Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ПРОТОТИПУ РЕЙТИНГОВОЇ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ ОБМІНУ МУЛЬТИМЕДІА

DEVELOPMENT OF A PROTOTYPE OF A RATING SOCIAL NETWORK FOR MULTIMEDIA EXCHANGE

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
Хомік Владислав Юрійович
_____
(підпис)

Керівник:
д.т.н., професор
Гордєєв Олександр Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«08» 06 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«02» 02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Хомік Владислав Юрійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розробка прототипу рейтингової соціальної мережі для обміну мультимедіа

Керівник роботи: д.т.н., професор Гордєєв Олександр Олександрович
затверджені наказом закладу вищої освіти від «*23*» *грудня* *2022* р. № *961-01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«8» червня 2023 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу, Мова програмування серверної частини – C#, Технології програмування клієнтської частини – HTML, CSS, TypeScript.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та надійністних параметрів проектованої системи, функціонально-структурна схема роботи об'єкта проектування, розробка моделі бази даних об'єкта проектування.

5. Перелік графічного матеріалу: 7 рис

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гордєєв О.О.*		
Специфікація вимог до розробленої системи	*Гордєєв О.О.*		
Розробка об'єкта проектування	*Гордєєв О.О.*		
Нормоконтроль	*Гордєєв О.О.*		
Гарант ОП	*Ліщина Н.М.*		
Показник запозичень тексту	8,51%		
Академічна доброчесність	*Ящук А.А.*		

7. Дата видачі завдання *«2» лютого 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*05.02.2023 р.*	Вик.
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2023 р.*	Вик.
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*10.03.2023 р.*	Вик.
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2023 р.*	Вик.
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2023 р.*	Вик.
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023 р.*	Вик.
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*08.06.2023 р.*	Вик.

Здобувач вищої освіти _____ Халіс В.Ю.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гордєєв О.О.
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти: Халік Владислав Юрійович
(ПІБ)

Тема кваліфікаційної роботи: Розробка прототипу рейтингової соціальної мережі для обміну мультимедіа

(повна назва згідно наказу)

Коротка характеристика кваліфікаційної роботи та прийнятих рішень:
Робота складається з трьох розділів, протягом яких дипломантом здійснюється огляд поставленої перед нами проблеми, пошук шляхів її вирішення та опис їх реалізації. Було використано наступні програмні засоби: Angular - клієнтська частина, ASP.NET Core Web API - серверна, SQL - база даних

Самостійні розробки і пропозиції автора:
Кандидат пропонує реалізувати рейтингову систему, яка дозволяє визначити популярний та цікавий контент на основі оцінок користувачів. Це допомагає фільтрувати контент і показувати його тим користувачам, які ймовірно більше зацікавлені в ньому.

Практичне значення роботи:
Кандидат створив просту соціальну мережу з рейтинговою системою, яка може стимулювати користувачів створювати високоякісний контент, оскільки вони отримують позитивну оцінку та визнання від інших користувачів. Це може призвести до підвищення загального рівня якості контенту.

Недоліки:
У роботі не виявлено новизни - вона більш походить на альтернативну існуючим варіантам. Проте складна структура роботи добре висвітлює вплив дипломанта.

Загальний висновок:
Рівень якості виконаної роботи є задовільним. Усі поставлені завдання виконані. Кваліфікаційна робота виконана на якісному рівні тому рекомендується до захисту і заслуговує на позитивну оцінку

Рецензент к. т. н. доц. (підпис) (науковий ступінь, вчене звання, посада, ПІБ) Поліщук М.М.

«28» червня 2023 р.

АНОТАЦІЯ

Хомік В. Ю. Розробка прототипу рейтингової соціальної мережі для обміну мультимедіа. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2023.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

В першому розділі було розглянуто існуючих рейтингових соціальних мереж і мультимедійних платформ, визначення основних функцій та вимог до системи обміну мультимедіа, вивчення потенційної цільової аудиторії та її потреб.

Другий розділ описує функціональні та нефункціональні вимоги, інтеграцію з іншими компонентами та вимоги до безпеки. Це допомагає забезпечити чітке розуміння вимог до системи та їх відповідність у процесі розробки та тестування.

В третьому розділі показана практична реалізація.

Ключові слова: соціальна мережа, Angular, ASP.NET Core API, SQL, контролер.

ANNOTATION

Khomik V. Yu. Development of a prototype of a rating social network for sharing multimedia. Manuscript.

Bachelor's qualifying thesis of the OP "Software Engineering". Lutsk National Technical University. Lutsk, 2023

The bachelor's qualification work consists of an introduction, three sections, conclusions and proposals, a list of used sources and appendices.

In the first section, the existing rating social networks and multimedia platforms were considered, the definition of the main functions and requirements for the multimedia exchange system, the study of the potential target audience and its needs.

The second section describes functional and non-functional requirements, integration with other components, and security requirements. This helps ensure that system requirements are clearly understood and met during development and testing.

The third section shows the practical implementation.

Keywords: social network, Angular, ASP.NET Core API, SQL, controllers.

ЗМІСТ

ВСТУП

За останні роки соціальні мережі значно розвинулися та зайняли важливе місце в інтернет-просторі. Вони стали популярними засобами спілкування, обміну інформацією та розвагами. Крім того, соціальні мережі надають можливість широкому колу людей ділитися своїми досягненнями та знайомитися з творчістю інших людей.

Незважаючи на широке поширення та популярність соціальних мереж, існують деякі прогалини в цій сфері, які можуть бути виправлені або поліпшені за допомогою нових розробок та інновацій. Соціальні мережі часто стають мішенню для кібератак та зловмисників, які можуть отримувати доступ до особистої інформації користувачів. У соціальних мережах часто з'являється шкідливий контент, який може включати фейки, насильство, образи та інші негативні матеріали. Недостатня модерація та контроль можуть призвести до шкідливого впливу на психіку та здоров'я користувачів. Використання соціальних мереж може стати залежністю та призвести до перенасиченості інформацією. Також за останні роки були виявлені випадки поширення фейкової та недостовірної інформації в соціальних мережах, що може мати серйозний вплив на суспільство та політику.

Світові тенденції розв'язку проблем соціальних мереж орієнтовані на різні напрямки. Одним з підходів є збільшення відповідальності соціальних мереж за збір та використання даних користувачів. Недавні скандали з популярними соціальними мережами довели, що компанії повинні докладати більше зусиль для захисту конфіденційної інформації своїх користувачів. Соціальні мережі повинні зробити більше, щоб забезпечити належну модерацію та контроль контенту. Наразі деякі компанії використовують штучний інтелект та інші технології для автоматизованої модерації, проте це може призвести до помилкового видалення деякого корисного контенту. Тому, більшість соціальних мереж потребують додаткового підходу до модерації та контролю. З'являються нові соціальні мережі,

які намагаються збільшити свою користувацьку базу, надаючи нові можливості та гарантуючи більшу приватність та безпеку даних користувачів. Такі платформи, наприклад, Mastodon, Diaspora, Minds та інші, пропонують більш захищену та приватну альтернативу для користувачів. Важливо надавати користувачам соціальних мереж можливість розуміти та розпізнавати фейкову та недостовірну інформацію.

Метою дослідження є програмний продукт для обміну мультимедіа, який буде відповідати вимогам користувачів та конкурентним перевагам на ринку.

Завдання дослідження:

– визначення вимог та потреб користувачів соціальних мереж, що пов’язані з обміном мультимедіа;

– аналіз конкурентного середовища та ринку соціальних мереж, зокрема, мереж, які спеціалізуються на обміні мультимедіа;

– визначення технічних можливостей розробки мережі, в тому числі вибір платформи, інструментів розробки та іншого програмного забезпечення;

– проектування інтерфейсу та взаємодії користувачів з мережею;

– розробка алгоритмів та технологій, необхідних для роботи мережі, в тому числі алгоритмів рейтингування та фільтрації контенту;

– розробка тестових завдань для оцінки роботи мережі та її можливостей;

– оцінка ефективності та розробка рекомендацій щодо подальшого розвитку мережі.

Об’єкт дослідження є процес створення нової соціальної мережі, яка буде спрямована на обмін мультимедіа (такі як фото, відео, аудіо тощо) та включатиме функціонал оцінювання контенту користувачами. Досліджується якість роботи мережі, її можливості, вигляд та інтерфейс, а також технології, що використовуються в процесі розробки. Вивчається соціальний аспект роботи мережі, а саме, як користувачі взаємодіють між собою та як впливають на розвиток

мережі. Проводиться аналіз ринку та конкурентів, щоб зрозуміти, як можна зробити мережу більш привабливою та конкурентоспроможною.

Предметом дослідження є розробка прототипу рейтингової соціальної мережі для обміну мультимедіа. Досліджується робота самої мережі, її можливості, вигляд та інтерфейс, а також технології, що використовуються в процесі розробки.

