і видає відповіді.

Класичні приклади: Web-технології (клієнт-браузер, Web-сервер), робота з розподіленими системами керування базами даних (СКБД).

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

2.5 Висновки

1. Проаналізовано інформаційне забезпечення. Наведено узагальнену модель системи, що складається з користувача, клієнтского додатку, серверної частини та бази даних.
2. Розглянуто варіанти практичної реалізації веб-додатків та технології, що для цього використовуються, принципи їх роботи і базові моделі реалізації.
3. Розроблено метод проведення багатокритеріального оцінювання робіт.
4. Розглянуто ключові персони, user story та user scenario. Створено модель роботи системи «Majestics».
5. Розроблено макети веб-інтерфейсу, продемонстровано вікна авторизації, профілю, вікно конкурсних робіт та подання заявок. Розроблені макети дозволяють оптимально розмістити елементи на екрані під час створення інтерфейсу.

6. Розглянуто архітектуру логіки веб-додатку, можливі реалізації зв'язку з серверною частиною і було обрано найбільш гнучкий і якісний підхід.

3 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Починаючи новий проект, часто стикаються з вибором: який JavaScript фреймворк вибрати для сайту - Vue.js, React або Angular? Відмінності між ними є і досить суттєві. Однак кожен із них підходить для вирішення завдань. Тому залишається відкритим питання ефективності роботи.

Відкритих даних щодо широти використання фреймворків немає, проте непрямий аналіз відкритих вакансій на сайті Indeed.com дає досить цікавий розподіл:

1 місце – React 78,1%

2 місце – Angular 21%

3 місце – Vue.js 0,8%

4 місце – без уточнення середовища 0,1%.

Javascript фреймворки насамперед варто порівняти за рендерингом сторінки. Сучасна архітектура допускає два види: за клієнта (сторінка відмальовується з допомогою потужностей ПК користувача) чи за сервера. Нижче наведена таблиця порівнянь обраних фреймворків [23].

Таблиця 3.1 – Порівняння фреймворків

	Angular	React	Vue
Тип	Фреймворк	Бібліотека для побудови інтерфейсу користувача	Фреймворк
Причина обрати	Якщо хочете використовувати TypeScript	Якщо дотримуєтеся підходу «пишемо все на JavaScript»	Легкий JavaScript та HTML

Продовження таблиці 3.1 – Порівняння фреймворків

	Angular	React	Vue
Типи додатків	Нативні, гібридні та веб-додатки	Односторінкові та мобільні програми	Просунуті односторінкові програми та підтримка нативних програм
Ідеально підходить для створення	Великомасштабних додатків із багатим функціоналом	Сучасних веб-додатків та нативних додатків для iOS та Android	Веб-додатків та односторінкових додатків
Простота вивчення	Важко	Трохи легше, ніж Angular	Просто
Модель	MVC	Віртуальна DOM	Віртуальна DOM
Мова системи	TypeScript	JavaScript	JavaScript
Підтримка спільнотою	Велика спільнота розробників	Спільнота розробників Facebook	Проект із відкритим вихідним кодом, який підтримує спільнота
Переважна мова розробки	TypeScript	JSX - JavaScript XML	HTML-шаблони та JavaScript

Найголовнішими перевагами Angular серед усіх інших альтернатив є те, що він був створений для розробки веб-додатків або сайтів, що імітують роботу десктопного додатку. Сам же фреймворк розробляється і підтримується компанією Google, що дає розробникам гарантію якості і швидкого вирішення проблем службою підтримки в разі необхідності. За допомогою його

фукнціоналу розробник може створювати HTML елементи самостійно, які будуть мати свій унікальний вигляд та поведінку. За рахунок цього більша частина часу витрачається лише на початку розробки, а саме на моменті налаштування та створення цих елементів, адже розробник може використовувати їх безліч разів, змінюючи їх вигляд або поведінку.

3.2 Розробка веб-компонентів графічного дизайну

Весь інтерфейс веб-додатку в реалізації на фреймворку Angular побудований шляхом створення своїх користувацьких компонентів. Нижче наведено HTML код одного з таких, а саме сторінки проведення конкурсів. На рисунку 3.1 зображено блок вибору категорії з випадаючого списку. На рисунку 3.2 можна побачити блок вибору категорії за допомогою екранних кнопок. Далі наведено код для відображення робіт.

```
<div class="main">
  <h3 class="title">XIX International Competition</h3>
  <!--<div class="f_select">
    <div class="btn-group">
      <button type="button" class="btn btn-primary">School</button>
      <button type="button" class="btn btn-primary">College</button>
      <button type="button" class="btn btn-primary">University </button>
    </div>
  </div>
  <div class="s_select">
    <div class="btn-group">
      <button type="button" class="btn btn-primary">3D animation </button>
      <button type="button" class="btn btn-primary">2D bitmap graphics </button>
      <button type="button" class="btn btn-primary">GIF animation </button>
      <button type="button" class="btn btn-primary">2D vector graphics </button>
      <button type="button" class="btn btn-primary">graphic implementation</button>
      <button type="button" class="btn btn-primary">3D graphics</button>
      <button type="button" class="btn btn-primary">information content</button>
      <button type="button" class="btn btn-primary">software implemetation</button>
      <button type="button" class="btn btn-primary">flash animation</button>

    </div>
  </div>-->
```

Рисунок 3.1 – Блок вибору категорії зі списку

```
<div class="buttons">
  <div class="f_button">
    <button type="button" class="btn btn-dark"> School</button>
    <button type="button" class="btn btn-dark"> College</button>
    <button type="button" class="btn btn-dark"> University </button>
  </div>
  <div class="s_button">
    <button type="button" class="btn btn-light">3D animation </button>
    <button type="button" class="btn btn-light">2D bitmap graphics </button>
    <button type="button" class="btn btn-light"> GIF animation </button>
    <button type="button" class="btn btn-light">2D vector graphics </button>
    <button type="button" class="btn btn-light"> graphic implementation</button>
    <button type="button" class="btn btn-light">3D graphics</button>
    <button type="button" class="btn btn-light"> information content</button>
    <button type="button" class="btn btn-light"> software implemetation</button>
    <button type="button" class="btn btn-light"> flash animation</button>

  </div>
</div>
```

Рисунок 3.2 – Блок вибору категорії екранними кнопками

```
  <div class="grid">
    <h4 class="total"> Table view:</h4>
    <div class="column">
      <imgsrc="../../assets/images/main/img1.png" style="width:100%">
      <img src="../../assets/images/main/img2.png" style="width:100%">
      <img src="../../assets/images/main/img1.png" style="width:100%">
      <img src="../../assets/images/main/img2.png" style="width:100%">
      <img src="../../assets/images/main/img1.png" style="width:100%">
      <img src="../../assets/images/main/img2.png" style="width:100%">
      <img src="../../assets/images/main/img1.png" style="width:100%">
      <img src="../../assets/images/main/img2.png" style="width:100%">
      <img src="../../assets/images/main/img1.png" style="width:100%">
    </div>
  </div>
 <button type="button" class="btn btn-info">Send work</button>
</div>
```

Рисунок 3.3 – Блок відображення робіт

На рисунку 3.4 зображено компонент головної інформації про веб-додаток. Для повноцінного фукціонування компоненту йому потрібна частина коду, яка буде виконувати клієнтську логіку, а тобто відповідний TypeScript файл.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.css']
})
export class MainComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }
}
```

Рисунок 3.4 – Компонент головної інформації

Оскільки цей компонент призваний лише відображати дані про конкурс, клієнтська логіка тут не потрібна. TypeScript файл визначає лише зв'язки між HTML кодом, логікою та стилями, які відповідають даному компоненту.

3.3 Розробка модуля ідентифікації користувача

Ідентифікація об’єкту є однією з функцій підсистеми захисту. Ця функція виконується першою у випадку спроби об’єкта ввійти в мережу (підключитися до мережі). Якщо процедура ідентифікації завершилася успішно, даний об’єкт вважається законним для даної мережі.

Наступний крок – автентифікація об’єкту (перевірка справжності об’єкту). Ця процедура встановлює чи є даний об’єкт саме таким, яким він себе оголошує.

Після того, коли об’єкт ідентифіковано та підтверджена його справжність, можна встановити сферу його дій та доступні йому ресурси. Таку процедуру називають наданням повноважень (авторизацією). Перелічені три процедури ініціалізації є процедурами захисту і відносяться до одного об’єкту інформаційної системи.

При захисті каналів передачі даних підтвердження справжності (автентифікація) об’єктів означає взаємне встановлення справжності об’єктів, які зв’язуються між собою. Процедура взаємної автентифікації зазвичай здійснюється на початку сеансу в процесі встановлення з’єднання абонентів. Мета даної процедури – забезпечити впевненість в тому,

що з’єднання встановлено з законним об’єктом і вся інформація дійде до місця призначення.

Одним із способів перешкодити таким діям є присвоєння об’єктам чи суб’єктам якогось унікального ідентифікатора (ознаки). Ідентифікація дозволяє користувачу назвати себе (повідомити своє ім’я). На рисунку 3.5 показано модуль ідентифікації користувача.

```
try
            {
                var result = await _dbContext.Works
                    .Include(x => x.User)
                    .Include(x => x.Marks)
                    .Where(x => x.Id == workId)
                    .Select(x => new WorkViewModel
                    {
                        User = new UserViewModel
                        {
                            FirstName = x.User.FirstName,
                            Institute = x.User.Institute,
                            LastName = x.User.LastName
                        },
                        Description = x.Description,
                        Title = x.Title,
                        Source = x.Source,
                        WorkId = x.Id
                    }).FirstOrDefaultAsync();

                var marks = _dbContext.Marks.Where(x => x.WorkId == workId).ToList();
                var juryMarks = marks.Where(a => a.UserType.HasValue && a.UserType == UserType.Jury).ToList();
                var userMarks = marks.Where(a => a.UserType.HasValue && a.UserType == UserType.User).ToList();
                var anonMarks = marks.Where(a => !a.UserType.HasValue).ToList();

                result.AnonMark = anonMarks.Any() ? Convert.ToInt32(anonMarks.Average(x => x.Value)) : 0;
                result.JuryMark = juryMarks.Any() ? Convert.ToInt32(juryMarks.Average(x => x.Value)) : 0;
                result.UsersMark = userMarks.Any() ? Convert.ToInt32(userMarks.Average(x => x.Value)) : 0;

                return result;

            }
            catch (Exception ex)
            {
                throw;
            }
```

Рисунок 3.5 – Ідентифікація користувача

3.4 Розробка модуля оцінювання користувачів

Модуль оцінювання робіт користувачів був розроблений для того, аби кожен міг по-своєму оцінити роботу за певними критеріями. Оскільки додаток дає можливість голосування не тільки для журі та організаторів, а і для зареєстрованих учасників та анонімних користувачів, то для кожного з них був введений свій коефіцієнт впливу на остаточну оцінку. Нижче наведено рисунки

модуля оцінки. На рисунку 3.6 зображено створення моделей результатів авторизації.

```
import { Injectable } from '@angular/core';
import { User, UserManager, WebStorageStateStore } from 'oidc-client';
import { BehaviorSubject, concat, from, Observable } from 'rxjs';
import { filter, map, mergeMap, take, tap } from 'rxjs/operators';
import { ApplicationPaths, ApplicationName } from './api-authorization.constants';

export type IAuthenticationResult =
  SuccessAuthenticationResult |
  FailureAuthenticationResult |
  RedirectAuthenticationResult;

export interface SuccessAuthenticationResult {
  status: AuthenticationResultStatus.Success;
  state: any;
}

export interface FailureAuthenticationResult {
  status: AuthenticationResultStatus.Fail;
  message: string;
}

export interface RedirectAuthenticationResult {
  status: AuthenticationResultStatus.Redirect;
}

export enum AuthenticationResultStatus {
  Success,
  Redirect,
  Fail
}

export interface IUser {
  name: string;
}

@Injectable({
  providedIn: 'root'
})
```

Рисунок 3.6 – Моделі результатів авторизації

На рисунку 3.7 показано базову логіку авторизації. Далі авторизуємо користувача в системі і генеруємо токени доступу (рисунок 3.8).

```
        public isAuthenticated(): Observable<boolean> {
          return this.getUser().pipe(map(u => !!u));
        }

        public getUser(): Observable<IUser | null> {
          return concat(
            this.userSubject.pipe(take(1), filter(u => !!u)),
            this.getUserFromStorage().pipe(filter(u => !!u), tap(u =>
this.userSubject.next(u))),
            this.userSubject.asObservable());
        }

        public getAccessToken(): Observable<string> {
          return from(this.ensureUserManagerInitialized())
            .pipe(mergeMap(() => from(this.userManager.getUser())),
              map(user => user && user.access_token));
        }
```

Рисунок 3.7 – Логіка авторизації

```
        // We try to authenticate the user in three different ways:
        // 1) We try to see if we can authenticate the user silently. This happens
        //    when the user is already logged in on the IdP and is done using a hidden
iframe
        //    on the client.
        // 2) We try to authenticate the user using a PopUp Window. This might fail if
there is a
        //    Pop-Up blocker or the user has disabled PopUps.
        // 3) If the two methods above fail, we redirect the browser to the IdP to
perform a traditional
        //    redirect flow.
        public async signIn(state: any): Promise<IAuthenticationResult> {
          await this.ensureUserManagerInitialized();
          let user: User = null;
          try {
            user = await this.userManager.signinSilent(this.createArguments());
            this.userSubject.next(user.profile);
            return this.success(state);
          } catch (silentError) {
            // User might not be authenticated, fallback to popup authentication
            console.log('Silent authentication error: ', silentError);

            try {
              if (this.popUpDisabled) {
                throw         new         Error('Popup         disabled.         Change
\'authorize.service.ts:AuthorizeService.popupDisabled\' to false to enable it.');
              }
              user = await this.userManager.signinPopup(this.createArguments());
              this.userSubject.next(user.profile);
              return this.success(state);
            } catch (popupError) {
              if (popupError.message === 'Popup window closed') {
                // The user explicitly cancelled the login action by closing an opened
popup.
                return this.error('The user closed the window.');
              } else if (!this.popUpDisabled) {
                console.log('Popup authentication error: ', popupError);
              }

              // PopUps might be blocked by the user, fallback to redirect
              try {
                await this.userManager.signinRedirect(this.createArguments(state));
                return this.redirect();
              } catch (redirectError) {
                console.log('Redirect authentication error: ', redirectError);
                return this.error(redirectError);
              }
            }
          }
        }
```

Рисунок 3.8 – Авторизація користувача в системі

3.5 Розробка сервісів зв'язку клієнтської частини з серверною частиною

Для зв'язку з серверною частиною використовуються окремі типи файлів, в Angular вони називаються сервісами.

Сервіси в Angular представляють досить широкий спектр класів, які виконують деякі специфічні завдання, наприклад, логування, роботу з даними і т.д.

На відміну від компонентів і директив сервіси не працюють з розміткою html, не роблять на неї прямого впливу. Вони виконують строго певне і досить вузьке завдання.

Стандартні завдання сервісів [24]:

- Надання даних з додатком. Сервіс може сам зберігати дані в пам'яті, або для отримання даних може звертатися до будь-якого джерела даних, наприклад, до сервера.
- Сервіс може представляти канал взаємодії між окремими компонентами програми.
- Сервіс може інкапсулювати бізнес-логіку, різні обчислювальні завдання, завдання по логуванню, які краще виносити з компонентів. Тим самим код компонентів буде зосереджений безпосередньо на роботі з поданням. Крім того, тим самим ми також можемо вирішити проблему повторення коду, якщо нам буде потрібно виконати одну і ту ж задачу в різних компонентах і класах

Сервіси мають окремий тип реєстрації. На рисунку 3.9 зображено імпорт залежностей, необхідних для роботи додатку.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { RouterModule } from '@angular/router';

import { AppComponent } from './app.component';
import { NavMenuComponent } from './nav-menu/nav-menu.component';
import { HomeComponent } from './home/home.component';
import { CounterComponent } from './counter/counter.component';
import { FetchDataComponent } from './fetch-data/fetch-data.component';
import { ApiAuthorizationModule } from 'src/api-authorization/api-
authorization.module';
import { AuthorizeGuard } from 'src/api-authorization/authorize.guard';
import { AuthorizeInterceptor } from 'src/api-
authorization/authorize.interceptor';
import { HeaderComponent } from './header/header.component';
```

Рисунок 3.9 – Імпорт залежностей

Далі можна продемонструвати реєстрацію компонентів, які реалізовані в додатку (рисунок3.10) та компонентів, що реалізовані в сторонніх бібліотеках.

```
@NgModule({
  declarations: [
    AppComponent,
    NavMenuComponent,
    HomeComponent,
    CounterComponent,
    FetchDataComponent,
    HeaderComponent,
    MainComponent,
    FooterComponent
  ],
```

Рисунок 3.10 – Компоненти, які реалізовані в додатку

```
    imports: [
      BrowserModule.withServerTransition({ appId: 'ng-cli-universal' }),
      HttpClientModule,
      FormsModule,
      ApiAuthorizationModule,
      RouterModule.forRoot([
        { path: '', component: HomeComponent, pathMatch: 'full' },
        { path: 'counter', component: CounterComponent },
        { path: 'fetch-data', component: FetchDataComponent, canActivate:
[AuthorizeGuard] },
      ])
    ],
    providers: [
      { provide: HTTP_INTERCEPTORS, useClass: AuthorizeInterceptor, multi: true }
    ],
    bootstrap: [AppComponent]
  })
export class AppModule { }
```

Рисунок 3.11 – Компоненти, які реалізовані в сторонніх бібліотеках

Сам сервіс має вигляд звичайного класу. На рисунку 3.12 показано саму реєстрація компоненту та створення конструктору.

```
import { Component, OnInit } from '@angular/core';
import { AuthorizeService, AuthenticationResultStatus } from
'../authorize.service';
import { ActivatedRoute, Router } from '@angular/router';
import { BehaviorSubject } from 'rxjs';
import { LoginActions, QueryParameterNames, ApplicationPaths, ReturnUrlType }
from '../api-authorization.constants';

@Component({
    selector: 'app-login',
    templateUrl: './login.component.html',
    styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
    public message = new BehaviorSubject<string>(null);

    constructor(
        private authorizeService: AuthorizeService,
        private activatedRoute: ActivatedRoute,
        private router: Router) { }
```

Рисунок 3.12 – Створення конструктору

Також визначено статус користувача та перенаправлено його на необхідну сторінку (рисунок 3.13). Наведено авторизацію користувача (рисунок 3.14) та створення моделі результату авторизації (рисунок 3.15).

```
                    case LoginActions.Login:
                        await this.login(this.getReturnUrl());
                        break;
                    case LoginActions.LoginCallback:
                        await this.processLoginCallback();
                        break;
                    case LoginActions.LoginFailed:
                        const                          message                          =
this.activatedRoute.snapshot.queryParamMap.get(QueryParameterNames.Message);
                        this.message.next(message);
                        break;
                    case LoginActions.Profile:
                        this.redirectToProfile();
                        break;
                    case LoginActions.Register:
                        this.redirectToRegister();
                        break;
                    default:
                        throw new Error(`Invalid action '${action}'`);
                }
            }
```

Рисунок 3.13 – Перенаправлення користувача

```
            private async login(returnUrl: string): Promise<void> {
                const state: INavigationState = { returnUrl };
                const result = await this.authorizeService.signIn(state);
                this.message.next(undefined);
                switch (result.status) {
                    case AuthenticationResultStatus.Redirect:
                        break;
                    case AuthenticationResultStatus.Success:
                        await this.navigateToReturnUrl(returnUrl);
                        break;
                    case AuthenticationResultStatus.Fail:
                        await
this.router.navigate(ApplicationPaths.LoginFailedPathComponents, {
                            queryParams: { [QueryParameterNames.Message]: result.message
}
                        });
                        break;
                    default:
                        throw   new   Error(`Invalid   status   result   ${(result   as
any).status}.`);
                }
            }

            private async processLoginCallback(): Promise<void> {
                const url = window.location.href;
                const result = await this.authorizeService.completeSignIn(url);
                switch (result.status) {
                    case AuthenticationResultStatus.Redirect:
                        throw new Error('Should not redirect.');
                    case AuthenticationResultStatus.Success:
                        await this.navigateToReturnUrl(this.getReturnUrl(result.state));
                        break;
                    case AuthenticationResultStatus.Fail:
                        this.message.next(result.message);
                        break;
                }
            }
```

Рисунок 3.14 – Авторизація користувача

```
            private redirectToRegister(): any {
                this.redirectToApiAuthorizationPath(
                    `${ApplicationPaths.IdentityRegisterPath}?returnUrl=${encodeURI('/'
+ ApplicationPaths.Login)}`);
            }

            private redirectToProfile(): void {

this.redirectToApiAuthorizationPath(ApplicationPaths.IdentityManagePath);
            }

            private async navigateToReturnUrl(returnUrl: string) {
                await this.router.navigateByUrl(returnUrl, {
                    replaceUrl: true
                });
            }

            private getReturnUrl(state?: INavigationState): string {
                const    fromQuery    =    (this.activatedRoute.snapshot.queryParams    as
INavigationState).returnUrl;
                if (fromQuery &&
                    !(fromQuery.startsWith(`${window.location.origin}/`) ||
                        /\/[^\/].*/.test(fromQuery))) {
                    throw new Error('Invalid return url. The return url needs to have the
same origin as the current page.');
                }
                return (state && state.returnUrl) ||
                    fromQuery ||
                    ApplicationPaths.DefaultLoginRedirectPath;
            }

            private redirectToApiAuthorizationPath(apiAuthorizationPath: string) {
                const redirectUrl = `${window.location.origin}${apiAuthorizationPath}`;
                window.location.replace(redirectUrl);
            }
        }

        interface INavigationState {
            [ReturnUrlType]: string;
        }
```

Рисунок 3.15 – Створення моделі результату авторизації

3.6 Висновки

1. Обґрунтовано вибір мови програмування та фреймворку для реалізації завдання.

