Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Веб-додаток для планування процедур користувача по догляду за собою і збереження даних про них, тематичного спілкування «Щоденник краси»
(тема)

Виконала:

студент 4 курсу, групи ПЗПІ-20-7

Круть А.Д.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Кафедри ПІ Кравець Н.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 – Інженерія програмного забезпечення

(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«___»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Круть Алісі Дмитрівні

(прізвище, ім’я, по батькові)

1. Тема роботи: Веб-додаток для планування процедур користувача по догляду за собою і збереження даних про них, тематичного спілкування «Щоденник краси» затверджена наказом університету від 20.05.2024 р. № 471Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 червня 2024 р.

3. Вихідні дані до роботи: *Веб-додаток має полегшити управління косметичними процедурами, зустрічами та взаємодією в рамках форуму спільноти. Вона має бути розроблена з використанням сучасних технологій, включаючи React.js, Node.js, Express.js та MongoDB. Додаток має на меті надати користувачам зручну та інтуїтивно зрозумілу платформу для документування своїх процедур краси, планування зустрічей та участі в дискусіях на форумі.*

4. Перелік питань, що потрібно опрацювати у роботі: *Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.04 – 17.04 2024	*виконано*
2	Створення специфікації програмного забезпечення	18.04 – 20.04 2024	*виконано*
3	Проєктування програмного забезпечення	20.04 – 25.04 2024	*виконано*
4	Розробка програмного забезпечення	25.04 – 21.05 2024	*виконано*
5	Тестування програмного забезпечення	21.05 – 23.05 2024	*виконано*
6	Оформлення пояснювальної записки	23.05 – 27.05 2024	*виконано*
7	Підготовка доповіді та презентації	28.05 – 31.05 2024	*виконано*
8	Перевірка на нормоконтроль	14.06.2024	*виконано*
9	Попередній захист	17.06.2024	*виконано*
10	Здача роботи у електронний архів	14.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	17.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024 р.

Студент _____ Круть А.Д.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. кафедри ПІ Кравець Н.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 85 стор., 54 рис., 28 джерел.

ВЕБ-ДОДАТОК, ЗУСТРІЧІ, КАЛЕНДАР, КОСМЕТИЧНІ ПРОЦЕДУРИ, ФОРУМ, ЩОДЕННИК КРАСИ, BACK-END, CSS, FRONT-END, HTML, JAVASCRIPT, MONGODB, NODEJS, REACTJS.

Об’єкт розробки – веб-додаток "Щоденник краси", який покликаний допомогти користувачам в управлінні своїми косметичними процедурами, зустрічами та взаємодією в рамках форуму спільноти.

Мета розробки – створити зручну платформу, яка дозволить людям документувати свої косметичні процедури, планувати зустрічі, брати участь у дискусіях на форумі та налаштовувати свої профілі користувачів. Додаток має на меті впорядкувати процеси управління красою та сприяти формуванню почуття спільноти серед користувачів.

Метод рішення – середовище розробки Visual Studio Code, JavaScript-бібліотека для створення фронт-енду ReactJS, мова гіпертекстової розмітки документів HTML, формальна мова опису зовнішнього вигляду документа CSS, для розробки бек-енду програмна платформа Node.js, мова програмування JavaScript, веб-фреймворк Express та СКБД MongoDB.

У результаті розробки створено веб-додаток "Щоденник краси", який має стильний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам записувати свої косметичні процедури, керувати зустрічами за допомогою інтерактивного календаря, брати участь у дискусіях на форумі та персоналізувати свої профілі користувачів. Додаток підтримує безперешкодну інтеграцію між записами щоденника, подіями календаря, повідомленнями на форумі та профілями користувачів, надаючи користувачам комплексну платформу для управління їхньою діяльністю, пов’язаною з красою.

WEB APPLICATION, APPOINTMENTS, CALENDAR, COSMETIC PROCEDURES, FORUM, BEAUTY DIARY, BACK-END, CSS, FRONT-END, HTML, JAVASCRIPT, MONGODB, NODEJS, REACTJS.

The object of development is the BeautyDiary web application, which is designed to help users manage their beauty treatments, appointments, and interactions within the community forum.

The goal of the development is to create a convenient platform that will allow people to document their beauty treatments, schedule appointments, participate in forum discussions, and customize their user profiles. The application aims to streamline beauty management processes and foster a sense of community among users.

The solution method used is the Visual Studio Code development environment, the ReactJS JavaScript library for creating the front-end, the HTML hypertext markup language for documents, the CSS formal language for describing the appearance of a document, the Node.js software platform for back-end development, the JavaScript programming language, the Express web framework, and the MongoDB database.

As a result of the development, the BeautyDiary web application was created, which has a stylish and intuitive interface that allows users to record their beauty treatments, manage appointments using an interactive calendar, participate in forum discussions, and personalize their user profiles. The application supports seamless integration between diary entries, calendar events, forum posts, and user profiles, providing users with a comprehensive platform for managing their beauty-related activities.

Я, Круть Аліса, студентка гр. ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Веб-додаток "Щоденник краси"», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в

електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У нашому сучасному взаємопов’язаному світі краса, здоров’я та догляд за собою перетворилися з простої розкоші на невід’ємні складові повсякденного життя. Оскільки суспільство все більше усвідомлює важливість самовираження, впевненості в собі та цілісного благополуччя, попит на доступні та персоналізовані б’юті-рішення стрімко зростає. Поява веб-додатків змінила спосіб взаємодії людей з практиками краси, забезпечивши безперешкодний доступ до інформації, ресурсів та послуг, пристосованих до індивідуальних потреб та вподобань.

Актуальність "Щоденника краси" підкреслюється зміною парадигм в індустрії краси та ширшими суспільними тенденціями. В епоху, що характеризується безпрецедентним цифровим зв’язком і підвищеним акцентом на індивідуальності, споживачі шукають б’юті-рішення, які виходять за рамки традиційних стандартів і охоплюють різноманітність у всіх її формах. "Щоденник краси" задовольняє цей попит, надаючи платформу, яка відзначає унікальність, сприяє інклюзивності та дає можливість користувачам визначати та приймати власні стандарти краси.

Оскільки соціальні медіа-платформи надають дуже багато б’юті-трендів, порад і продуктів, вибагливі користувачі прагнуть мати цілісну платформу, яка не лише дозволяє їм документувати свої процедури, а й сприяє залученню спільноти, обміну знаннями та експертним порадам. "Щоденник краси" заповнює цю прогалину, об’єднуючи функціональність щоденника, календаря та форуму в єдину цифрову екосистему, що дозволяє користувачам розпочати трансформаційну подорож до самовираження та догляду за собою.

Крім того, після таких глобальних подій, як пандемія COVID-19, що змінили поведінку та вподобання споживачів, значення онлайн-платформ для діяльності, пов’язаної з красою, лише посилилося. В умовах обмежень на особисте спілкування та зростаючої залежності від цифрових каналів, "Щоденник краси" стає важливим ресурсом для людей, які прагнуть орієнтуватися в складнощах догляду за красою, не виходячи з дому.

Метою роботи є надання людям комплексної та зручної платформи, яка

призначена для документування, організації та вдосконалення їхніх ритуалів краси. Завдяки інтеграції щоденникових записів, інтерактивного календаря та функцій залучення спільноти, "Щоденник краси" має на меті змінити сприйняття, практику та оцінку краси користувачами. Сприяючи розширенню можливостей та цілісному благополуччю, "Щоденник краси" прагне надихнути користувачів розпочати трансформаційну подорож до самовираження та догляду за собою.

Завданням роботи є розробка веб-додатку "Щоденник краси", який охоплює ряд функціональних можливостей, покликаних полегшити документування та вдосконалення б’юті-рутин. Це передбачає створення зручного інтерфейсу для реєстрації та управління профілем, що дозволить користувачам робити детальні записи в щоденнику, включаючи ритуали краси, режими догляду за шкірою та вподобання щодо макіяжу. Крім того, додаток має містити інтерактивний календар для планування та управління подіями і зустрічами, пов’язаними з красою, а також надійну систему сповіщень, яка інформуватиме користувачів про нові записи в щоденнику та майбутні події. Крім того, додаток має бути оптимізовано для адаптивного дизайну, що забезпечить зручність використання на різних пристроях.

Галузь застосування результатів розробки веб-додатку охоплює широкий спектр індустрії краси, догляду за шкірою та особистої гігієни. «Щоденник краси» призначений для людей різних демографічних груп та способу життя, які шукають персоналізовані рішення та досвід у сфері краси та догляду за собою. Також галузь застосування результатів розробки охоплює широкий спектр секторів у сфері краси, догляду за шкірою та особистої гігієни, пропонуючи різні можливості як для окремих осіб, так і для професіоналів, підприємств та організацій.

Отже, завдяки своїм багатогранним можливостям, дизайну, орієнтованому на користувача та прагненню до розширення можливостей, веб-додаток переосмислює спосіб, у який люди взаємодіють та документують свої подорожі у світі краси. Сприяючи залученню спільноти та доступу до експертних думок, "Щоденник краси" не лише покращує індивідуальний досвід користувачів, але й каталізує позитивні зміни в індустрії краси в цілому.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Практика ведення щоденника здавна вважається потужним інструментом для саморефлексії, особистісного зростання та емоційного благополуччя. У сучасну цифрову епоху поява додатків для ведення щоденників змінила спосіб залучення людей до цієї вічної практики, пропонуючи безпрецедентну зручність, гнучкість та інноваційність. Сфера додатків для ведення щоденників є широкою і різноманітною, надаючи користувачам безліч можливостей для легкого і ефективного документування своїх думок, досвіду та повсякденної діяльності.

Серед помітних претендентів у цій сфері – Day One, Diarium, Grid Diary та Dabble Me. Кожна програма може похвалитися унікальними можливостями, функціоналом і користувацьким досвідом, задовольняючи різноманітні вподобання та потреби у сфері ведення особистого щоденника. Від елегантних інтерфейсів до крос-платформної сумісності, від структурованих форматів до ведення щоденників електронною поштою – ці додатки втілюють еволюцію ведення щоденників у цифрову епоху.

Day One – це програма для ведення щоденників, відома своїм інтерфейсом, потужним набором функцій і бездоганним користувацьким інтерфейсом (див. рис. 1.1). Він дозволяє користувачам легко документувати свої думки, досвід і повсякденну діяльність, пропонуючи при цьому цілий ряд можливостей для налаштування та інтеграції.

Переваги Day One:

– Day One може похвалитися візуально привабливим та інтуїтивно зрозумілим інтерфейсом, що робить ведення щоденника чудовим досвідом;

– користувачі можуть форматувати свої записи за допомогою Markdown, що забезпечує більшу гнучкість та кастомізацію;

– Day One пропонує настроювані нагадування, що дозволяє користувачам підтримувати постійні звички ведення щоденника;

– додаток підтримує синхронізацію між пристроями, гарантуючи, що користувачі можуть отримати доступ до своїх записів з будь-якого місця;

– користувачі можуть додавати фотографії та відео до своїх записів, збагачуючи свій досвід ведення щоденника візуальним контентом;

– Day One пропонує налаштування конфіденційності, що дозволяє користувачам контролювати, хто може отримати доступ до їхніх записів у журналі.

Недоліки Day One:

– деякі основні функції, такі як синхронізація та необмежена кількість фотографій, вимагають підписки на Day One Premium;

– хоча базова версія пропонує основні функції безкоштовно, доступ до преміум-функцій є платним, що може відлякати деяких користувачів;

– інтеграція з іншими програмами та сервісами може бути обмежена порівняно з деякими іншими програмами для ведення журналів.

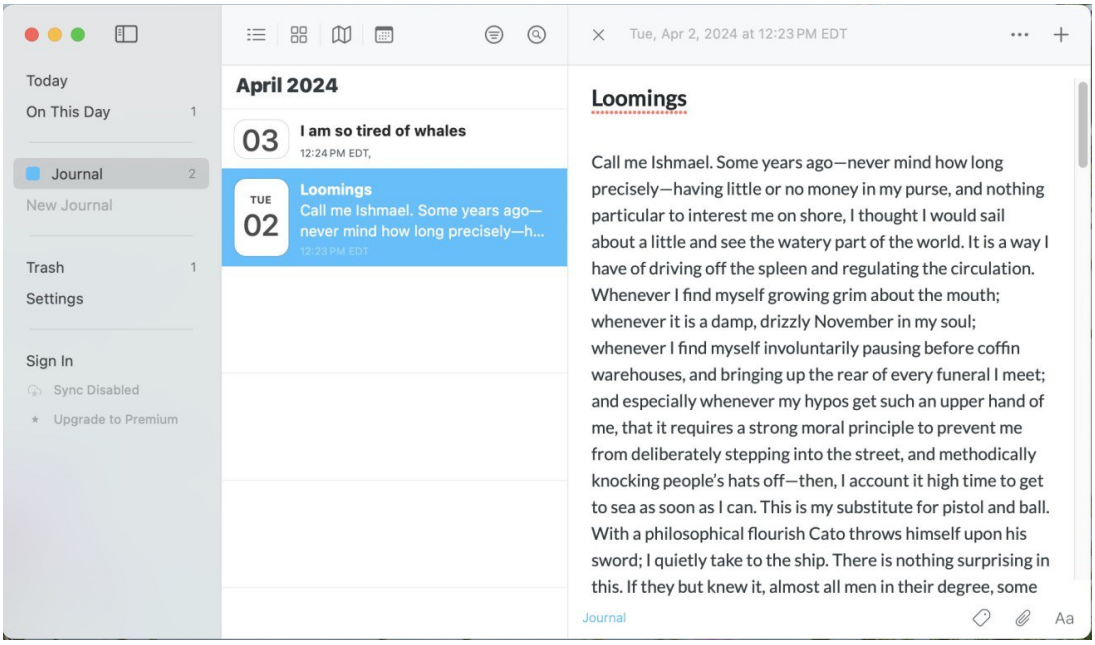

Рисунок 1.1 – Day One [1]

Diarium – це універсальний додаток для ведення журналів, доступний на різних платформах, що пропонує широкі можливості налаштування, підтримку мультимедіа та безперебійну синхронізацію (див. рис. 1.2).

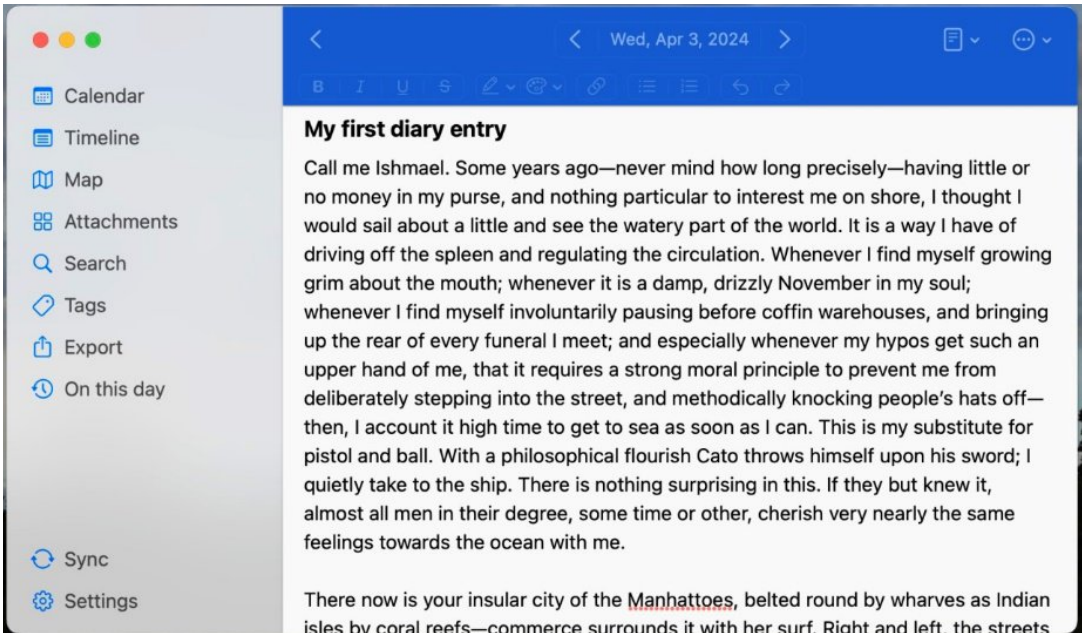

Рисунок 1.2 – Diarium [2]

Переваги Diarium:

– Diarium пропонує нативні додатки для Windows, macOS, iOS та Android, завдяки чому користувачі можуть отримати доступ до своїх записів з будь-якого пристрою;

– додаток легко синхронізується між пристроями за допомогою різних хмарних сервісів, включаючи OneDrive, Google Drive, Dropbox і WebDAV;

– користувачі можуть додавати до своїх записів різні типи мультимедіа, включаючи фотографії, аудіозаписи та файли;

– Diarium дозволяє користувачам налаштовувати нагадування, допомагаючи їм підтримувати регулярні звички ведення щоденника;

– користувачі можуть експортувати свої записи в різні формати, такі як DOCX, HTML, RTF і TXT, забезпечуючи портативність та доступність даних;

– Diarium може автоматично підтягувати стрічки з соціальних мереж, фітнес-додатків, календарів і погодних сервісів, збагачуючи записи в щоденнику контекстною інформацією.

Недоліки Diarium:

– додаток може потребувати певного часу на освоєння для деяких користувачів через широкі можливості кастомізації та функції;

– широкий набір функцій Diarium може виявитися надто складним для користувачів, які шукають простішого способу ведення щоденника.

Grid Diary пропонує структурований підхід до ведення щоденника, спонукаючи користувачів відповідати на повторювані запитання, організовані у сітчастому форматі (див. рис. 1.3).