Робота є новизною у зв’язку зі зростанням популярності використання мультимедійного контенту в соціальних мережах та потребою у забезпеченні безпеки та конфіденційності користувачів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

За останні кілька років соціальні мережі стали важливою частиною нашого життя. Вони дозволяють нам спілкуватись з друзями, знайомитись з новими людьми, ділитись своїми думками, ідеями та фотографіями. Однак, з поширенням соціальних мереж з'явилися деякі проблеми, такі як недостатній захист персональних даних, залежність від соціальних мереж та поширення фейкових новин.

Однією з найбільш значущих проблем у сфері соціальних мереж є недостатній захист персональних даних. Користувачі соціальних мереж часто надають особисту інформацію, таку як ім'я, адресу електронної пошти та номер телефону, що може бути використана для спаму та шахрайства [1]. Крім того, соціальні мережі можуть використовувати персональну інформацію користувачів для спрямування реклами. Зараз у світі немає єдиного законодавства, яке б визначало правила збереження та обробки персональних даних користувачів соціальних мереж. Це створює ризик витоку конфіденційної інформації та може призвести до неправомірних дій з використанням цих даних.

На сьогоднішній день соціальні мережі мають величезну кількість мультимедійного контенту, який завантажують користувачі. Проте часто зустрічається небажаний контент, який може бути образливим, нецензурним, агресивним тощо. Для боротьби з цією проблемою необхідно забезпечити контроль якості мультимедійного контенту та вчасну реакцію на порушення правил користування.

Реклама є неодмінною складовою соціальних мереж, але часто користувачі відчувають дискомфорт від її перенасиченості та нав'язливості. Крім того, реклама

може мати негативний вплив на психологічний стан користувачів та їх поведінку в соціальних мережах.

Ще однією проблемою у сфері соціальних мереж є залежність від них. Це може вплинути на наше здоров'я та здатність до соціального взаємодії в реальному житті. Люди можуть проводити безліч часу в соціальних мережах, замість того, щоб відвідувати родину та друзів, займатись спортом або розвивати свої професійні навички.

Також, поширення фейкових новин стало проблемою в соціальних мережах. Багато людей отримують свої новини з соціальних мереж, де фейкові новини можуть поширюватись дуже швидко і займати значну частину новинної стрічки.

Дослідження також показало, що багато користувачів соціальних мереж більше довіряють рекомендаціям від своїх друзів та знайомих, ніж рекламним матеріалам. Це створює потребу у соціальних мережах, які можуть забезпечувати обмін думками та рекомендаціями між користувачами.

Окрім того, з розвитком мультимедійних технологій зростає інтерес користувачів до відео та аудіо контенту. Відео стає все більш популярним форматом для споживання інформації в соціальних мережах. Наразі такі соціальні мережі, як YouTube та Instagram, надають користувачам можливість ділитися відео та іншим мультимедіа контентом. Однак, існує потреба у соціальних мережах, які спеціалізуються на обміні мультимедійним контентом та забезпечують користувачам зручний та ефективний спосіб його пошуку та споживання.

Також, з ростом кількості користувачів соціальних мереж збільшується потреба в ефективному управлінні та аналізі великих обсягів даних, що генеруються цими мережами. Саме тому, виникає потреба у розробці нових технологій, які можуть забезпечити ефективний аналіз та управління соціальними мережами.

Враховуючи всі вищезазначені фактори, розробка прототипу рейтингової соціальної мережі для обміну мультимедіа є актуальним та перспективним напрямком дослідження.

Другим етапом аналізу було вивчення сучасних соціальних мереж, що надають можливості обміну мультимедіа контентом. Основними з них є Facebook, Instagram, Snapchat, TikTok та Pinterest.

Facebook є найбільшою соціальною мережею у світі зі значною кількістю активних користувачів. Однак, Facebook зосереджений на текстових та фото контентах, тоді як мультимедіа контент може бути зручніше обмінюватись на інших платформах. Крім того, Facebook має проблему з захистом даних користувачів, що може вплинути на їхню довіру до платформи.

Instagram є однією з найпопулярніших соціальних мереж у світі, особливо серед молоді. Instagram дозволяє обмінюватися фото та відео контентом, а також містить функцію «Stories», де користувачі можуть додавати короткі відео та зображення, які зникають через 24 години. Однак, Instagram має проблему з низькою якістю контенту, оскільки користувачі часто редагують фотографії за допомогою фільтрів, що може призвести до спотворення реальності.

Snapchat є платформою, яка спеціалізується на обміні короткими відео та зображеннями, які також зникають через короткий проміжок часу. Snapchat має популярність серед молодіжної аудиторії, оскільки має дещо більші можливості для креативного використання контенту. Однак, Snapchat не є платформою для більш широкої аудиторії, що може знизити його потенціал для комерційних цілей.

Однією з найбільш популярних соціальних мереж сьогодні є TikTok. Ця мережа пропонує користувачам створювати короткі відео на різноманітні теми, додавати музику та ефекти, та ділитися ними зі своїми підписниками. TikTok набув великої популярності серед молоді і його активність та обсяги користувачів продовжують зростати. Ця мережа має кілька недоліків, які варто врахувати. Один з головних недоліків – це потенційні проблеми з безпекою і захистом приватності.

Наприклад, деякі користувачі можуть ділитися непристойним або небезпечним контентом, а також створювати фальшиві профілі або спамити користувачів. Інший недолік TikTok – це ризик залучення до поганого контенту, зокрема реклами та вірусів. Також можливість залучення дітей та підлітків до небезпечних тенденцій і викликів або, як їх ще називають, «челенджів».

Ще одною популярною соціальною мережею є Pinterest, яка пропонує користувачам зберігати та ділитися фотографіями та ідеями на різноманітні теми, такі, як мода, краса, дизайн інтер'єру, кулінарія та багато іншого. Pinterest є популярним серед осіб, які зацікавлені в творчій інспірації та знаходженні нових ідей. Pinterest також має свої недоліки. Один з них полягає в тому, що більшість користувачів – жінки, тому це може бути менш привабливим для компаній, що працюють з більш широкою аудиторією. Також, як і в інших соціальних мережах, Pinterest може бути платформою для розповсюдження фейкових новин та неперевіреної інформації, що може завдати шкоди користувачам. Крім того, іноді Pinterest може бути недостатньо інтерактивною платформою, де користувачі більше споживають контент, а не створюють його. Однак, це залежить від того, як користувачі використовують платформу.

Загалом, сучасний стан предметної області свідчить про те, що соціальні мережі мають свої переваги та недоліки. У зв'язку з цим, є потреба у розробці нових технологій та підходів до створення соціальних мереж, що враховуватимуть побажання користувачів та забезпечуватимуть їхню безпеку та конфіденційність даних. У дослідженні буде використовуватись досвіду вищеперечислених та інших соціальних мереж.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Наступні цілі є важливою основою для розробки проекту:

– розробка системи реєстрації користувачів, яка дозволить створювати нові облікові записи з унікальними ідентифікаторами та паролями;

– створення профілю користувача. Розробка функціоналу для створення та управління профілями користувачів, включаючи відображення основної інформації, фотографій та налаштування конфіденційності;

– завантаження мультимедіа-контенту. Реалізація можливісті завантаження різних типів мультимедіа-контенту, таких як фотографії або відео. Створення зручного інтерфейсу для завантаження та обробки цього контенту;

– коментування публікацій і встановлення рейтингу. Реалізація функцій рейтингової системи, яка дозволить користувачам виражати свою думку щодо контенту, а також обговорювати його з іншими користувачами;

– пошук та фільтрація контенту. Надання можливісті пошуку та фільтрації мультимедіа-контенту на основі різних критеріїв, таких як користувачі, хештеги або категорії;

– захист і безпека даних. Встановлення високого рівня захисту та безпеки для користувачів, зокрема шляхом застосування механізмів аутентифікації, авторизації та обмеження доступу до конфіденційної інформації;

– тестування та налагодження. На фінальній стадії потрібно провести ретельне тестування прототипу, виявляючи та виправляючи неполадки. Всі функції мають працювати належним чином. Інтерфейс користувача повинен бути інтуїтивно зрозумілим.

Висновки до розділу 1

В першому розділі проведено ретельний аналіз та огляд сучасного стану предметної області, що становить основу для подальшого виконання

кваліфікаційної роботи. Під час проведення аналізу були використані різноманітні джерела інформації, такі як наукові статті, книги, інтернет-ресурси.

У результаті проведеного аналізу були ідентифіковані основні проблеми, тенденції та виклики, що виникають у предметній області. Особлива увага була приділена актуальним проблемам, що потребують негайного вирішення, а також новим напрямкам розвитку, які можуть стати перспективними у майбутньому.

З метою досягнення поставлених цілей кваліфікаційної роботи було сформульовано серію завдань. Кожне завдання відображає конкретну проблему або аспект предметної області, який потребує розгляду та дослідження.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Однією з головних функціональних вимог для розробки прототипу рейтингової соціальної мережі є можливість користувачів розміщувати різноманітний мультимедіа контент на своїх сторінках [2]. Крім того, користувачі повинні мати можливість переглядати контент, який був розміщений іншими користувачами.