2. Розглянуто і наведено приклади реалізації графічного інтерфейсу у вигляді компонентів.

3. Розглянуто різні типи реєстрації файлів та їх структуру і використання. Розроблено сервіси зв'язку клієнтської частини з серверною частиною.

4. Розролено модулі ідентифікації користувачів та розглянуто і наведено приклад модуля оцінювання конкурсних робіт.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування програмного забезпечення

Тестування програмного забезпечення – процес аналізу програмного засобу та супутньої документації з метою виявлення дефектів і підвищення якості продукту [25].

Тестування програмного забезпечення – широке поняття, яке включає планування, проектування та, власне, виконання тестів (рисунок 4.1).

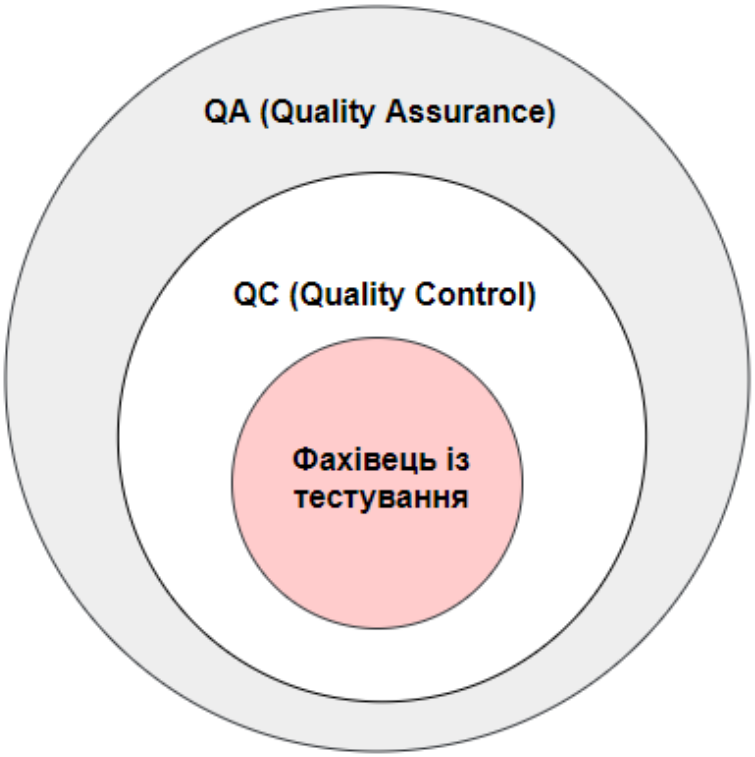


Рисунок 4.1 – Тестування ПЗ

QA (Quality Assurance) – забезпечення якості продукту. QA-фахівець контролює та забезпечує якість роботи продукту компанії. Він відповідає і окремі етапи розробки софта. Зокрема, за вибір інструментів для розробки, запобігання можливим проблемам. Ще він бере участь у процесі вдосконалення продукту. QA охоплює всі етапи розробки, включаючи опис проекту, власне, тестування, реліз і часто пост-релізний етап.

QC (Quality Control) – контроль якості продукту. Завдання QC-фахівця - перевірка конкретного продукту, що включає аналіз коду продукту, дизайну плюс тестування. QC-інженер розробляє стратегію тестування цілком певного тестування, взаємодіє з розробниками та організує саме тестування.

Фахівець із тестування займається виконанням тестів. Тестуванням називають перевірку відповідності результатів роботи програмного продукту на відповідність заданим критеріям. Тестувальники займаються тестуванням всього продукту в цілому або окремих компонентів. Тестування грає найважливішу роль забезпечення якості продукту.

Цілі тестування:

- Підвищити ймовірність того, що додаток, призначений для тестування, буде працювати правильно при будь-яких обставинах.
- Підвищити ймовірність того, що додаток, призначене для тестування, буде відповідати всім описаним вимогам [24].
- Надання актуальної інформації про стан продукту на даний момент.

Протестуємо відкриття власного профілю веб-системи. В боковому меню натиснемо на кнопку «My Profile». Має відкритись сторінка, де наведена вся інформація про користувача. Тут знаходиться: пошта, повне ім’я, вік, адреса, телефон, навчальний заклад, навички та інтереси учасника (рисунок 4.2).

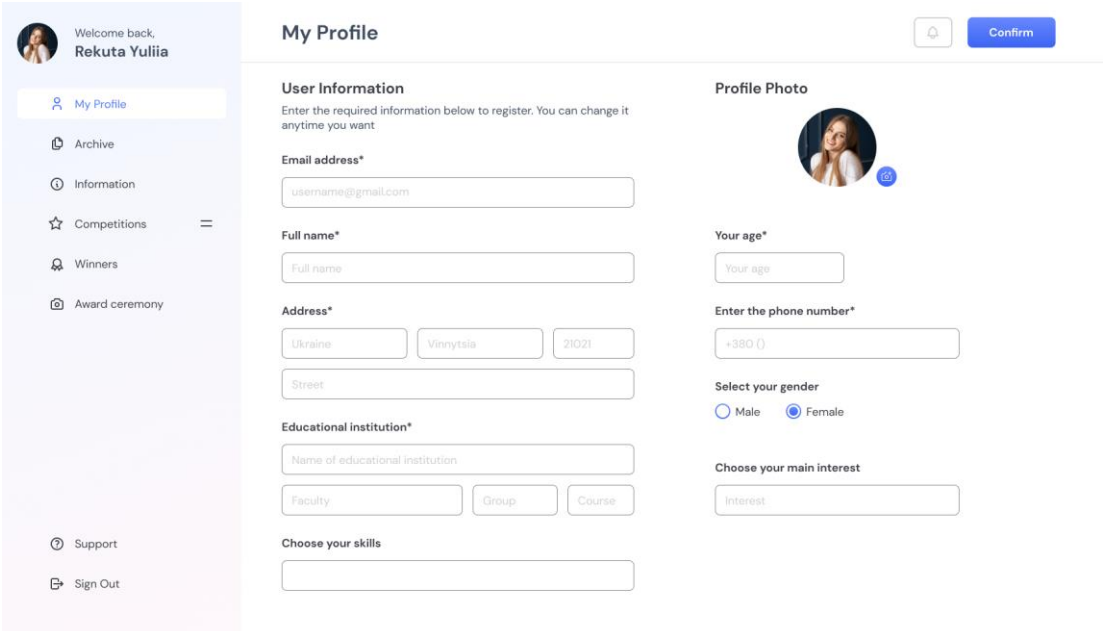


Рисунок 4.2 – Профіль користувача

Розглянемо більш детально саме бокове меню. Воно відкривається при наведенні, знаходяться там такі пункти як «My Profile», «Archive», «Information» та «Competitions», «Winners», «Award ceremony» (рис. 4.3).

Спробуємо натиснути на кнопку «Information». Відкривається вкладка з інформацією про загальні положення конкурсу, які також можна завантажити, переходячи за посиланням (рисунок 4.4)

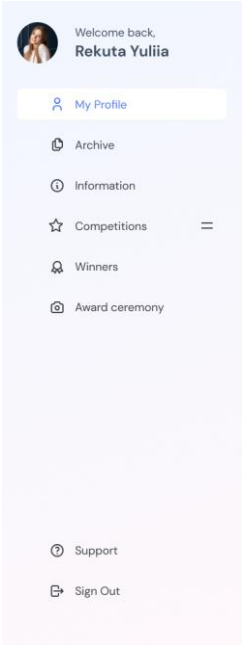


Рисунок 4.3 – Бокове меню

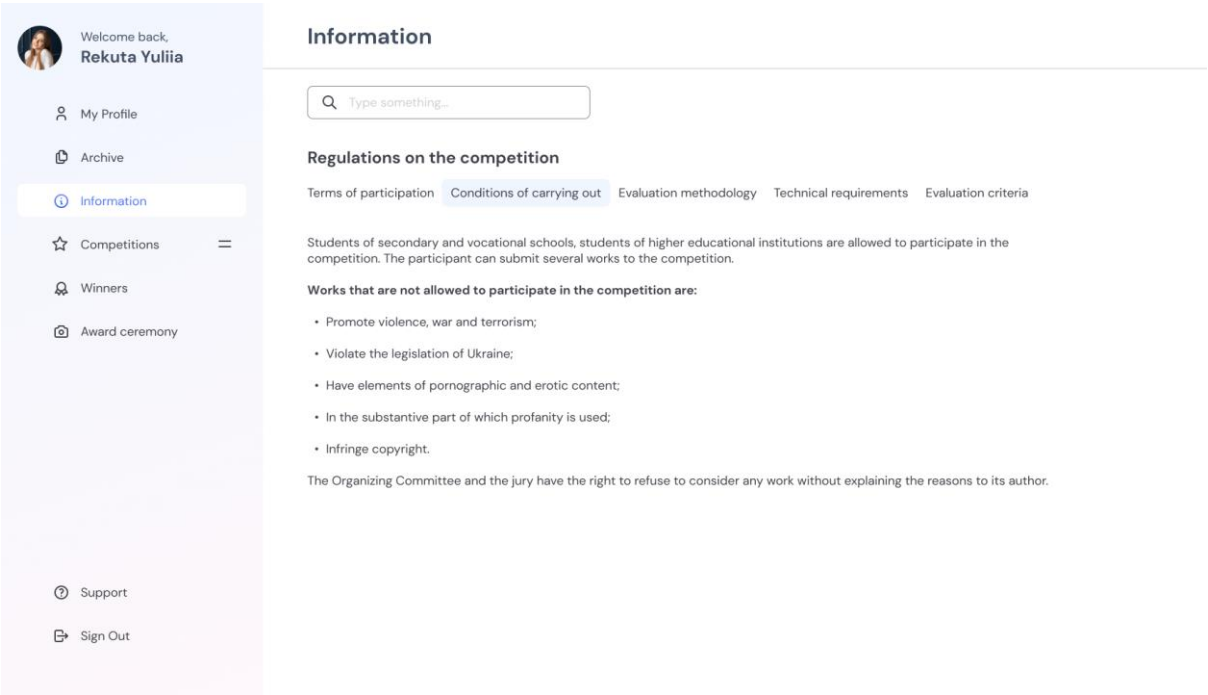


Рисунок 4.4 – Вікно інформації про конкурс

Також можна протестувати перегляд робіт на сайті. Для цього необхідно перейти на сторінку «Competitions», і, якщо конкурс активний, можна побачити роботи конкурсантів. При цьому, попередньо, потрібно виконати фільтрацію, наприклад, натиснувши на GIF-animation (рисунок 4.5).

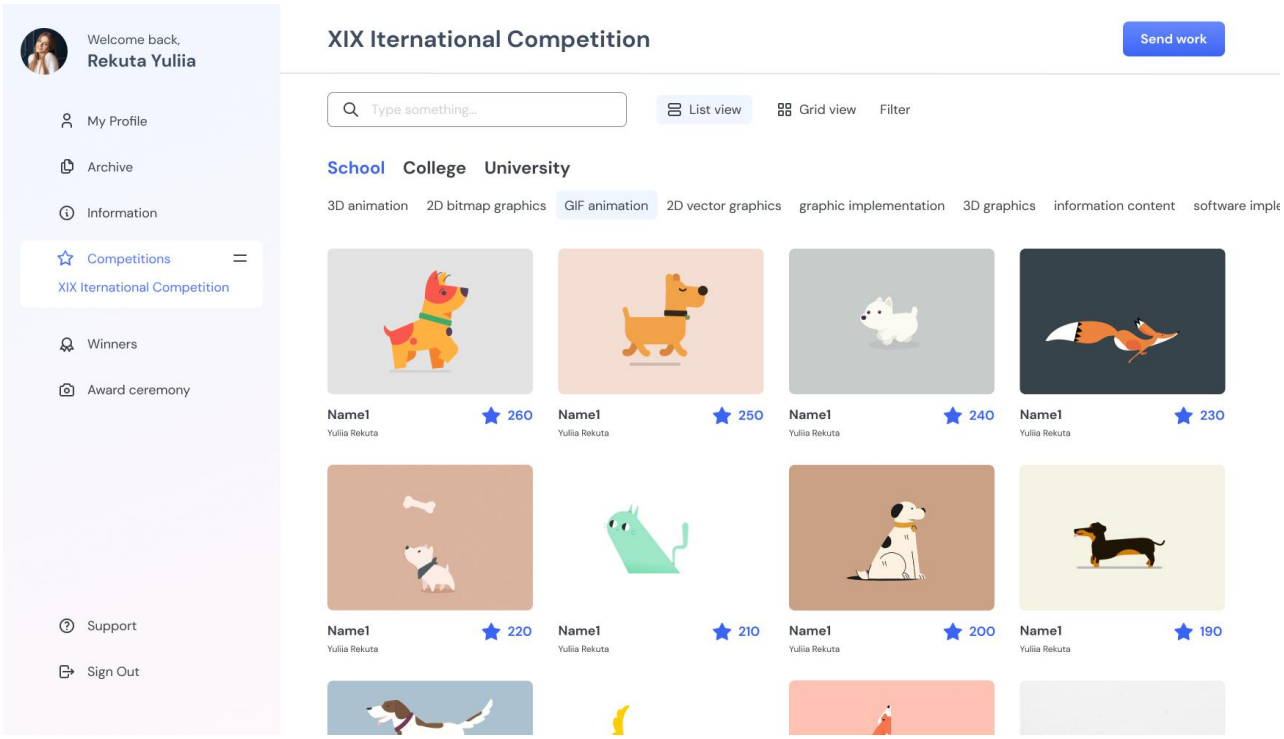


Рисунок 4.5 – Роботи конкурсантів

Тепер можна вибрати роботу, яку ми хочемо переглянути більш детально. Необхідно просто нажати на потрібну роботу і вона відкриється у новому вікні (рисунок 4.6). Також ми можемо її оцінити за вказаними критеріями (рисунок 4.7).

Натиснемо на кнопку «Archive». В даному вікні відображаються всі роботи учасника, які були подані раніше (рисунок 4.8). Є можливість переглянути їх як у вигляді списку, так і карточками. Тут наведена така інформація: Назва роботи, назва конкурсу, зайняте місце, керівник, дата проведення, архів з самою роботою, яку можна завантажити, та сертифікат (якщо було зайняте призове місце).

Також можна переглянути такі вікна як «Winners» (рисунок 4.9 та 4.10) та «Award ceremony» (рисунок 4.11). У вікні «Winners» зображено список конкурсів та їх переможців. «Award ceremony» демонструє фото та відеозвіт.

Результати тестування показали що програмний додаток працює вірно.