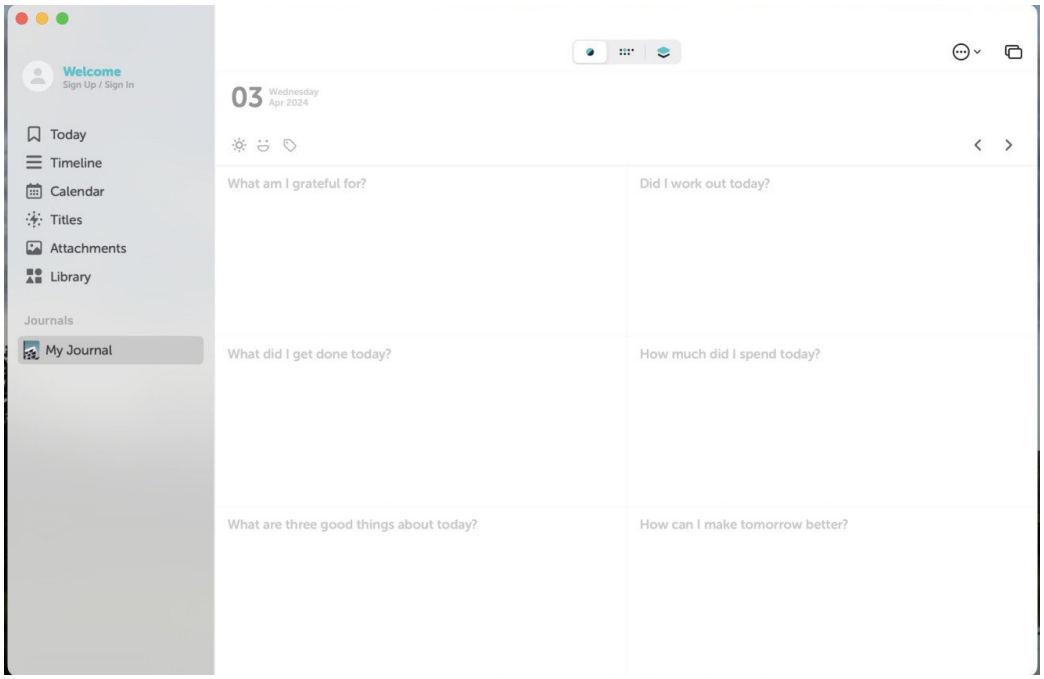

Рисунок 1.3 – Grid Diary [3]

Він має на меті сприяти рефлексії та самоусвідомленню за допомогою підказок і категорій запитань, що налаштовуються.

Переваги Grid Diary:

– сітчастий формат Grid Diary надає користувачам структуровану основу для ведення щоденника, що робить його ідеальним для початківців або тих, хто шукає керівництва;

– користувачі можуть налаштовувати питання в сітці відповідно до своїх уподобань та розмірковувати про різні аспекти свого життя;

– простота і доступність додатку дозволяють користувачам з будь-яким рівнем досвіду легко долучатися до регулярного ведення щоденника;

– зосередженість Grid Diary на рефлексії та самоаналізі заохочує користувачів розвивати уважність та самосвідомість;

– незважаючи на те, що Grid Diary в основному базується на тексті, він також підтримує вкладення, дозволяючи користувачам додавати до своїх записів фотографії, посилання та інші медіафайли.

Недоліки Grid Diary:

– структурований формат Grid Diary може не підходити для користувачів, які віддають перевагу довгим записам у журналі;

– хоча користувачі можуть налаштовувати підказки, Grid Diary може не мати таких широких можливостей налаштування, як деякі інші програми для ведення журналів.

Dabble Me відрізняється від традиційних додатків для ведення щоденників тим, що використовує електронну пошту як основне середовище для ведення щоденників (див. рис. 1.4).

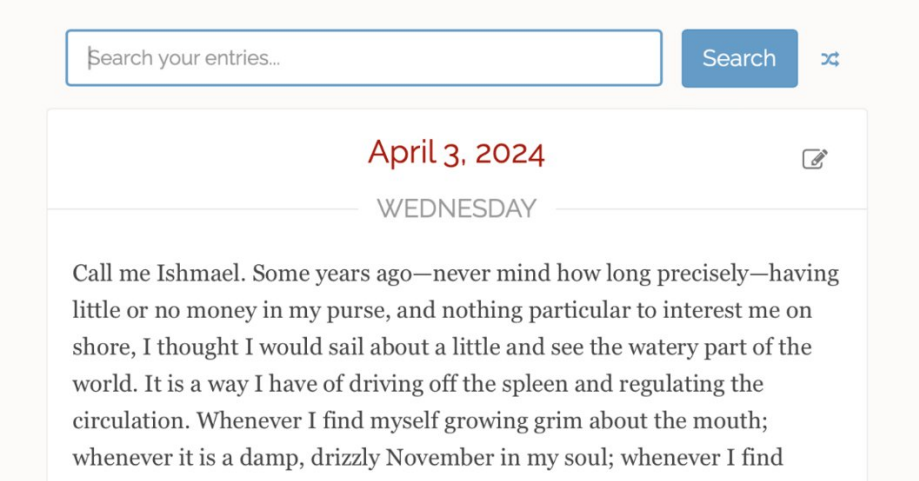

Рисунок 1.4 – Dabble Me [4]

Він пропонує спрощений та легкий підхід до ведення журналів, що робить його доступним для користувачів, які перебувають у дорозі.

Переваги Dabble Me:

– використання електронної пошти для ведення щоденника в Dabble Me усуває необхідність відкривати окремий додаток, що спрощує процес ведення щоденника;

– користувачі можуть вести щоденник, просто відповівши на запит електронною поштою, що усуває необхідність формувати нову звичку або не

забувати відкривати додаток для ведення щоденника;

– Dabble Me надає користувачам повний архів їхніх записів на сайті, пропонуючи функції пошуку, перегляду календаря та експорту;

– користувачі можуть бути впевнені, що їхні записи в журналі захищені, оскільки Dabble Me пропонує зашифровані приватні журнали та опції експорту для перенесення даних;

– додаток може отримувати зовнішні дані з соціальних мереж, фітнес-програм і календарів, збагачуючи записи в журналі контекстною інформацією.

Недоліки Dabble Me:

– у безкоштовній версії Dabble Me відсутні нагадування, що може вплинути на залученість користувачів і послідовність у звичках ведення щоденника;

– хоча підхід на основі електронної пошти спрощує ведення щоденника, деякі користувачі можуть віддати перевагу спеціальному інтерфейсу додатку для більшого занурення в процес;

– доступ до преміум-функцій, таких як щоденні підказки на електронну пошту, вимагає платної підписки, що може відлякати деяких користувачів.

My Diary - Daily Diary Journal – це безкоштовний онлайн-щоденник з функцією блокування, що дозволяє користувачам записувати щоденні записи, таємні думки, подорожі, настрій та особисті моменти (див. рис. 1.5). Він пропонує різні варіанти налаштування, такі як теми, наклейки та шрифти, щоб зробити особисті щоденники яскравими та безпечними.

Переваги My Diary - Daily Diary Journal:

– My Diary - Daily Diary Journal дозволяє користувачам встановлювати пароль до щоденника, що гарантує безпеку їхніх приватних записів. Крім того, підтримка блокування щоденника за відбитками пальців забезпечує швидкий доступ до нього;

– додаток пропонує кілька художніх тем для щоденників, що дозволяє користувачам персоналізувати свої щоденники;

– користувачі можуть синхронізувати свої щоденники з Google Drive або Dropbox для безпечного зберігання та доступу до них з різних пристроїв;

– My Diary - Daily Diary Journal слугує додатком для фотожурналу, дозволяючи користувачам писати щоденникові записи з фотографіями, записами та відео, щоб запам’ятовувати їх надовго;

– завдяки режиму захисту очей користувачі можуть писати, не турбуючись про перенапруження очей, що покращує процес ведення щоденника;

– додаток пропонує відстеження настрою за допомогою смайликів і стікерів, що дозволяє користувачам висловлювати свої почуття та відстежувати зміни настрою з плином часу;

– додавання тегів до записів у щоденнику полегшує їхню організацію та пошук за категоріями.

Недоліки My Diary - Daily Diary Journal:

– деякі функції, такі як нагадування та певні теми, можуть бути обмежені у безкоштовній версії;

– синхронізація з Google Drive або Dropbox вимагає підключення до Інтернету, яке може бути недоступним не завжди;

– преміум-функції можуть вимагати передплати або одноразової покупки.

Рисунок 1.5 – My Diary - Daily Diary Journal [5]

MSKD – Skincare Diary – це комплексний додаток для управління доглядом за шкірою, розроблений для спрощення відстеження б’юті-режимів (див. рис. 1.6).

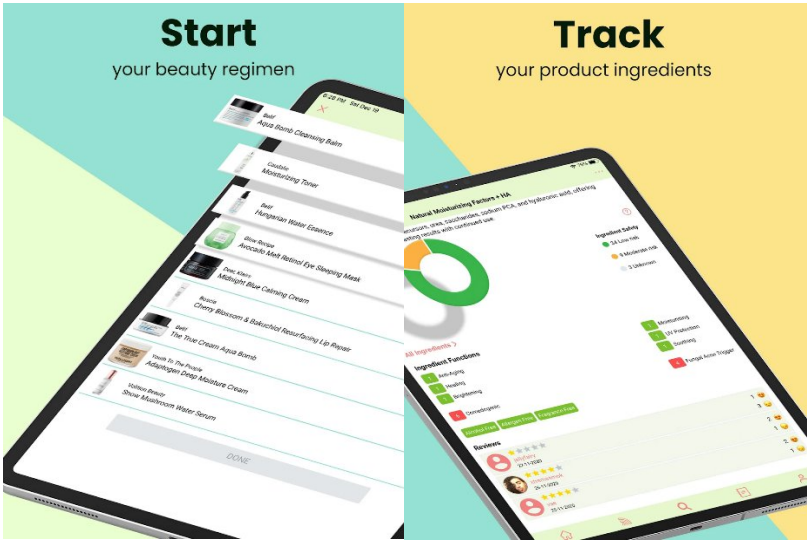

Рисунок 1.6 – MSKD – Skincare Diary [6]

У ньому є менеджер процедур, база даних інгредієнтів продуктів, нагадування, щоденник для обличчя з порівнянням фотографій, відгуки про продукти, трекер заначок і списки бажань.

Переваги MSKD - Skincare Diary:

– користувачі можуть створювати персоналізовані процедури догляду за шкірою, обираючи продукти з величезної бібліотеки брендів, з можливістю сканування штрих-кодів для швидкого додавання;

– додаток надає інформацію про інгредієнти для догляду за шкірою, включаючи потенційну токсичність, щоб допомогти користувачам зробити усвідомлений вибір косметичних засобів;

– MSKD пропонує настроювані нагадування для процедур догляду за шкірою та таймери для нанесення масок і повторного нанесення сонцезахисного крему;

– користувачі можуть робити та порівнювати фотографії свого обличчя, щоб відстежувати прогрес у догляді за шкірою та виявляти будь-які зміни;

– MSKD має унікальний шаблон для детальних оглядів продуктів, що допомагає користувачам зрозуміти ефективність та придатність продукту;

– додаток допомагає користувачам організувати свої засоби по догляду за шкірою, відстежувати терміни придатності та створювати списки бажань для майбутніх покупок.

Недоліки MSKD - Skincare Diary:

– хоча додаток містить величезну бібліотеку засобів для догляду за шкірою, деякі нішеві або міжнародні бренди можуть бути відсутніми, і для їх додавання потрібно подавати заявки від користувачів;

– користувачам може знадобитися деякий час для ознайомлення з усіма можливостями та функціями додатку, особливо якщо вони є новачками у відстеженні догляду за шкірою;

– точність та повнота бази даних продуктів залежать від даних користувачів, що може призвести до прогалин у покритті певних продуктів.

Загалом, кожна програма для ведення щоденників пропонує унікальний набір можливостей, функцій та користувацького досвіду, що задовольняє різноманітні вподобання та потреби у сфері особистої рефлексії та самовираження. Незалежно від того, чи шукають вони мінімалістичний інтерфейс, потужний набір функцій, структурований підхід або ведення щоденника електронною поштою, серед широкого спектру цифрових додатків для ведення щоденників знайдеться відповідне рішення.

Отже, ринок додатків для ведення журналів характеризується різноманітністю, інноваціями та орієнтованим на користувача дизайном. Незалежно від того, чи шукають вони мінімалістичний інтерфейс, потужний набір функцій або спрощений процес ведення щоденника, користувачі можуть знайти відповідне рішення серед безлічі доступних варіантів. Коли користувачі орієнтуються в цифровому світі ведення щоденників, вони отримують можливість розвивати уважність, самоаналіз і глибший зв’язок зі своїм внутрішнім «я» завдяки інтуїтивно зрозумілим інструментам і функціям, які надають ці інноваційні додатки.

1.2 Виявлення та вирішення проблем

Виявлення та вирішення проблем у сфері щоденників краси створює певні труднощі, насамперед через обмежену доступність всеохоплюючих рішень, розроблених спеціально для цієї ніші. Дефіцит спеціалізованих платформ, що

задовольняють тонкі вимоги та бажання шанувальників краси, підкреслює потребу у веб-додатку, такому як «Щоденник краси», який би заповнив ці прогалини та надав комплексне рішення. Уважно вивчивши конкретні больові точки та недоліки, що переважають у нинішньому середовищі, нижче перелічені ключові проблеми, з якими стикаються користувачі, а також наведено, як «Щоденник краси» їх вирішить.

Однією з головних проблем, з якою стикаються прихильники краси, є дефіцит спеціалізованих платформ, орієнтованих виключно на косметичні процедури, догляд за шкірою, волоссям та макіяжем. Хоча загальні додатки для ведення щоденників існують, їм часто бракує спеціалізованих функцій та можливостей, необхідних для всебічного документування та відстеження б’юті-процедур. Як наслідок, користувачам доводиться комбінувати розрізнені інструменти або вдаватися до традиційних методів, таких як ручка та папір, що призводить до неефективності та фрагментарності ведення записів.

Ще одна значна проблема пов’язана з фрагментарним характером управління даними в індустрії краси. Оскільки користувачі покладаються на різні платформи, додатки та носії для документування своїх б’юті-процедур, відсутність інтеграції та узгодженості створює значну перешкоду. Керування записами в щоденнику, подіями в календарі, нагадуваннями та інформаційними ресурсами на різних платформах не лише обтяжливе, але й підвищує ймовірність втрати даних або недогляду за ними.

Багато існуючих додатків для щоденників не пропонують достатніх можливостей персоналізації та кастомізації, пристосованих саме до контенту, пов’язаного з красою. Користувачі часто обмежені жорсткими шаблонами та типовими макетами, які не враховують усіх нюансів б’юті-процедур. Неможливість додавати фотографії, коментарі або класифікувати записи за певними категоріями краси ускладнює користувацький досвід і зменшує цінність щоденника як комплексного інструменту відстеження.

Окрім записів у щоденнику та управління календарем, б’юті-прихильники часто шукають доступ до інформативних статей, порад експертів та кураторських

ресурсів, щоб поглибити свої знання та навички. Однак на сьогоднішній день бракує спеціалізованих платформ, які б пропонували надійне сховище контенту, пов’язаного з красою, що змушує користувачів шукати потрібну інформацію в різних джерелах. Відсутність централізованого місця для доступу до інформативних статей та ресурсів загострює проблему постійного інформування та освіти у сфері краси, що постійно розвивається.

З поширенням використання мобільних пристроїв, забезпечення безперешкодного доступу на різних пристроях, включаючи смартфони та планшети, стало необхідністю. Однак багато існуючих додатків для ведення щоденників не мають адаптивного дизайну, оптимізованого для мобільних пристроїв. Така розбіжність у користувацькому досвіді між комп’ютерним та мобільним інтерфейсами знижує зручність та простоту платформи, перешкоджаючи користувачам документувати свої б’юті-процедури на ходу.

У відповідь на ці виклики «Щоденник краси» має на меті змінити спосіб, у який шанувальники краси документують, відстежують та взаємодіють зі своїми б’юті-процедурами. Пропонуючи комплексний набір функцій, розроблений спеціально для потреб шанувальників краси, «Щоденник краси» намагається усунути недоліки існуючих рішень та забезпечити бездоганну, інтегровану платформу для управління всіма аспектами догляду за собою. «Щоденник краси» прагне стати найбільш підходящою платформою для ентузіастів краси, які прагнуть підняти свої б’юті-подорожі на нові висоти.

1.3 Постановка задачі

Задача передбачає комплексну розробку веб-додатку «Щоденник краси», покликаного задовольнити різноманітні потреби та вподобання ентузіастів краси. Це завдання включає в себе ретельну реалізацію різних функціональних вимог, спрямованих на надання користувачам безперешкодного та захоплюючого досвіду документування своїх процедур краси, доступу до інформаційних ресурсів, управління подіями календаря та сприяння залученню спільноти до сфери особистої гігієни та краси. Нижче наведені категорії, які представляють різні

функціональні аспекти веб-додатку «Щоденник краси». Кожна категорія окреслює певну сферу функціональності, яка сприяє загальному користувацькому досвіду та корисності платформи:

а) реєстрація та профілі користувачів:

1) розробити функціонал, який дозволить користувачам створювати облікові записи з унікальними іменами та паролями, що забезпечить безпечний доступ до платформи;

2) впровадити функції, які дозволять користувачам редагувати свої профілі, включаючи особисту інформацію та налаштування конфіденційності, тим самим полегшуючи кастомізацію та контроль користувачів над своїми обліковими записами;

б) щоденникові записи:

1) створити надійну систему щоденників, яка дозволить користувачам вести детальні записи, що описують їхні косметичні процедури, практики догляду за шкірою, режими догляду за волоссям та вподобання щодо макіяжу;

2) передбачити можливість для користувачів легко додавати фотографії та коментарі до своїх записів у щоденнику, покращуючи візуальну репрезентацію та персоналізацію;

в) інтерактивний календар:

1) розробити та впровадити модуль інтерактивного календаря, який дозволить користувачам створювати, керувати та організовувати календарні події, пов’язані з доглядом за собою, візитами до лікаря, покупками косметики та іншими важливими заходами;

2) інтегрувати функції нагадування, щоб повідомляти користувачів про майбутні події, забезпечуючи своєчасне та ефективне планування;

г) сповіщення: розробити систему сповіщень, яка дозволить користувачам підписатися на сповіщення про нові записи в щоденнику, майбутні події та інші важливі оновлення, тим самим підвищуючи залученість користувачів та їхню взаємодію з платформою;

д) адаптивний дизайн:

1) забезпечити реалізацію підходу адаптивного дизайну, що робить веб-додаток доступним і зручним для користувачів на різних пристроях, включаючи настільні комп’ютери, ноутбуки, мобільні телефони та планшети;

2) використовувати принципи адаптивного дизайну для оптимізації користувацького досвіду та зручності інтерфейсу на різних розмірах екрану та роздільній здатності;

е) статті та ресурси:

1) створити всеосяжне сховище інформативних статей та ресурсів, що охоплюють різні теми, пов’язані з красою, особистим доглядом та порадами експертів;

2) впровадити пошукові функції, які дозволять користувачам легко орієнтуватися та отримувати доступ до статей та ресурсів за категоріями або ключовими словами, що сприятиме ефективному пошуку інформації.

Отже, розробка «Щоденника краси» передбачає ретельну інтеграцію цих функціональних рішень для розробки універсальної та орієнтованої на користувача платформи, яка дає можливість ентузіастам краси ефективно документувати, організовувати та вдосконалювати свої подорожі у світі краси. Завдяки ретельній увазі до деталей, інноваційному дизайну та бездоганній функціональності «Щоденник краси» має на меті встановити новий стандарт у сфері веб-додатків, орієнтованих на красу, надаючи користувачам цілісне рішення для їхніх потреб та прагнень, пов’язаних з красою.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Розробка веб-додатку «Щоденник краси» вимагає ретельного формування вимог для забезпечення його функціональності, зручності використання, продуктивності та надійності відповідно до передбачених цілей та очікувань користувачів. Цей процес включає комплексний аналіз функціональних і нефункціональних вимог, а також міркувань щодо розгортання, підтримки та тестування. Нижче наведено вимоги до програмної системи «Щоденник краси».