Також, система має надавати можливість користувачам позначати контент позитивними або негативними відгуками, а також залишати коментарі на сторінках інших користувачів. Крім того, повинна бути можливість здійснювати пошук контенту за різними критеріями, такими як категорія, тематика, автор тощо.

Іншими важливими функціональними вимогами є можливість створювати групи користувачів, в яких можна ділитися контентом, інформацією, ідеями тощо. Також, система повинна мати можливість надавати статистику та аналітику про використання користувачами мережі, яка може допомогти покращити функціонал мережі та зрозуміти інтереси користувачів.

Отже, функціональні вимоги до розроблюваної соціальної мережі включають такі функції:

– реєстрація та авторизація користувачів;

– створення профілів користувачів з можливістю додавання інформації про себе;

– перегляд профілів інших користувачів;

– можливість додавання і видалення друзів;

– обмін повідомленнями та файлами між друзями;

– створення публікацій (зображень, відео, аудіо) з можливістю додавання опису та тегів;

– перегляд та пошук публікацій інших користувачів за тегами та категоріями;

– можливість відзначати подобається та коментувати публікації інших користувачів;

– розміщення реклами в соціальній мережі;

– налаштування приватності профілю та публікацій.

Ці функції дозволять користувачам створювати та обмінюватися мультимедіа-контентом, знаходити нових друзів, спілкуватися з іншими користувачами, а також забезпечать комфортні умови використання соціальної мережі.

Для розробки прототипу рейтингової соціальної мережі необхідно визначити технічні вимоги, які дозволять реалізувати всі необхідні функції та забезпечити надійну та швидку роботу системи.

Основні технічні вимоги до розроблюваної системи можна сформулювати наступним чином:

– мова програмування та фреймворки. Для розробки прототипу рейтингової соціальної мережі можна використовувати різні мови програмування та фреймворки. Для забезпечення високої продуктивності та ефективності системи можна використовувати мови програмування, такі як Python, Ruby, Java, або Node.js. Для розробки користувацького інтерфейсу можна використовувати фреймворки, такі як React або Vue.js;

– база даних. Для зберігання інформації про користувачів, їх контент та іншу необхідну інформацію, необхідно використовувати реляційну базу даних. Наприклад, можна використовувати MySQL або PostgreSQL. Для швидкого доступу до даних можна використовувати кешування даних в оперативній пам’яті;

– безпека. Система повинна бути захищеною від зламів та зловмисних дій. Для цього можна використовувати різні методи захисту, такі як хешування паролів,

захист від SQL-ін’єкцій та XSS-атак, аутентифікацію та авторизацію користувачів, а також шифрування даних;

– масштабованість. Система має бути здатна працювати з великою кількістю користувачів та забезпечувати швидкий доступ до контенту в будь-який час;

– мультиплатформеність. Система має бути доступна на різних пристроях та операційних системах, таких як iOS, Android та Windows;

– керованість: Система повинна бути легко керованою, з можливістю зміни налаштувань, модерації контенту та користувачів;

– функціональність. Система повинна мати різноманітні функції, такі як пошук, коментування, лайки, підписки, сповіщення та інші, щоб забезпечити належний рівень зручності та задоволення користувачів.

До процесних вимог можна віднести такі:

– реєстрація користувачів. У процесі реєстрації потрібно збирати базову інформацію про користувача, включаючи електронну пошту, пароль, ім’я та прізвище. Також можна додатково запитувати інформацію, наприклад, про вік, стать, місце проживання, захоплення та інші, що допоможуть підібрати контент для користувача;

– створення та редагування профілю. Після реєстрації користувач може створити свій профіль, додати фото, опис, посилання на інші соціальні мережі та інші дані. Також користувач повинен мати можливість редагувати свій профіль у будь-який момент;

– додавання та перегляд контенту. Головною функцією соціальної мережі для обміну мультимедіа є можливість додавання та перегляду різних типів контенту, таких як фото, відео, аудіо та інші. Користувач повинен мати можливість додавати контент, описувати його та встановлювати настройки приватності;

– пошук та фільтрація контенту. З великою кількістю контенту, доступного на соціальній мережі, важливо мати можливість швидко знайти потрібний контент. Пошук може бути здійснений за різними параметрами, такими як хештеги, назви,

автори, теги та інші. Крім того, користувач повинен мати можливість фільтрувати контент за різними параметрами;

– розробка та тестування прототипу. Після визначення вимог до прототипу необхідно розробити та протестувати прототип рейтингової соціальної мережі для обміну мультимедіа. Розробка може бути проведена за допомогою різних технологій та інструментів, включаючи мови програмування, фреймворки, бази даних тощо. Тестування прототипу допоможе перевірити, чи відповідає розроблений прототип вимогам та функціональним вимогам, а також виявити та виправити помилки та недоліки;

– розгортання та налаштування. Після успішного тестування прототипу рейтингової соціальної мережі, необхідно розгорнути його на сервері та налаштувати для користування. Це може включати встановлення необхідного програмного забезпечення та налаштування бази даних, сервера та інших компонентів;

– підтримка та розвиток. Після успішного розгортання та налаштування прототипу необхідно забезпечити його подальшу підтримку та розвиток. Це може включати виправлення помилок, оновлення функціональності та забезпечення безпеки;

– тестування та вдосконалення. Після розгортання та підтримки прототипу необхідно проводити регулярне тестування та вдосконалення рейтингової соціальної мережі. Це допоможе забезпечити безперебійну роботу мережі та забезпечити її функціональність та ефективність.

Нижче наведено перелік нефункціональних вимог:

– безпека. Система повинна забезпечувати високий рівень безпеки даних користувачів та їх приватності. Для цього необхідно використовувати шифрування та інші методи захисту інформації;

– швидкість роботи. Ресурс повинен швидко відповідати на запити користувачів та надавати можливість швидко завантажувати мультимедійний контент. Це дуже важливо для забезпечення комфортного користування сервісом;

– надійність. Ресурс повинен бути максимально стійким до збоїв та відмов. Це допоможе забезпечити безперебійну роботу сервісу та уникнути втрати даних;

– масштабованість. Ресурс повинен бути готовий до зростання обсягів роботи з часом. Система повинна легко масштабуватися для забезпечення оптимальної роботи при збільшенні кількості користувачів та обсягів даних;

– доступність. Ресурс повинен бути доступним для користувачів у будь-який час та з будь-якого пристрою. Система повинна працювати стабільно та надавати можливість користувачам взаємодіяти з нею з будь-якого місця та на будь-якому пристрої з доступом до Інтернету;

– інтуїтивно зрозумілий інтерфейс. Інтерфейс повинен бути простим та зрозумілим для користувачів. Він повинен дозволяти швидко та легко знаходити необхідні функції та взаємодіяти з системою.

Користувацький інтерфейс є однією з найважливіших складових будь-якого додатку, в тому числі і соціальної мережі. У розробка відбуватиметься відповідно до сучасних тенденцій та кращих практик розробки інтерфейсу для забезпечення максимальної зручності та задоволення користувачів. Основні вимоги до користувацького інтерфейсу такі:

– інтуїтивно зрозумілий інтерфейс – користувач повинен з легкістю знайти потрібні функції та можливості без додаткових пояснень;

– привабливий та сучасний дизайн – візуальна привабливість інтерфейсу є ключовим фактором для залучення та утримання користувачів;

– адаптивність - інтерфейс повинен працювати коректно на різних розмірах екранів та пристроях;

– швидкість та ефективність – користувачі очікують, що робота з соціальною мережею буде швидкою та ефективною, тому інтерфейс повинен бути оптимізований для максимальної продуктивності;

– забезпечення приватності – користувачі повинні мати можливість контролювати свої дані та налаштування приватності через інтерфейс соціальної мережі.

Для досягнення цих вимог будуть використовуватися сучасні технології та підходи до розробки інтерфейсу, такі як розробка на базі компонентів та створення макетів у вигляді компонентів для швидкої і ефективної розробки. Крім того, буде проводитися тестування інтерфейсу з метою виявлення можливих проблем та покращень.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Обрання технологій та інструментів для розробки програмного продукту є важливим етапом в процесі його створення. Вибір певної технології повинен базуватися на потребах проекту, а також на можливостях команди розробників. Для серверної частини я обрав .NET, оскільки це надійна технологія, яка надає високу продуктивність та безпеку. Крім того, .NET має широку підтримку спільноти розробників, що забезпечує наявність різноманітних інструментів для розробки та тестування програмного продукту [3]. Однією з основних переваг використання .NET є те, що він є повноцінною платформою, яка дозволяє створювати як десктопні, так і веб-додатки. .NET є зручним інструментом для розробки великих та складних систем, оскільки він забезпечує високу гнучкість та підтримує багатофункціональність. Ще однією важливою перевагою .NET є його висока продуктивність та ефективність. Завдяки використанню компіляції перед виконанням, програми на .NET виконуються значно швидше, ніж інші інтерпретовані мови програмування [4]. .NET також підтримує багатопоточність,

що дозволяє ефективно використовувати ресурси обчислювальних систем. Крім того, .NET є платформою з великою кількістю розширень та бібліотек, що значно полегшує розробку програмного забезпечення. Розробники можуть використовувати готові бібліотеки для виконання рутинних завдань, що дозволяє зосередитися на більш складних задачах. Також, .NET має вбудовану систему безпеки, яка забезпечує захист від хакерських атак та інших загроз безпеці. Більшість компонентів, що входять до .NET, підтримують підхід «безпеки за замовчуванням», що дозволяє уникнути багатьох звичайних вразливостей [5].