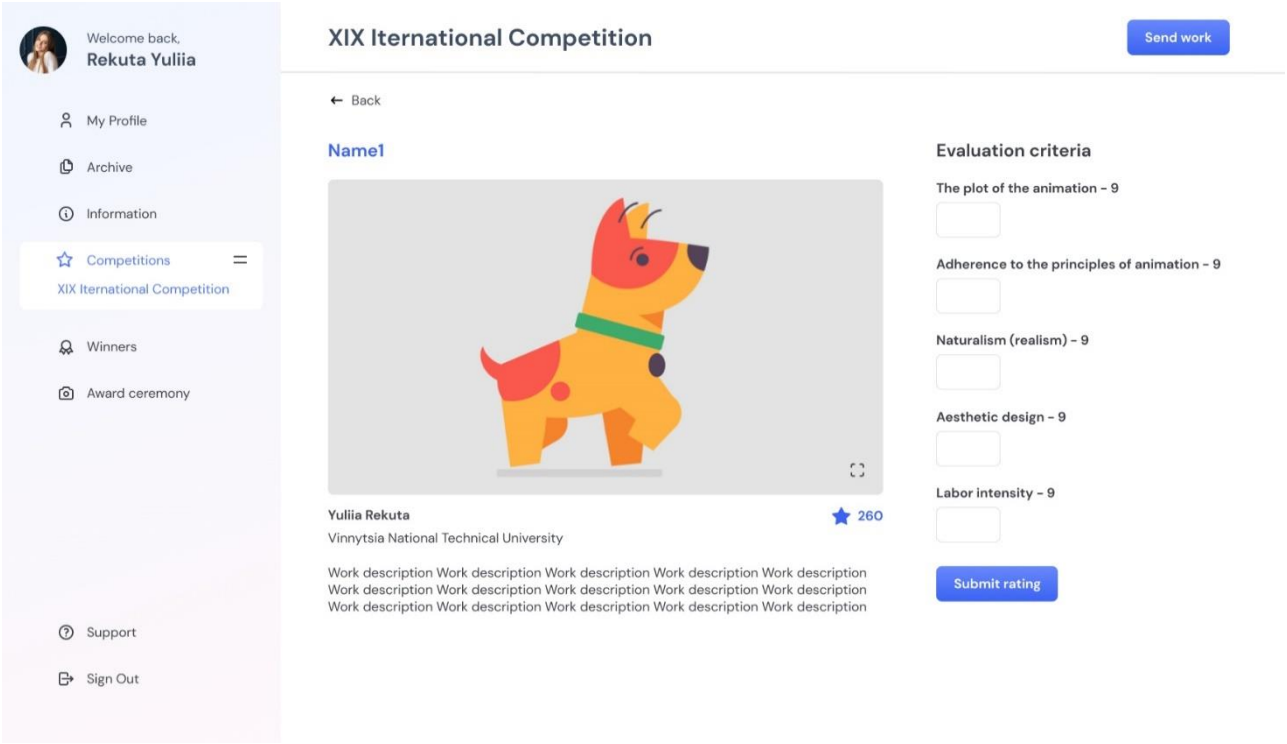


Рисунок 4.6 – Перегляд роботи

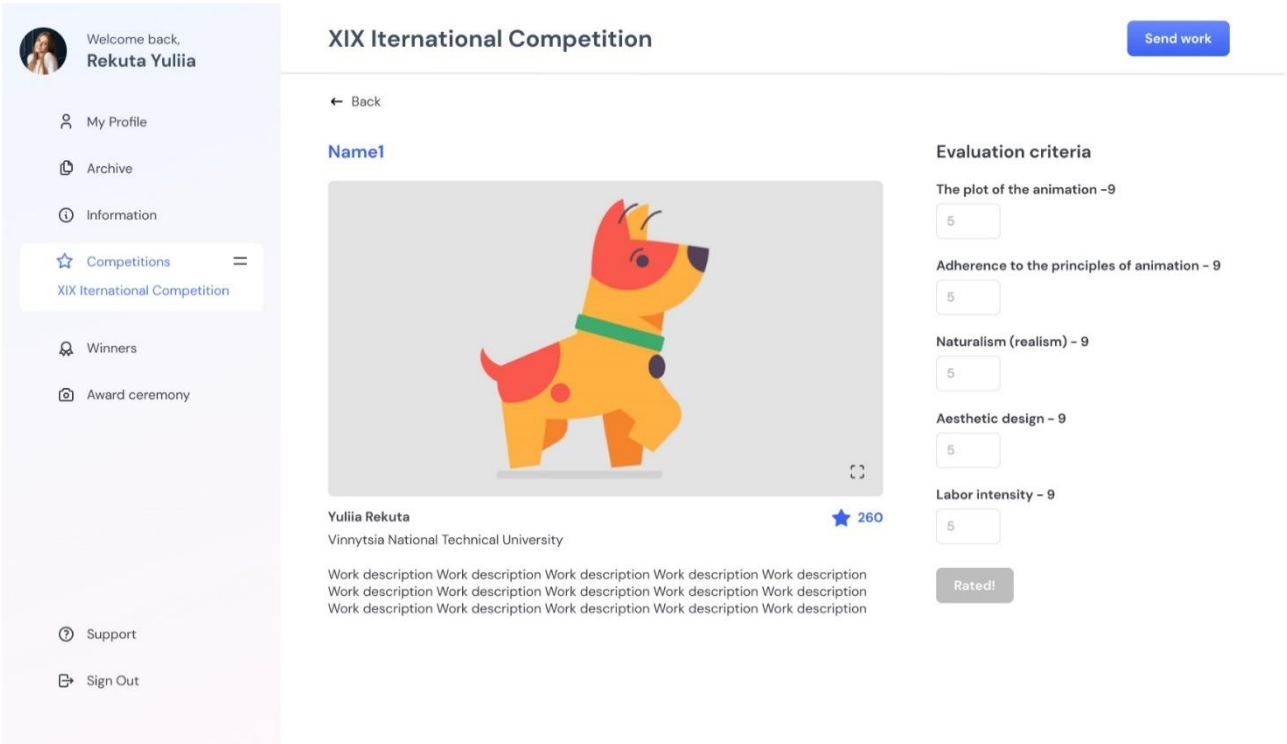


Рисунок 4.7 – Оцінка роботи

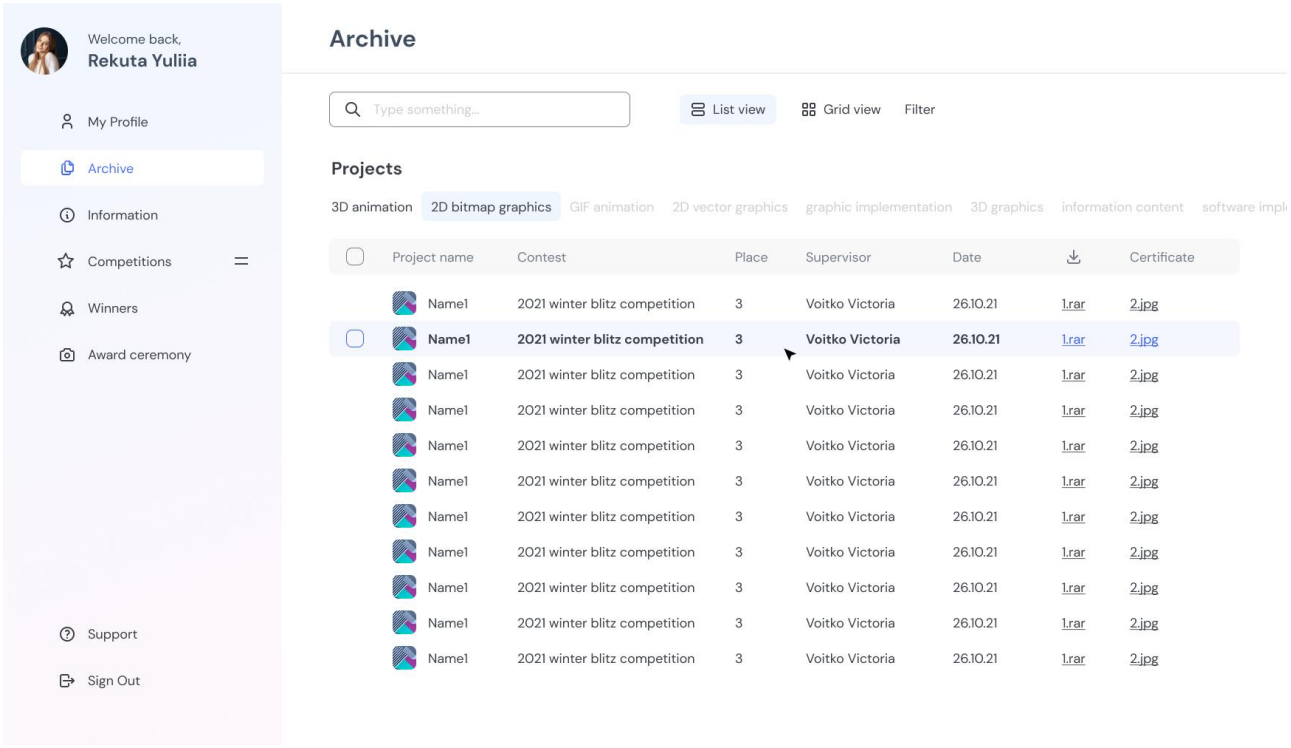


Рисунок 4.8 – Архів

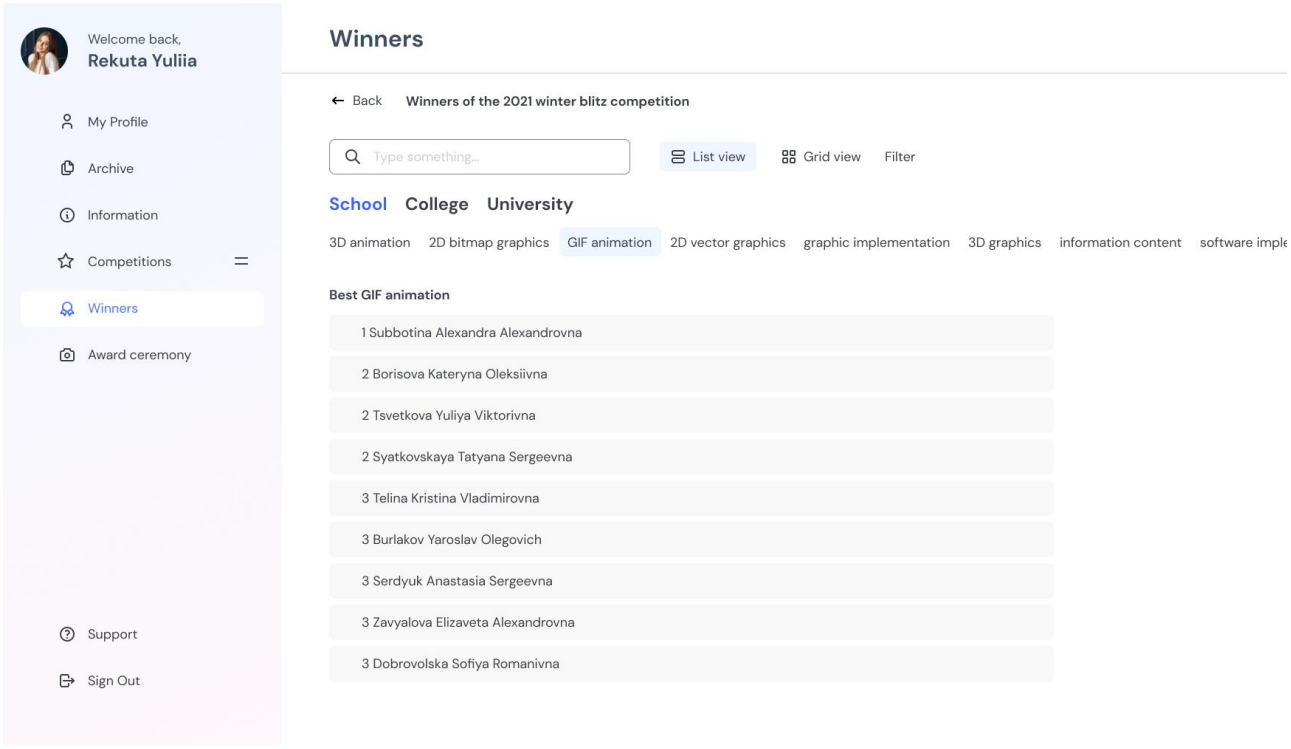


Рисунок 4.9 – Переможці вибраного конкурсу. Список

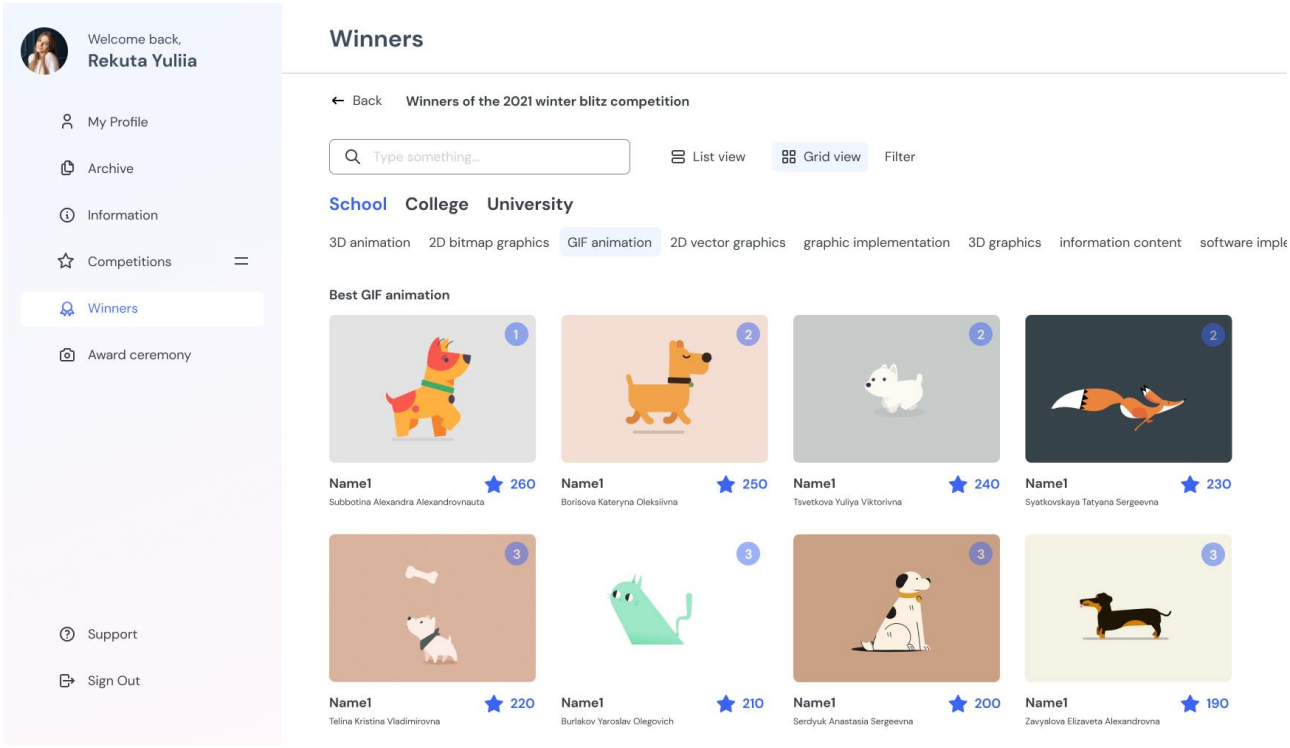


Рисунок 4.10 – Переможці вибраного конкурсу. Таблиця

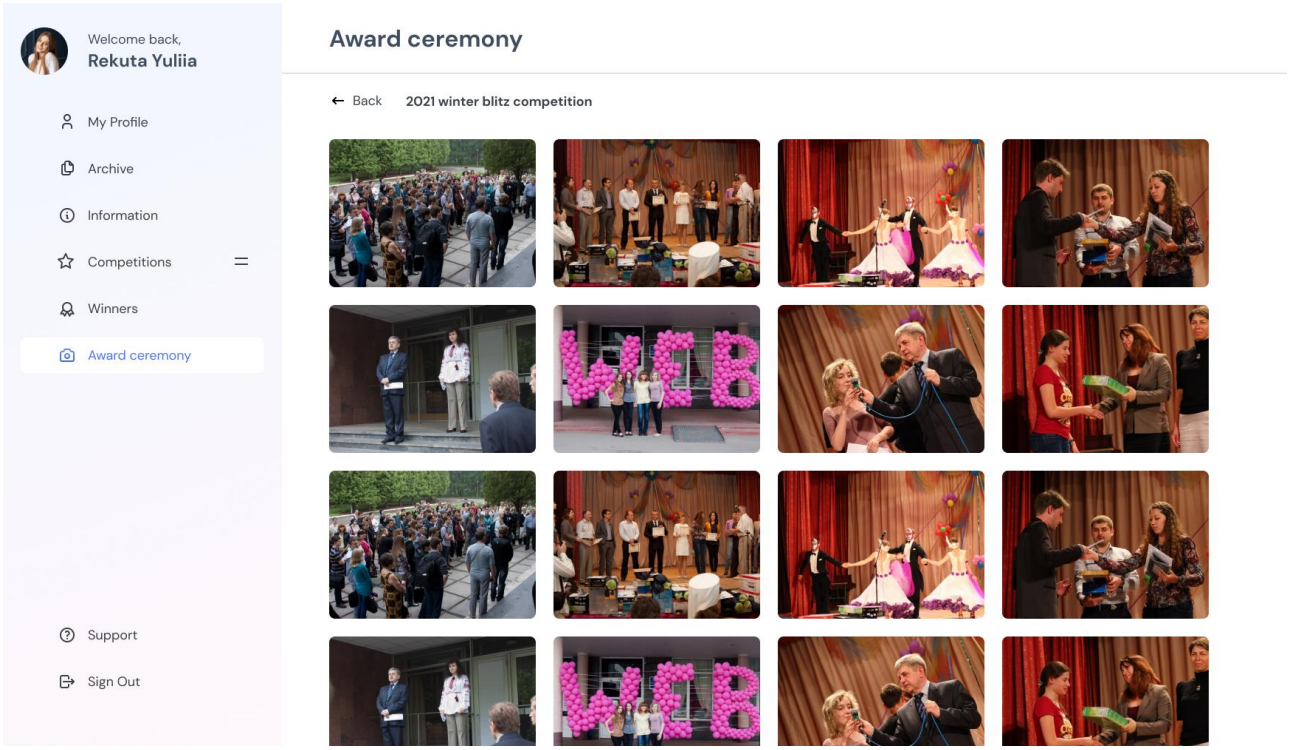


Рисунок 4.11 – Фото та відеозвіт

Під час тестування було виконано такі перевірки:

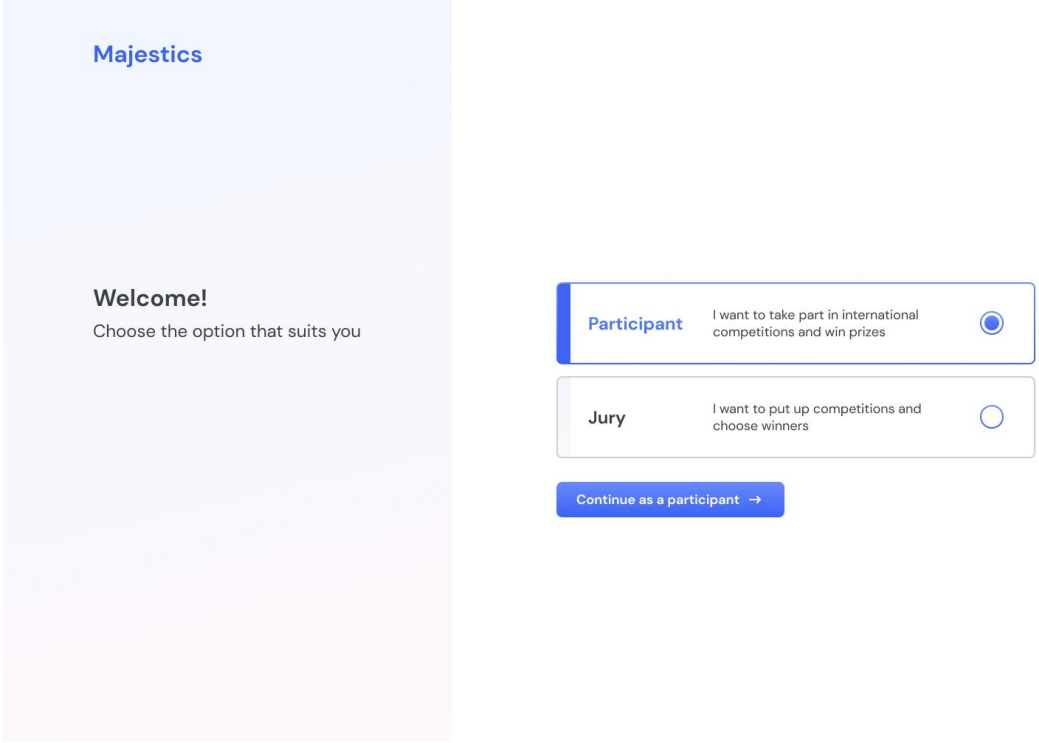
- перевірено коректність відображення кольорів на сайті, відображення шрифтів та зображень;
- перевірено зміст кожної сторінки;

- перевірено модулі авторизації та реєстрації;
- перевірено швидкість завантаження;
- перевірено можливість додавання робіт.

Тестування програмного продукту показало повну відповідність поставленому технічному завданню.

4.2 Тестування модуля авторизації та реєстрації

Виконаємо реєстрацію в якості учасника. Для цього зайдемо на сайт і побачимо вікно привітання (рисунок 4.12). Там обираємо учасника і тиснемо кнопку «Continue as a participant». Після того відкриється форма для заповнення необхідних даних (рис. 4.13). Тепер ми можемо ввести усі необхідні дані для реєстрації. Редагувати або ж добавити інформацію можна у вкладці профілю.



Рисунко 4.12 – Вікно привітання

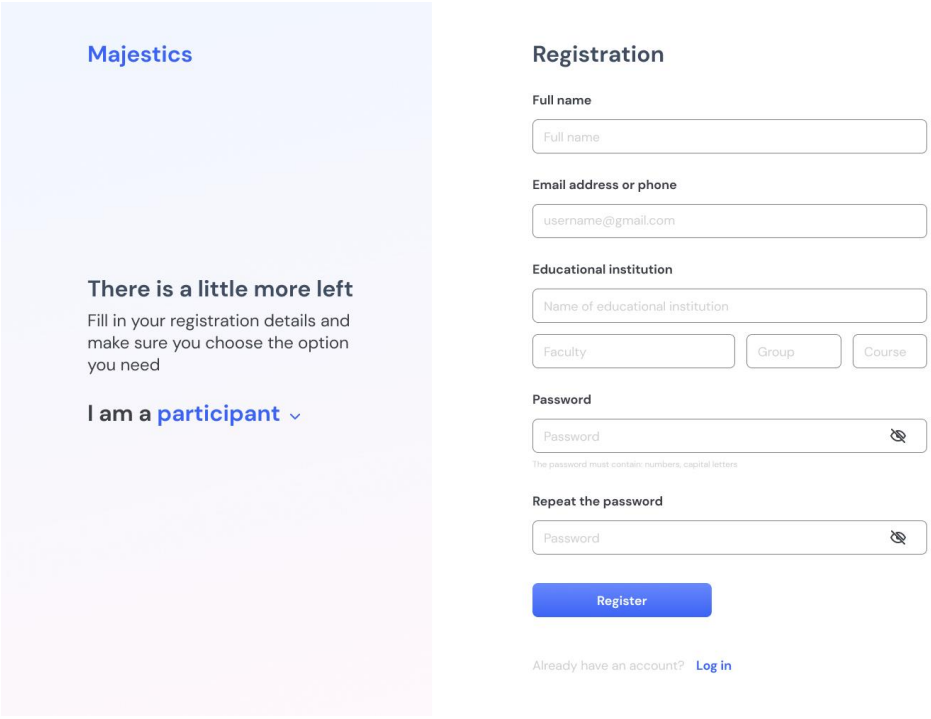


Рисунок 4.13 – Вікно реєстрації

Отже, веб-система працює коректно. На рисунку 4.14 зображено результат реєстрації – аккаунт зареєстрованого учасника. В даному акаунті користувач може переглянути всі свої дані, такі як прізвище, ім’я, по батькові, пошта, номер телефону, додати дані місця навчання. Якщо якесь з полів вказано не вірно – можемо змінити необхідне поле у будь-який момент.