Функціональні вимоги:

а) реєстрація та профілі користувачів:

1) система повинна дозволяти користувачам створювати облікові записи з унікальними іменами користувачів та паролями для забезпечення безпечного доступу;

2) користувачі повинні мати можливість редагувати свої профілі, включаючи особисту інформацію та налаштування конфіденційності, з метою кастомізації та контролю;

б) записи в щоденнику:

1) користувачі повинні мати можливість вести щоденник, в якому детально описувати свої косметичні процедури, догляд за шкірою, волоссям та вподобання щодо макіяжу;

2) система повинна підтримувати додавання фотографій та коментарів до записів у щоденнику для покращення персоналізації та візуального представлення;

в) інтерактивний календар:

1) додаток повинен дозволяти користувачам створювати та керувати календарними подіями, пов’язаними з доглядом за собою, зустрічами та покупками;

2) користувачі повинні отримувати нагадування про майбутні події для ефективного планування;

г) перегляд профілю іншого користувача: система повинна дозволяти користувачам переглядати профіль іншого користувача, щоб мати

можливість підписатися або додати користувача форуму у друзі;

д) спілкування зареєстрованих користувачів: користувачі повинні мати можливість спілкуватися з іншими користувачами форуму;

е)сповіщення:

1) користувачі повинні мати можливість підписатися на сповіщення про нові записи в щоденнику та майбутні події, щоб залишатися в курсі подій та бути залученими до роботи з платформою;

2) нагадування про майбутні події, позначені відповідними категоріями, повинні надаватися користувачам для ефективного планування та складання розкладу;

є) адаптивний дизайн: веб-сайт повинен бути розроблений таким чином, щоб забезпечити простоту використання та доступність на різних пристроях, включаючи мобільні телефони та планшети, за допомогою принципів адаптивного дизайну;

ж) статті та ресурси:

1) користувачі повинні мати доступ до сховища інформативних статей та ресурсів на теми краси, догляду за шкірою та особистої гігієни;

2) повинні бути реалізовані пошукові функції для полегшення навігації та пошуку відповідного контенту за категоріями або ключовими словами.

Нефункціональні вимоги:

а) продуктивність веб-сайту:

1) система повинна відповідати вимогам до продуктивності, щоб забезпечити швидке завантаження та швидкість відгуку навіть у пікові періоди використання;

2) показники продуктивності, такі як швидкість завантаження сторінок і час відгуку сервера, повинні відстежуватися та оптимізуватися за необхідності;

б) інтерфейс користувача та юзабіліті:

1) інтерфейс користувача повинен бути інтуїтивно зрозумілим, візуально привабливим і зручним для навігації, що покращує загальний

користувацький досвід;

2) юзабіліті-тестування повинно проводитися для виявлення та вирішення будь-яких проблем або викликів, пов’язаних з юзабіліті;

в) вимоги до підтримки, а саме резервне копіювання даних та оновлення програмного забезпечення:

1) слід регулярно виконувати резервне копіювання даних, щоб запобігти їх втраті та забезпечити цілісність даних;

2) для підтримки безпеки та функціональності системи слід встановити процедури оновлення програмного забезпечення та патчів;

г) вимоги до тестування:

1) необхідно розробити комплексні процедури тестування для перевірки функціональності та безпеки програми;

2) функціональне тестування повинно підтвердити, що всі функції та можливості працюють належним чином, в той час як тестування безпеки повинно виявити та зменшити потенційні вразливості.

Отже, формування вимог до веб-додатку «Щоденник краси» є багатогранним процесом, який передбачає ретельний розгляд функціональних і нефункціональних аспектів, міркувань щодо процедур тестування. Дотримуючись цих вимог, можна розробити програмний комплекс «Щоденник краси», який ефективно відповідатиме потребам користувачів і забезпечить безперебійний та приємний досвід для прихильників краси.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

У сфері сучасної веб-розробки створення надійного та багатофункціонального додатку вимагає ретельного планування та дизайну. Для веб-додатку "BeautyDiary", призначеного для ентузіастів краси, які шукають персоналізовану платформу для документування своїх б’юті-рутин, режимів догляду за шкірою та вподобань щодо макіяжу, процес проєктування має вирішальне значення для забезпечення інтуїтивно зрозумілого користувацького досвіду та бездоганної функціональності.

Уніфікована мова моделювання (UML) слугує фундаментальним інструментом для проектування та візуалізації складних програмних систем, відіграючи вирішальну роль у розробці веб-додатку "BeautyDiary". UML забезпечує стандартизований підхід до представлення архітектури, дизайну та поведінки системи [7]. Використовуючи різні діаграми UML, можна створити комплексний план "BeautyDiary", гарантуючи, що його функціональність, структура та взаємодія будуть чітко визначені та зрозумілі.

На рисунку 3.1 наведена діаграма варіантів використання, яка ілюструє варіанти використання проектованої системи, укладені в межу системи, та зовнішні актори, а також певні відносини між акторами та варіантами використання.

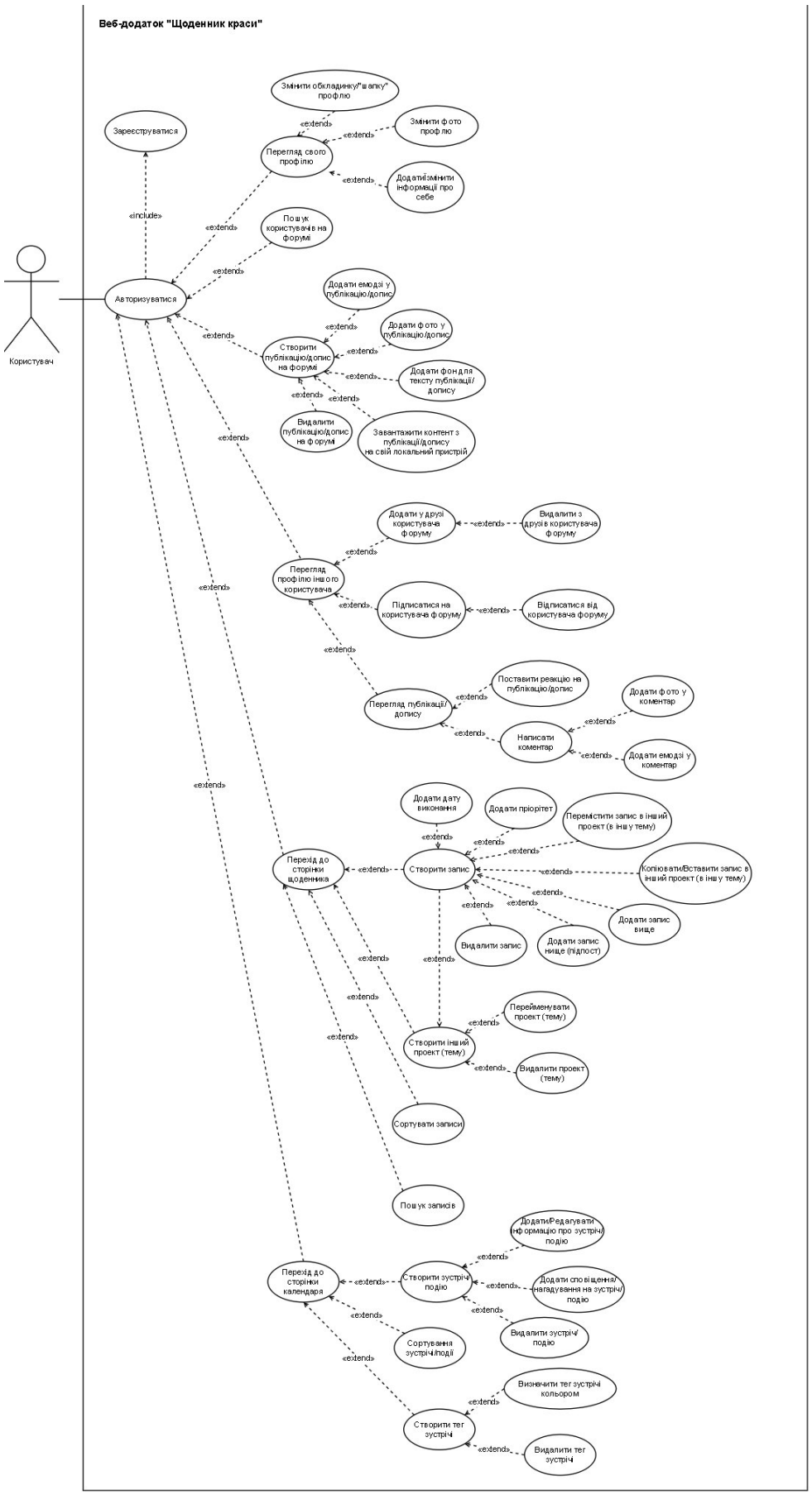

Рисунок 3.1 – Діаграма варіантів використання (рисунок виконаний самостійно)

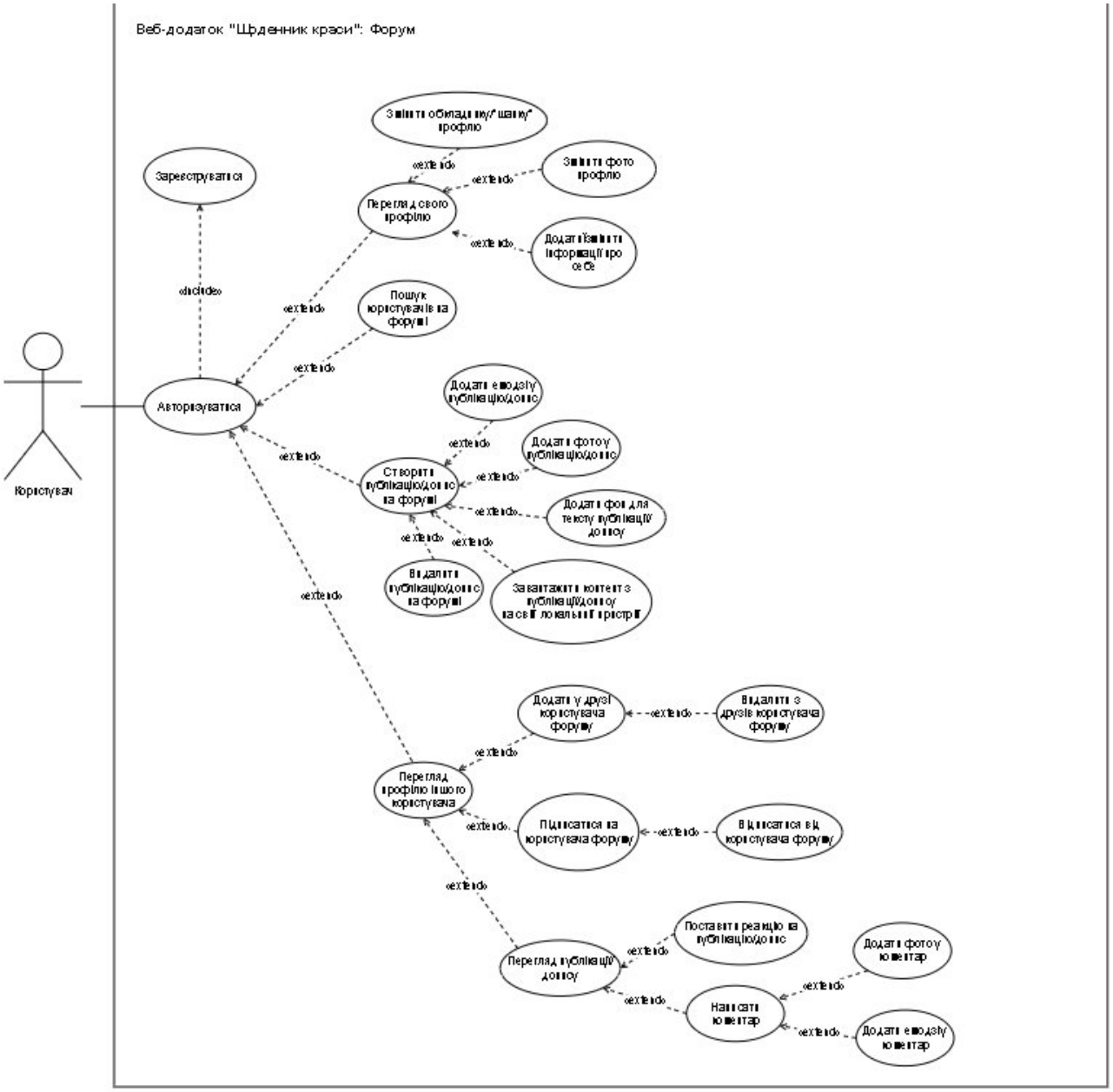

Рисунок 3.2 – Діаграма варіантів використання на форумі (рисунок виконаний самостійно)

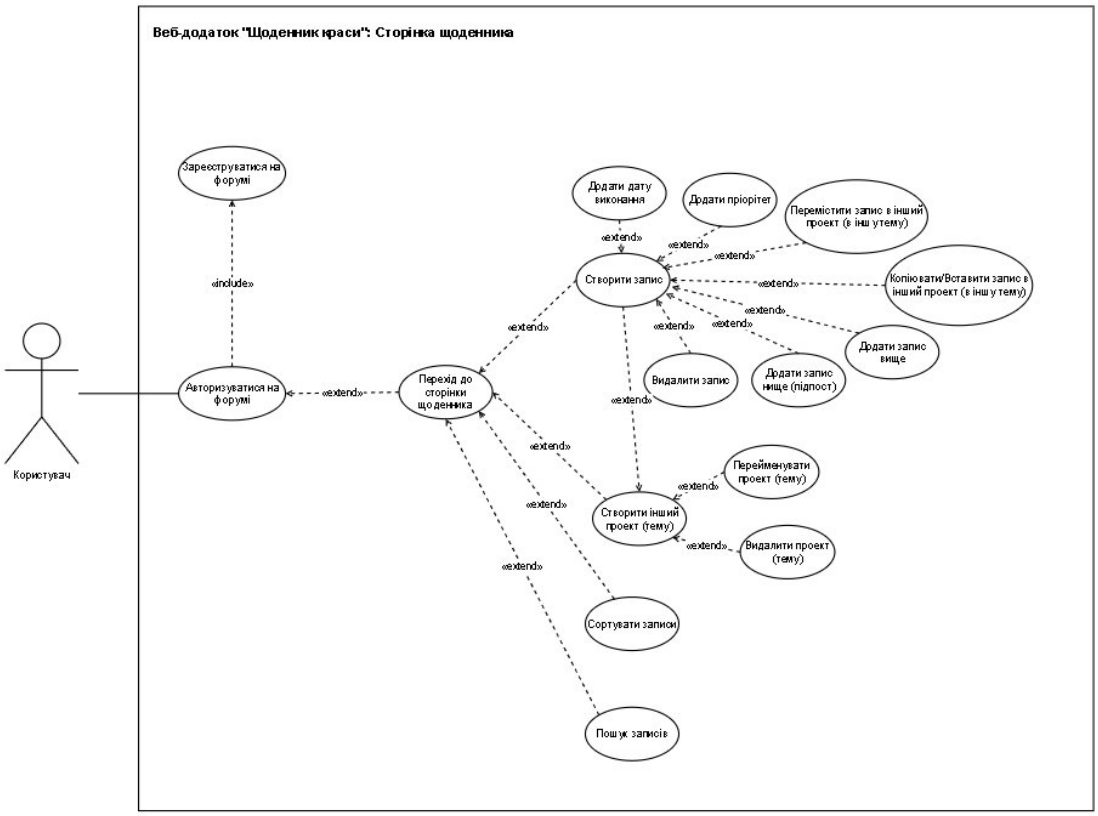

Рисунок 3.3 – Діаграма варіантів використання на сторінці щоденника (рисунок виконаний самостійно)

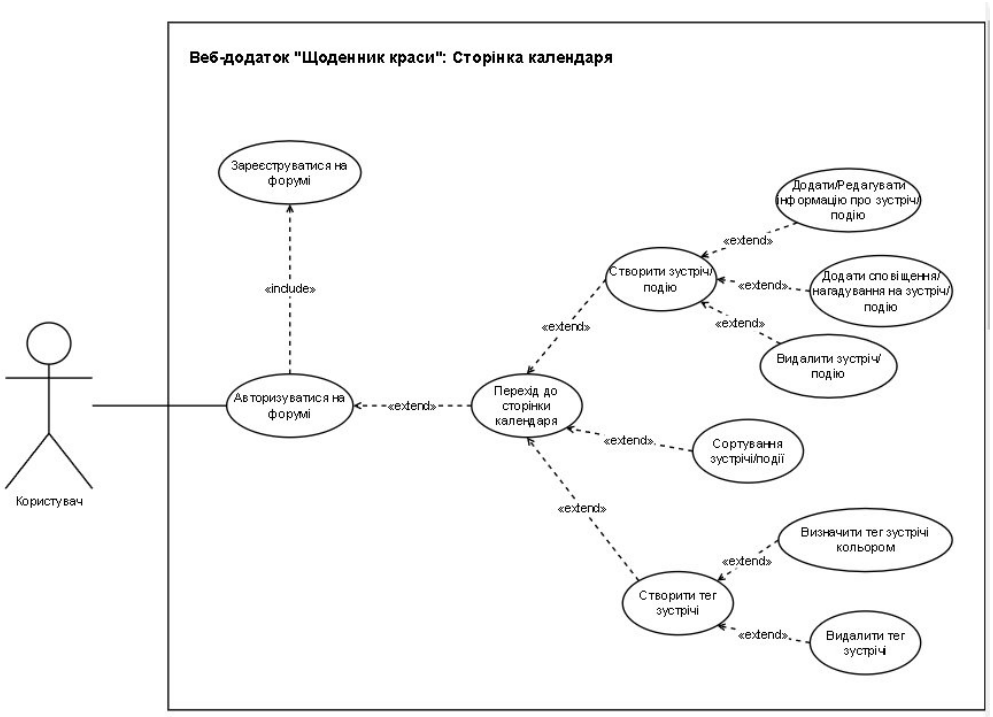

Рисунок 3.4 – Діаграма варіантів використання на сторінці календаря (рисунок виконаний самостійно)

Діаграма класів ілюструє статичну структуру "BeautyDiary", показуючи класи,

їхні атрибути, методи та зв’язки (див. рис. 3.5).