Отже, .NET є досить потужним та гнучким інструментом для розробки програмного забезпечення, який дозволяє створювати високопродуктивні та ефективні додатки з безпечним кодом.

Для створення користувацького інтерфейсу обрано Angular. Це фреймворк для розробки веб-додатків, який дозволяє створювати багатокомпонентні та масштабовані додатки [6]. Angular забезпечує високу продуктивність та ефективність, а також зручну структуру для розробки. Крім того, Angular має широку спільноту розробників, що забезпечує наявність багато різних компонентів та модулів для розробки користувацького інтерфейсу. Основними перевагами Angular є:

– підтримка структурованого підходу до розробки. Angular пропонує компонентну архітектуру, яка дозволяє розбити додаток на окремі компоненти і зменшити залежності між ними. Це полегшує розробку та підтримку додатку;

– багатофункціональність. Angular надає багато корисних функцій, таких як маршрутизація, форми, валідація даних, обробка подій, і багато іншого. Це дозволяє розробникам швидко створювати високоякісні додатки з мінімальною кількістю коду;

– розширюваність. Angular підтримує модульну архітектуру, що дозволяє додавати нові функціональність, не переписуючи весь додаток [7]. Це зменшує ризик появи помилок і забезпечує легкість розширення додатку в майбутньому;

– підтримка TypeScript. TypeScript – це підмножина JavaScript, яка додає сильну типізацію та інші корисні функції. Angular повністю підтримує TypeScript, що дозволяє розробникам створювати більш безпечний та структурований код [8];

– активна спільнота. Angular має широку та активну спільноту розробників, які надають підтримку, створюють нові інструменти та розширення, а також допомагають вирішувати проблеми. Це дозволяє розробникам швидко знайти відповіді на свої запитання та вирішити проблеми під час розробки додатку.

Для збереження даних обрано SQL, оскільки вона є надійною та стабільною технологією для збереження даних. SQL є дуже популярною системою керування базами даних, що забезпечує високу швидкість роботи з даними та безпеку їх збереження [9]. Крім того, SQL має велику спільноту розробників, що забезпечує наявність багато різних інструментів та бібліотек для розробки та підтримки баз даних. SQL (Structured Query Language) є стандартною мовою для керування реляційними базами даних, і має кілька переваг в порівнянні з іншими системами керування базами даних. Основні переваги SQL такі:

– ефективність. SQL забезпечує швидкий доступ до даних і є ефективним виконавцем запитів. Багато операцій можуть бути виконані на більш високому рівні, що дозволяє значно скоротити час виконання запитів до бази даних;

– простота використання. SQL має простий синтаксис, що дозволяє легко створювати та модифікувати бази даних. Це дає змогу швидко розгорнути систему бази даних та швидко виконувати операції з даними [10];

– масштабованість. SQL може бути використаний для роботи з даними будь-якого розміру, що дозволяє легко масштабувати бази даних зі зростанням обсягу даних;

– надійність. SQL забезпечує високу рівень надійності, що робить його найпопулярнішою системою керування базами даних серед підприємств та організацій;

– підтримка. SQL має велику спільноту розробників та користувачів, що забезпечує підтримку та розвиток системи. Це дає змогу використовувати нові технології та функції, які постійно додаються до системи [11].

У загальному, SQL забезпечує ефективний та простий спосіб управління даними, має високу надійність та широку підтримку. Це робить його ідеальним вибором для багатьох проектів та додатків, які потребують керування базами даних.

Висновки до розділу 2

У другому розділі були детально описані вимоги системи, що включають функції, які система повинна виконувати, та їх залежності. Кожна функція була розглянута з точки зору її важливості та пріоритетності, що дозволяє забезпечити ефективне планування та розробку системи.

Крім того, були визначені нефункціональні вимоги, такі як надійність, швидкодія, безпека, масштабованість та інші параметри якості системи. Ці вимоги визначаються з урахуванням потреб користувачів і стандартів, які повинні бути виконані системою.

При проведенні аналізу предметної області та постановці завдань на кваліфікаційну роботу були розглянуті різні засоби і методи, які можуть бути використані для розв'язання поставленої проблеми.

РОЗДІЛ 3
РОЗРОБКА ПРОТОТИПУ РЕЙТИНГОВОЇ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ ОБМІНУ МУЛЬТИМЕДІА

3.1 Практична реалізація об’єкта проектування

Для реалізації фронтенду і бекенду використовуються окремі проекти. Цей підхід дозволяє зберігати фронтендову та бекендову логіку в окремих кодових базах, що дозволяє краще організувати розробку, розширення і тестування проекту. Цей підхід дозволяє забезпечити відокремленість фронтенду і бекенду, спрощує розгортання та масштабування окремих частин проекту. Кожен проект може мати свою власну структуру, залежності, конфігурацію і тестування. Як видно на рисунку 3.1, запускається одночасно два проекти.

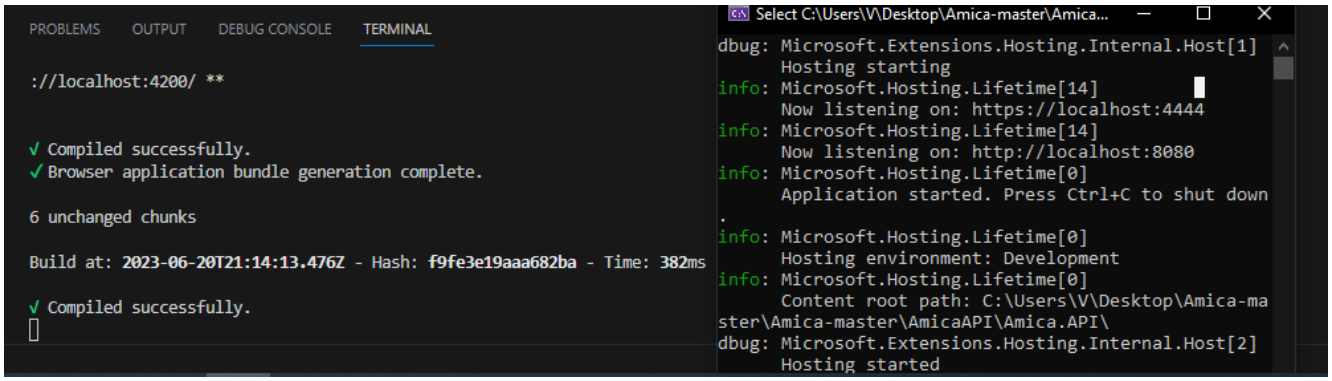

Рисунок 3.1 – Запуск серверів

Angular може взаємодіяти з серверною частиною, яка реалізована з використанням ASP.NET Core API, за допомогою HTTP запитів. Завдяки HTTP-протоколу, Angular може відправляти запити на сервер та отримувати відповіді з даними, що дозволяє реалізувати обмін даними між клієнтом та сервером. На рисунку 3.2 зображена схема такої взаємодії.

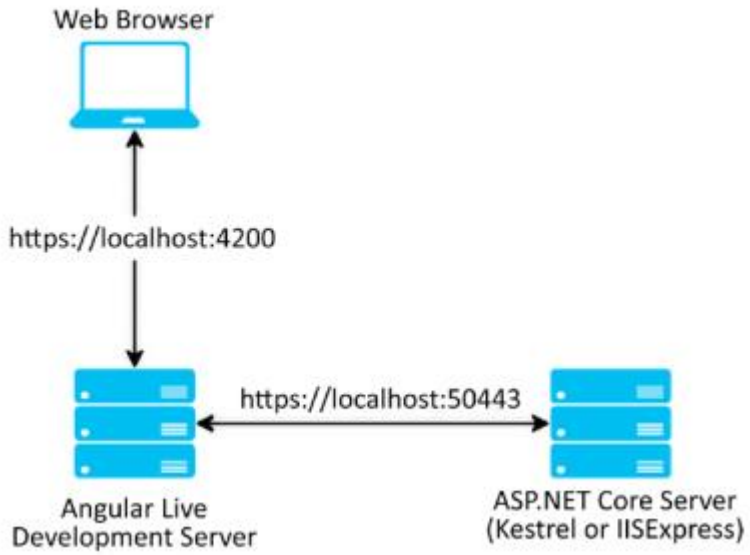

Рисунок 3.2 – Схема взаємодії клієнтської і серверної частини

Для взаємодії з серверною частиною в Angular використовуються Angular сервіси та модуль HttpClient. Angular сервіси дозволяють групувати логіку взаємодії з сервером в окремі модулі, які можуть бути використані в різних компонентах додатку. HttpClient модуль надає можливість відправляти HTTP запити і обробляти їх відповіді.