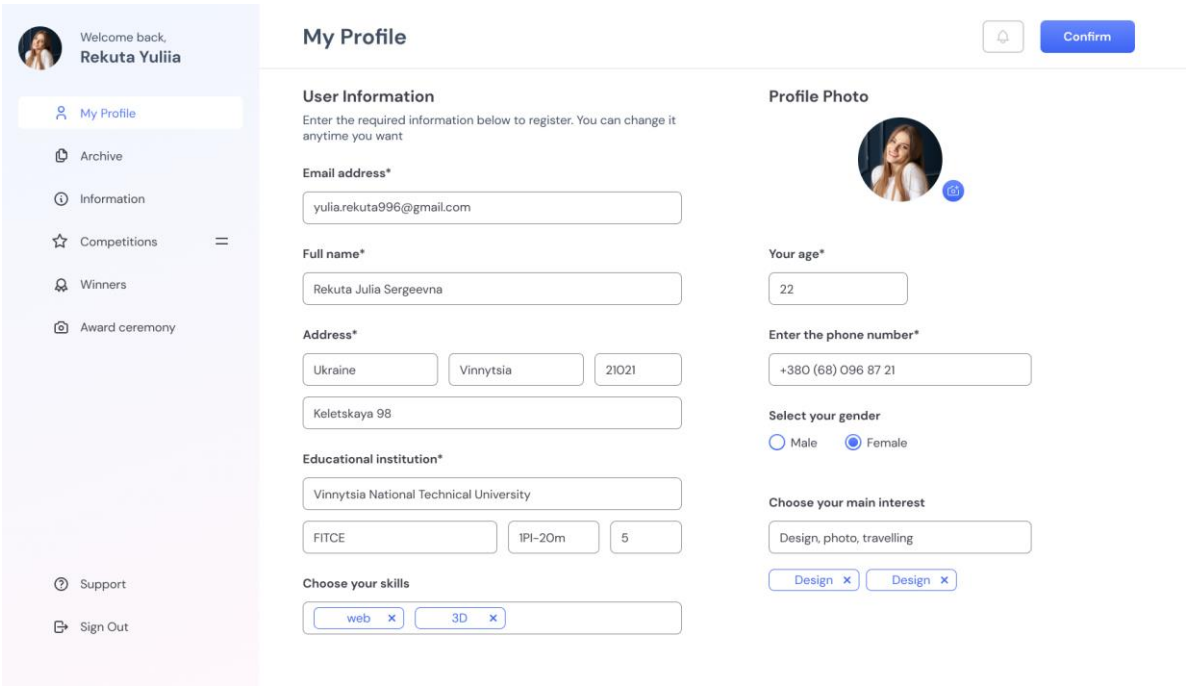


Рисунок 4.14 – Аккаунт зареєстрованого учасника

Перевіримо форму авторизації. Після введення даних і кліку на кнопку «Log in» відбувається перевірка даних і перехід на сторінку користувача з виведенням його інформації. Проте якщо одне з полів буде вказано не вірно, воно засвітиться червоним кольором (рисунок 4.15).

Зареєстрований учасник може переглянути, чи є активні конкурси та додати свою роботу. Натискаємо кнопку «Send work» і бачимо вікно введення інформації про свої роботу (рисунок 4.16). Мають бути заповнені такі поля як назва роботи, категорія, до якої відноситься, опис, керівник. Якщо робота виконувалась кількома учасниками, натискаємо кнопку «Add participants». На рисунку 4.17 бачимо вірно введені дані про роботу та вікно із повідомленням про успіх надсилання.

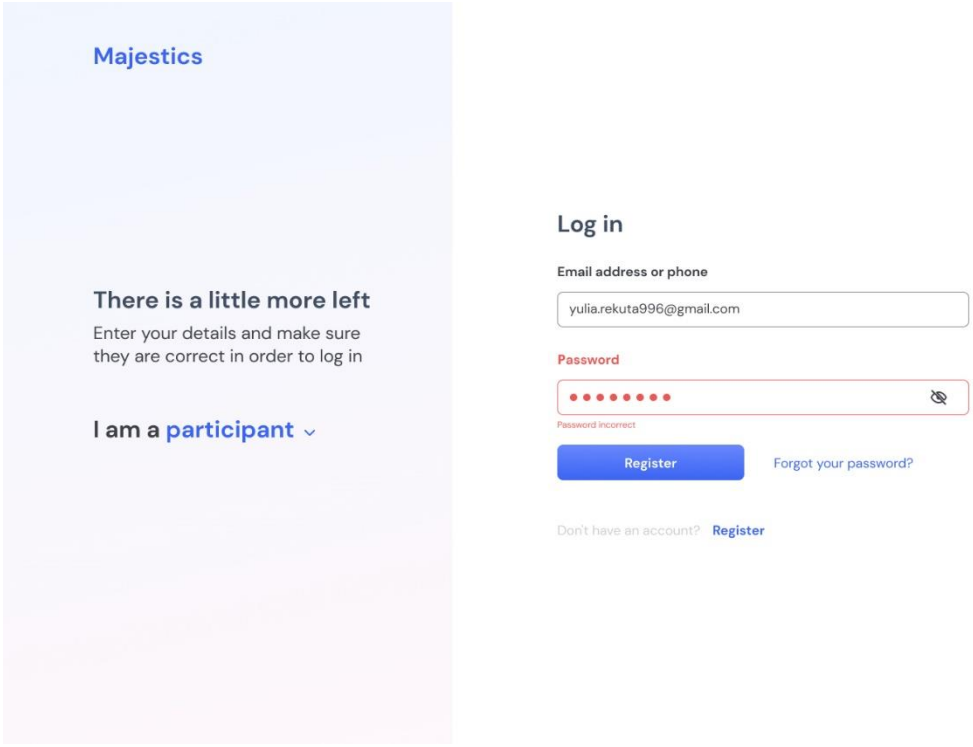


Рисунок 4.15 – Невірно введений пароль

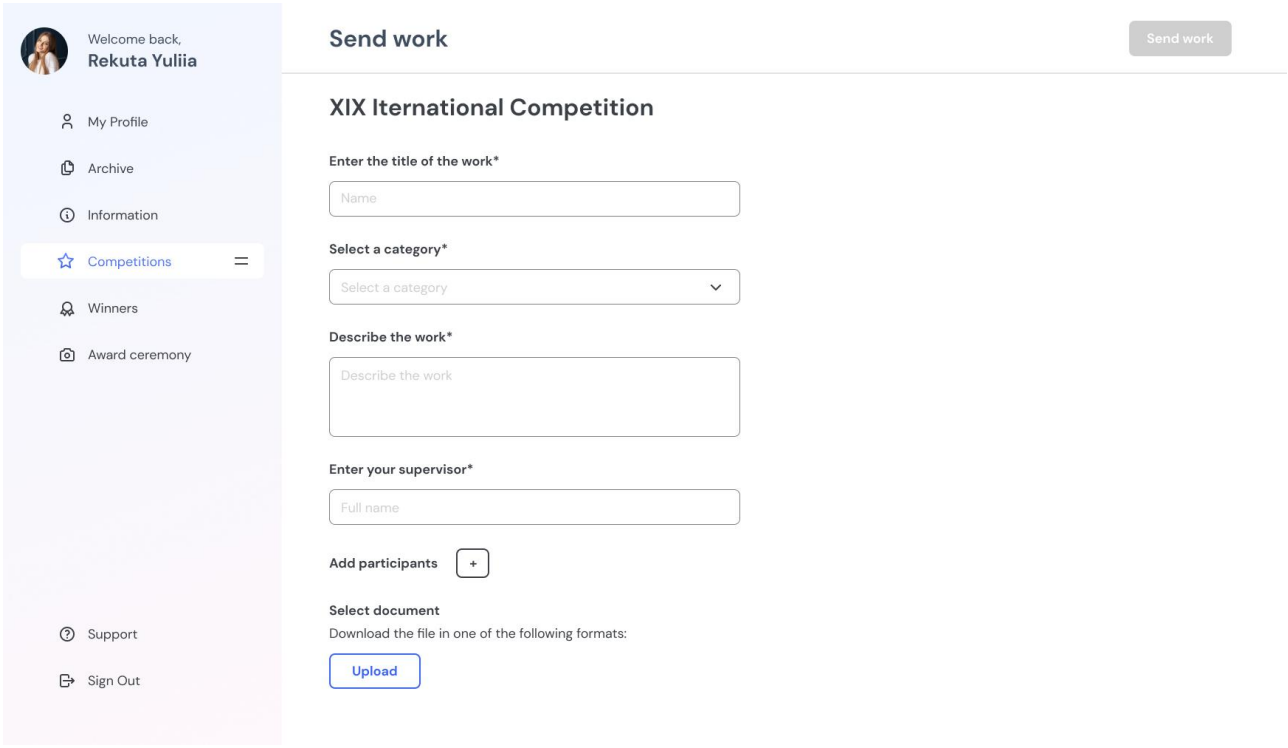


Рисунок 4.16 – Подача роботи на конкурс

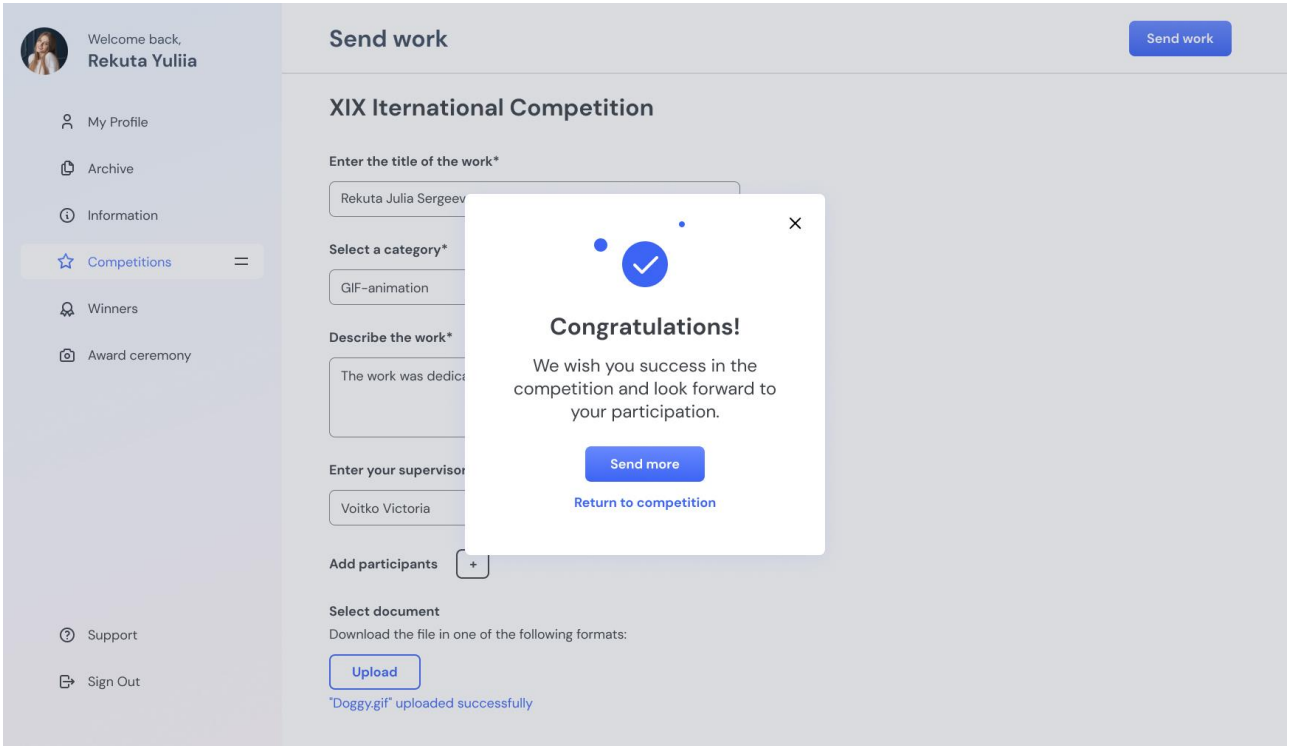


Рисунок 4.17 – Успіх надсилання

Після тестування модуля авторизації та реєстрації, та перегляду різних видів акаунтів стало зрозуміло, що веб-система працює відповідно поставленому завданню.

4.3 Висновки

Одним з важливих етапів розробки додатку є тестування, так як воно є гарантією якості розроблюваного додатку. Було протестовано вхід на сайт та перехід на інші його сторінки. Для перевірки працездатності було створено профілі журі та учасника, а також введення неправильних даних при вході в систему. Було переглянуто як усі роботи на сторінці, так і окрему роботу зі всіма критеріями оцінки в якості анонімного користувача.

Тестування програми показало повну її працездатність та відповідність поставленому технічному завданню. Розроблено веб-система відповідає усім поставленим завданням.

РОЗДІЛ 5

ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи на тему розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки.

Для проведення комерційного та технологічного аудиту залучимо 3-х експертів. Такими експертами будуть Войтко Вікторія Володимирівна (к.т.н., доц. кафедри ПЗ ВНТУ), Бурбело Сергій Михайлович (доц. кафедри ПЗ ВНТУ), Черноволик Галина Олександрівна (доц. кафедри ПЗ ВНТУ).

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями [26], які наведені в додатку Г.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Войтко В.В.	2. Бурбело С. М.	3. Черноволик Г. О.
	Бали, виставлені експертами:		
1	3	3	3
2	2	2	2
3	4	4	4
4	4	3	4
5	4	3	4
6	3	4	4
7	2	3	3

Продовження таблиці 5.1

8	4	4	4
9	4	4	4
10	4	4	4
11	4	4	4
12	3	4	4
Сума балів	$СБ_1 = 41$	$СБ_3 = 42$	$СБ_3 = 42$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = 41.6$		

За результатами розрахунків, наведених в таблиці 5.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. Середньоарифметична сума балів СБ, розрахована на основі висновків експертів дорівнює 41.6, а це означає, що науково-технічний рівень та комерційний потенціал нової розробки високий.

5.2 Прогнозування витрат на здійснення науково-дослідної роботи.

Витрати, пов’язані з проведенням науково-дослідної роботи під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- матеріали;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- інші витрати;
- накладні (загальновиробничі) витрати.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням

конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками.

Витрати на основну заробітну плату розробників (*Зо*) розраховують відповідно до посадових окладів працівників, за формулою 5.1.

$$З_о = \sum_{i=1}^{k} \frac{М}{Т_р} \cdot t \qquad (5.1)$$

де *k* – кількість посад розробників, залучених до процесу досліджень;

М – місячний посадовий оклад конкретного розробника, грн;

$Т_р$ – середня кількість робочих днів у місяці, $Т_р$ = 21 день;

t - кількість днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	10000	714,28	5	2 380,9
Розробник	7000	476,19	45	14 999,9
			Всього:	17 380,8

Розрахуємо додаткову заробітну плату. Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників:

$$З_{дод} = 0,1 \cdot 17\,380,8 = 1\,738,08 \text{ (грн)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників (формула 5.2):

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Нзп}{100}, \quad (5.2)$$

Нзп – норма нарахування на заробітну плату.

$$З_н = (17380,8+1738,08)*0,22 = 4\ 206,15 \text{ (грн)}.$$

Розрахуємо витрати на комплектуючі вироби *(Кв)* (таблиця 5.3), які використовують при дослідженні нового технічного рішення за формулою 5.3.

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.3)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.3 – Витрати на комплектуючі

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	190	1	190
Пачка паперу	уп.	132	1	132
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				365,2

До статті «Амортизація обладнання, програмних засобів та приміщень» відносять амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проведення науково-дослідної роботи, за його наявності в дослідній

організації або на підприємстві. Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою (5.4):

$$А=\frac{Ц}{Тв}\cdot\frac{Т}{12}, \quad (5.4)$$

де Ц – балансова вартість обладнання, грн;

Тв – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Т – Термін використання (Т=1,5 міс.).

Розрахунки для персонального комп'ютера наведено у таблиці 5.4.

Таблиця 5.4 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп'ютер	28000	2	1,5	1 750
			Всього:	1 750

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень. Витрати на силову електроенергію розраховуються за формулою 5.5:

$$В_е=\sum_{i=1}^{n}\frac{Wyi\cdot ti\cdot Це\cdot Квпі}{\eta_i}; \quad (5.5)$$

Де Wyi – встановлена потужність обладнання на певному етапі розробки, кВт (Wyi = 0,6кВт)

ti – тривалість роботи обладнання на етапі дослідження, год (ti = 190 год);

Це – вартість 1 кВт-години електроенергії, грн (це = 4,1 грн/кВт);

Квпі – коефіцієнт, що враховує використання потужності, Квпі <1 ($К_{впі} = 0,8$).

$$В_е = 0,6 \cdot 190 \cdot 4.1 = 467,4 \text{ (грн.)}$$

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{К} Ц_{іпрг} \cdot С_{прг.і} \cdot К_і \quad (5.6)$$

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

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

Кі – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (Кі = 1,10…1,12);

k – кількість найменувань програмних засобів.

Розрахунки для персонального комп’ютера наведено у таблиці 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Сервер	1	600	600
Доменне ім’я	1	120	120
		Всього:	792

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати розраховуються як 50…100% від суми основної заробітної плати дослідників (формула 5.7).

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} \quad (5.7)$$

$$I_в = (17380{,}8) \cdot \frac{50}{100\%} = 8\,690{,}4$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$Взаг = З_о + З_р + З_{дод} + З_н + А + К + В_е + В_{прг} + I_в$$

$В_{заг}$ = 2 380,9 + 14 999,9 + 1 738,08 + 4 206,15 + 1 750 + 365,2 + 467,4+ + 792 + 8 690,4 = 35 390,03 (грн)

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою 5.8:

$$ЗВ = \frac{Взаг}{\eta} \quad (5.8)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи; впровадження, то $\eta = 0{,}9$.

$$ЗВ = \frac{35\,390{,}03}{0{,}5} = 39\,322{,}3 \text{ (грн.)}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо розробником (замовником)

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

Збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання по-зитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою (5.9):

$$\Delta П_i = (\Delta П_я \cdot N + П_я \Delta N_i)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки в аналізованому році;

N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів наукової розробки;

ΔN – зміна основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 20 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 20 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 700 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_{2021} = 20 \cdot 700 + (200 + 20) \cdot 200 = 58000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_{2022} = 20 \cdot 700 + (200 + 20) \cdot (200 + 150) = 91000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_{2023} = 20 \cdot 700 + (200 + 20) \cdot (200 + 150 + 100) = 113000 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати розробник (замовник) від можливого впровадження науково-технічної розробки на власному підприємстві:

$$ПП = \sum_{i=1}^{T} \frac{\Delta \pi i}{(1+\tau)^{t}} \quad (5.10)$$

де $\Delta \pi i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, = *0,05…0,15*;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання підприємством збільшеної величини чистого прибутку в аналізованому році.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{58000}{(1+0,1)^1} + \frac{91000}{(1+0,1)^2} + \frac{113000}{(1+0,1)^3} = 218\ 105,2 \text{ (грн.)}$$

Далі розраховують величину початкових інвестицій, які розробник (замовник) має вкласти для здійснення науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{розр} \cdot ЗВ \quad (5.11)$$

де $k_{розр}$ – коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково-технічної розробки, зазвичай = *1...5.*

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

$$PV = 1 \cdot 39\,322{,}3 = 39\,322{,}3$$

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $\mathrm{E}_{\text{абс}}$ вкладених інвестицій розраховується за формулою (5.12):

$$\mathrm{E}_{\text{абс}} = (\text{ПП} - PV), \qquad (5.12)$$

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

PV – теперішня вартість початкових інвестицій, грн.

$$\mathrm{E}_{\text{абс}} = 232\,806{,}3 - 39\,322{,}3 = 178\,782{,}9 \text{ грн.}$$

Оскільки $E_{\text{абс}} > 0$, то це свідчить про потенційну доцільність у впровадженні цієї науково-технічної розробки.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $\mathrm{E}_{\text{в}}$ за формулою (5.13):

$$\mathrm{E}_{\text{в}} = \sqrt[\text{Тж}]{1 + \frac{\mathrm{E}_{\text{абс}}}{\mathrm{PV}}} - 1 \qquad (5.13)$$

де $\mathrm{E}_{\text{абс}}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$\mathrm{T}_{\text{ж}}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{178\,782,9}{39\,322,3}} - 1 = 0,77 \text{ або } 77\ \%$$

Далі розраховуємо період окупності інвестицій, які можуть бути вкладені розробником (замовником) у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} \qquad (5.14)$$

$$T_{ок} = \frac{1}{0,77} = 1,3 \text{ року}$$

Якщо $T_{ок}$ < 3-х років, то це свідчить про економічну ефективність впровадження науково-технічної розробки її розробником (замовником).