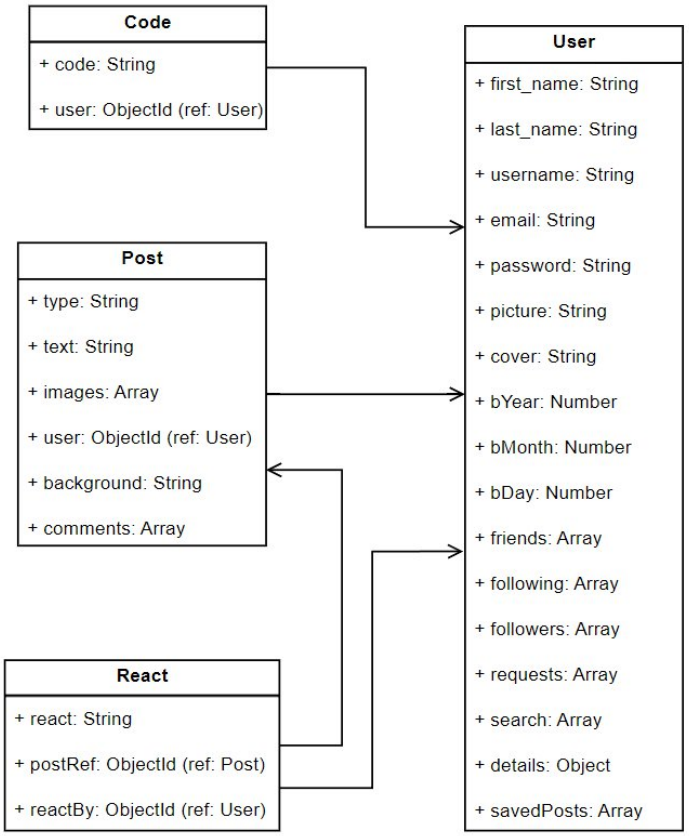

Рисунок 3.5 – Діаграма класів (рисунок виконаний самостійно)

Діаграми класів допомагають визначити план об’єктно-орієнтованого дизайну програми [8].

Атрибути класу Code:

– code: представляє код, пов’язаний з користувачем;

– user: посилається на клас User, вказуючи, що кожен екземпляр коду пов’язаний з певним користувачем.

Атрибути класу Post:

– type: вказує на тип допису (наприклад, зображення профілю, обкладинка).

– text: містить текстовий вміст публікації;

– images: зберігає зображення, пов’язані з дописом;

– user: посилається на клас User, вказуючи на користувача, який створив публікацію;

– background: відображає фон допису;

– comments: зберігає коментарі, пов’язані з публікацією.

Атрибути класу React:

– react: вказує на тип реакції (наприклад, like, love, haha) на допис;

– postRef: посилається на клас Post, вказуючи на пост, до якого належить реакція;

– reactBy: посилається на клас User, вказуючи на користувача, який відреагував на публікацію.

Атрибути класу User:

– first_name: представляє ім’я користувача;

– last_name: представляє прізвище користувача;

– username: представляє унікальне ім’я користувача;

– email: відображає адресу електронної пошти користувача;

– password: пароль користувача;

– picture: відображає зображення профілю користувача;

– cover: відображає зображення обкладинки користувача;

– bYear: відображає рік народження користувача;

– bMonth: відображає місяць народження користувача;

– bDay: відображає день народження користувача;

– friends: зберігає посилання на друзів користувача;

– following: зберігає посилання на користувачів, яких користувач відстежує;

– followers: зберігає посилання на користувачів, які підписані на користувача;

– requests: зберігає посилання на запити на додавання в друзі, що знаходяться на розгляді;

– search: зберігає історію пошуку;

– details: містить додаткову інформацію про користувача (наприклад, біографію, місце роботи);

– savedPosts: зберігає посилання на повідомлення, які користувач зберіг.

Клас Code представляє кодову сутність, наприклад, код верифікації. Він містить посилання на клас User, що вказує на те, що кожен екземпляр коду пов’язаний з конкретним користувачем, який згенерував або отримав код.

Клас Post представляє допис у додатку "BeautyDiary". Він містить такі атрибути, як текст, зображення та коментарі. Подібно до класу Code, він також має посилання на

клас User, що вказує на те, що кожен пост асоціюється з конкретним користувачем, який його створив.

Клас React представляє реакції або відповіді на публікації в додатку "BeautyDiary". Він містить такі атрибути, як тип реакції (наприклад, like, love) і посилання на класи Post і User. Це означає, що кожна реакція пов’язана з конкретним дописом і користувачем, який на нього відреагував.

Клас User представляє користувачів додатку "BeautyDiary". Він містить такі атрибути, як ім’я, прізвище, електронна пошта, пароль тощо. Він слугує центральною сутністю в системі та на нього посилаються екземпляри класів Code, Post і React, вказуючи на зв’язки між користувачами і різними діями в додатку.

Ці класи та їхні зв’язки на діаграмі класів забезпечують чітке представлення структури даних і взаємодії в додатку "BeautyDiary". Користувачі можуть створювати дописи, реагувати на дописи та взаємодіяти з іншими користувачами за допомогою коментарів та реакцій – все це відображено в структурі діаграми класів.

Діаграма розгортання зображує фізичне розгортання додатку, показуючи, як різні компоненти розподілені по серверах і вузлах [9]. Діаграма розгортання веб-додатку "BeautyDiary" ілюструє, як його компоненти розгортаються на різних вузлах або апаратних пристроях у розподіленому обчислювальному середовищі (див. рис. 3.6).

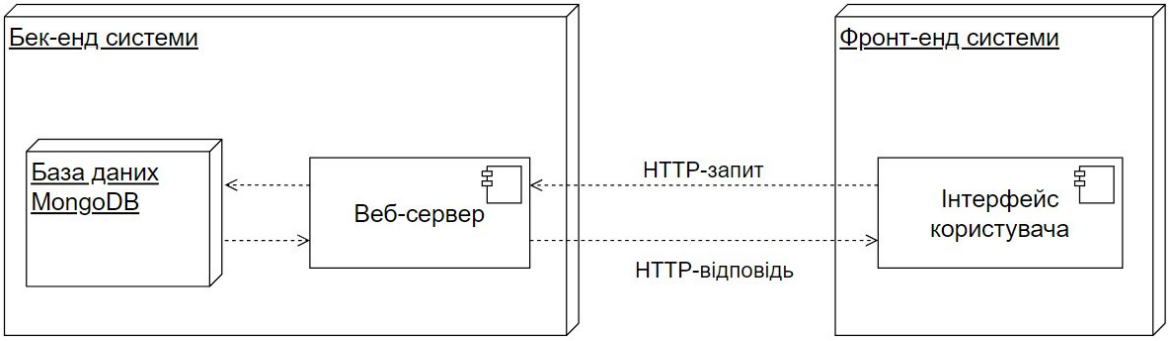

Рисунок 3.6 – Діаграма розгортання (рисунок виконаний самостійно)

MongoDB – це система баз даних NoSQL, яка використовується для зберігання та управління структурованими та неструктурованими даними. Вона зберігає дані, пов’язані з профілями користувачів, косметичними послугами, контентом та іншими даними програми. На веб-сервері розміщується внутрішня логіка додатку і

обслуговуються запити від клієнтів. Він виконує такі завдання, як маршрутизація, обробка запитів і зв’язок з базою даних.

Фронт-енд системи представляє клієнтські компоненти, що відповідають за представлення користувацького інтерфейсу та взаємодію з користувачами [10]. Компонент інтерфейсу користувача охоплює інтерфейсні елементи веб-додатку, включаючи сторінки, форми, кнопки та інші інтерактивні елементи. Він взаємодіє з бекендом за допомогою HTTP-запитів для отримання даних, надсилання форм і виконання дій на основі даних, введених користувачем. HTTP-запит передає дані або інструкції від фронт-енду до бек-енду, наприклад, запитує дані користувача, надсилає форму або отримує вміст [11]. Після обробки запиту клієнта бекенд генерує HTTP-відповідь, що містить запитувані дані або підтверджує завершення дії. Фронтенд отримує HTTP-відповідь і відповідно оновлює інтерфейс користувача, відображаючи відповідну інформацію або повідомляючи користувача про результат.

Діаграма розгортання показує, як веб-додаток "BeautyDiary" розподілений між різними рівнями: бек-енд і фронт-енд, і як потоки даних між цими компонентами передаються за допомогою HTTP-зв’язку. Вона дає уявлення про архітектуру та конфігурацію розгортання системи, допомагаючи зрозуміти її масштабованість, надійність та характеристики продуктивності.

Діаграма компонентів ілюструє високорівневі компоненти та їхні залежності в додатку, забезпечуючи візуальне представлення архітектури програми [12]. На діаграмі компонентів компоненти представляють різні аспекти веб-додатку "BeautyDiary" і взаємодіють з серверним компонентом для запису даних. Кожен компонент відповідає за управління певним типом даних у системі (див. рис. 3.7).

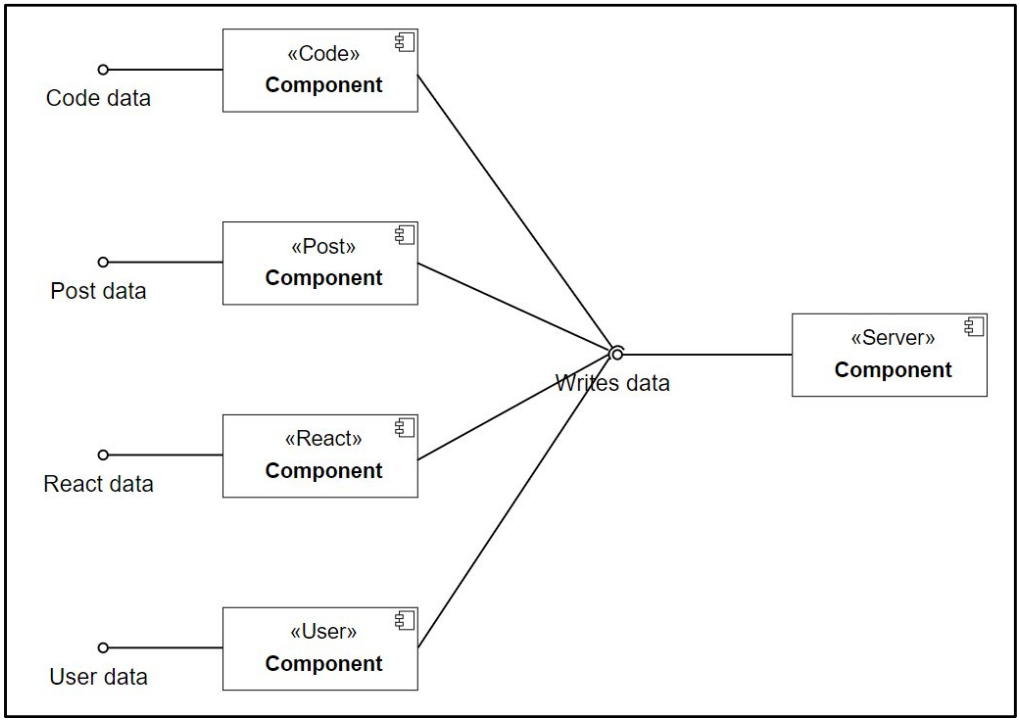

Рисунок 3.7 – Діаграма компонентів (рисунок виконаний самостійно)

Компонент коду відповідає за управління даними, пов’язаними з кодом, у програмі та взаємодіє з серверним компонентом для запису даних коду.

Компонент публікації керує даними, пов’язаними з публікацією, зокрема текстом, зображеннями та коментарями. Взаємодіє з серверним компонентом для написання даних публікації.

Компонент реакції обробляє реакції на дописи, зроблені користувачами. Взаємодіє з серверним компонентом для запису даних про реакцію.

Компонент користувача керує даними, пов’язаними з користувачами, включаючи особисті дані, друзів і збережені повідомлення. взаємодіє з серверним компонентом для запису даних користувача.

Серверний компонент діє як центральний компонент, який отримує дані від інших компонентів і керує записом даних на сервер. Він взаємодіє з кожним з інших компонентів, щоб отримувати дані і зберігати їх належним чином.

Діаграма компонентів ілюструє, як різні компоненти веб-додатку "BeautyDiary" взаємодіють один з одним і з сервером для управління та запису різних типів даних, забезпечуючи коректне функціонування додатку та збереження цілісності даних.

Діаграма послідовності відображає взаємодію між об’єктами всередині додатку з

плином часу, демонструючи потік повідомлень і порядок подій під час виконання [13]. На рисунку 3.8 наведена діаграма послідовності, яка описує послідовність взаємодії між різними компонентами веб-додатку "BeautyDiary" при взаємодії користувача з системою для відвідування та навігації по його сторінках.

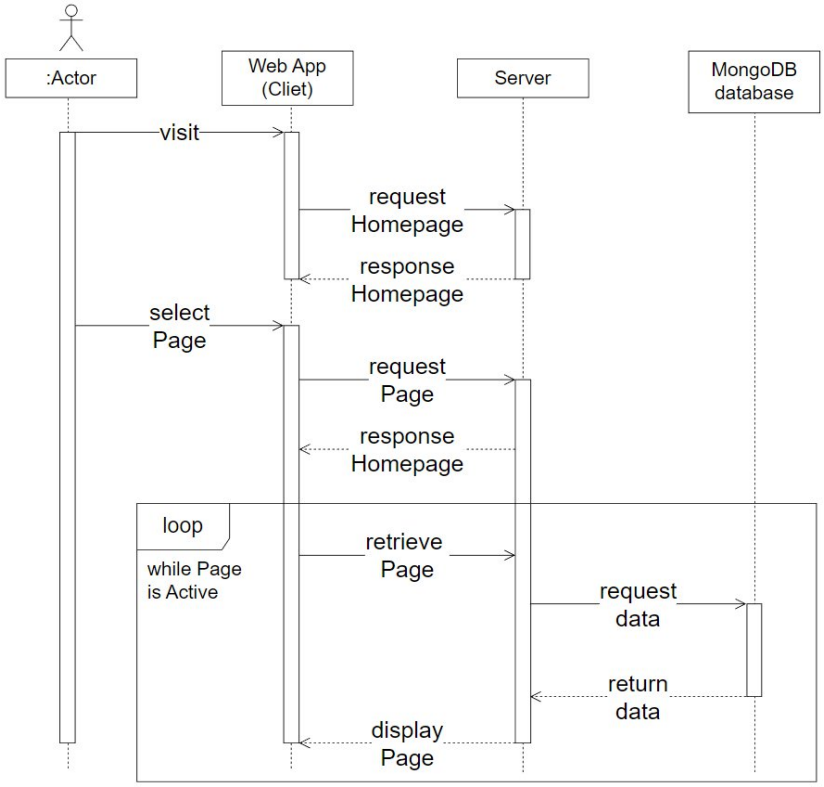

Рисунок 3.8 – Діаграма послідовності (рисунок виконаний самостійно)

Актор представляє користувача, який взаємодіє з веб-додатком. Компонент Web App (Client) представляє інтерфейс веб-додатку, включаючи користувальницький інтерфейс і скрипти на стороні клієнта. Компонент Server символізує внутрішній сервер, на якому розміщено веб-додаток і який обробляє клієнтські запити. База даних MongoDB зображена як внутрішня база даних, що відповідає за зберігання та управління даними додатку.

Актор (користувач) ініціює взаємодію, відвідуючи веб-додаток (на стороні клієнта) через веб-браузер. Клієнт (веб-браузер) надсилає запит на сервер, щоб отримати домашню сторінку веб-додатку. Отримавши запит, сервер обробляє його і повертає клієнту вміст домашньої сторінки. Актор взаємодіє з клієнтським веб-додатком, вибираючи певну сторінку в додатку. Клієнт надсилає ще один запит до сервера, цього

разу запитуючи конкретну сторінку, обрану користувачем. Після отримання запиту на обрану сторінку, сервер обробляє запит і повертає клієнту вміст запитуваної сторінки. Секція "цикл – поки сторінка активна " триває доти, доки обрана сторінка залишається активною або відкритою в браузері користувача. В рамках циклу клієнт періодично отримує з сервера оновлення або додаткові дані, пов’язані з активною сторінкою. Отримавши запит на дані, пов’язані зі сторінкою, сервер взаємодіє з базою даних MongoDB, щоб отримати необхідну інформацію. База даних MongoDB обробляє запит і повертає запитувані дані назад на сервер. З отриманими даними з бази даних сервер оновлює вміст активної сторінки і відправляє її назад клієнту для відображення. Цикл триває доти, доки користувач залишається на активній сторінці, а клієнт періодично отримує оновлення з сервера.

Діаграма послідовності ілюструє потік взаємодії між актором (користувачем), клієнтським веб-додатком, сервером і базою даних MongoDB в процесі відвідування і навігації по сторінках веб-додатку "BeautyDiary". Він надає детальне уявлення про те, як відбувається обмін даними та їх відображення користувачеві під час взаємодії.

Діаграма пакетів надає високорівневий огляд організації та модуляризації "BeautyDiary". Вона дозволяє групувати пов’язані класи, компоненти та функції в цілісні пакети, тим самим сприяючи модульності, багаторазовому використанню та підтримці програми [14]. На рисунку 3.9 наведена діаграма пакетів, яка ілюструє високорівневу організацію та залежності між різними компонентами або модулями в межах веб-додатку "BeautyDiary".

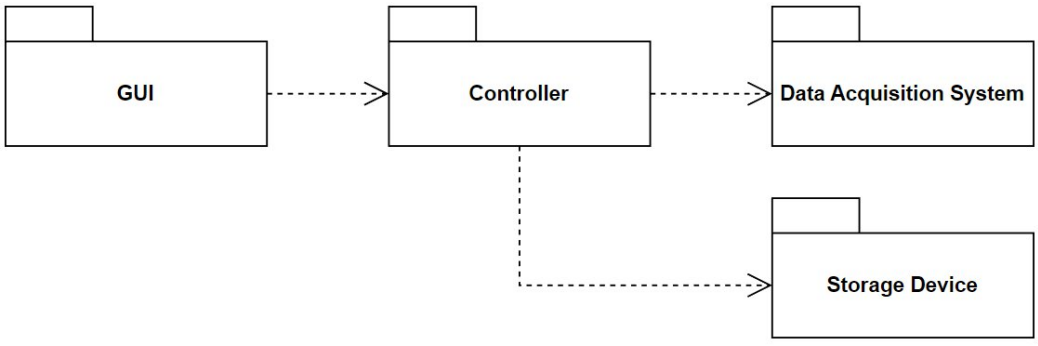

Рисунок 3.9 – Діаграма пакетів (рисунок виконаний самостійно)

Пакет GUI (Graphical User Interface) представляє компонент графічного

інтерфейсу користувача додатку. Він охоплює всі візуальні елементи та функції взаємодії з користувачем, з якими користувачі взаємодіють під час використання додатку. Пакет Controller слугує посередником між графічним інтерфейсом та іншими компонентами системи. Він керує введенням даних користувачем, логікою програми та потоком даних між графічним інтерфейсом та базовими компонентами. Пакет Data Acquisition System представляє систему, що відповідає за отримання або збір даних з різних джерел. Він може включати функції, пов’язані з отриманням, обробкою та інтеграцією даних із зовнішніх джерел або датчиків. Пакет Storage Device інкапсулює функції, пов’язані зі зберіганням даних та управлінням ними. Він може включати модулі, що відповідають за зберігання, отримання та управління даними в постійній системі зберігання, такій як база даних або файлова система.