Для виконання запитів до сервера, Angular використовує HTTP методи, такі як GET, POST, PUT, DELETE тощо. За допомогою цих методів можна отримувати дані з сервера (GET), відправляти дані на сервер (POST, PUT), а також видаляти дані з сервера (DELETE).

При відправці запитів, Angular може передавати параметри запиту, які можуть бути використані на сервері для обробки запиту. Наприклад, при отриманні списку користувачів, можна передати параметри, такі як фільтри, сортування або сторінкування, для отримання відповіді зі списком користувачів, що відповідають заданим умовам.

Після відправки запиту, Angular може отримати відповідь в форматі JSON або іншому, залежно від налаштувань серверної частини. Отримані дані можуть

бути оброблені в Angular та використані для відображення відповідних даних в компонентах та шаблонах.

Крім того, Angular також підтримує обробку помилок, які можуть виникнути під час взаємодії з сервером. При виникненні помилки, Angular може відповідно обробити цю помилку та відобразити користувачу повідомлення про помилку або зробити необхідні дії для її виправлення.

У проекті ми використовувася Swagger для документування веб-сервісу API. Swagger - це набір інструментів для створення, документування та використання веб-сервісів API. В основі Swagger лежить специфікація OpenAPI, яка дозволяє описувати структуру та функціональні можливості API. Swagger надає зручний спосіб описати API, включаючи доступні маршрути, параметри, типи даних, можливі відповіді та помилки. Це дозволяє створити чітку та зрозумілу документацію, яка може бути використана для спілкування між розробниками, тестувальниками та іншими зацікавленими сторонами. На рисунку 3.3 зображена документація API серверної частини.

Рисунок 3.3 – Документація API серверної частини

Таким чином, за допомогою HTTP запитів та використанням Angular сервісів та модуля HttpClient, можна забезпечити взаємодію між Angular фронтендом та серверною частиною, реалізованою з використанням ASP.NET Core API.

ASP.NET Core API може взаємодіяти з Angular шляхом надання RESTful API для забезпечення обміну даними між сервером і клієнтом. Використовуючи стандартні HTTP-методи, ASP.NET Core API надає можливість Angular здійснювати запити на сервер для отримання, створення, оновлення та видалення даних.

ASP.NET Core API використовує атрибути маршрутизації для визначення шляхів (URL) до різних ресурсів, доступних на сервері. Ці маршрути дозволяють Angular звертатись до відповідних ендпоінтів API.

Коли Angular відправляє запит на відповідний URL на сервер, ASP.NET Core API реагує на цей запит і виконує відповідну логіку. Це може включати виклик методів контролерів або служб, обробку даних, взаємодію з базою даних тощо.

Після обробки запиту, ASP.NET Core API генерує відповідь, яка надсилається назад до Angular. Відповідь може містити різноманітні дані, такі як JSON-об’єкти, статус коди, заголовки тощо.

Angular отримує відповідь від сервера і може обробити отримані дані, відображати їх на сторінці, здійснювати необхідні дії на основі цих даних або повідомляти користувача про стан операції.

Якщо під час взаємодії виникають помилки, ASP.NET Core API може повертати відповідні статус коди помилок, а Angular може виконати необхідні дії для їх обробки та відображення користувачу.

ASP.NET Core API може забезпечувати механізми аутентифікації та авторизації для захисту ресурсів та забезпечення безпеки взаємодії між Angular та серверною частиною.

Загалом, взаємодія між Angular та ASP.NET Core API здійснюється за допомогою HTTP-запитів, де Angular відправляє запити на сервер, а ASP.NET Core

API обробляє ці запити, виконує необхідні операції і повертає відповіді назад до Angular для подальшої обробки та відображення даних.

Серверна частина, реалізована з використанням ASP.NET Core API, взаємодіє з базою даних за допомогою ORM (Object-Relational Mapping) технологій, таких як Entity Framework Core. Цей підхід дозволяє зв’язати моделі даних, визначені у коді, з таблицями у базі даних і забезпечити зручну взаємодію між ними.

Розробник створює класи моделей даних, які відповідають сутностям у базі даних. Кожна модель може мати властивості, що відображають стовпці таблиць у базі даних. Визначення моделей дозволяє встановити правильність структури даних та відношень між ними.

Розробник створює контекст бази даних, який відповідає підключенню до конкретної бази даних. У контексті вказуються деталі підключення до бази даних, такі як рядок підключення, тип бази даних та інші налаштування, а також зв’язок між моделями даних та таблицями бази даних.

Entity Framework Core дозволяє використовувати міграції для автоматичного створення або оновлення структури бази даних, відповідно до змін у моделях даних. Міграції допомагають зберігати схему бази даних у синхронізованому стані з моделями даних у коді, уникати ручних змін у базі даних та забезпечити міграцію даних без втрати інформації.

За допомогою Entity Framework Core, можна створювати запити до бази даних з використанням LINQ (Language Integrated Query) або методів розширення. Запити можуть включати вибірку, фільтрацію, сортування, групування та інші операції для отримання потрібних даних з бази даних.

Entity Framework Core дозволяє використовувати транзакції для забезпечення консистентності даних та забезпечення атомарності операцій. Також можна використовувати різні стратегії завантаження даних (eager loading, lazy loading) та кешування, щоб забезпечити оптимальну продуктивність та швидкодію взаємодії із базою даних.

Контролери в ASP.NET Core API відповідають за обробку HTTP-запитів та повернення відповідей до клієнтів. Вони є посередниками між маршрутизатором, який визначає, який контролер буде обробляти запит, та моделями даних та послугами, що потрібні для виконання запиту.

Розробник створює класи контролерів, які успадковуються від базового класу ControllerBase. Вони можуть бути розміщені у відповідних папках проекту для організації коду.

Контролери пов’язані з маршрутизацією, яка визначає, який контролер та дія будуть обробляти певний HTTP-запит. Маршрутизація може бути налаштована за допомогою атрибутів на класі контролера або за допомогою маршрутизаційних шаблонів у конфігурації маршрутів. Це наведено в лістингу 3.1.

Лістинг 3.1 – Маршрутизаційний шаблон

```
[Route("api/[controller]/[action]")]
    [ApiController]
    public class AccountsController
```

кінець лістингу 3.1

Контролер може містити різні дії (actions), які відповідають різним HTTP-запитам, таким як GET, POST, PUT, DELETE і т.д. Кожна дія є методом контролера, який обробляє запит, виконує необхідні операції і повертає результат, який може бути відправлений клієнту. Наприклад, можна використовувати атрибут [Route] на класі контролера, щоб вказати базовий шлях до контролера, а потім застосовувати атрибути [HttpGet], [HttpPost], [HttpPut], [HttpDelete] тощо на методах дій для визначення конкретних маршрутів та HTTP-методів. Дії контролера можуть мати параметри, які отримують дані від клієнта. Це можуть бути параметри запиту (query parameters), параметри маршруту (route parameters), параметри тіла запиту (request body) та інші. ASP.NET Core API автоматично зв’язує значення параметрів з

даними, отриманими від клієнта. Контролер може повертати різні типи відповідей до клієнта. Це може бути простий текст, HTML-сторінка, JSON-об’єкт або навіть файл. Для повернення відповіді можна використовувати різні класи результатів, такі як OkResult, BadRequestResult, JsonResult, FileResult тощо, які допомагають зручно повертати різні типи відповідей у відповідності до стандартів HTTP. Приклад контролера наведений в Додатку А в лістингу А.1.

Це загальна схема роботи контролерів у взаємодії з Angular. Контролери обробляють HTTP-запити, виконують необхідні операції і повертають відповіді, які можуть бути оброблені фронтендом у Angular для подальшого відображення або обробки даних.

Для передачі даних між різними частинами системи використовується патерн DTO (Data Transfer Object). DTO є об’єктом, який містить дані, необхідні для передачі, і надає спосіб упакування та розпакування цих даних. В лістингу 3.2 наведено DTO для передачі інформації про коментар під постом.

Лістинг 3.2 – Об’єкт для передачі даних між клієнтською і серверною частиною

```
public class CommentDTO {
        public string? ID { get; set; }
        public string? Text { get; set; }
        public DateTime? Date { get; set; }
        public ProfileDTO? Author { get; set; }
    }
```

кінець лістингу 3.2

Для розробки системи авторизації і аутентифікації був використаний стандарт JWT. Він складається з трьох основних компонентів: заголовку (header), навантаження (payload) і підпису (signature).

Заголовок (header) містить два основні поля: «type» (тип токена, зазвичай встановлений як «JWT») і «alg» (алгоритм шифрування, який використовується для підпису токена, наприклад, HMAC SHA256 або RSA).

Навантаження (payload) містить корисну інформацію, яка передається між сторонами. Це може бути інформація про користувача, його ролі, додаткові дані тощо. Навантаження також може містити стандартні заявки (наприклад, «sub» – суб'єкт, «iss» – видавець, «exp» – термін дії) або власні заявки, які визначені користувачем. Навантаження зображене на рисунку 3.4.