5.4 Висновки

1. Оцінювання комерційного потенціалу розробки показало, що розробка має високий рівень комерційного потенціалу, оскільки середньоарифметична сума балів СБ, розрахована на основі висновків експертів дорівнює 41.6.

2. Розраховано витрати на виконання науково-дослідної та конструкторсько-технологічної роботи. Розраховано збільшення чистого прибутку протягом трьох років. Вартість чистого прибутку дорівнюватиме 218 105,2 (грн.)

3. Обрахунок терміну окупності розробки показав, що фінансування даної наукової розробки буде доцільним. Ток < 3-х років, а це свідчить про економічну ефективність впровадження науково-технічної розробки її розробником (замовником).

ВИСНОВКИ

У під час виконання магістерської кваліфікаційної роботи було розроблено методи та програмні засоби багатокритеріального оцінювання робіт з комп'ютерної графіки. Для розробки було використано мову програмування з TypeSctipt платформою, для виконання якої фреймворк Node.js, а фреймворком для створення графічної і складової і реалізації її логіки було обрано Angular 8.

Було проаналізовано стан даної проблеми на сьогоднішній день. Здійснено детальний аналіз предметної області та розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом.

Були вирішенні такі задачі:

- проведено аналіз типів веб-додатків та методів їх створення;
- розроблено інтерфейс програмного продукту;
- розроблено модель автоматизованої системи оцінювання з використанням різнотипної системи оцінки;
- розроблено багатокритеріальний метод формування загальної оцінки роботи з урахуванням експертних оцінок кількох членів журі за системою кількісних і якісних критеріїв оцінювання за 9-ти бальною шкалою Сааті;
- створено взаємодію між веб-додатком та серверною частиною;
- розроблено інформаційне наповнення системи;
- проведено тестування програмного продукту;
- розроблено програмний продукт, призначений для вирішення проблеми.

Також було розроблено структуру програмного інтерфейсу, удосконалено макети дизайну та графічну складову веб-додатку, обрано необхідні бібліотеки та фреймворки для реалізації даного завдання.

Перед написанням програмного продукту було покращено схеми користування веб-додатком, обрано кольорові гамми та стилістику інтерфейсу.

Результати роботи доповідалися на конференціях: XLIX науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії Вінницького національного технічного університету (2020), Міжнародній науково-практичній конференції молодих вчених та студентів «Молодь у світі сучасних технологій» (МССТ-2020), Міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ. Пам’яті Олексія Петровича Стахова» (2021).

Було проведено основні етапи тестування клієнтської частини веб-системи. Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конкурсы для веб-дизайнеров и студий. URL: https://web-valley.ru/articles/konkursy-dlya-dizajnerov (дата звернення: 09.09.2020)

2. Конкурс сайтів «WEB AWARDS UA». URL: https://awards.it-rating.in.ua/ua/ (дата звернення: 05.05.2020)

3. Конкурс сайтів «Webby Awards». URL: https://www.webbyawards.com/ (дата звернення: 05.05.2020)

4. Favourite Website Awards — FWA. URL: https://thefwa.com/awards/page/1/ (дата звернення: 05.05.2020)

5. European Design Awards. URL: https://europeandesign.org/ed-awards/winners/?award-year=2021/ (дата звернення: 05.05.2020)

6. В.В. Войтко, О.А. Боднар, Ю.С. Рекута Розробка веб-системи для розміщення і оцінювання робіт оцінювання робіт міжнародного конкурсу з комп'ютерної графіки / Матеріали XLIX науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2020). [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9621/8009

7. В.В. Войтко, О.А. Боднар, Ю.С. Рекута. Розробка веб-системи для оцінювання конкурсних графічних робіт / Матеріали міжнародної науково-практичної конференції молодих вчених та студентів «Молодь у світі сучасних технологій». – Херсон, 2020. [Електронний ресурс] – Режим доступу до ресурсу: http://kntu.net.ua/ukr/content/view/full/58984

8. В. В. Войтко, С.В. Бевз, С.М. Бурбело, Ю.С. Рекута. Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки / Електронні інформаційні ресурси: створення, використання, доступ. Пам'яті Олексія Петровича Стахова. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021. С. 38-40

9. Что такое SMTP-сервер. URL: https://www.unisender.com/ru/support/about/glossary/chto-takoe-smtp/ (дата звернення: 05.05.2020)

10. Переваги та недоліки веб-додатків. URL: https://stud.com.ua/97611/informatika/perevagi_nedoliki_dodatkiv (дата звернення: 11.05.2020)

11. Основні етапи веб-розробки. Способи створення сайтів. URL: https://web-systems.solutions/blog/veb-rozrobka-etapy-i-standarty/ (дата звернення: 11.05.2020)

12. Модели жизненного цикла, принципы и методологии разработки программного обеспечения (ПО). URL: https://evergreens.com.ua/ru/articles/software-development-metodologies (дата звернення: 11.05.2020)

13. Введение в Angular. URL: https://webformyself.com/vvedenie-v-angular-chto-eto-za-frejmvork-i-zachem-ego-ispolzovat/ (дата звернення: 11.05.2020)

14. J. Houser. Learn With: Angular 8, Bootstrap, and NodeJS: Enterprise Application Development with Angular 8 and NodeJS. DotComIt, LLC, 2019. C. 265

15. WebReference. URL: https://webref.ru/layout/bootstrap (дата звернення: 11.09.2020)

16. Visual Studio Code. URL: https://bizzapps.ru/p/visual-studio-code/ (дата звернення: 11.09.2020)

17. Критерии выбора и алгоритм принятия эффективного решения. URL: http://www.elitarium.ru/prinyatie-resheniya-alternativa-risk-ocenka-kriterij-variant-vybor-plan-veroyatnost-razrabotka-ispolnenie/ (дата звернення: 21.09.2020)

18. Способы сравнения многокритериальных альтернатив. URL: https://www.arhivinfo.ru/2-102092.htmlhttps://www.arhivinfo.ru/2-102092.html (дата звернення: 13.10.2020)

19. Інтерфейси ПК. Основне призначення та характеристики. URL: http://infuha.ru/news/a-206.html (дата звернення: 13.10.2020)

20. Плоский дизайн сайта. URL: https://tilda.education/articles-flat-design (дата звернення: 13.10.2020)

21. What is TypeScript? URL: https://www.typescriptlang.org/ (дата звернення: 22.10.2020)

22. The Onion Architecture : part 1. URL: https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/ (дата звернення: 23.10.2020)

23. Я. Файн. Angular и TypeScript, 2018. С. 464

24. Сервисы и dependency injection. URL: https://metanit.com/web/angular2/4.1.php (дата звернення: 27.10.2020)

25. Э. Дастин, Дж. Рэшка, Дж. Пол.Дастин Э. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. – Москва: Лори, 2003. С. 567

26. В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт – Вінниця : ВНТУ, 2021. С. 42

ДОДАТКИ

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доц. В. В. Войтко

"____"_____2021 р.

Виконала:

_____студентка гр. 1ПІ-20м Ю. С. Рекута

"____"_____2021 р.

Вінниця – 2021 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки».

Галузь застосування – проведення конкурсів та олімпіад онлайн.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 277 від « 24 » вересня 2021 р.

3. Мета та призначення розробки.

Метою роботи розширення можливостей оцінювання графічних робіт з використанням кількісних і якісних критеріїв оцінювання, що дозволить більш глибоко оцінити характеристики роботи.

Призначення роботи – надання користувачам можливості розміщення та перегляду конкурсних робіт та новин, та реєстрації під різними видами акаунтів.

4. Вихідні дані для проведення НДР

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

1. Э. Дастин, Дж. Рэшка, Дж. Пол.Дастин Э. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. Москва: Лори, 2003. 567 с

2. Криспин Л., Дж. Грегори. Гибкое тестирование. ИД "Вильямс", 2010. 464 с.

3. Я. Файн. Angular и TypeScript, 2018. 464с

4. Alex Banks, Eve Porcello. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 с

5. Технічні вимоги

- Середовище виконання – Visual Studio Code

- Мова програмування – TypeScript, HTML, CSS.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до магістерської кваліфікаційної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії і етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз, вибір та актуальність розробки	15.09.21 – 30.10.21	Вик.
2	Аналіз вибраних аналогів	01.10.21 – 12.10.21	Вик.
3	Аналіз інформаційного забезпечення	12.10.21 – 17.10.21	Вик.
4	Розробка методу та моделей автоматизаціїроботи системи	17.10.21 – 28.10.21	Вик.
5	Розробка макетів графічного інтерфейсу	28.10.21 – 04.11.21	Вик.
6	Програмна реалізація модулів додатку	04.11.21 – 15.11.21	Вик.
7	Тестування роботи додатку	15.11.21 – 25.11.21	Вик.
8	Економічна частина	25.11.21 – 30.11.21	Вик.

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б.

Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 20м

Науковий керівник: к.т.н. доц. Войтко В.В.

Unicheck	
Оригінальність	**95,8 %**
Схожість	4,2 %

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.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу та програмних засобів багатокритеріального оцінювання робіт з комп'ютерної графіки».

Автор _____ Рекута Юлія Сергіївна

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)

ДОДАТОК В

Лістинг програми

appmodule.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { RouterModule } from '@angular/router';

import { AppComponent } from './app.component';
import { NavMenuComponent } from './nav-menu/nav-menu.component';
import { HomeComponent } from './home/home.component';
import { CounterComponent } from './counter/counter.component';
import { FetchDataComponent } from './fetch-data/fetch-data.component';
import { ApiAuthorizationModule } from 'src/api-authorization/api-authorization.module';
import { AuthorizeGuard } from 'src/api-authorization/authorize.guard';
import { AuthorizeInterceptor } from 'src/api-authorization/authorize.interceptor';
import { HeaderComponent } from './header/header.component';
import { MainComponent } from './main/main.component';
import { FooterComponent } from './footer/footer.component';

@NgModule({
  declarations: [
    AppComponent,
    NavMenuComponent,
    HomeComponent,
```

```
    CounterComponent,
    FetchDataComponent,
    HeaderComponent,
    MainComponent,
    FooterComponent
  ],
  imports: [
    BrowserModule.withServerTransition({ appId: 'ng-cli-universal' }),
    HttpClientModule,
    FormsModule,
    ApiAuthorizationModule,
    RouterModule.forRoot([
      { path: '', component: HomeComponent, pathMatch: 'full' },
      { path: 'counter', component: CounterComponent },
      { path: 'fetch-data', component: FetchDataComponent, canActivate: [AuthorizeGuard] },
    ])
  ],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: AuthorizeInterceptor, multi: true }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

api-authorization.constants.ts

```
export const ApplicationName = 'Majestics';
export const ReturnUrlType = 'returnUrl';
export const QueryParameterNames = {
```

```
  ReturnUrl: ReturnUrlType,
  Message: 'message'
};

export const LogoutActions = {
  LogoutCallback: 'logout-callback',
  Logout: 'logout',
  LoggedOut: 'logged-out'
};

export const LoginActions = {
  Login: 'login',
  LoginCallback: 'login-callback',
  LoginFailed: 'login-failed',
  Profile: 'profile',
  Register: 'register'
};

let applicationPaths: ApplicationPathsType = {
  DefaultLoginRedirectPath: '/',
  ApiAuthorizationClientConfigurationUrl:
`/_configuration/${ApplicationName}`,
  Login: `authentication/${LoginActions.Login}`,
  LoginFailed: `authentication/${LoginActions.LoginFailed}`,
  LoginCallback: `authentication/${LoginActions.LoginCallback}`,
  Register: `authentication/${LoginActions.Register}`,
  Profile: `authentication/${LoginActions.Profile}`,
  LogOut: `authentication/${LogoutActions.Logout}`,
  LoggedOut: `authentication/${LogoutActions.LoggedOut}`,
  LogOutCallback: `authentication/${LogoutActions.LogoutCallback}`,
  LoginPathComponents: [],
```

```
  LoginFailedPathComponents: [],
  LoginCallbackPathComponents: [],
  RegisterPathComponents: [],
  ProfilePathComponents: [],
  LogOutPathComponents: [],
  LoggedOutPathComponents: [],
  LogOutCallbackPathComponents: [],
  IdentityRegisterPath: '/Identity/Account/Register',
  IdentityManagePath: '/Identity/Account/Manage'
};

applicationPaths = {
  ...applicationPaths,
  LoginPathComponents: applicationPaths.Login.split('/'),
  LoginFailedPathComponents: applicationPaths.LoginFailed.split('/'),
  RegisterPathComponents: applicationPaths.Register.split('/'),
  ProfilePathComponents: applicationPaths.Profile.split('/'),
  LogOutPathComponents: applicationPaths.LogOut.split('/'),
  LoggedOutPathComponents: applicationPaths.LoggedOut.split('/'),
  LogOutCallbackPathComponents: applicationPaths.LogOutCallback.split('/')
};

interface ApplicationPathsType {
  readonly DefaultLoginRedirectPath: string;
  readonly ApiAuthorizationClientConfigurationUrl: string;
  readonly Login: string;
  readonly LoginFailed: string;
  readonly LoginCallback: string;
  readonly Register: string;
  readonly Profile: string;
  readonly LogOut: string;
```

```
  readonly LoggedOut: string;
  readonly LogOutCallback: string;
  readonly LoginPathComponents: string [];
  readonly LoginFailedPathComponents: string [];
  readonly LoginCallbackPathComponents: string [];
  readonly RegisterPathComponents: string [];
  readonly ProfilePathComponents: string [];
  readonly LogOutPathComponents: string [];
  readonly LoggedOutPathComponents: string [];
  readonly LogOutCallbackPathComponents: string [];
  readonly IdentityRegisterPath: string;
  readonly IdentityManagePath: string;
}

export const ApplicationPaths: ApplicationPathsType = applicationPaths;
```

login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthorizeService, AuthenticationResultStatus } from '../authorize.service';
import { ActivatedRoute, Router } from '@angular/router';
import { BehaviorSubject } from 'rxjs';
import { LoginActions, QueryParameterNames, ApplicationPaths, ReturnUrlType } from '../api-authorization.constants';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
```

```
    public message = new BehaviorSubject<string>(null);

    constructor(
        private authorizeService: AuthorizeService,
        private activatedRoute: ActivatedRoute,
        private router: Router) { }

    async ngOnInit() {
        const action = this.activatedRoute.snapshot.url[1];
        switch (action.path) {
            case LoginActions.Login:
                await this.login(this.getReturnUrl());
                break;
            case LoginActions.LoginCallback:
                await this.processLoginCallback();
                break;
            case LoginActions.LoginFailed:
                const message =
this.activatedRoute.snapshot.queryParamMap.get(QueryParameterNames.Message);
                this.message.next(message);
                break;
            case LoginActions.Profile:
                this.redirectToProfile();
                break;
            case LoginActions.Register:
                this.redirectToRegister();
                break;
            default:
                throw new Error(`Invalid action '${action}'`);
        }
    }
```

```
    private async login(returnUrl: string): Promise<void> {
        const state: INavigationState = { returnUrl };
        const result = await this.authorizeService.signIn(state);
        this.message.next(undefined);
        switch (result.status) {
            case AuthenticationResultStatus.Redirect:
                break;
            case AuthenticationResultStatus.Success:
                await this.navigateToReturnUrl(returnUrl);
                break;
            case AuthenticationResultStatus.Fail:
                await
this.router.navigate(ApplicationPaths.LoginFailedPathComponents, {
                    queryParams:        {        [QueryParameterNames.Message]:
result.message }
                });
                break;
            default:
                throw new Error(`Invalid status result ${(result as any).status}.`);
        }
    }

    private async processLoginCallback(): Promise<void> {
        const url = window.location.href;
        const result = await this.authorizeService.completeSignIn(url);
        switch (result.status) {
            case AuthenticationResultStatus.Redirect:
                throw new Error('Should not redirect.');
            case AuthenticationResultStatus.Success:
```

```
                await this.navigateToReturnUrl(this.getReturnUrl(result.state));
                break;
            case AuthenticationResultStatus.Fail:
                this.message.next(result.message);
                break;
        }
    }

    private redirectToRegister(): any {
        this.redirectToApiAuthorizationPath(
            `${ApplicationPaths.IdentityRegisterPath}?returnUrl=${encodeURI('/'
+ ApplicationPaths.Login)}`);
    }

    private redirectToProfile(): void {

this.redirectToApiAuthorizationPath(ApplicationPaths.IdentityManagePath);
    }

    private async navigateToReturnUrl(returnUrl: string) {
        await this.router.navigateByUrl(returnUrl, {
            replaceUrl: true
        });
    }

    private getReturnUrl(state?: INavigationState): string {
        if (fromQuery &&
            !(fromQuery.startsWith(`${window.location.origin}/`) ||
                /\/[^\/].*/.test(fromQuery))) {
        }
        return (state && state.returnUrl) ||
```

```
            fromQuery ||
            ApplicationPaths.DefaultLoginRedirectPath;
        }

        private redirectToApiAuthorizationPath(apiAuthorizationPath: string) {
          const redirectUrl = `${window.location.origin}${apiAuthorizationPath}`;
          window.location.replace(redirectUrl);
        }
      }

      interface INavigationState {
        [ReturnUrlType]: string;
      }

      header.component.html
      <div class="wrapper">
        <div class="header_info">
          <h1 class="title">Majestics.</h1>
          <p class="header_text">Міжнародні відкриті конкурси з комп'ютерної<br />графіки та веб-дизайну серед студентів та учнів</p>
          <div class="buttons">
            <div class="button1">
              <a href="./Identity/Account/Register?returnUrl=/authentication/login"><button type="button" class="btn btn-success">Реєстрація</button></a>
            </div>
            <div class="button2">
              <button type="button" class="btn btn-outline-secondary">Про конкурс</button>
            </div>
```

```
      </div>
    </div>
    <div class="header_img"><img src="../../assets/images/header/head.jpg"></div>
  </div>
```

main.component.html

```
  <div class="main">
    <h3 class="title">XVII Міжнародний конкурс з веб-дизайну та комп'ютерної графіки</h3>
    <!--<div class="f_select">
      <div class="btn-group">
        <button type="button" class="btn btn-primary">Школа</button>
        <button type="button" class="btn btn-primary">Коледж</button>
        <button type="button" class="btn btn-primary">ВНЗ</button>
      </div>
    </div>
    <div class="s_select">
      <div class="btn-group">
        <button type="button" class="btn btn-primary">2D Раст</button>
        <button type="button" class="btn btn-primary">2D Вектор</button>
        <button type="button" class="btn btn-primary">2D Фотоколаж</button>
        <button type="button" class="btn btn-primary">3D</button>
      </div>
    </div>-->
    <div class="buttons">
      <div class="f_button">
        <button type="button" class="btn btn-dark">Школа</button>
        <button type="button" class="btn btn-dark">Коледж</button>
        <button type="button" class="btn btn-dark">ВНЗ</button>
      </div>
```

```
            <div class="s_button">
              <button type="button" class="btn btn-light">2D Раст</button>
              <button type="button" class="btn btn-light">2D Вектор</button>
              <button type="button" class="btn btn-light">2D Фотоколаж</button>
              <button type="button" class="btn btn-light">3D</button>
            </div>
          </div>
          <div class="grid">
            <h4 class="total">Всього:</h4>
            <div class="column">
              <img src="../../assets/images/main/img1.png" style="width:100%">
              <img src="../../assets/images/main/img2.png" style="width:100%">
              <img src="../../assets/images/main/img1.png" style="width:100%">
              <img src="../../assets/images/main/img2.png" style="width:100%">
              <img src="../../assets/images/main/img1.png" style="width:100%">
              <img src="../../assets/images/main/img2.png" style="width:100%">
              <img src="../../assets/images/main/img1.png" style="width:100%">
              <img src="../../assets/images/main/img2.png" style="width:100%">
              <img src="../../assets/images/main/img1.png" style="width:100%">
            </div>
          </div>
          <button type="button" class="btn btn-info">Переглянути Усі</button>
        </div>
        </div>
```

main.component.css