GUI залежить від функціональних можливостей, наданих контролером для обробки користувацького вводу та логіки програми. Модуль Controller взаємодіє з Data Acquisition System для отримання або обробки даних, необхідних для функціональності програми. Модуль Controller взаємодіє з пристроєм зберігання даних для зберігання або отримання даних додатку.

Діаграма пакетів надає стислий огляд основних компонентів/модулів додатку "BeautyDiary" та їх взаємозв’язків, підкреслюючи, як різні частини системи співпрацюють для надання користувачам запланованої функціональності.

3.2 Проєктування архітектури ПЗ

Розробка архітектури програмного забезпечення для веб-додатку "BeautyDiary" передбачає створення надійної клієнт-серверної архітектури для полегшення ефективної комунікації, управління ресурсами та надання послуг. Ця архітектура слугуватиме основою для управління взаємодією з користувачами, зберігання та пошуку даних, а також забезпечення безперебійної функціональності на різних пристроях та платформах.

Архітектура клієнт-сервер складається з двох основних компонентів: клієнта та сервера. Кожен з них відіграє певну роль у забезпеченні зв’язку та управлінні ресурсами в системі. Сервер – це те, що надає запитувані послуги, а клієнти – це ті, хто запитує послуги. Архітектура клієнт-сервер – це обчислювальна модель, в якій сервер розміщує,

надає та керує більшістю ресурсів і послуг, запитуваних клієнтом [15]. Вона також відома як мережева обчислювальна модель або мережа клієнт-сервер, оскільки всі запити та послуги надаються через мережу (див. рис. 3.10).

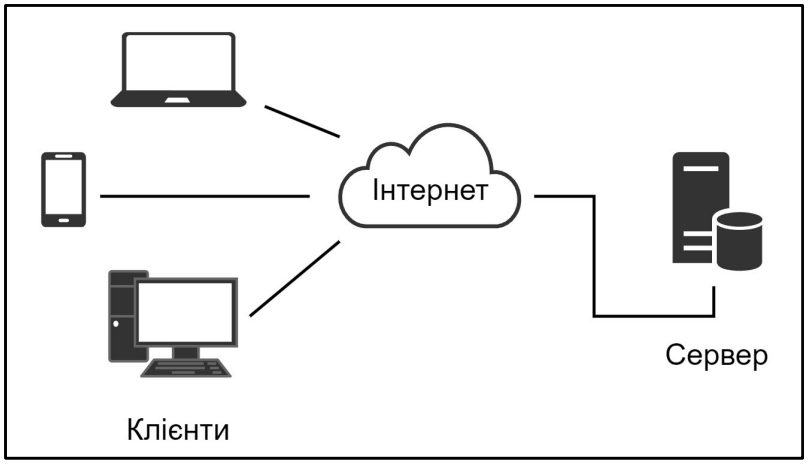

Рисунок 3.10 – Мережа клієнт-сервер (рисунок виконаний самостійно)

Архітектура або модель клієнт-сервер має інші системи, підключені через мережу, де ресурси спільно використовуються різними комп’ютерами.

У клієнт-серверній архітектурі веб-додатку "BeautyDiary" клієнт являє собою інтерфейс кінцевого користувача, за допомогою якого користувачі взаємодіють з додатком. Сюди входять веб-браузери на стаціонарних комп’ютерах, мобільні браузери на смартфонах і планшетах, а також будь-які спеціалізовані програмні додатки, розроблені для конкретних платформ. Клієнти ініціюють зв’язок, надсилаючи запити на сервер, як правило, через протоколи HTTP або HTTPS. Ці запити можуть включати такі дії, як вхід в систему, створення записів у щоденнику, доступ до подій календаря або перегляд статей і ресурсів. На стороні сервера розгортається надійна інфраструктура для управління зберіганням даних, обробки запитів і надання послуг клієнтам. Це може включати розгортання додатку на хмарних серверах або виділених хостингових середовищах для забезпечення масштабованості, надійності та продуктивності [16]. Сервер відповідає за обробку вхідних клієнтських запитів, обробку та перевірку вхідних даних користувача, взаємодію з базами даних для отримання та зберігання створеного користувачем контенту, реалізацію бізнес-логіки та робочих процесів додатків, а також інтеграцію із зовнішніми системами або сервісами для розширення функціональності та покращення користувацького досвіду.

Нижче наведена архітектура клієнт-сервер для веб-додатку "BeautyDiary" (див. рис. 3.11).

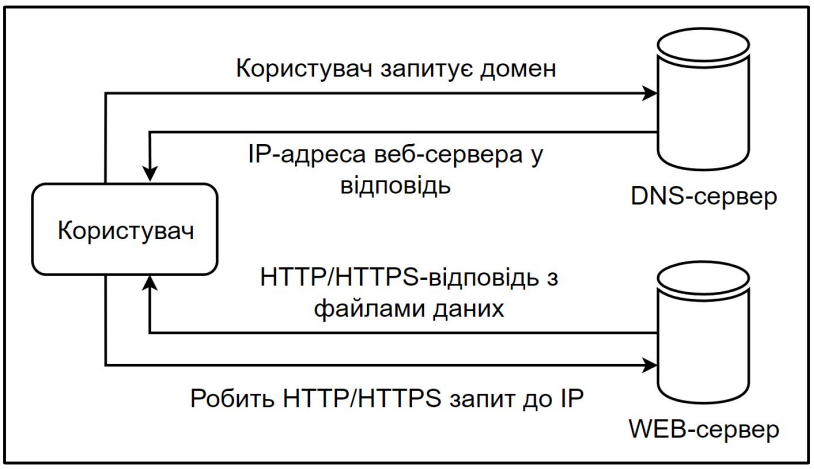

Рисунок 3.11 – Архітектура клієнт-сервер для веб-додатку "BeautyDiary" (рисунок виконаний самостійно)

Користувач вводить уніфікований локатор ресурсів (URL) веб-сайту або файлу, а браузер надсилає запит до сервера системи доменних імен (DNS). DNS-сервер відповідає за пошук і отримання IP-адреси, пов’язаної з веб-сервером, а потім ініціює дії з використанням цієї IP-адреси [17]. Після відповіді DNS-сервера браузер надсилає HTTP- або HTTPS-запит на IP-адресу веб-сервера, яка була надана DNS-сервером. Після отримання запиту сервер починає передавати необхідні файли веб-сайту. Зрештою, файли обробляються браузером, після чого веб-сайт стає доступним для перегляду.

Ключові особливості клієнт-серверної архітектури:

– клієнт-серверна архітектура дозволяє здійснювати горизонтальне масштабування шляхом додавання нових серверів, щоб впоратися зі зростаючим користувацьким навантаженням. Це гарантує, що додаток може пристосуватися до зростання без шкоди для продуктивності та надійності;

– зберігання та управління даними централізовано на стороні сервера, що забезпечує узгоджений доступ і синхронізацію між кількома клієнтами. Це забезпечує цілісність даних, полегшує процеси резервного копіювання та відновлення, а також спрощує обслуговування та оновлення;

– клієнти в рамках архітектури можуть ефективно обмінюватися ресурсами та

співпрацювати над завданнями, що виконуються сервером. Це включає спільний доступ до записів у щоденнику, подій календаря та статей/ресурсів, а також доступ до спільних сховищ даних і сервісів;

– розподілена між кількома серверами, клієнт-серверна архітектура підвищує надійність та відмовостійкість системи. Резервні системи та механізми обходу відмов забезпечують безперебійне надання послуг навіть у випадку збоїв у роботі серверів чи мережі.

Завдяки реалізації добре продуманої клієнт-серверної архітектури, веб-додаток "BeautyDiary" може забезпечити користувачам безперебійну та швидку роботу, дозволяючи їм керувати своїми процедурами краси, отримувати доступ до інформативних ресурсів та взаємодіяти з б’юті-спільнотою на різних пристроях та платформах.

3.3 Проєктування структури зберігання даних

При проєктуванні структури зберігання даних для веб-додатку "BeautyDiary" MongoDB виявилася надійним і універсальним вибором, пропонуючи гнучке і масштабоване рішення для управління різноманітними даними, що генеруються користувачами. Підхід MongoDB, орієнтований на роботу з документами, легко узгоджується з характером записів у щоденнику краси, які часто складаються з різноманітного та мультимедійного контенту.

MongoDB – це база даних NoSQL, відома своєю документно-орієнтованою моделлю даних, що забезпечує динамічний та безсхемний підхід до зберігання даних [18]. На відміну від традиційних реляційних баз даних, MongoDB організовує дані в колекції документів, кожен з яких представлений у форматі JSON (JavaScript Object Notation). Це дозволяє зберігати складні та ієрархічні структури даних, що ідеально підходить для розміщення різноманітної інформації, наприклад, пов’язаної з косметичними процедурами, режимами догляду за шкірою та вподобаннями в макіяжі. Крім того, вбудована підтримка MongoDB BSON (Binary JSON), формату серіалізації з двійковим кодуванням, підвищує ефективність і продуктивність зберігання та пошуку даних. BSON розширює JSON,

додаючи підтримку додаткових типів даних і оптимізацій, що дозволяє MongoDB легко обробляти різноманітні набори даних, забезпечуючи при цьому сумісність з додатками та інструментами на основі JSON [19]. Структура бази даних MongoDB наведена на рисунку 3.12.

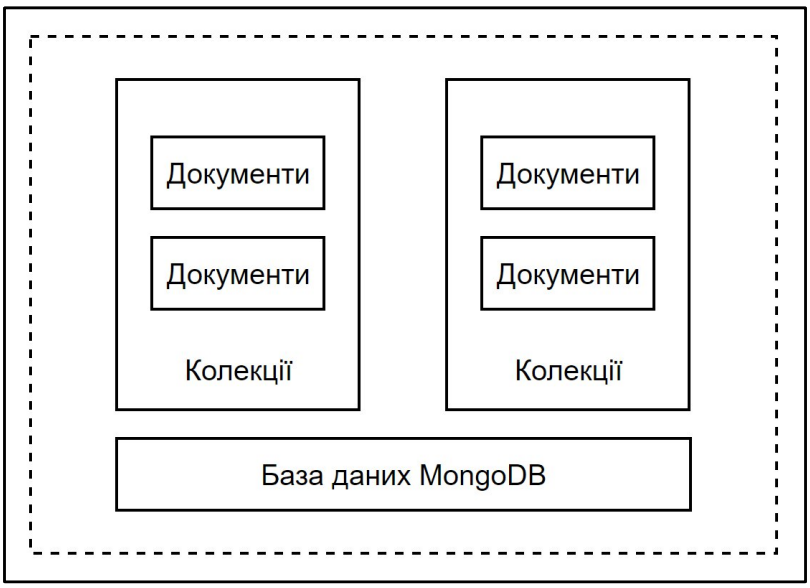

Рисунок 3.12 – Структура бази даних MongoDB (рисунок виконаний самостійно)

Колекції MongoDB слугують контейнерами для пов’язаних наборів документів. Кожна колекція представляє окрему категорію або тип даних, наприклад, записи щоденника, профілі користувачів, події календаря або ресурси статей. Колекції створюються динамічно по мірі додавання документів, що забезпечує гнучкість і масштабованість у міру зростання додатку.

Документи в MongoDB – це фундаментальні одиниці даних, структуровані як пари ключ-значення у форматі JSON. Ці документи інкапсулюють окремі записи в колекції, забезпечуючи багате та виразне представлення даних [20]. Наприклад, запис у щоденнику може містити такі поля, як дата, заголовок, зміст, додані фотографії та коментарі користувачів, що дає змогу користувачам отримати детальну інформацію про свої косметичні процедури.

Особливості MongoDB:

– безсхемний дизайн MongoDB забезпечує гнучку та ітеративну розробку, що дозволяє безперешкодно еволюціонувати моделям даних з часом;

– MongoDB підтримує горизонтальне масштабування за допомогою

шардингу, розподіляючи дані між декількома серверами, щоб впоратися зі зростаючими обсягами даних і трафіком користувачів;

– MongoDB надає надійні можливості індексування, що дозволяє ефективно запитувати та отримувати дані на основі різних критеріїв, таких як дата, користувач або ключові слова контенту;

– MongoDB пропонує вбудовані функції реплікації для забезпечення доступності та відмовостійкості даних, з підтримкою автоматичного обходу збоїв та резервування даних на декількох вузлах.

Для веб-додатку "BeautyDiary" MongoDB слугує основою архітектури зберігання даних, полегшуючи ефективне управління записами щоденника, профілями користувачів, подіями календаря та статтями ресурсів. Використовуючи багатий функціонал та гнучку модель даних MongoDB, додаток може задовольнити різноманітні потреби та вподобання ентузіастів краси, дозволяючи їм легко фіксувати, організовувати та знаходити інформацію, пов’язану з красою.

Отже, MongoDB є потужним та адаптивним рішенням для проектування структури зберігання даних веб-додатку "BeautyDiary", пропонуючи масштабованість, гнучкість та продуктивність, необхідні для підтримки динамічного характеру даних, пов’язаних з красою, та взаємодії з користувачами.

3.4 Приклади найцікавіших алгоритмів та методів

У сучасному швидкоплинному світі ефективне управління подіями та зустрічами має вирішальне значення для особистої та професійної продуктивності. Веб-додаток "BeautyDiary" використовує можливості Node.js, Express.js, MongoDB та інших технологій, щоб забезпечити у багатофункціональній платформі створення, керування та нагадування користувачам про заплановані події у календарі, в якому пріоритетами є безпека, масштабованість та зручність для користувача.

Нижче наведено код створення інтерактивного календаря за допомогою Node.js з Express.js для бекенду і FullCalendar.js для фронтенду. Цей приклад демонструє, як обробляти CRUD-операції для подій календаря та реалізувати нагадування про майбутні

події за допомогою фонової бібліотеки обробки завдань, такої як Agenda.js.

Проєкт вимагає декількох npm-пакетів, включаючи Express.js для обробки HTTP-запитів, Mongoose для взаємодії з MongoDB, Agenda для планування завдань, bcrypt для хешування паролів і jsonwebtoken для генерації токенів JWT:

```
const express = require('express');
const mongoose = require('mongoose');
const Agenda = require('agenda');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const app = express();
const port = 3000;
```

Програма підключається до бази даних MongoDB за допомогою Mongoose. Обробник події db.once('open') записує повідомлення, коли з’єднання з базою даних встановлено:

```
mongoose.connect('mongodb://localhost/calendar', { useNewUrlParser: true, useUnifiedTopology: true });
const db = mongoose.connection;
db.once('open', () => {
  console.log('Connected to MongoDB');
});
```

Програма використовує Agenda для визначення та обробки фонових завдань. У цьому випадку він визначає завдання з назвою send reminder для надсилання нагадувань про події календаря:

```
const agenda = new Agenda({ db: { address: 'mongodb://localhost/calendar' } });
agenda.define('send reminder', async job => {
  const { eventId, userId } = job.attrs.data;
  const event = await Event.findById(eventId);
  const user = await User.findById(userId);
  console.log(`Sending reminder for event "${event.title}" to user "${user.username}"`); });
```

Визначення схем визначає схеми Mongoose для моделей User і Event. Схема події містить такі поля, як назва, опис, дата початку, дата завершення і createdBy (посилання на користувача, який створив подію):

```
app.use(express.json())
const eventSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: String,
  start: { type: Date, required: true },
  end: { type: Date, required: true },
  createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }});
const Event = mongoose.model('Event', eventSchema);
const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
```

```
  password: { type: String, required: true },
  events: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Event' }]
});
userSchema.methods.generateAuthToken = function() {
  return jwt.sign({ _id: this._id }, 'secretkey');};
const User = mongoose.model('User', userSchema);
```

Схема користувача включає такі поля, як ім’я користувача, електронна пошта, пароль і масив подій. Метод generateAuthToken, визначений у схемі User, генерує JWT-токен для користувача. Цей токен надсилається клієнту після успішної автентифікації і може бути використаний для доступу до захищених маршрутів.

Забезпечення автентифікації визначає функцію проміжного програмного забезпечення з назвою authenticateUser для автентифікації користувачів за допомогою токенів JWT:

```
const authenticateUser = async (req, res, next) => {  try {
    const token = req.header('Authorization').replace('Bearer ', '');
    const decoded = jwt.verify(token, 'secretkey');
    const user = await User.findOne({ _id: decoded._id });
    if (!user) {
      throw new Error();    }
    req.user = user;
    next();  } catch (error) {
    res.status(401).json({ error: 'Please authenticate' });  }};
```

Ця частина коду застосовується до захищених маршрутів, щоб гарантувати, що тільки автентифіковані користувачі можуть отримати до них доступ.