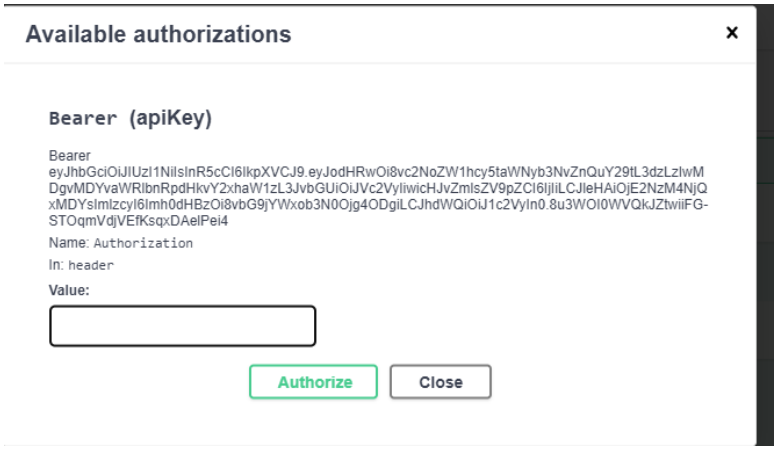

Рисунок 3.4 – Навантаження (payload)

Підпис (signature) використовується для перевірки автентичності токена. Він генерується на основі заголовка, навантаження і секретного ключа, який відомий тільки серверу. Підпис дозволяє перевірити, чи були дані в токені змінені після його створення.

Було встановлено та налаштовано необхідні засоби розробки, включаючи Node.js, Angular CLI та Visual Studio або інше IDE для розробки .NET проекту.

За допомогою Angular CLI було створено новий проект Angular, який включає необхідні компоненти, модулі та сервіси для розробки інтерфейсу.

Були створені компоненти, які представляють окремі частини інтерфейсу, такі як заголовок, меню, форми, списки тощо. Кожен компонент включає шаблон

HTML для відображення та логіку TypeScript для обробки подій та взаємодії з бекендом. На рисунку 3.5 зображено сторінку профілю.

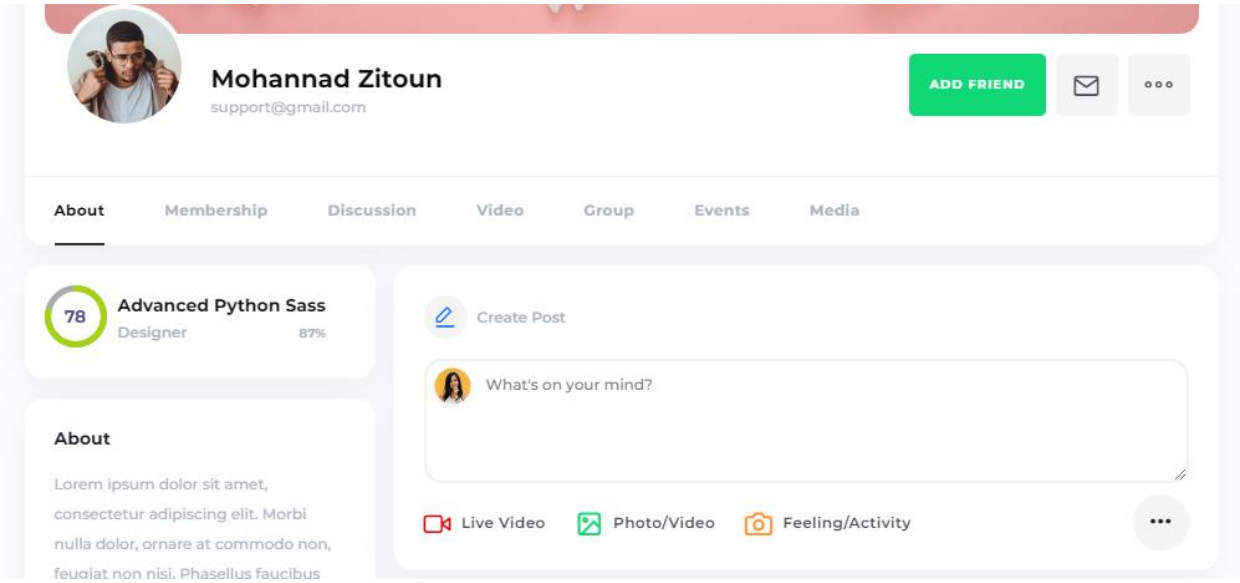

Рисунок 3.5 – Сторінка профілю

Було налаштовано маршрутизацію в Angular для можливості навігації між різними компонентами нашого інтерфейсу. Для цього було використано сервіс навігації, який наведено в лістингу 3.3.

Лістинг 3.3 – Сервіс навігації

```
public toHome()
    {
        this.router.navigate(['/']);
    }
    public toLogin()
    {
        this.router.navigate(['/accounts/login']);
    }
    public toRegistration()
    {
        this.router.navigate(['/accounts/registration']);
    }
    public toNotFound()
    {
        this.router.navigate(['/404']);
    }
```

Продовження лістингу 3.3

```
    public toPost(postId: number)
        {
            this.router.navigate(['/posts'], { queryParams: { id: postId } });
        }
    }
```

кінець лістингу 3.3

За допомогою HTTP сервісів Angular було забезпечено взаємодію з бекендом на основі ASP.NET Core API. Було створено сервіси, які взаємодіють з API для отримання та надсилання даних. В Додатку А в лістингу А.2 наведено приклад сервісу для передачі інформації для входу чи реєстрації.

За допомогою CSS і можливо Sass або іншого CSS-препроцесора, були стилізовані компоненти та елементи інтерфейсу для досягнення відповідного зовнішнього вигляду. На рисунку 3.6 зображена стилізація компонента посту.

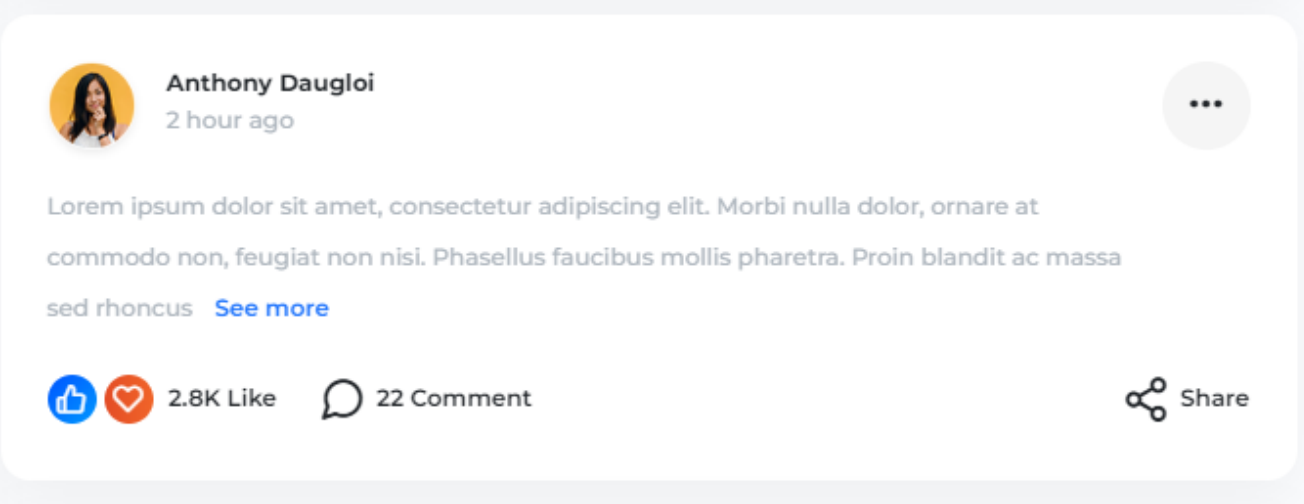

Рисунок 3.6 – Стилізація посту

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування та налагодження інформаційно-комп’ютерної системи є важливою частиною розробки проекту. Цей процес допомагає виявити та виправити помилки, переконатися в правильності роботи системи і забезпечити якість продукту перед його впровадженням.

Були обрані статегії тестування функціональності та продуктивності. Їхньою метою була перевірка правильності реалізації всіх функцій соціальної мережі і їх взаємодію з користувачем, виявлення помилок, недоліків та некоректної роботи системи. Також це включало оцінку продуктивності системи під різними навантаженнями. Для тестування використовувався фреймворк для написання тестів Jasmin і Karma – інструмент для запуску тестів. Інтерфейс Karma зображено на рисунку 3.7.