```
.main{
  margin: 30px;
  width: 90%;
}
```

```
.title{
  color: black;
  text-align: center;
  text-transform: uppercase;
  margin-top: 40px;
}

/*.btn-hig {
  background-color: white;
  color: black;
  margin-top: 40px;
  border: none;
}
btn-hig:active {
  border-bottom: 2px;
  border-color: blue;
  border-top: none;
  border-left: none;
  border-right: none;
}
.btn-low {
  background-color: white;
  color: black;
  margin-top: 40px;
  border: none;
}
.btn-low:active {
  border-bottom: 2px;
  border-color: black;
  border-top: none;
```

```
  border-left: none;
  border-right: none;
}*/

/*.btn-group {
  margin-top: 20px;
}
.btn-primary {
  background-color: white;
  color: black;
  border: none;
}*/

.buttons{
  width: 100%;
}

.f_button{
  margin-left: 40%;
}

.s_button {
  margin-left: 34%;
}

.btn-dark {
  margin-top: 20px;
  background-color: white;
  color: black;
  border: none;
```

```
}

.btn-light {
  margin-top: 20px;
  background-color: white;
  color: black;
  border: none;
}

.btn-info {
  border-radius: 70px;
  margin-top: 15px;
  width: 170px;
  height: 40px;
  background-color: #2f809a;
  border-color: #2f809a;
  color: white;
  text-transform: uppercase;
  position: relative;
  margin-left: 43%;
}

.btn-info:hover{
  border-radius: 70px;
  margin-top: 15px;
  width: 170px;
  height: 40px;
  background-color: #adbce6;
  color: #18404d;
  border-color: #adbce6;
```

```
  text-transform: uppercase;
  position: relative;
  margin-left: 43%;
}
```

nav-menu.component.html

```
<header>
  <nav
    class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3"
  >
    <div class="container">
      <a class="navbar-brand" [routerLink]="['/']">Majestics</a>
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target=".navbar-collapse"
        aria-label="Toggle navigation"
        [attr.aria-expanded]="isExpanded"
        (click)="toggle()"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div
        class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse"
        [ngClass]="{ show: isExpanded }"
      >
        <app-login-menu></app-login-menu>
        <ul class="navbar-nav flex-grow">
```

```
        <li
          class="nav-item"
          [routerLinkActive]="['link-active']"
          [routerLinkActiveOptions]="{ exact: true }"
        >
          <a class="nav-link text-dark" [routerLink]="['/']">Home</a>
        </li>
        <li class="nav-item" [routerLinkActive]="['link-active']">
          <a class="nav-link text-dark" [routerLink]="['/counter']"
            >Counter</a
          >
        </li>
        <li class="nav-item" [routerLinkActive]="['link-active']">
          <a class="nav-link text-dark" [routerLink]="['/fetch-data']"
            >Fetch data</a
          >
        </li>
      </ul>
    </div>
  </div>
 </nav>
</header>
<!--<div class="navbar">
 <div class="sidenav">
 </div>
 <div class="title"><p class="tile">Majestics</p></div>
 <div class="sign_in">
  <button type="button" class="btn btn-outline-dark">Реєстрація</button>
 </div>
</div>-->
```

nav-menu.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-nav-menu',
  templateUrl: './nav-menu.component.html',
  styleUrls: ['./nav-menu.component.css']
})
export class NavMenuComponent {
  isExpanded = false;

  collapse() {
    this.isExpanded = false;
  }

  toggle() {
    this.isExpanded = !this.isExpanded;
  }
}
```

ДОДАТОК Г

Рекомендовані критерії оцінювання комерційного потенціалу розробки

Таблиця Г.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка.

	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці Г.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка.

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реаліза-ції ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти ро-ків	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвес-тицій більший 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти ро-ків	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію	Необхідно отримання ве-ликої кількості дозвільних документів на виробництво та реалізацію продукту, що потребує значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту потребує незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

	продукту				

ДОДАТОК Д

Ілюстративна частина

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНИХ ЗАСОБІВ БАГАТОКРИТЕРІАЛЬНОГО ОЦІНЮВАННЯ РОБІТ З КОМП’ЮТЕРНОЇ ГРАФІКИ

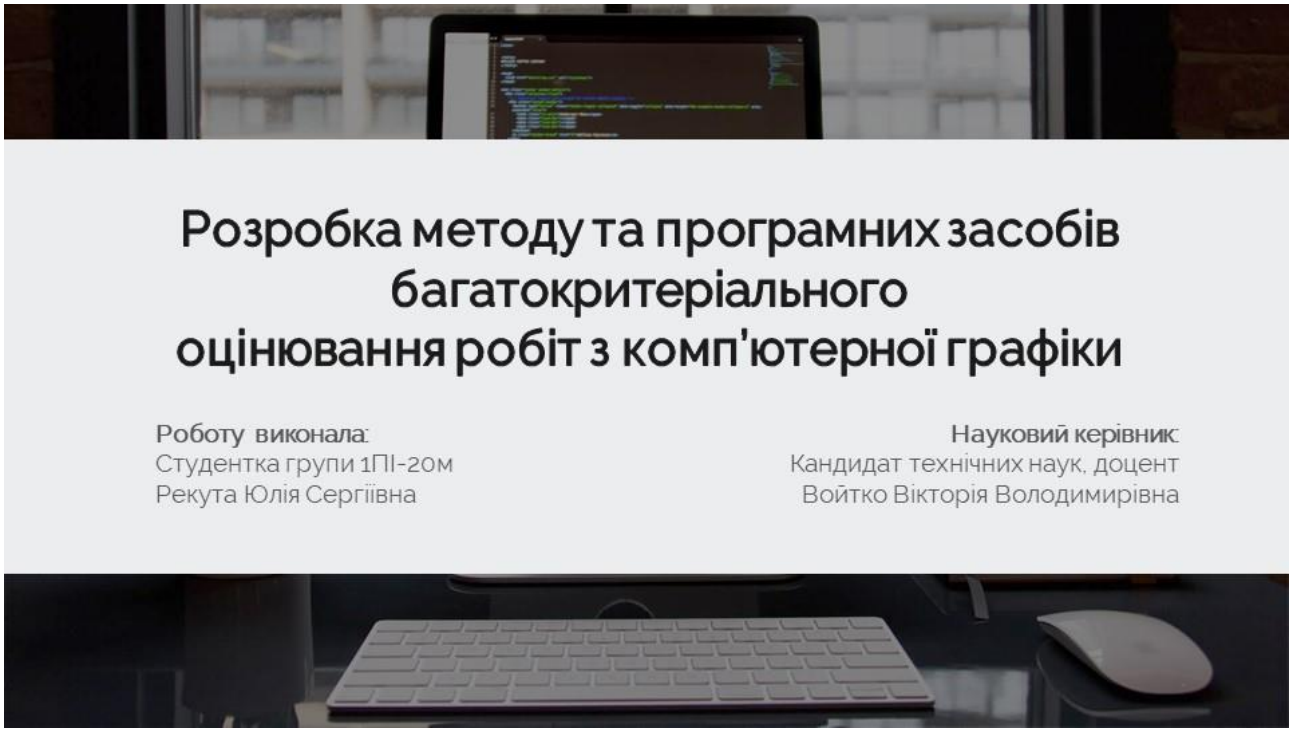


Рисунок Д.1 – Назва роботи

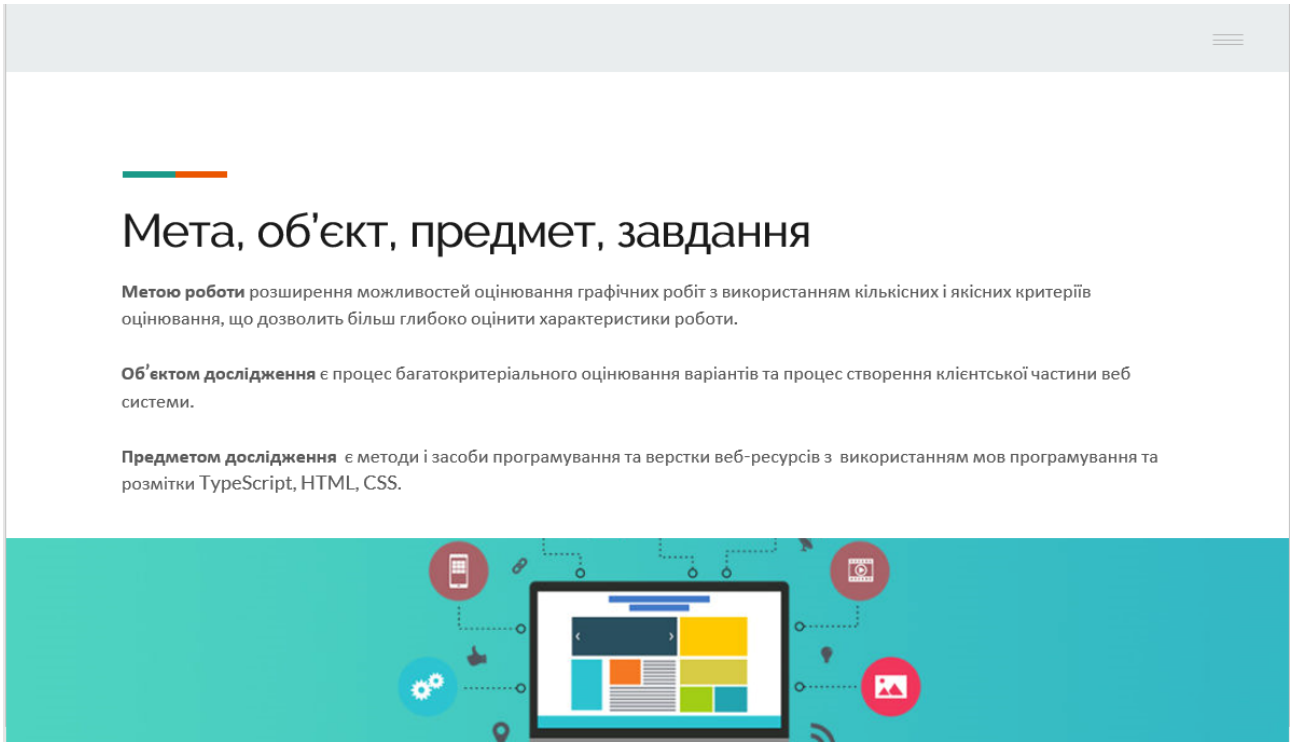


Рисунок Д.2 – Мета, об’єкт і предмет дослідження

Рисунок Д.3 – Завдання

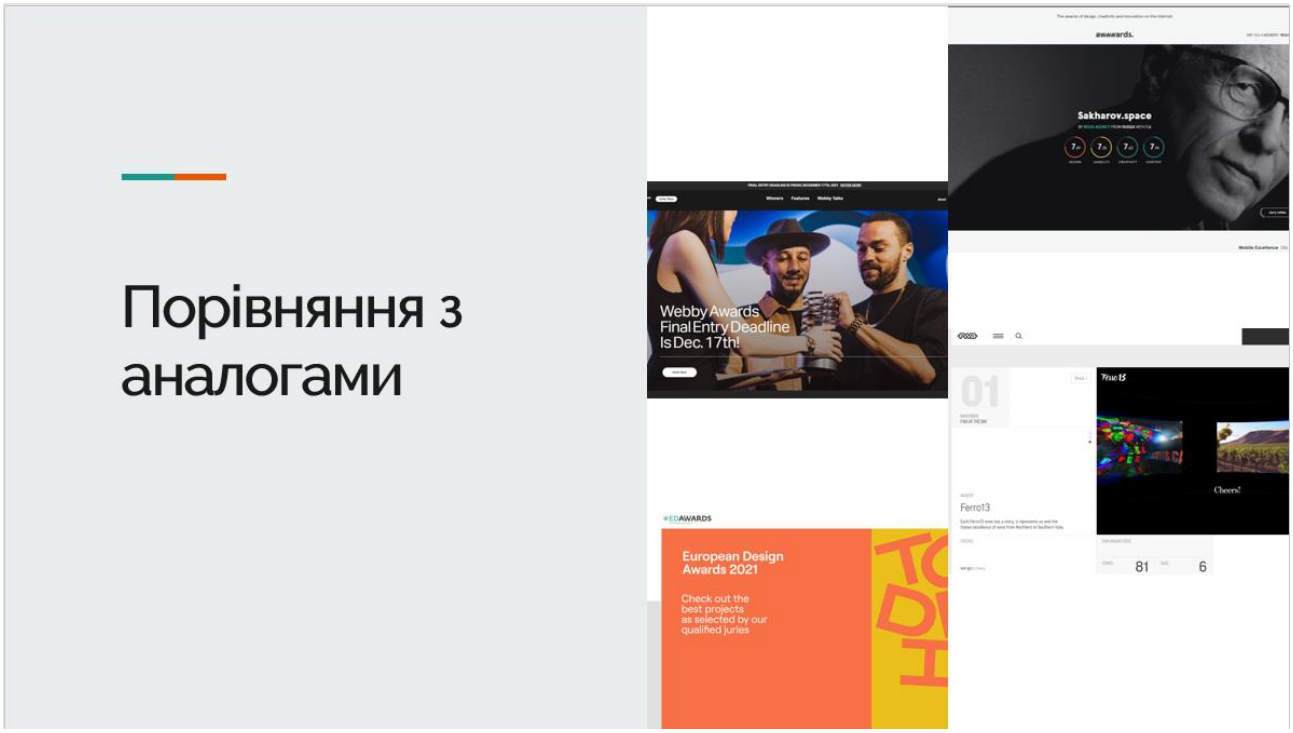


Рисунок Д.4 – Порівняння з аналогами

Система «Awwwards»

«Awwwards» – конкурс сайтів та мобільних додатків, який проводиться щорічно, починаючи з 2017-го року. Спочатку триває народне голосування, в якому може взяти участь кожен, після чого експертна комісія та члени журі оцінюють роботи. Призові місця присуджуються щодня як веб-дизайнерам за зручність використання сайту, креативний підхід, так і розробникам за коди.

За умовами конкурсу, на розгляд приймаються лише реалізовані проекти. Оцінка враховує 4 напрямки: дизайн - 40%; креативність - 30%; юзабіліті - 20%; контент – 10%.

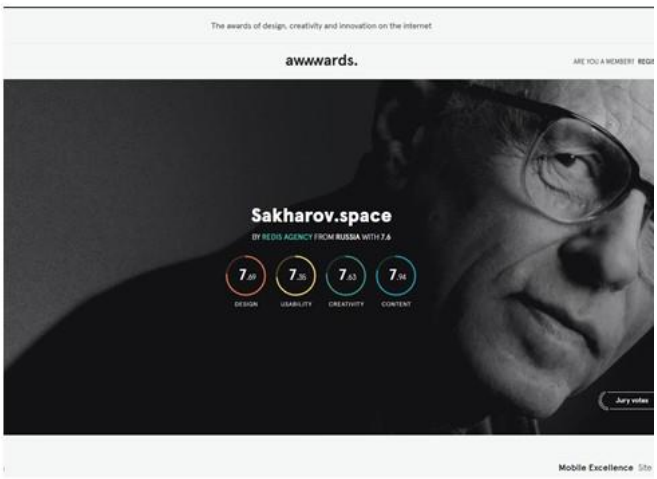


Рисунок Д.5 – Система «Awwwards»

«Webby Awards»

це міжнародна інтернет-премія для дизайнерів та розробників сайтів, мобільних програм, а також фахівців з реклами.

Критерії оцінювання: опрацьована структура та навігація сайту; сучасний дизайн; функціональність сайту; успішність проекту.

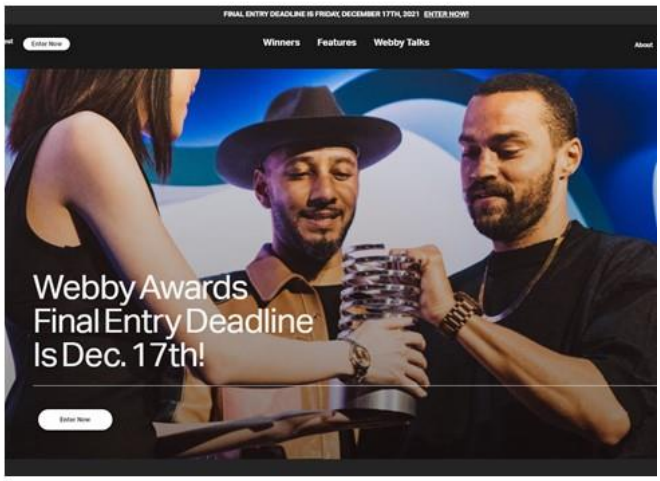


Рисунок Д.6 – Система «Webby Awards»

«FWA»

«Favourite Website Awards — FWA» – це конкурс найкреативніших та передових робіт з усього світу.

Сайт оцінюється не за особистими відчуттями суддів, а за чек-листом:

- 40 балів - непрацюючий сайт, бите посилання;
- 60 балів – звичайний середньостатистичний сайт;
- До 80 балів – круто, але до FWA не дотягує;
- Від 85 балів - топовий сайт;
- Від 95 - FAW.

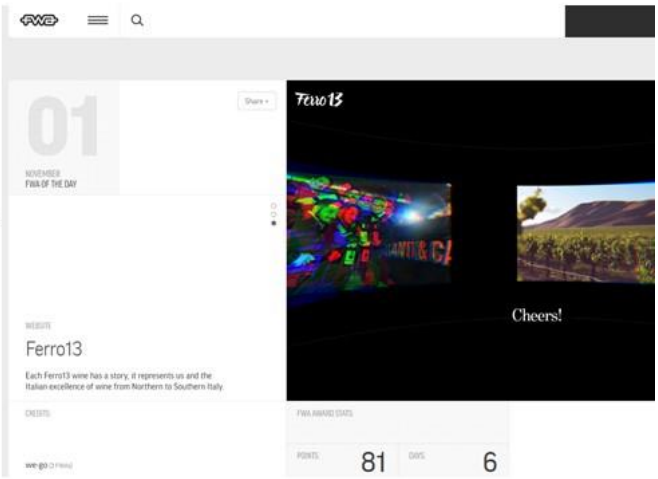


Рисунок Д.7 – Система «FWA»

«European Design Awards»

Конкурс проводиться за 8 категоріями: брендинг; упакування; внутрішнє виробництво; публікації; діджитал (тут 2 номінації — найкращий сайт та найкращий мобільний додаток); просування; ілюстрації; різне.

Проект оцінюють за такими критеріями: якість виконання; креативність; юзабіліті; технічне виконання.
Місто проведення заходу щороку змінюється. Переможця запрошують до однієї з європейських країн для вручення призу.

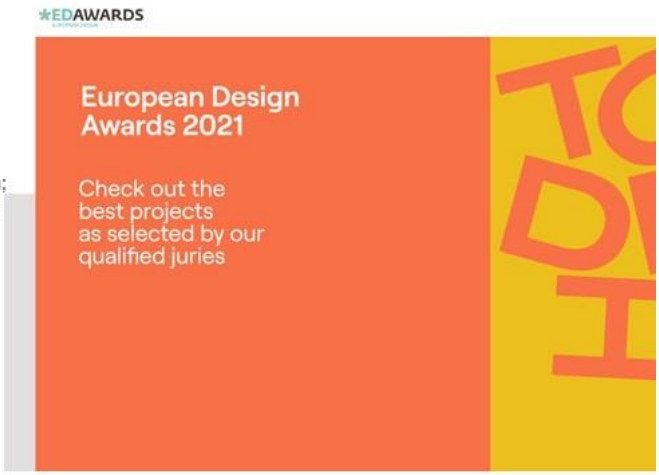


Рисунок Д.8 – Система «European Design Awards»

Порівняльні характеристики програмних продуктів

Критерій	«awwwards»	«Webby Awards»	« FWA»	«ED Awards»	«Majestics»
Умова	реальний робочий проєкт	реальний робочий проєкт	креативний проєкт	реальний робочий проєкт, креативний дизайн	креативний проєкт
Доступність для усіх користувачів	+	-	-	-	+
Надійність	-	+	+	+	+
Народне голосування	-	-	-	-	+
Критерії оцінювання робіт	+	+	-	+	+

Рисунок Д.9 – Порівняння аналогів

Запропонована система та метод включають в себе вирішення певних задач, зокрема:

- перевірку голосів на унікальність;
- коригування відсоткового впливу голосу на загальну оцінку в залежності від типу користувача;
- модифікація критеріїв оцінки та їх вплив на загальну оцінку кожної роботи або групи робіт;
- можливість анонімного оцінювання;
- обмеження можливості оцінювання для різних типів користувачів.