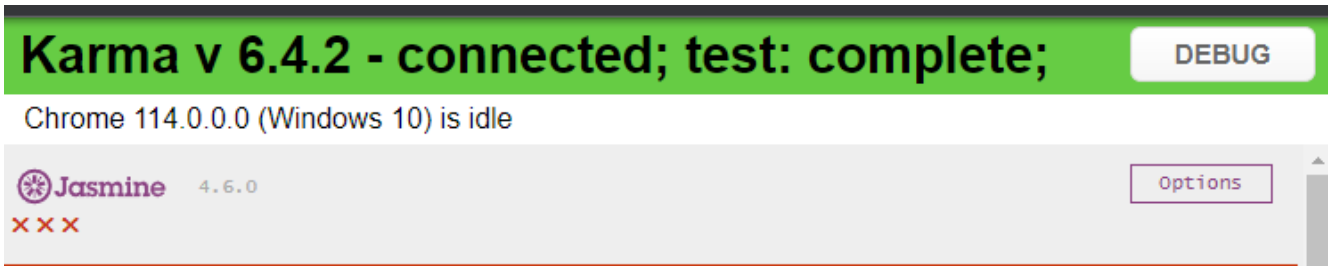

Рисунок 3.7 – Інтерфейс Karma

Під час функціонального тестування була здійснена перевірка кожного функціонального модуля, включаючи реєстрацію користувачів, створення та редагування профілю, завантаження, перегляд та оцінювання мультимедіа-контенту, пошук та взаємодію з іншими користувачами.

Під час тестування безпеки було первірено безпеку вразливість системи, включаючи аутентифікацію, авторизацію, захист від SQL-ін’єкцій, перехоплення даних та інших загроз безпеки.

Під час тестування продуктивності вимірювався час відгуку системи, при різному навантаженні, перевірка реакції системи на великий обсяг даних та багатокористувацьку взаємодію.

3.3 Захист інформаційно-комп’ютерної системи

Забезпечення захисту нашої соціальної мережі було досягнуто шляхом впровадження різноманітних заходів безпеки.

Використання криптографічних протоколів для захисту комунікацій між клієнтами і сервером. Це дозволяло шифрувати передані дані та забезпечувати конфіденційність і цілісність інформації.

Реалізація механізмів аутентифікації і авторизації користувачів. Користувачі мусили пройти процедуру аутентифікації, використовуючи унікальний логін і пароль, щоб мати доступ до своїх облікових записів. Авторизація використовувала різні рівні доступу для обмеження прав користувачів на виконання певних операцій.

Застосування механізмів захисту від атак, таких як SQL-ін’єкції, перехоплення сесій, хибні запити тощо. Це включало валідацію введених даних, використання параметризованих запитів до бази даних, застосування механізмів перевірки достовірності тощо.

Резервне копіювання та відновлення даних. Була встановлена система регулярного резервного копіювання даних, щоб у разі втрати даних або виникнення системних збоїв їх можна було відновити.

Моніторинг системи та виявлення підозрілих активностей. Застосовувалися механізми моніторингу системи для виявлення незвичних або підозрілих дій, що можуть свідчити про можливі атаки або вторгнення.

Навчання та підвищення свідомості користувачів. Користувачам надавалися рекомендації та навчальні матеріали щодо безпечного використання соціальної мережі, виявлення шахрайства та захисту особистої інформації.

3.5 SEO інформаційно-комп’ютерної системи

У проекті була проведена ретельна оптимізація для покращення SEO показників і підвищення видимості нашої інформаційно-комп’ютерної системи у пошукових системах.

Були проведені дослідження, щоб визначити набір ключових слів, які найкраще відображають тематику та функціонал нашої інформаційної системи. Ці ключові слова були використані у заголовках сторінок, мета-тегах, URL-адресах, описах сторінок та контенті.

Був створений унікальний та цікавий контент для нашої інформаційної системи. Вміст мав високу якість, був релевантний та містив ключові слова. Були написані описи, блоги, статті та інструкції, які привертали увагу користувачів та пошукових систем.

URL-адреси сторінок нашої інформаційної системи були оптимізовані для кращої розпізнаваності та рейтингування. Вони були структуровані, зрозумілі та містили ключові слова, уникавши зайвих параметрів.

Були встановлені відповідні мета-теги для опису сторінок. Кожна сторінка мала унікальний мета-тег, що містив ключові слова та короткий опис контенту. Ці мета-теги допомагали залучити увагу користувачів у пошукових системах.

Була розроблена стратегія отримання зовнішніх посилань на нашу інформаційну систему. Ми співпрацювали з релевантними веб-сайтами та блогерами, щоб отримати якісні посилання на нашу систему. Це допомагало підвищити авторитет нашої інформаційної системи в очах пошукових систем.

Була проведена інтеграція зі соціальними медіа, щоб сприяти поширенню нашого вмісту та збільшити його видимість. Користувачі могли легко ділитися сторінками та контентом нашої інформаційної системи через соціальні мережі, що сприяло збільшенню трафіку та популярності.

Ми використовували різні інструменти аналітики, такі як Google Analytics, для відстеження трафіку, поведінки користувачів, показників конверсії та інших метрик. Це надавало нам змогу аналізувати дані та вносити зміни у стратегію оптимізації, щоб досягти кращих результатів.

Ці оптимізаційні заходи сприяють поліпшенню SEO нашої інформаційно-комп'ютерної системи, забезпечуючи більшу видимість, зростання трафіку та покращення рейтингу в пошукових системах. Вони допомагають залучити більше цільової аудиторії та підвищити успішність нашого проекту.

3.6 Економічне обгрунтування розробки

Витрати на розробку: 50 000 грн

Очікувані доходи за перший рік: 80 000 грн

Річний зріст доходів: 5%

Витрати на підтримку та обслуговування на рік: 10 000 грн

Період окупності: 2 роки

Ставка дисконту: 10%

Ризиковий коефіцієнт: 0.9

За допомогою цих даних ми можемо розрахувати деякі показники економічної ефективності проекту:

Чистий приведений дохід (ЧПД):

Розрахуємо чистий дохід (ЧД) за кожен рік:

Рік 1: 80 000 грн - 10 000 грн = 70 000 грн

Рік 2: 80 000 грн * 1.05 - 10 000 грн = 83 000 грн

Розрахуємо чистий приведений дохід (ЧПД) за допомогою формули:

ЧПД = (ЧД1 / (1 + ставка дисконту)^1) + (ЧД2 / (1 + ставка дисконту)^2)

ЧПД = (70 000 грн / (1 + 0.1)^1) + (83 000 грн / (1 + 0.1)^2) = 139 091.74 грн

Внутрішня норма повернення (ВНП):

Розрахуємо ВНП за допомогою методу інтерполяції:

ВНП = нижня ставка дисконту + (різниця ставок дисконту * (ВНП нижня ставка / (ВНП нижня ставка – ВНП верхня ставка)))

ВНП = 0.1 + ((0.05 - 0.1) * (139091.74 грн / (139091.74 грн - 120000 грн))) = 0.078

ВНП дорівнює 7.8%

Період окупності:

Розрахуємо період окупності, який дорівнює кількості років, необхідних для повернення витрат:

Період окупності = Витрати на розробку / Чистий дохід за рік

Період окупності = 50 000 грн / 70 000 грн = 0.71 років (близько 8 місяців)

Ці показники допомагають нам зрозуміти фінансову вигоду та рентабельність проекту. Високе значення ЧПД та ВНП показує, що проект є економічно вигідним, а короткий період окупності свідчить про швидкий темп повернення витрат.

Висновки до розділу 3

В рамках третього розділу було описано розробку прототипу рейтингової соціальної мережі для обміну мультимедіа. Прототип є початковою версією системи, яка демонструє основні функціональні можливості та інтерфейс користувача. Це важливий крок у відпрацюванні концепції та перевірці працездатності системи перед її повноцінною розробкою.

Прототип містить такі основні функції соціальної мережі, як створення профілів користувачів, завантаження та обмін мультимедійним контентом, рейтингування та коментування вмісту. Ці функції дозволяють користувачам взаємодіяти, спілкуватись та оцінювати контент, що створюється у мережі.

Розроблений прототип становить основу для подальшого розвитку та розширення рейтингової соціальної мережі. На основі результатів тестування та отриманого фідбеку можна внести необхідні зміни та додати нові функції, що покращать досвід користувачів та зроблять систему більш привабливою.

ВИСНОВКИ

В процесі аналізу предметної області було проведено детальне вивчення характеристик і вимог, пов’язаних зі створенням інформаційної системи. Результатом аналізу було визначено основні проблеми, які виникають в даній області, а також потреби та очікування користувачів.

На основі проведеного аналізу були поставлені завдання на кваліфікаційну роботу, які відповідали вимогам та потребам предметної області.

Під час визначення вимог та потреб користувачів соціальних мереж, що пов’язані з обміном мультимедіа, було з’ясовано, що користувачі хочуть мати зручний та ефективний спосіб обміну різними типами мультимедіа, такими як фотографії, відео та аудіо. Користувачі очікують швидкого та зручного процесу завантаження файлів різного розміру без втрати якості. Крім того, вимоги до безпеки також важливі, оскільки користувачі хочуть бути впевнені, що їхні мультимедійні файли захищені від несанкціонованого доступу. Також було встановлено, що повинна бути можливість керувати приватністю своїх мультимедійних файлів, встановлюючи рівні доступу для різних груп користувачів або обмежуючи доступ лише для обраних осіб.

Був проведений аналіз конкурентного середовища та ринку соціальних мереж, зокрема мереж, які спеціалізуються на обміні мультимедіа. Було встановлено, що деякі конкуренти вже займають сильні позиції на ринку та мають велику активну користувацьку базу. Це може створювати виклики для нових учасників, оскільки потрібно знайти спосіб виділитись та привернути увагу користувачів. Тому важливими факторами успіху в цьому сегменті є інтуїтивний та зручний інтерфейс користувача, якість обробки та відображення мультимедійного контенту, а також наявність унікальних функцій і можливостей, які ще не були реалізовані в уже існуючих програмних продуктах конкурентів.

В процесі визначення технічних можливостей розробки мережі і вибору програмного забезпечення було розглянуто різні інструменти розробки, такі як редактори коду, інтегровані середовища розробки (IDE) та системи контролю версій, які допоможуть ефективно керувати розробкою. Були використанні такі технології та інструменти:

– ASP.NET Core – для розробки серверної частини;
– Angular – для створення динамічного клієнтського інтерфейсу;
– SQL – мова запитів, яка використовувалася для роботи з базою даних;
– Visual Studio – основне інтегроване середовище розробки (IDE);
– SQL Server Management Studio – для управління та взаємодії з базою даних SQL;
– GitHub – платформа для спільної розробки та управління версіями коду.

Було спроектовано інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко орієнтуватись у системі та знаходити необхідні функції та розділи. Використано сучасний та привабливий дизайн, з використанням зручних шрифтів, кольорів та графічних елементів, що створюють приємне візуальне враження та відповідають сучасним трендам. Забезпечено зручну навігацію, яка дозволяє користувачам швидко переміщатись між різними розділами, профілем, повідомленнями та іншими функціями платформи.

Було розроблено алгоритми, які оцінюють та ранжують контент, щоб забезпечити його відображення користувачам на основі їхніх інтересів, популярності та інших факторів. Також були реалізовані алгоритми фільтрації, які виявляють і відсіюють небажаний або неприйнятний контент, такий як спам, образливі повідомлення або небажаний матеріал.

В рамках розробки тестових завдань для оцінки роботи мережі було визначено набір сценаріїв. Ці сценарії включали створення та редагування профілів, взаємодію з дописами та коментарями, пошук та фільтрацію контенту. Для виконання цих сценаріїв було створено реалістичні тестові набори даних, які

включали ізноманітні профілі користувачів, дописи, зображення та інші елементи контенту.

Оцінюючи роботу мережі, було виявлено потенційні проблеми, такі як затримки завантаження, нестабільність, складність використання певних функцій. Тому було розроблено рекомендації щодо вдосконалення та розвитку мережі, які включають у себе розширення функціональності, вдосконалення інтерфейсу, оптимізацію продуктивності та безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bhajaria N. Data Privacy: A runbook for engineers. Shelter Island: Manning Publications. 2022. 384 p.

2. Zammetti F. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. New York: Apress. 2020. 395 p.

3. De Sanctis V. Building Web APIs with ASP.NET Core. Shelter Island: Manning Publications. 2023. 472 p.

4. Jackson L. The Complete ASP.NET Core 3 API Tutorial: Hands-On Building, Testing, and Deploying. New York: Apress. 2020. 484 p.

5. Helland A., Durano V., Chilberto J., Price E. ASP.NET Core 5 for Beginners: Kick-start your ASP.NET web development journey with the help of step-by-step tutorials and examples. Birmingham: Packt Publishing. 2020. 602 p.

6. Fain Y., Moiseev A. Angular Development with TypeScript. Shelter Island: Manning. 2018. 560 p.

7. Wilken J. Angular in Action. Shelter Island: Manning. 2018. 320 p.

8. Seshadri S. Angular: Up and Running: Learning Angular, Step by Step. Sebastopol: O’Reilly Media. 2018. 309 p.

9. DeBarros A. Practical SQL, 2nd Edition: A Beginner's Guide to Storytelling with Data. San Francisco: No Starch Press. 2022. 464 p.

10. Blokdyk G. Microsoft Sql Server Management Studio A Clear and Concise Reference. Toronto: 5STARCooks. 2021. 312 p.

11. Smith P. J. Entity Framework Core in Action, Second Edition. Shelter Island: Manning Publications. 2021. 624 p.

ДОДАТКИ

Додаток А

Лістинг А.1 – контролер для сторінки профілю

```
namespace Amica.API.WebServer.Controllers {
    [Route("api/[controller]")]
    [ApiController]
    public class ProfilesController : ControllerBase {
        private readonly IProfilesRepository profiles;
        private readonly ILogger<ProfilesController> logger;

        public ProfilesController(IProfilesRepository profiles,
ILogger<ProfilesController> logger) {
            this.profiles = profiles;
            this.logger = logger;
        }

        #region Controller Services
        private long? GetProfileByJwt(IEnumerable<Claim> claims)
{
            var raw = claims.SingleOrDefault(c => {
                return c.Type == JwtClaims.TokenIdentity;
            })?.Value;

            if ( raw is null )
                return null;

            var userId = long.Parse(raw);
            return userId;
        }
        #endregion

        /// <summary>
        /// Get profile by profile_nickname
        /// </summary>
        /// <param name="profile_nickname"></param>
        /// <response code="200">Success</response>
        /// <response code="404">Profile not found</response>
        ///  <response  code="500">Any  server  problem.  Until
release returns Exception text</response>
        [HttpGet]
        [Route("nickname/{profile_nickname}")]
        public            async            Task<IActionResult>
GetProfileByNickName([FromRoute] string profile_nickname) {
            try {
                // get profile id from JWT token
                var userId = GetProfileByJwt(User.Claims);

                var           res           =           await
profiles.GetProfileByNickNameAsync(profile_nickname, userId ?? -1);
```

```
                    if ( res is null )
                        return NotFound();
                    return Ok(res);
                }
                catch ( Exception e ) {
                    logger.LogError(e, "Unexpected server error");
                    return Problem(e.Message, null, 500);
                }
            }

            /// <summary>
            /// Match profiles by part
            /// </summary>
            /// <param name="part"></param>
            /// <response code="200">Success</response>
            /// <response code="404">Profiles not found</response>
            ///  <response code="500">Any server problem. Until
release returns Exception text</response>
            [HttpGet]
            [Route("find/{part}")]
            public async Task<IActionResult>
FindListByNickNameAsync([FromRoute] string part) {
                try {
                    // get profile id from JWT token
                    //var userId = GetProfileByJwt(User.Claims);

                    var res = await
profiles.MatchProfilesByNickNameAsync(part);
                    if ( res is null )
                        return NotFound();
                    return Ok(res);
                }
                catch ( Exception e ) {
                    logger.LogError(e, "Unexpected server error");
                    return Problem(e.Message, null, 500);
                }
            }

            /// <summary>
            /// Get profile by id
            /// </summary>
            /// <param name="id"></param>
            /// <response code="200">Success</response>
            /// <response code="404">Profile not found</response>
            ///  <response code="500">Any server problem. Until
release returns Exception text</response>
            [HttpGet]
            [Route("{id}")]
            public async Task<IActionResult>
GetProfileById([FromRoute] long id) {
```

```
                try {
                    // get profile id from JWT token
                    var userId = GetProfileByJwt(User.Claims);

                    var res = await profiles.GetProfileByIdAsync(id,
userId ?? -1);
                    if ( res is null )
                        return NotFound();
                    return Ok(res);
                }
                catch ( Exception e ) {
                    logger.LogError(e, "Unexpected server error");
                    return Problem(e.Message, null, 500);
                }
            }
        }
                                    }
```

кінець лістингу А.1

Лістинг А.2 – Сервіс акаунту

```
    export class AccountsService {
        private client: AccountsApiClient;
        private cookie: CookieService;
        constructor(
            @Inject(AccountsApiClient) client: AccountsApiClient,
            @Inject(CookieService) cookie: CookieService,
            private router: Router,
            private navigate: NavigationService
        ) {
            this.client = client;
            this.cookie = cookie;
        }
        public getToken(): string | undefined {
            const token = this.cookie.getCurrentAuth()?.token;
            return token;
        }
        public getCurrentProfile(): Profile | undefined {
            const profile = this.cookie.getCurrentAuth()?.profile;
            return profile;
        }
        public isSignedIn(): boolean {
            const token = this.cookie.getCurrentAuth()?.token;
            return token != undefined && token != null;
        }
        public signOut() {
            this.cookie.deleteCurrentAuth();
            this.navigate.toLogin();
        }
```

```
        public async signInAsync(req: SignInRequestDTO):
Promise<SignInResponceDTO> {
            var res = await this.client.signInAsync(req.login,
req.password);
            if (res.isSignedIn == true) {
                this.cookie.setCurrentAuth(res);
                this.navigate.toHome();
            }
            return res;
        }
        public async signUpAsync(
            req: SignUpRequestDTO,
            needToSignIn: boolean = false
        ): Promise<SignUpResponceDTO> {
            var res = await this.client.signUpAsync(req);
            if (res.isSignedUp == true) {
                if (needToSignIn) {
                    var authRes = await this.signInAsync({
                        login: req.nickname,
                        password: req.password,
                    });
                }
            }
            return res;
        }
    }
```

кінець лістингу А.2