Рисунок Д.10 – Вирішення задач

Модель розробленої системи

Перед тим як розробити прототип, було створено модель роботи системи «Majestics».

Рисунок Д.11 – Модель розробленої системи

Метод багатокритеріального оцінювання

Метод багатокритеріального оцінювання полягає у виконанні такої послідовності дій:

1. Визначення системи критеріїв оцінювання робіт, які можуть бути як кількісними, так і якісними; система є відкритою і може модифікуватися і доповнюватися за потреби.
2. Кожен експерт проводить оцінювання робіт за методом парних порінянь за 9-бальною шкалою Сааті та формує матриці парних порівнянь.
3. За матрицями парних порівнянь визначаються функції належності, що дозволяють отримати цифрові оцінки за кількісними критеріями оцінювання з використанням цифрових еквівалентів 9-бальної шкали Сааті.
4. Визначаються оцінки всіх робіт за кожним критерієм окремо як множина цифрових значень отриманих в результаті визначених функцій належності.
5. Визначення інтегрального критерію визначається шляхом усереднення результату з відкиданням межових (найбільшої і найменшої) оцінок експертів з метою підвищення об'єктивності оцінення шляхом виключення можливостей зумисного збільшення чи зменшення рейтингу конкретної роботи.

Використання методу парних порівнянь за 9-бальною шкалою Сааті дає змогу використовувати систему якісних і кількісних критеріїв оцінення.

Рисунок Д.12 – Метод багатокритеріального оцінювання

Рисунок Д.13 – Схема завантаження роботи

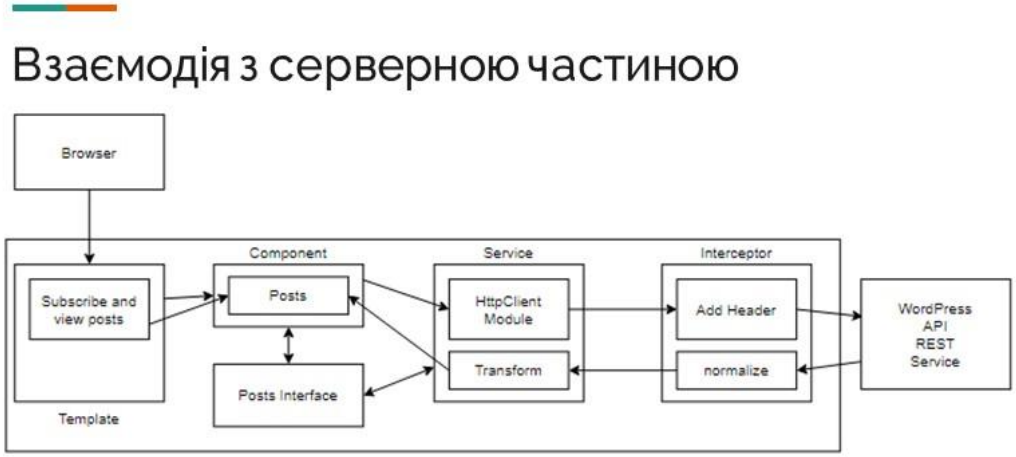


Рисунок Д.14 – Модель взаємодії клієнтської та серверної частини

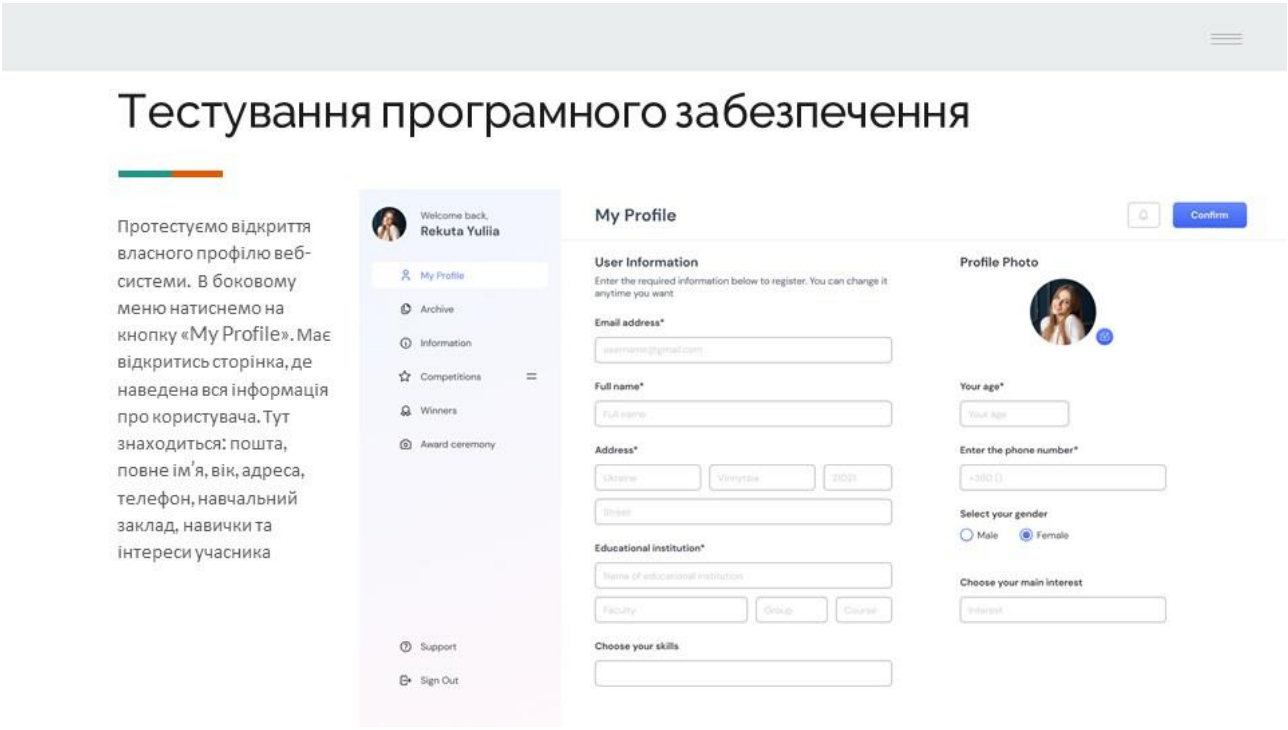


Рисунок Д.15 – Профіль користувача

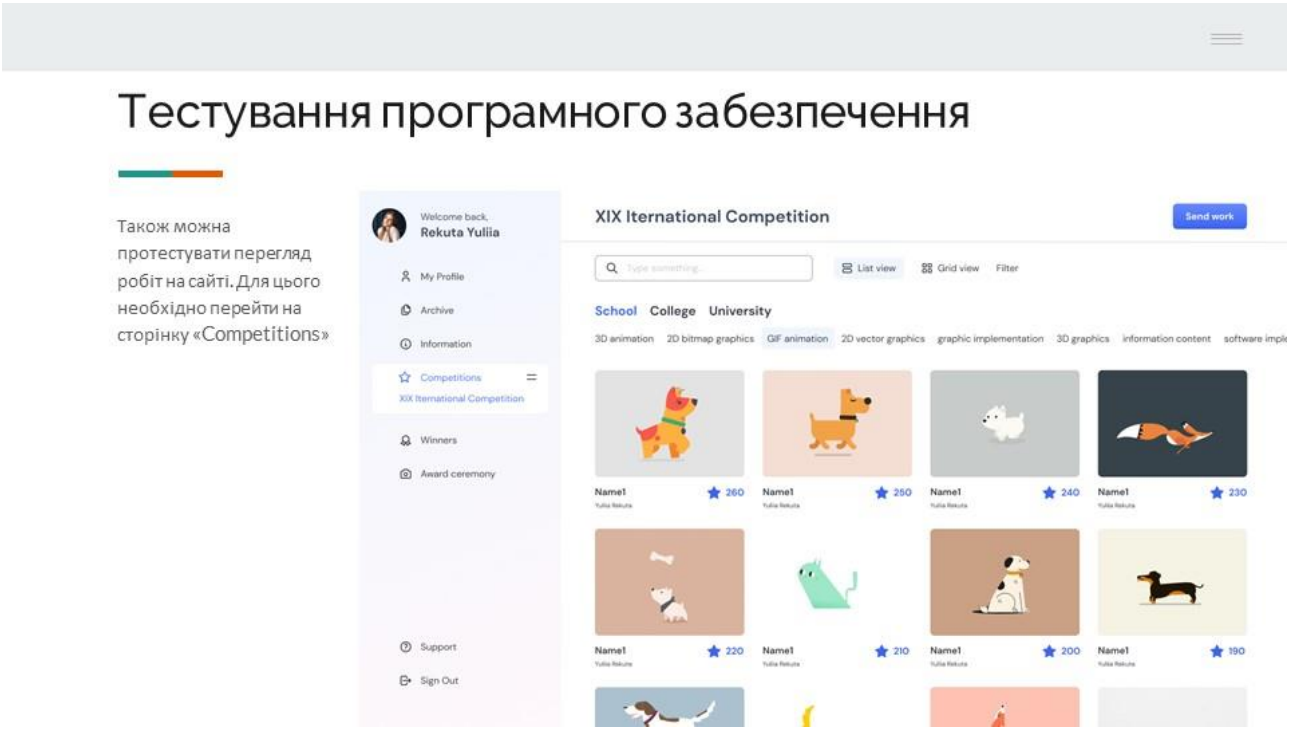


Рисунок Д.16 – Сторінка активного конкурсу

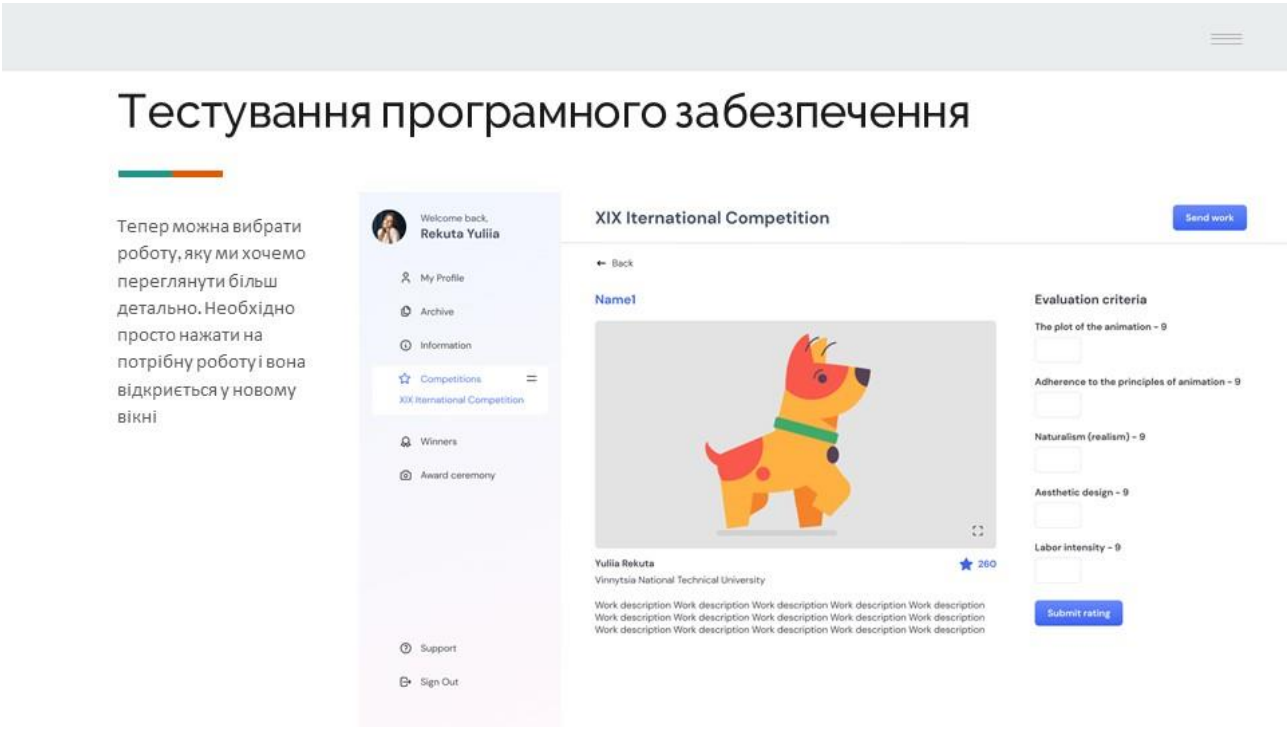


Рисунок Д.17 – Перегляд роботи

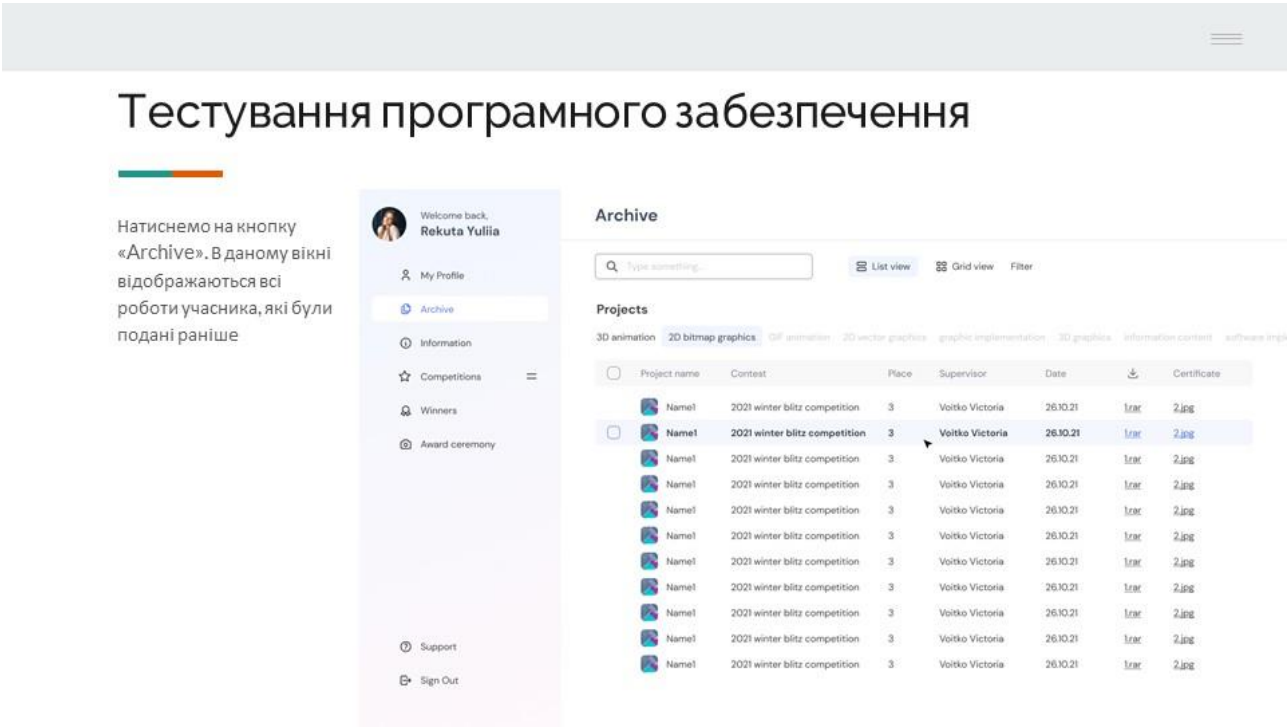


Рисунок Д.18 – Архів

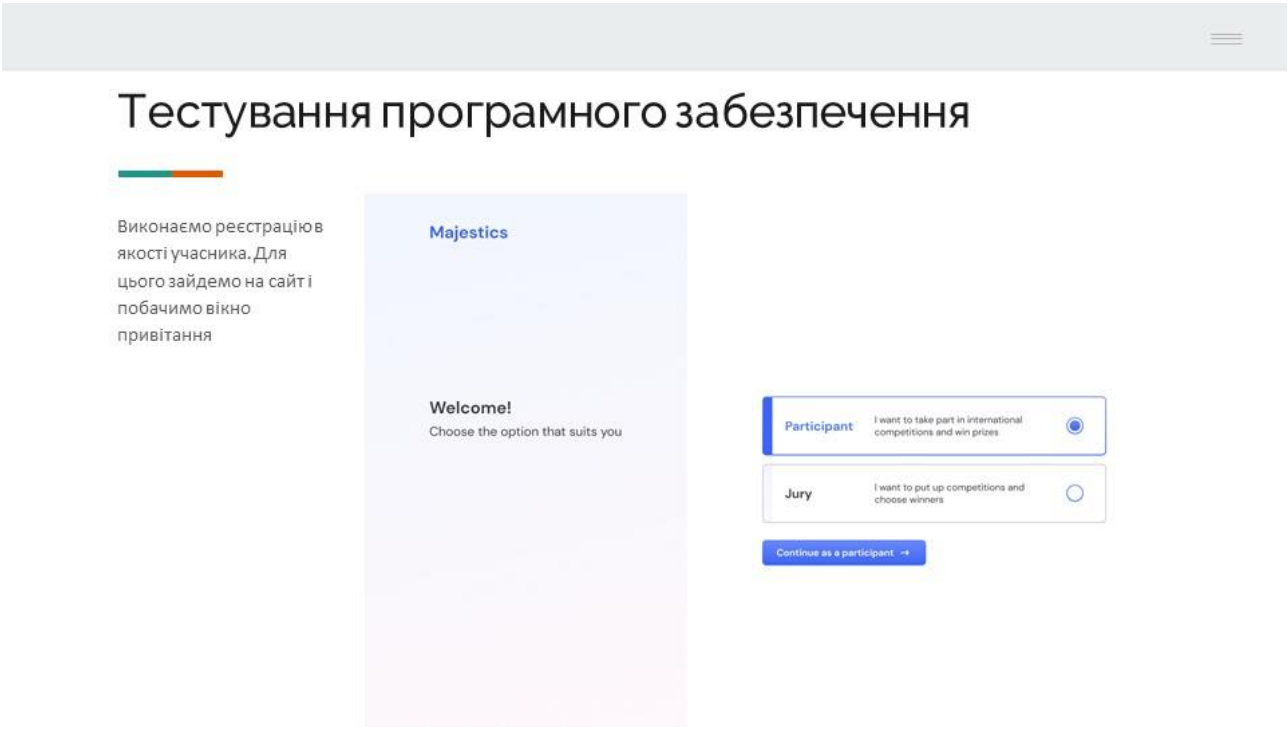


Рисунок Д.19 – Вікно привітання

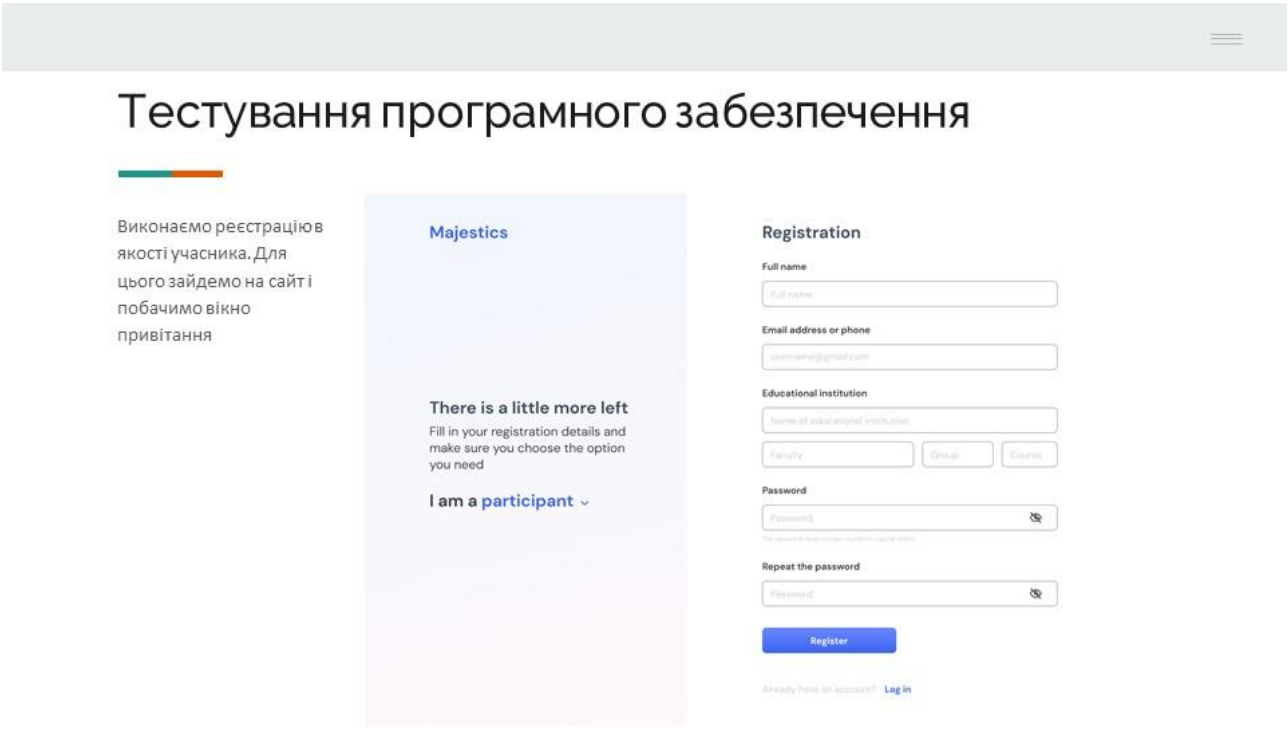


Рисунок Д.20 – Форма реєстрації

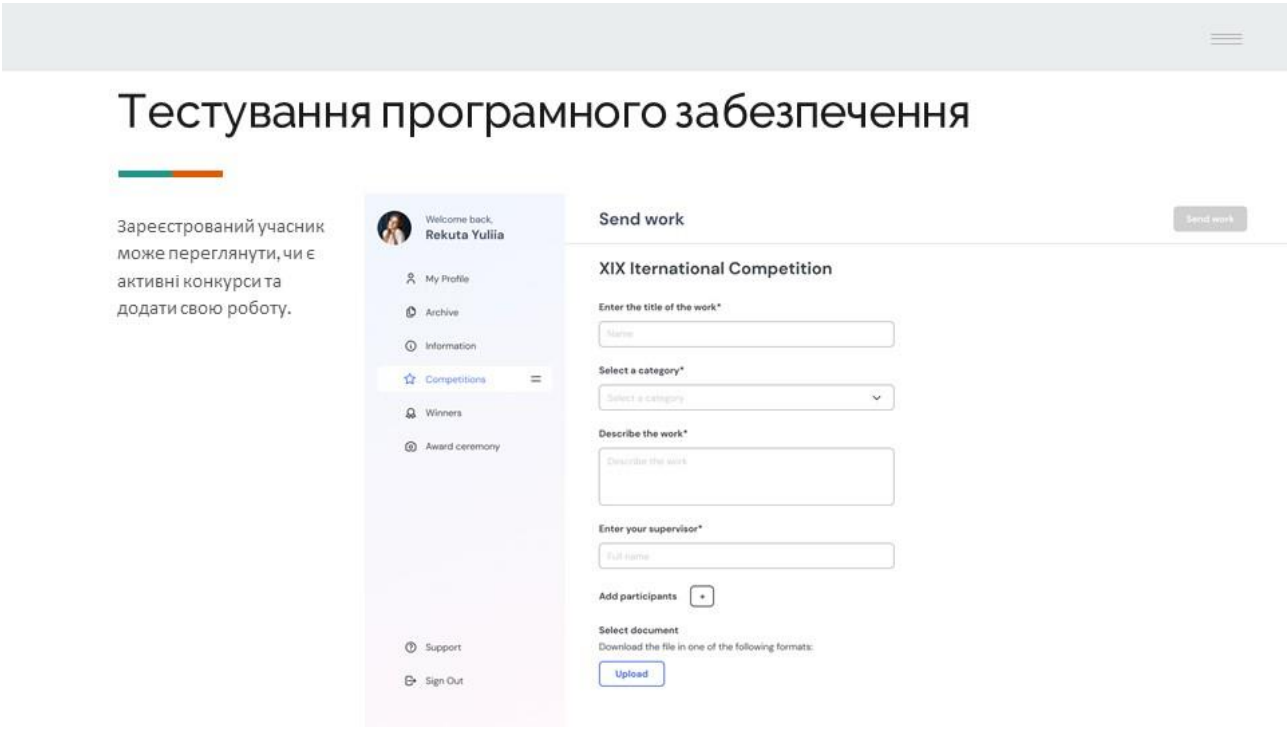


Рисунок Д.21 – Форма заповнення даних про роботу

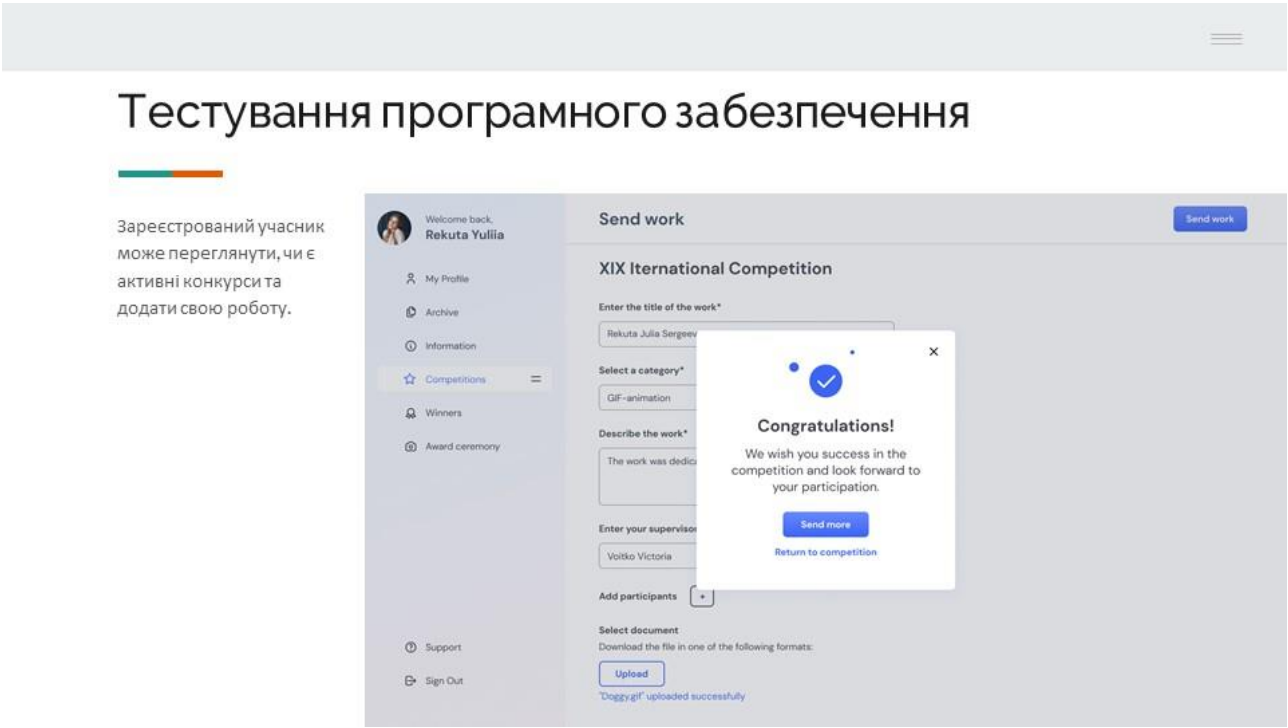


Рисунок Д.22 – Успішна відправка роботи

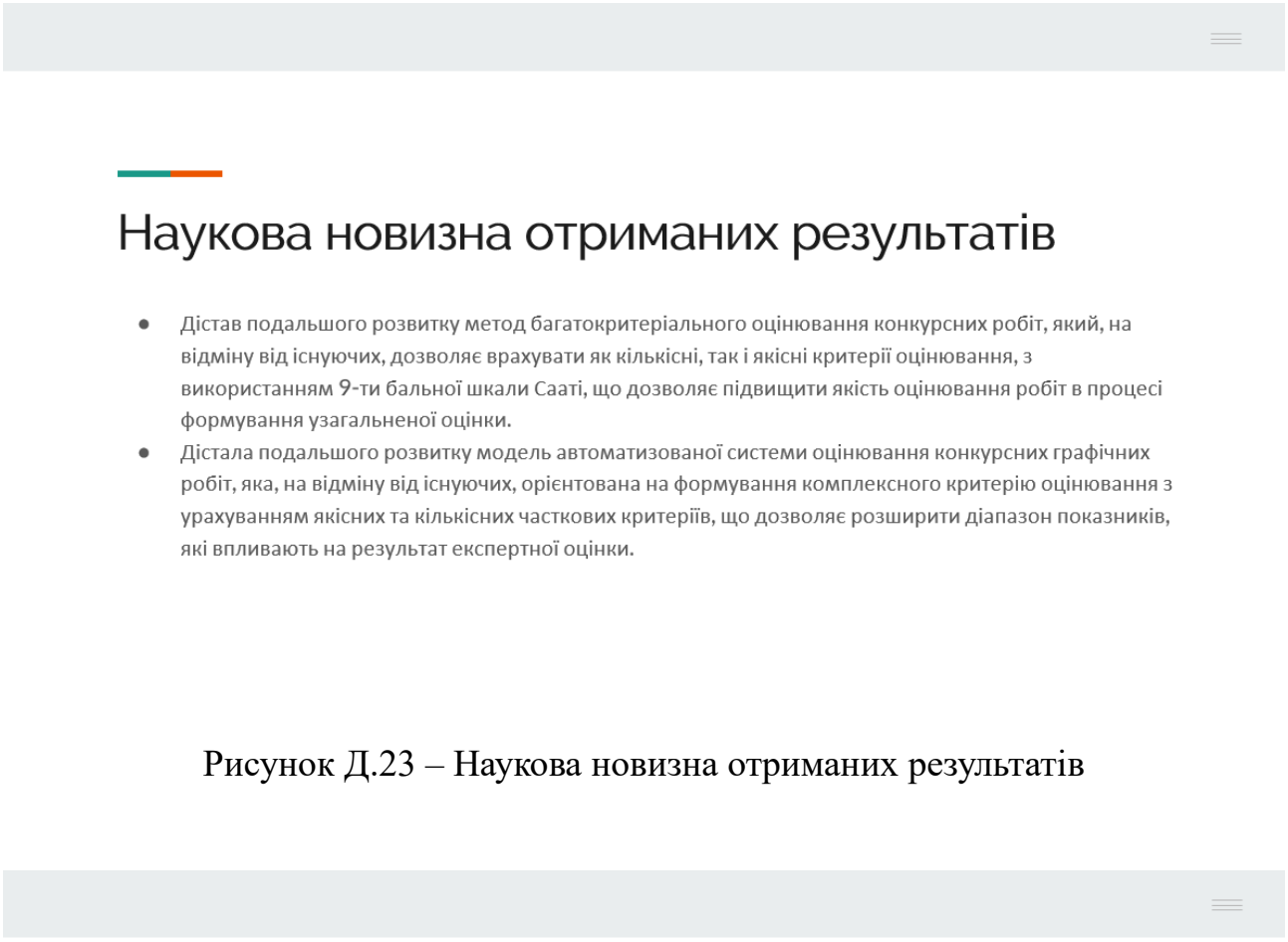


Рисунок Д.23 – Наукова новизна отриманих результатів

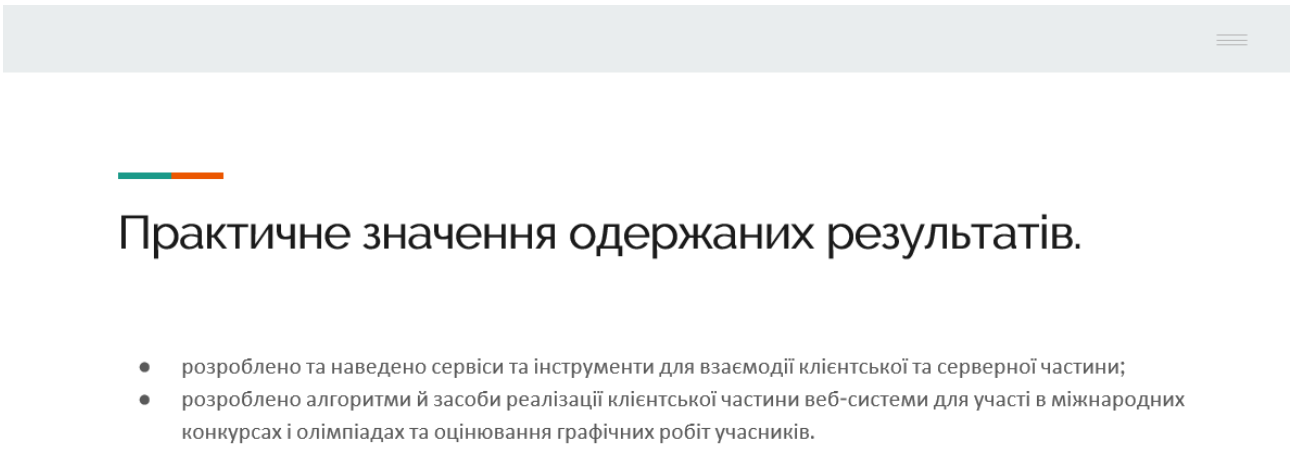


Рисунок Д.24 – Практичне значення одержаних результатів

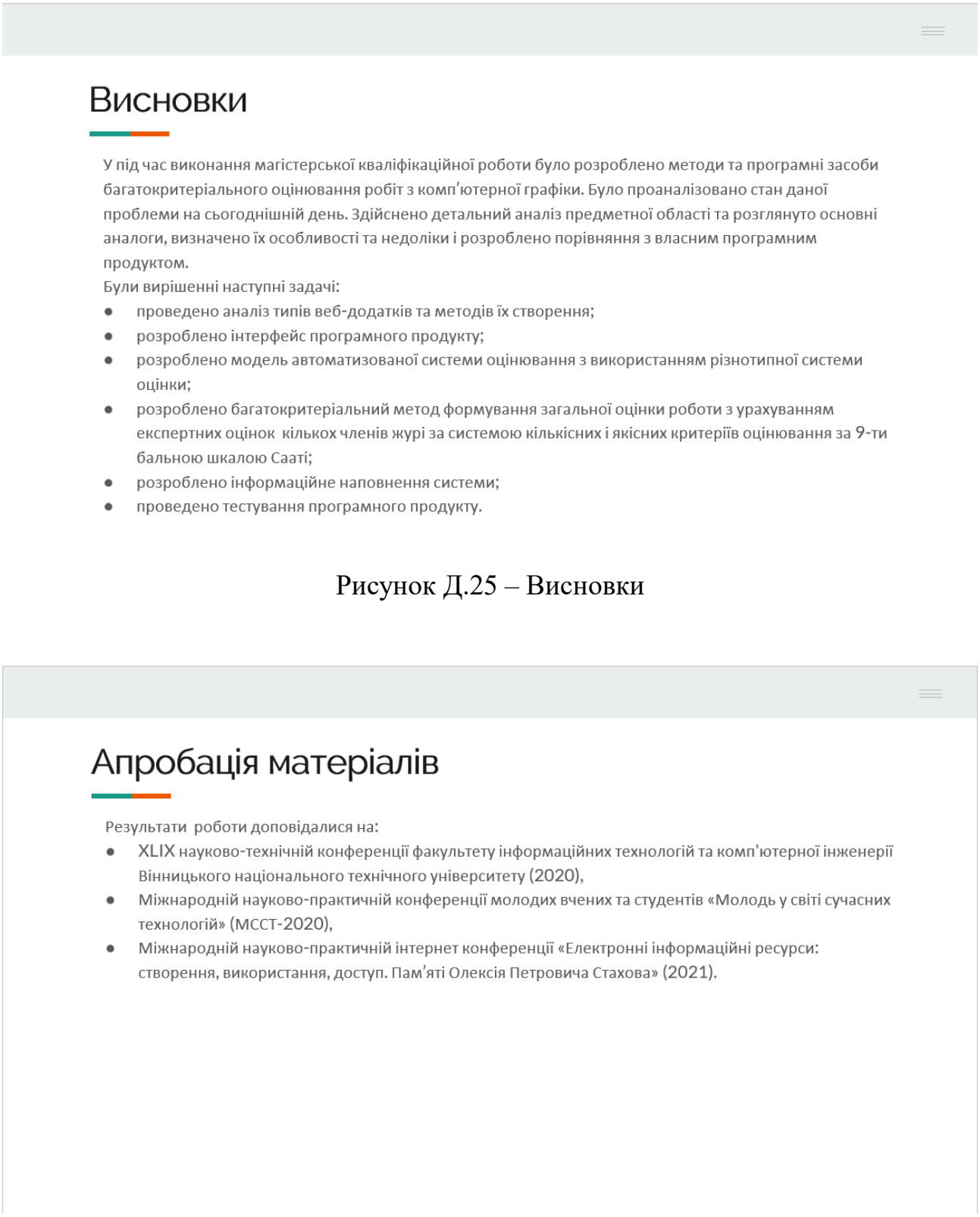
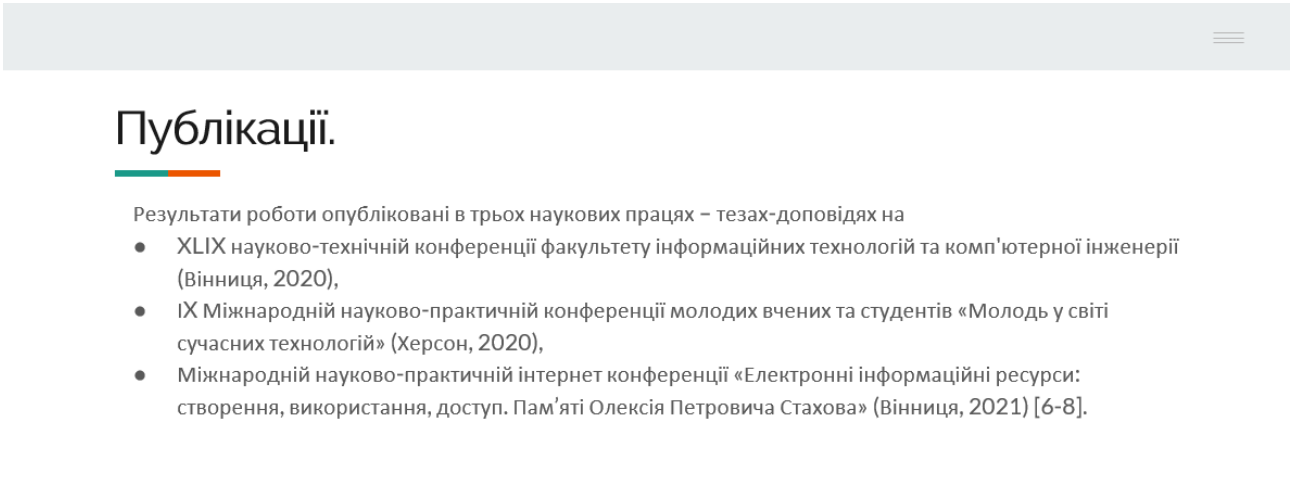


Рисунок Д.25 – Висновки

Рисунок Д.26 – Апробація матеріалів



Публікації.

Результати роботи опубліковані в трьох наукових працях – тезах-доповідях на

- XLIX науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (Вінниця, 2020),
- IX Міжнародній науково-практичній конференції молодих вчених та студентів «Молодь у світі сучасних технологій» (Херсон, 2020),
- Міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ. Пам'яті Олексія Петровича Стахова» (Вінниця, 2021) [6-8].

Рисунок Д.27 – Публікація матеріалів