CRUD-операції для подій визначає маршрути для створення (POST /events) та отримання (GET /events) подій:

```
app.post('/events', authenticateUser, async (req, res) => {
  try {
   const event = new Event({ ...req.body, createdBy: req.user._id });
    await event.save();
    req.user.events.push(event);
    await req.user.save();
    res.status(201).json(event);  } catch (err) {
    res.status(400).json({ message: err.message });  }});


app.get('/events', authenticateUser, async (req, res) => {
  try {  const events = await Event.find({ createdBy: req.user._id });
    res.json(events);  } catch (err) {
    res.status(500).json({ message: err.message });  }});
agenda.start();
app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

Код також включає логіку обробки помилок з використанням блоків try-catch

для обробки винятків і реагування на них відповідними повідомленнями про помилки.

Що цікавого в цьому коді:

– додаток реалізує автентифікацію користувачів та хешування паролів для забезпечення безпечного доступу до ресурсів;

– використовує Agenda для планування та обробки фонових завдань, таких як надсилання нагадувань про події календаря;

– код структурований для підтримки масштабованості, з розділенням завдань і проміжного програмного забезпечення для автентифікації;

– він відповідає сучасним практикам, таким як використання JWT для автентифікації та Mongoose для моделювання баз даних, що робить його актуальним та сучасним відповідно до поточних стандартів розробки.

Отже, цей код демонструє надійну внутрішню архітектуру для управління подіями календаря з автентифікацією користувачів і фоновою обробкою завдань, що робить його комплексним рішенням для створення інтерактивних календарів у додатки.

3.5 Створення UI / UX дизайну системи

Концепція UI/UX, що розшифровується як User Interface/User Experience, відіграє ключову роль у розвитку та успіху "BeautyDiary", веб-додатку, присвяченого процедурам краси, догляду за шкірою, волоссям, макіяжу та суміжним темам.

UI дизайн включає в себе візуальні аспекти інтерфейсу, включаючи макет, кольори, типографіку та інтерактивні елементи, спрямовані на створення естетично привабливих і зручних для навігації інтерфейсів [21]. З іншого боку, UX-дизайн фокусується на загальному користувацькому досвіді, гарантуючи, що інтерфейс є інтуїтивно зрозумілим, ефективним і приємним у використанні завдяки розумінню поведінки та потреб користувачів [22].

Форми реєстрації та авторизації користувача наведені на рисунку 3.13.

Чисті та інтуїтивно зрозумілі форми реєстрації та авторизації користувача з полями для імені користувача, електронної пошти, пароля та додатковими полями для додаткової особистої інформації.

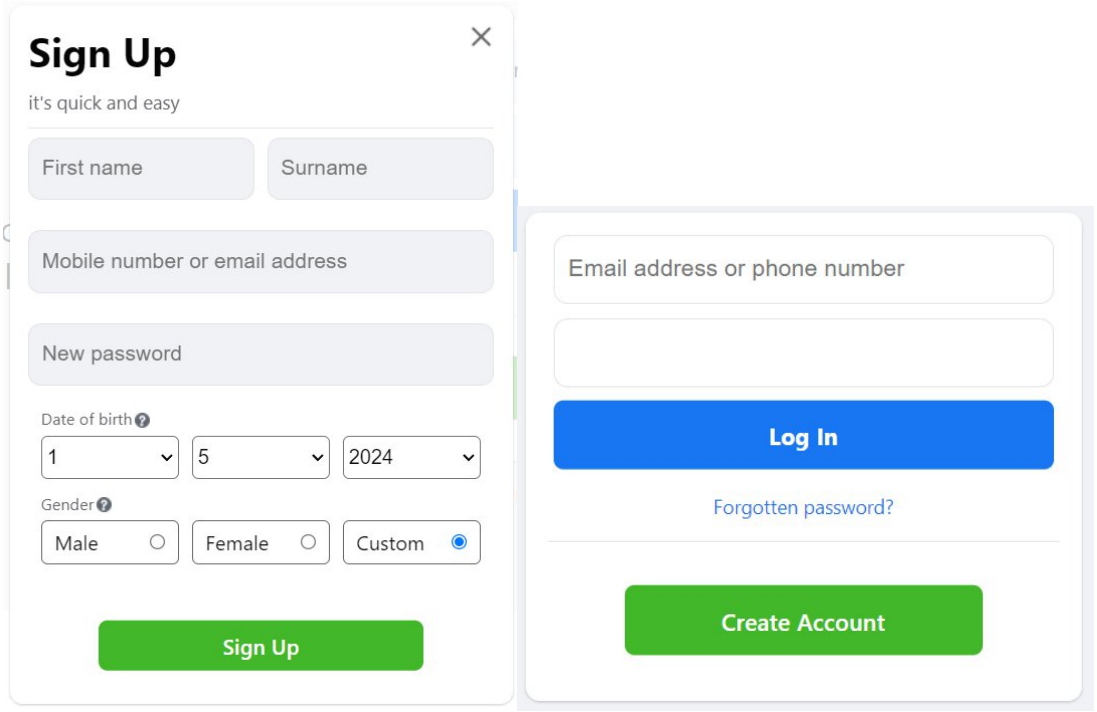

Рисунок 3.13 – Форми реєстрації та авторизації (рисунок виконаний самостійно)

Сторінка редагування профілю наведена на рисунку 3.14.

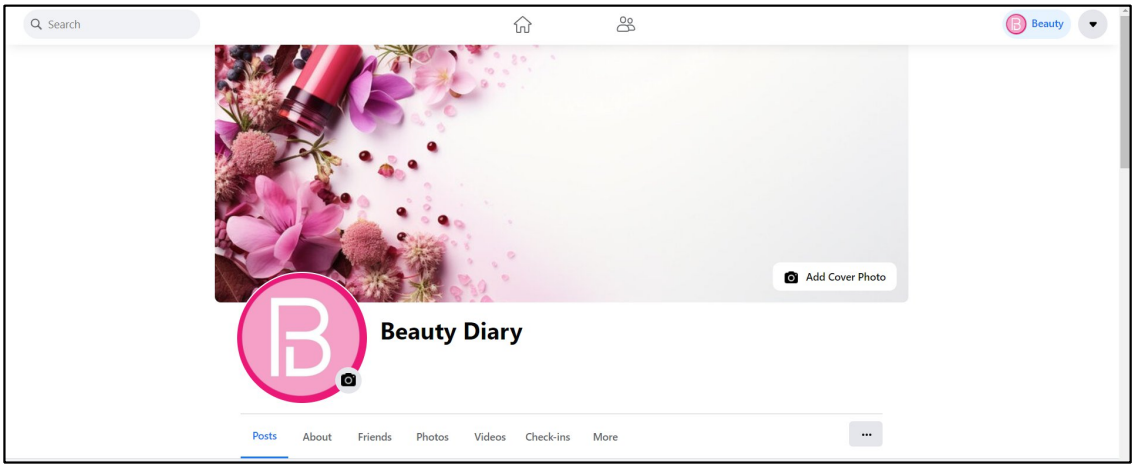

Рисунок 3.14 – Сторінка редагування профілю (рисунок виконаний самостійно)

Зрозумілий та доступний інтерфейс для користувачів для редагування своїх профілів, включаючи опції для оновлення особистої інформації, зображення профілю та обкладинки.

Вкладка для організації різних розділів профілю, таких як особисті дані наведено на рисунку 3.15.

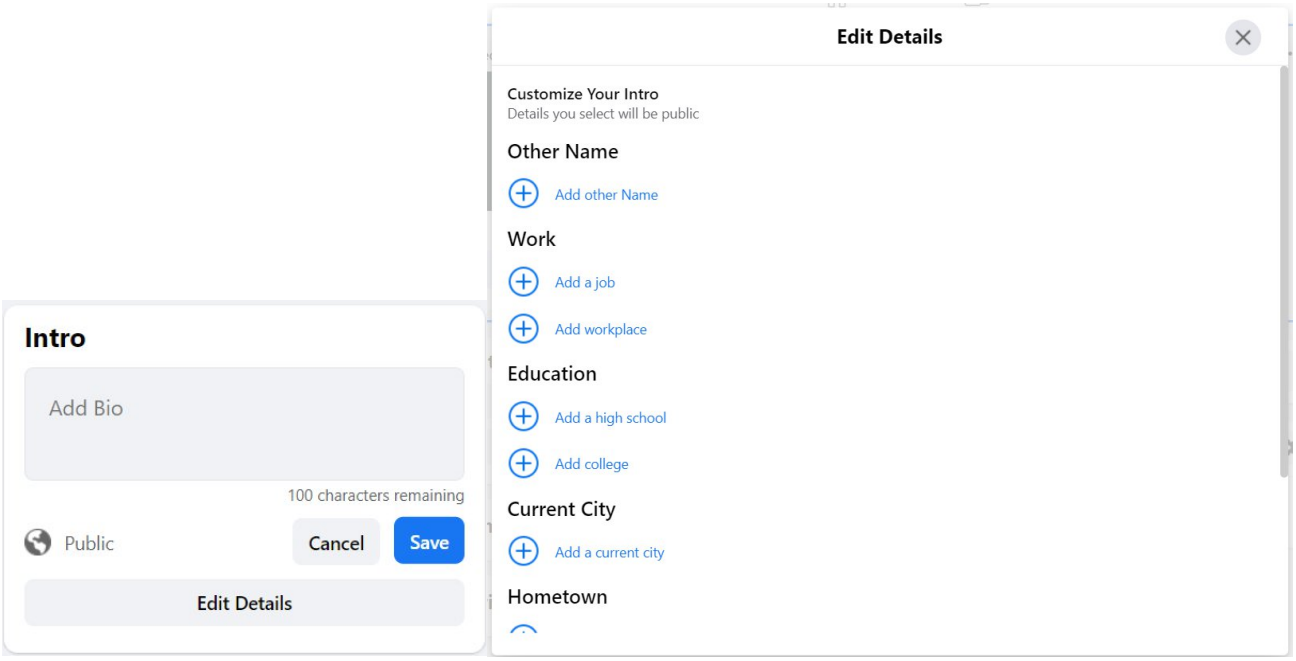

Рисунок 3.15 – Вкладка для організації різних розділів профілю (рисунок виконаний самостійно)

Сторінка для створення записів у щоденнику наведена на рисунку 3.16.

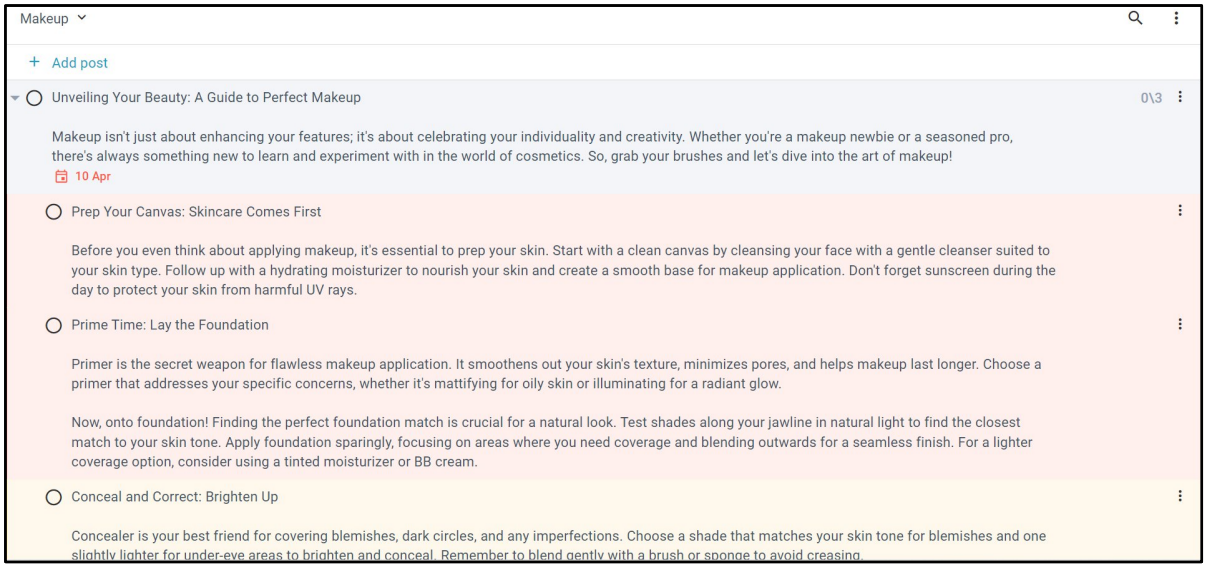

Рисунок 3.16 – Сторінка для створення записів у щоденнику (рисунок виконаний самостійно)

Інтуїтивно зрозуміла сторінка для створення записів у щоденнику з полями для опису, дати та вибору категорії (наприклад, догляд за шкірою, волоссям, макіяж).

Інтерфейс календаря наведено на рисунку 3.17.

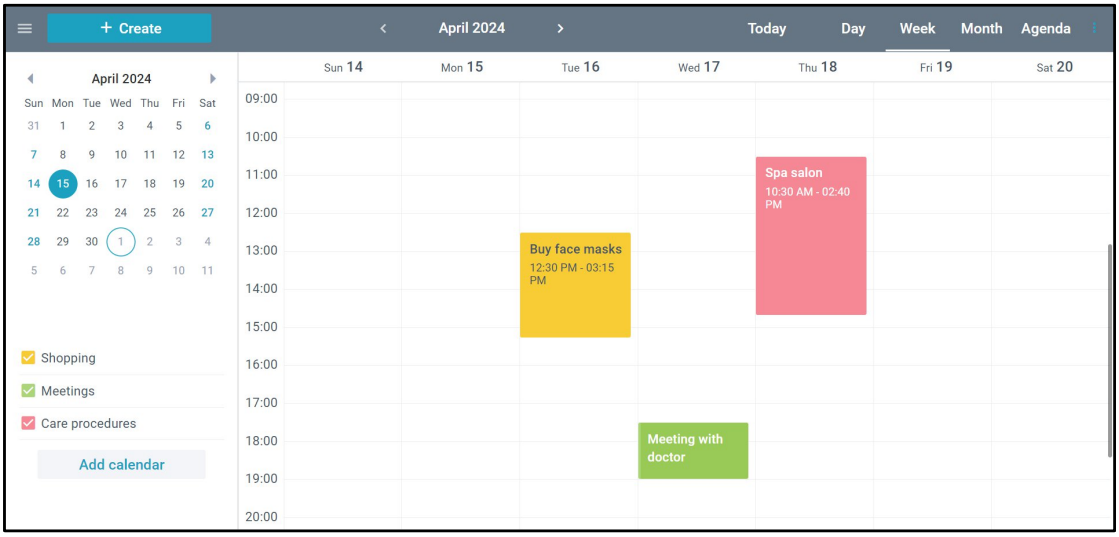

Рисунок 3.17 – Інтерфейс календаря (рисунок виконаний самостійно)

Адаптивний перегляд календаря з можливістю щоденної, щомісячної та щотижневої навігації.

Наявна можливість додавати, редагувати та видаляти події безпосередньо з інтерфейсу календаря, а також кольорове тегування категорій подій для легкої ідентифікації (див. рис. 3.18).

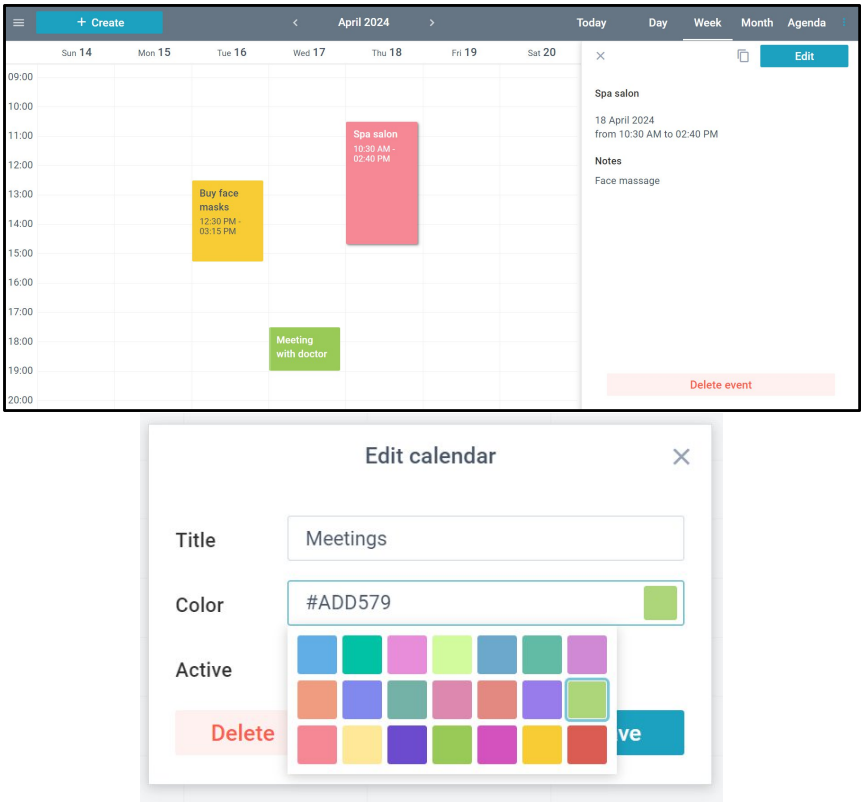

Рисунок 3.18 – Додавання, редагування та видалення події та тегів (рисунок виконаний самостійно)

Система нагадувань наведена на рисунку 3.19.

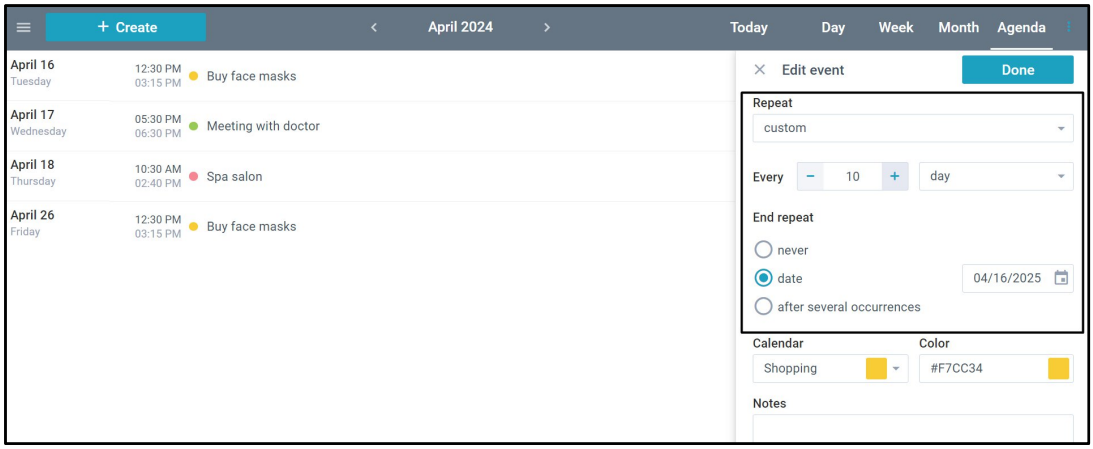

Рисунок 3.19 – Система нагадувань (рисунок виконаний самостійно)

Параметри нагадувань про майбутні події, що дозволяють користувачам встановлювати сповіщення в додатку. Можливість увімкнути або вимкнути нагадування безпосередньо зі сповіщень.

Керування підписками наведено на рисунку 3.20.

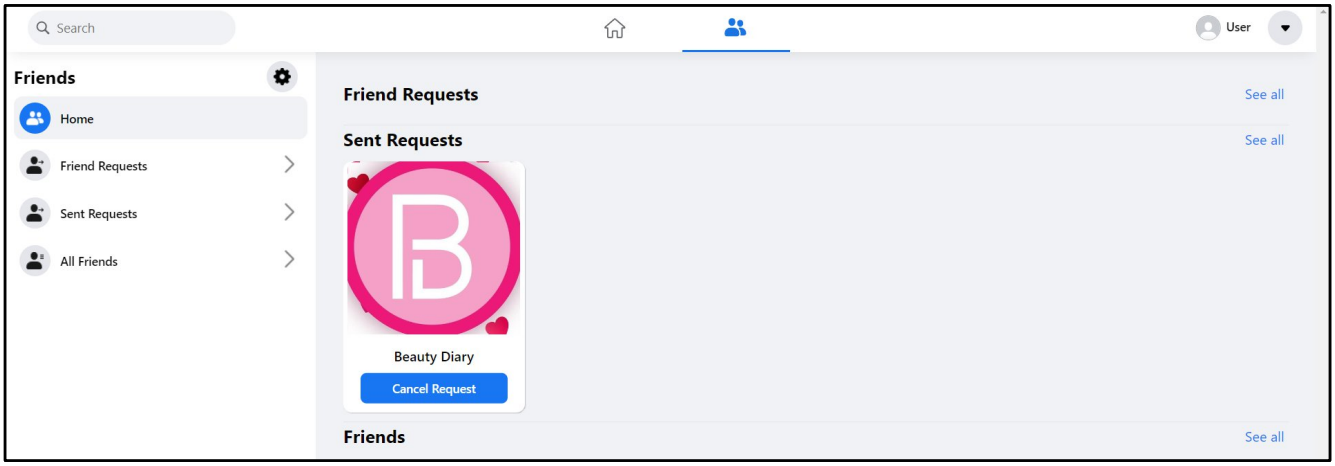

Рисунок 3.20 – Керування підписками (рисунок виконаний самостійно)

Функція додавання у друзі наведена на рисунку 3.21.

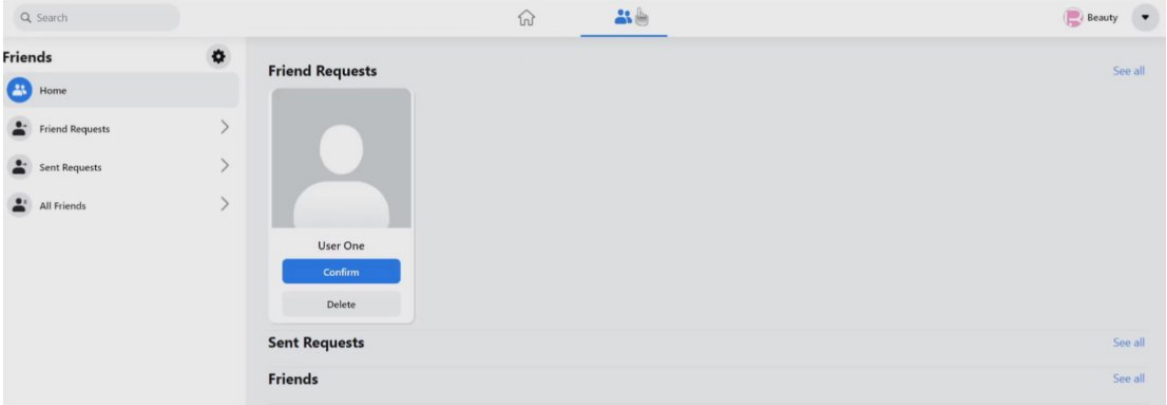

Рисунок 3.21 Додавання у друзі (рисунок виконаний самостійно)

Зручний інтерфейс для керування налаштуваннями сповіщень, включаючи опції підписки та відписки на різні типи сповіщень.

Центр сповіщень, тобто централізоване місце для перегляду всіх вхідних сповіщень, відсортованих за категоріями та мітками часу, наведено на рисунку 3.22.

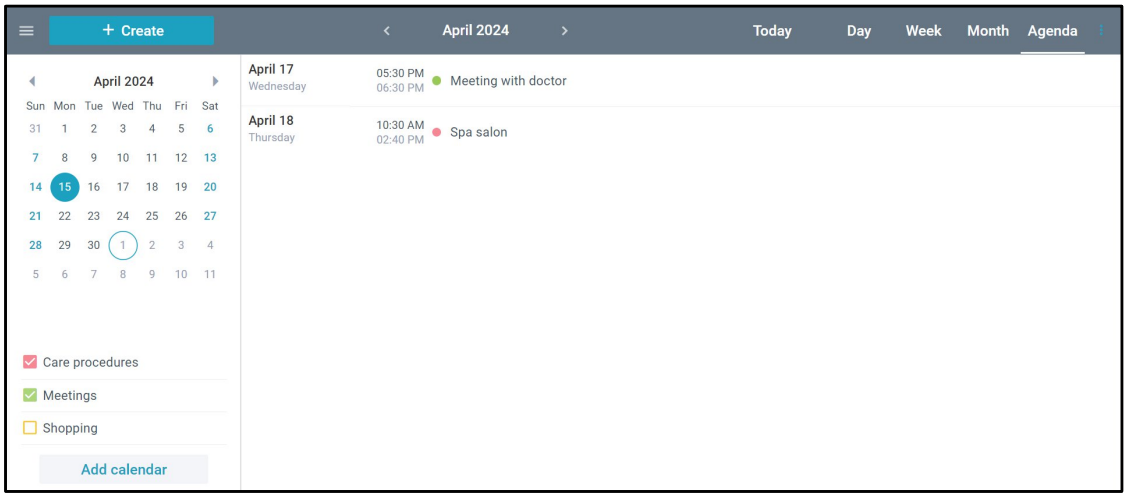

Рисунок 3.22 – Центр сповіщень (рисунок виконаний самостійно)

Мобільна адаптивність, тобто адаптивного макета, який легко адаптується до різних розмірів екрану та роздільної здатності наведено на рисунку 3.23.

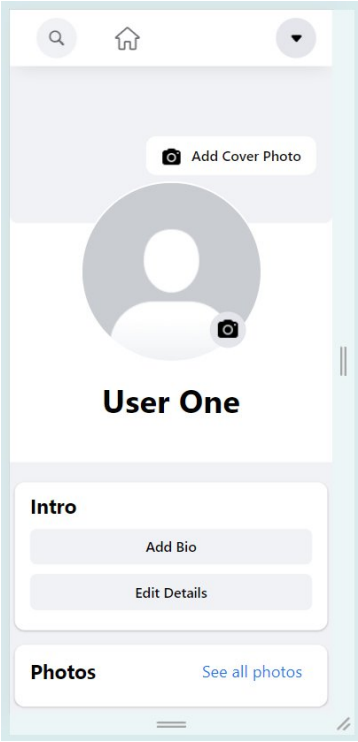

Рисунок 3.23 – Мобільна адаптивність (рисунок виконаний самостійно)

Пріоритетність основних функцій і контенту для мобільних користувачів, щоб забезпечити оптимальну зручність використання на невеликих пристроях.

Функція пошуку людей наведена на рисунку 3.24.

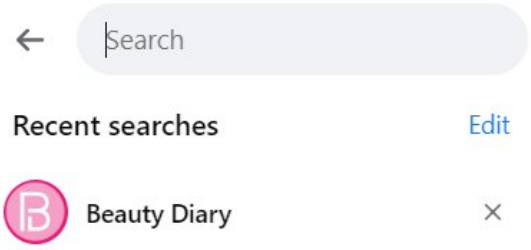

Рисунок 3.24 – Функція пошуку людей (рисунок виконаний самостійно)

Надійна функція пошуку, що дозволяє користувачам знаходити людей, які викладають інформативні статті та ресурси за категоріями, ключовими словами або тегами.

Презентація контенту з візуально привабливими фонами для посилення залучення користувачів наведена на рисунку 3.25.

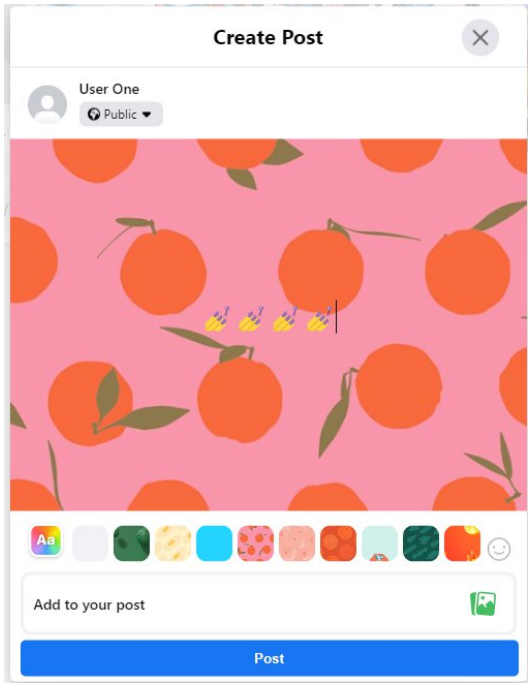

Рисунок 3.25 – Презентація контенту (рисунок виконаний самостійно)

Важливість UI / UX дизайну для "BeautyDiary":

– добре продуманий UI/UX підвищує залученість користувачів, забезпечуючи візуально привабливий і зручний інтерфейс. "BeautyDiary" може залучати та утримувати користувачів завдяки інтуїтивно зрозумілому та візуально приємному інтерфейсу, заохочуючи їх досліджувати платформу та взаємодіяти з її функціями;

– зосередившись на UX-дизайні, "BeautyDiary" може забезпечити користувачам легку навігацію різними розділами, доступ до інформації про косметичні процедури, поради по догляду за шкірою, навчальні посібники з макіяжу та багато іншого. Інтуїтивно зрозуміла навігація підвищує задоволеність користувачів і заохочує їх проводити більше часу на платформі;

– впроваджуючи принципи UX, такі як чітка структура контенту, ефективні робочі процеси та адаптивний дизайн, "BeautyDiary" може забезпечити покращений користувацький досвід. Користувачі повинні відчувати підтримку і мотивацію документувати свої косметичні процедури, ділитися досвідом і безперешкодно взаємодіяти з іншими користувачами;

– UI/UX дизайн може сприяти функціям персоналізації, дозволяючи користувачам налаштовувати свої профілі, вподобання та щоденники краси відповідно до їхніх унікальних потреб та вподобань. Такий рівень персоналізації підвищує

задоволеність користувачів та їхню лояльність до платформи;

– добре продуманий UI/UX враховує принципи доступності, гарантуючи, що "BeautyDiary" буде доступним для користувачів з різними здібностями. Це включає надання альтернативного тексту для зображень, забезпечення контрастності кольорів для зручності читання та реалізацію клавіатурної навігації для користувачів, які покладаються на допоміжні технології;

– враховуючи поширеність мобільних пристроїв, "BeautyDiary" повинен надавати пріоритет мобільній адаптивності у своєму UI/UX дизайні. Мобільний інтерфейс гарантує, що користувачі можуть безперешкодно отримувати доступ до платформи на різних пристроях, підвищуючи зручність і доступність.

Отже, UI/UX дизайн слугує основою успіху "BeautyDiary", створюючи орієнтовану на користувача платформу, яка сприяє залученню, задоволенню та довірі серед користувачів.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Веб-додаток "BeautyDiary" – це комплексна платформа, яка розроблена для задоволення потреб прихильників краси, що надає їм інструменти для документування своїх косметичних процедур у щоденнику, керування розкладом у календарі та участі в дискусіях та інформування про актуальні теми на форумі. Для досягнення цієї функціональності було прийнято кілька ключових програмних рішень, що використовують можливості React.js, Node.js, Express.js та MongoDB.

Вибір технологічного стеку для веб-додатку "BeautyDiary" був зумовлений кількома факторами, зокрема потребою в масштабованості, гнучкості та безшовній інтеграції між фронт-енд та бек-енд компонентами.

React.js був обраний в якості фронт-енд фреймворку через його компонентну архітектуру, яка дозволяє створювати багаторазові елементи інтерфейсу користувача. Компоненти в React інкапсулюють як структуру, так і поведінку елементів інтерфейсу, що полегшує підтримку та масштабування інтерфейсної кодової бази додатку [23]. За допомогою React.js можна створювати багаторазові компоненти, які можна використовувати в різних частинах програми, що призводить до чистоти коду та підвищення продуктивності розробників. Віртуальна DOM (Document Object Model) React дозволяє ефективно рендерити динамічний контент, мінімізуючи кількість необхідних маніпуляцій з DOM. Це призводить до кращої продуктивності та більш плавного користувацького досвіду, особливо для додатків зі складними та інтерактивними інтерфейсами.

Node.js та Express.js були обрані для створення внутрішньої інфраструктури веб-додатку BeautyDiary. Node.js – це середовище виконання, яке дозволяє запускати код JavaScript на стороні сервера, тоді як Express.js – це легкий веб-фреймворк для створення веб-додатків та API. Node.js дозволяє писати серверну логіку за допомогою JavaScript, забезпечуючи узгоджену мову та екосистему як для фронт-енд, так і для бек-енд розробки [24].

Express.js надає надійну систему маршрутизації для визначення кінцевих точок API та обробки HTTP-запитів. За допомогою Express.js можна легко створювати RESTful API для зв'язку з фронт-ендом і виконувати операції CRUD

(створення, читання, оновлення, видалення) над даними, що зберігаються у внутрішній базі даних [25].

MongoDB було обрано як рішення для внутрішньої бази даних завдяки її гнучкості, масштабованості та сумісності з JSON-подібними структурами документів. MongoDB зберігає дані у гнучких документах без схем, що робить її придатною для зберігання різноманітних типів даних, таких як профілі користувачів, записи в щоденнику та події календаря [26]. MongoDB розроблена для горизонтального масштабування між декількома вузлами, що дозволяє додатку "BeautyDiary" обробляти все більші обсяги даних і трафіку в міру його зростання. Завдяки таким функціям, як шардінг та набори реплік, MongoDB може забезпечити високу доступність та відмовостійкість, гарантуючи надійність додатку. Інтеграція MongoDB з JavaScript та Node.js спрощує процес розробки, дозволяючи використовувати узгоджену мову та набір інструментів у всьому стеку. Ця безшовна інтеграція впорядковує робочий процес розробки та дозволяє пришвидшити ітерації та розгортання нових функцій.

Створення детальної структури бек-енду передбачає організацію контролерів, помічників, проміжного програмного забезпечення, моделей і маршрутизаторів у масштабований і підтримуваний спосіб. Контролери керують логікою програми та взаємодіють з базою даних. Кожен контролер відповідає за певний набір функцій. Помічники містять утиліти, які допомагають у виконанні різних завдань у програмі. Проміжне програмне забезпечення перехоплює HTTP-запити і виконує завдання попередньої або подальшої обробки. Моделі визначають структуру та поведінку даних у додатку. Кожна модель представляє сутність даних і взаємодіє з базою даних. Маршрутизатори визначають кінцеві точки і направляють запити до відповідних контролерів. Ця внутрішня структура забезпечує міцний фундамент для створення масштабованого, підтримуваного та багатофункціонального веб-додатку. Вона організовує кодову базу в модульні компоненти, що полегшує її розуміння, налагодження та розширення. Крім того, вона забезпечує розподіл завдань і сприяє повторному використанню коду, що має важливе значення для довгострокової стійкості проекту.

Створення детальної структури фронт-енду передбачає організацію компонентів, даних, функцій, помічників, сторінок, редукторів, маршрутів і стилів таким чином, щоб сприяти повторному використанню коду, масштабованості та зручності обслуговування. Компоненти представляють собою багаторазові елементи інтерфейсу, які інкапсулюють власну логіку та стилістику. Керування даними включає обробку стану, отримання даних з API та керування локальним сховищем. Функції інкапсулюють логіку, яку можна повторно використовувати в різних компонентах і на різних сторінках. Помічники містять утиліти, які допомагають у виконанні різних завдань у програмі. Сторінки представляють різні види або розділи веб-сайту, що складаються з компонентів і даних. Редуктори визначають, як змінюється стан програми у відповідь на дії, надіслані до сховища Redux. Маршрутизатори визначають шляхи навігації та відображають URL-адреси на певні компоненти або сторінки. Таблиці стилів визначають візуальний вигляд і розташування елементів інтерфейсу на сайті. Ця детальна структура інтерфейсу забезпечує міцну основу для створення адаптивного, інтерактивного та зручного для користувача веб-додатку. Вона організовує код у модульні компоненти, розділяє проблеми між презентацією та логікою, а також забезпечує узгодженість і ремонтопридатність усієї кодової бази.

Однією з найважливіших в контролері користувача є функція реєстрації. Ця функція обробляє процес реєстрації нових користувачів у веб-додатку "BeautyDiary". Перевірка вхідних даних:

```
exports.register = async (req, res) => {
  try {
    const {
      first_name,
      last_name,
      email,
      password,
      username,
      bYear,
      bMonth,
      bDay,
      gender,
```

```
} = req.body
```

Витягуються реєстраційні дані користувача з тіла запиту, включаючи ім’я, прізвище, електронну пошту, пароль, ім’я користувача, рік народження, місяць народження, день народження та стать.

Перевірка адреси електронної пошти:

```
if (!validateEmail(email)) {
  return res.status(400).json({
    message: 'invalid email address',      })   }
```

Перевіряється, чи надана адреса електронної пошти має правильний формат за допомогою функції validateEmail. Якщо вона не дійсна, то повертає відповідь 400 з повідомленням про помилку.

Перевірка унікальності адреси:

```
const check = await User.findOne({ email })
if (check) {
  return res.status(400).json({
    message:
    'This email address already exists, try with a different email address',     })  }
```

Надходить запит до бази даних, щоб перевірити, чи існує ця адреса електронної пошти. Якщо вона існує, то повертає відповідь 400 з повідомленням про помилку.

Перевірка довжини імені, прізвища та пароля:

```
if (!validateLength(first_name, 3, 30)) {
  return res.status(400).json({
  message: 'first name must be between 3 and 30 characters.',})}
```

Перевіряється довжина імені, прізвища та пароля. Якщо довжина невірна, то повертає відповідь 400 з повідомленням про помилку.

Хешування пароля:

```
const cryptedPassword = await bcrypt.hash(password, 12)
```

Хешує пароль за допомогою bcrypt з коефіцієнтом 12 для безпечного зберігання його в базі даних.

Створення та перевірка імені користувача:

```
let tempUsername = first_name + last_name
let newUsername = await validateUsername(tempUsername)
```

Створюється тимчасове ім’я користувача на основі імені та прізвища. Перевіряється унікальність імені користувача за допомогою функції validateUsername.

Створення користувача:

```
const user = await new User({
  first_name,
  last_name,
  email,
  password: cryptedPassword,
  username: newUsername,
  bYear,
  bMonth,
  bDay,
  gender,
}).save()
```

Створюється новий документ користувача в базі даних з наданими реєстраційними даними користувача.

Генерація токенів підтвердження електронної пошти:

```
const emailVerificationToken = generateToken(
  { id: user._id.toString() },
  '30m' )
```

Генерується токен підтвердження електронної пошти за допомогою функції generateToken з терміном дії 30 хвилин.

Побудова URL-адреси перевірки:

```
const url = `${process.env.BASE_URL}/activate/${emailVerificationToken}`
```

Створюється URL-адреса перевірки, використовуючи згенерований токен перевірки електронної пошти та базову URL-адресу зі змінних оточення.

Відправлення листа з підтвердженням електронної пошти:

```
sendVerificationEmail(user.email, user.first_name, url)
```

Надсилається лист з підтвердженням електронної пошти на адресу користувача за допомогою функції sendVerificationEmail.

Генерація токенів автентифікації:

```
const token = generateToken({ id: user._id.toString() }, '7d')
```

Генерується маркер автентифікації для новозареєстрованого користувача з терміном дії 7 днів за допомогою функції generateToken.

Відповідь:

```
  res.send({
      id: user._id,
      username: user.username,
      picture: user.picture,
      first_name: user.first_name,
      last_name: user.last_name,
      token: token,
      verified: user.verified,
      message: 'Register Success !', })
```

Відповідає об’єктом JSON, що містить ідентифікатор користувача, ім’я користувача, зображення профілю, ім’я, прізвище, маркер автентифікації, статус перевірки та повідомлення про успіх.

Обробка помилок:

```
  } catch (error) {
    res.status(500).json({ message: error.message })  }}
```

Перехоплює будь-які помилки, що виникають під час процесу реєстрації, і повертає відповідь 500 з повідомленням про помилку.

Ця функція register інкапсулює основну логіку реєстрації користувачів у веб-додатку "BeautyDiary", забезпечуючи цілісність даних, безпеку та безперебійну роботу користувачів.

Однією з важливих частин коду є функція getAllPosts у контролері Post, яка отримує повідомлення від користувачів, за якими стежить поточний користувач, а також повідомлення, створені поточним користувачем. Ця функція демонструє ефективний підхід до отримання та об’єднання дописів з різних джерел, гарантуючи при цьому, що найсвіжіші дописи відображатимуться першими.

Отримання користувачів, які підписані на іншу людину:

```
exports.getAllPosts = async (req, res) => {
  try {
    const followingTemp = await
User.findById(req.user.id).select("following");
      const following = followingTemp.following;
```

Отримується список людней, за якими стежить поточний користувач. Це робиться за допомогою запиту до наступного поля документа поточного користувача у базі даних.

Створення масиву обіцянок для отримання дописів:

```
const promises = following.map((user) => {
  return Post.find({ user: user })
  .populate("user", "first_name last_name picture username cover")
    .populate("comments.commentBy", "first_name last_name picture username")
    .sort({ createdAt: -1 })
    .limit(10);});
```

Створюється масив обіцянок, де кожна обіцянка вибирає дописи користувача, за яким стежить поточний користувач. Функція map використовується для ітерації по масиву, і для кожного користувача створюється обіцянка для отримання його дописів.

Отримання власних дописів користувача:

```
const userPosts = await Post.find({ user: req.user.id })
 .populate("user", "first_name last_name picture username cover")
  .populate("comments.commentBy", "first_name last_name picture username")
  .sort({ createdAt: -1 })
  .limit(10);
```

Отримуються дописи, створені поточним користувачем. Це робиться за допомогою запиту до поля user моделі Post з ідентифікатором поточного користувача.

Об’єднання та сортування дописів:

```
const allPosts = [...followingPosts, ...userPosts];
allPosts.sort((a, b) => b.createdAt - a.createdAt);
```

Об’єднує дописи, отримані від користувачів, яких ви відстежили, і поточного користувача в один масив. Потім масив буде відсортовано за спаданням на основі поля createdAt у дописах так, щоб найновіші дописи з’являлися першими.

Відповідь:

```
res.json(allPosts);
```

Відповідає масивом JSON, що містить об’єднані та відсортовані дописи, які

представляють дописи від підписаних користувачів та поточного користувача.

Обробка помилок:

```
  } catch (error) {
    return res.status(500).json({ message: error.message });  }};
```

Перехоплює будь-які помилки, що виникають під час процесу, і повертає статус 500 з повідомленням про помилку у форматі JSON.

Контролер Post відіграє важливу роль у полегшенні взаємодії з дописами в додатку BeautyDiary, надаючи користувачам можливість ділитися, переглядати, коментувати та зберігати контент, пов’язаний з красою. Надійна обробка помилок у контролері гарантує, що будь-які проблеми, які виникають під час операцій, пов’язаних з публікаціями, будуть належним чином вирішені, підтримуючи надійність і стабільність роботи програми.

Контролер React керує реакціями, пов’язаними з публікаціями в додатку. Він дозволяє користувачам реагувати на пости різними емоціями, такими як like, love, haha, sad, wow і angry. Найбільш значущою є функція getReacts, яка отримує реакції на певний пост і агрегує їх у структурований формат.

Отримання реакції на публікацію:

```
exports.getReacts = async (req, res) => {
  try {
    const reactsArray = await React.find({ postRef: req.params.id });
```

Функція починається із запиту до бази даних, щоб знайти всі реакції, пов’язані з вказаним ідентифікатором допису (req.params.id).

Групування реакцій:

```
    const newReacts = reactsArray.reduce((group, react) => {
      let key = react["react"];
      group[key] = group[key] || [];
      group[key].push(react);
      return group;
    }, {});
```

Далі функція групує реакції на основі їхнього типу (like, love, haha, sad, wow, angry). Вона використовує метод reduce для ітерації над масивом реакцій і створення об’єкта, де кожен ключ представляє тип реакції, а його значення - масив

реакцій цього типу.

Будування масиву реакцій:

```
const reacts = [
    {
      react: "like",
      count: newReacts.like ? newReacts.like.length : 0,
    },
    {
      react: "love",
      count: newReacts.love ? newReacts.love.length : 0,
    },
    {
      react: "haha",
      count: newReacts.haha ? newReacts.haha.length : 0, },
    {
      react: "sad",
      count: newReacts.sad ? newReacts.sad.length : 0,
    },
    {
      react: "wow",
      count: newReacts.wow ? newReacts.wow.length : 0,
    },
    {
      react: "angry",
      count: newReacts.angry ? newReacts.angry.length : 0,
    },
  ];
```

Після групування реакцій функція створює масив об’єктів, що містить кожен тип реакції разом з кількістю реакцій для цього типу. Якщо немає реакцій певного типу, то за замовчуванням кількість реакцій дорівнює нулю.

Перевірка реакції користувача та збережений статус:

```
const check = await React.findOne({
  postRef: req.params.id,
  reactBy: req.user.id,
});
const user = await User.findById(req.user.id);
const checkSaved = user?.savedPosts.find(
```

```
    (x) => x.post.toString() === req.params.id    );
```

Функція перевіряє, чи поточний користувач відреагував на повідомлення. Вона також перевіряє, чи користувач зберіг допис для подальшого перегляду.

Повернення відповіді:

```
res.json({
  reacts,
  check: check?.react,
  total: reactsArray.length,
  checkSaved: checkSaved ? true : false,    });
```

Функція повертає відповідь у форматі JSON, що містить побудований масив реакцій (reacts), реакцію користувача (check.react), загальну кількість реакцій на допис (total) та логічне значення, що вказує на те, чи було збережено допис користувачем (checkSaved).

Обробка помилок:

```
} catch (error) {
  return res.status(500).json({ message: error.message });  }};
```

Перехоплює будь-які помилки, що виникають під час процесу, і повертає статус 500 з повідомленням про помилку у форматі JSON.

Контролер React ефективно керує реакціями на публікації, надає інформацію про взаємодію з користувачем і забезпечує надійну обробку помилок для безперебійної роботи користувача.

Отже, вибір React.js для фронт-енду, Node.js та Express.js для бек-енду та MongoDB для бек-енду бази даних був зумовлений міркуваннями гнучкості, масштабованості, продуктивності та продуктивності. Разом ці технології утворюють потужний і згуртований стек, який дозволяє веб-додатку "BeautyDiary" забезпечувати безперебійний та цікавий користувацький досвід для прихильників краси по всьому світу.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування розробленого програмного забезпечення передбачає перевірку того, що всі функції та можливості працюють так, як очікується, і відповідають вимогам, визначеним для кожного компонента. Для цього буду використовувати комбінацію методів автоматизованого тестування, включаючи модульне та інтеграційне тестування, щоб систематично перевіряти поведінку та продуктивність програмного забезпечення.

Модульне тестування фокусується на перевірці функціональності окремих модулів або компонентів програмного забезпечення ізольовано. Кожен блок, такий як функція або метод, тестується незалежно, щоб переконатися, що він видає очікуваний результат при різних вхідних даних [27].

Тестування записів у щоденнику гарантує точний запис косметичних процедур та іншої пов’язаної з ними інформації (див. рис. 5.1).

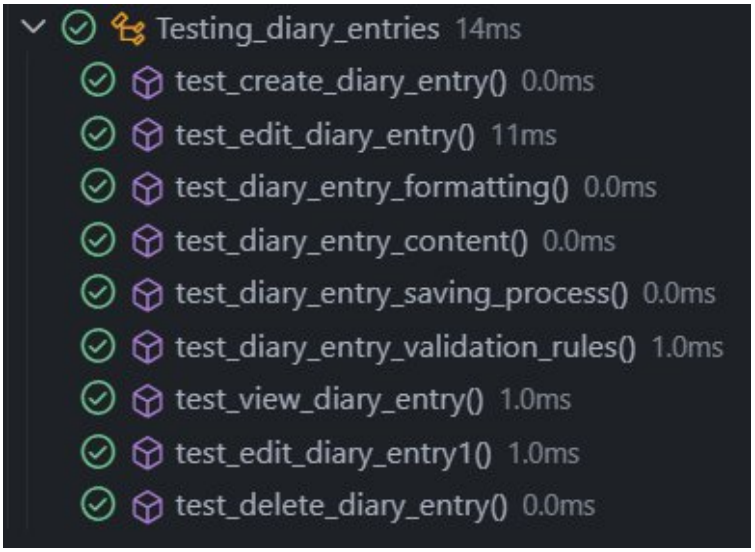

Рисунок 5.1 – Тестування записів у щоденнику (рисунок виконаний самостійно)

Функція test_create_diary_entry тестує створення записів у щоденнику, щоб переконатися, що процес працює коректно, включаючи форматування, введення вмісту та збереження. Функція test_edit_diary_entry перевіряє, чи можуть користувачі редагувати існуючі записи щоденника без помилок і чи правильно зберігаються зміни.

Функція test_diary_entry_formatting перевіряє можливості форматування

записів у щоденнику, переконуючись, що користувачі можуть форматувати текст (наприклад, напівжирний, курсив, списки) відповідно до призначення. Функція test_diary_entry_content перевіряє, що різні типи контенту, такі як косметичні процедури, догляд за шкірою, волоссям та макіяж, можуть бути точно описані в записах щоденника. Функція test_diary_entry_saving_process перевіряє, чи правильно функціонує процес збереження записів у щоденнику, запобігаючи втраті даних та підтримуючи їхню послідовність. Функція test_diary_entry_validation_rules тестує будь-які правила валідації, що застосовуються до записів у щоденнику, наприклад, обмеження мінімальної/максимальної довжини або вимога заповнення певних полів. Функція test_view_diary_entry перевіряє, чи можуть користувачі переглядати свої записи у щоденнику, як очікувалося, і чи точно відображається їхній вміст. Функція test_edit_diary_entry1 тестує функціональність редагування користувачами своїх записів у щоденнику, підтверджуючи, що зміни застосовані правильно і збережені. Функція test_delete_diary_entry перевіряє, чи можуть користувачі видаляти свої записи в щоденнику без проблем, і чи функціонує процес видалення належним чином.

Тестування інтерактивного календаря забезпечує належне планування та нагадування про косметичні процедури та зустрічі (див. рис. 5.2).

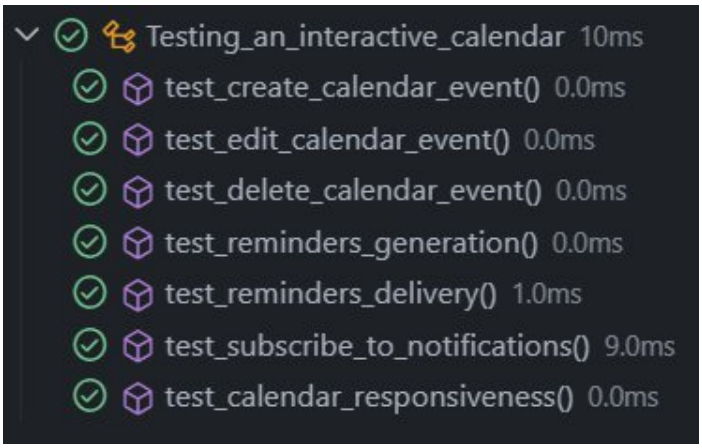

Рисунок 5.2 – Тестування інтерактивного календаря (рисунок виконаний самостійно)

Функція test_create_calendar_event тестує створення подій календаря,

переконуючись, що користувачі можуть додавати події з точними деталями і що вони зберігаються належним чином. Функція test_edit_calendar_event перевіряє, чи можуть користувачі редагувати існуючі події календаря без помилок і чи правильно відображаються зміни. Функція test_delete_calendar_event тестує функціонал видалення подій календаря, переконуючись, що події коректно видаляються з календаря. Функція test_reminders_generation перевіряє коректність генерації нагадувань для майбутніх подій на основі налаштувань користувача та деталей події. Функція test_reminders_delivery перевіряє доставку нагадувань користувачам через обрані канали сповіщень (наприклад, електронну пошту, сповіщення в застосунку). Функція test_subscribe_to_notifications перевіряє, що користувачі можуть підписатися на сповіщення про нові записи в щоденнику, майбутні події тощо, і що вони отримують сповіщення, як очікується. Функція test_calendar_responsiveness перевіряє адаптивність інтерфейсу календаря на різних пристроях, гарантуючи, що користувачі можуть легко взаємодіяти з ним незалежно від розміру екрану або типу пристрою.

Для форуму тестування забезпечує безперебійну взаємодію з користувачами та цілісність даних на платформі спільноти (див. рис. 5.3).

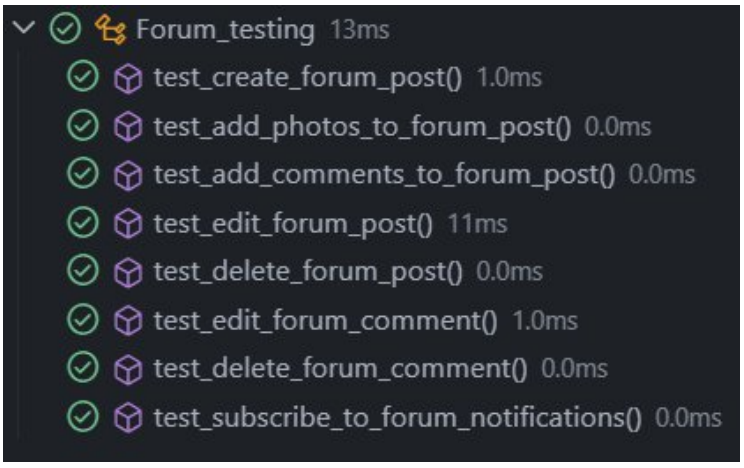

Рисунок 5.3 – Тестування форуму(рисунок виконаний самостійно)

Функція test_create_forum_post тестує створення повідомлень на форумі, включаючи додавання фотографій і коментарів, щоб переконатися, що вони відображаються коректно. Функція test_add_photos_to_forum_post перевіряє, чи можуть користувачі додавати фотографії до повідомлень форуму без проблем і чи

правильно вони відображаються. Функція test_add_comments_to_forum_post тестує функціональність додавання коментарів до повідомлень на форумі, переконуючись, що коментарі відображаються коректно і пов’язані з відповідним повідомленням. Функція test_edit_forum_post перевіряє, що користувачі можуть редагувати свої повідомлення на форумі без помилок і що зміни зберігаються належним чином. Функція test_delete_forum_post тестує функціонал видалення повідомлень на форумі, перевіряючи, що повідомлення коректно видаляються з форуму. Функція test_edit_forum_comment перевіряє, чи можуть користувачі без проблем редагувати свої коментарі на форумі, і чи правильно відображаються зміни. Функція test_delete_forum_comment перевіряє, що користувачі можуть видаляти свої коментарі на форумі без помилок і що процес видалення функціонує коректно. Функція test_subscribe_to_forum_notifications перевіряє функціональність підписки користувачів на сповіщення про нові повідомлення або відповіді на форумі, гарантуючи, що вони отримують сповіщення, як очікувалося.

Тестування реєстрації та профілів користувачів гарантує, що процес створення облікового запису працює безперебійно, облікові записи створюються з унікальними обліковими даними, інформація в профілі може бути точно оновлена, а налаштування конфіденційності є ефективними. Також перевіряється, чи можуть користувачі без проблем додавати та оновлювати фотографії профілю (див. рис. 5.4).

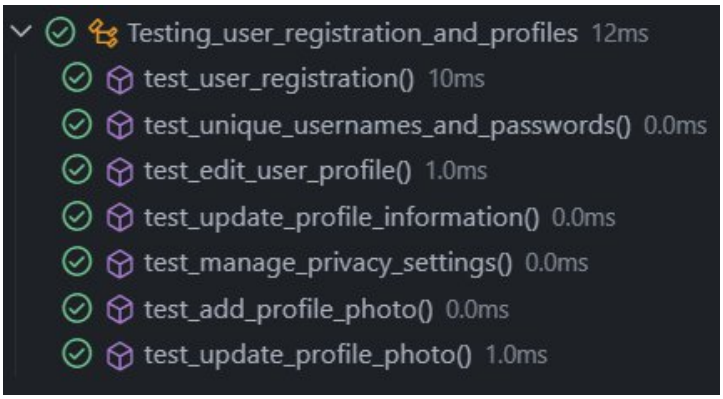

Рисунок 5.4 – Тестування реєстрації та профілів користувачів (рисунок виконаний самостійно)

Функція test_user_registration тестує процес реєстрації користувачів,

переконуючись, що користувачі можуть створювати облікові записи з унікальними іменами користувачів та паролями. Функція test_unique_usernames_and_passwords перевіряє, чи система забезпечує унікальність імен користувачів та паролів, щоб запобігти їхньому повторенню та підвищити безпеку. Функція test_edit_user_profile тестує функціонал для редагування користувачами своїх профілів, включаючи оновлення інформації та керування налаштуваннями конфіденційності. Функція test_update_profile_information перевіряє, чи можуть користувачі оновлювати інформацію свого профілю без будь-яких проблем і чи правильно зберігаються зміни. Функція test_manage_privacy_settings перевіряє, що користувачі можуть ефективно керувати своїми налаштуваннями конфіденційності, контролюючи, хто може переглядати інформацію про їхній профіль та дії. Функція test_add_profile_photo тестує функціонал додавання фотографій профілю, перевіряючи правильність завантаження та відображення фотографій у профілі. Функція test_update_profile_photo перевіряє, чи можуть користувачі оновлювати фотографії профілю без помилок і чи правильно відображаються зміни.

Модульне тестування, проведене для записів у щоденнику, інтерактивного календаря, форуму, реєстрації та профілів користувачів, було дуже успішним. Кожен компонент пройшов ретельне тестування, щоб забезпечити його функціональність, надійність та зручність для користувача.

Щодо записів у щоденнику, тестування підтвердило, що користувачі можуть створювати, редагувати та видаляти записи безперешкодно, з точним форматуванням і змістом. Правила валідації записів у щоденнику, такі як мінімальна/максимальна довжина та обов’язкові поля, також були ретельно перевірені, щоб гарантувати цілісність даних. Аналогічно, інтерактивний календар був ретельно протестований для перевірки створення, редагування та видалення подій, а також генерації та доставки нагадувань. Також було підтверджено адаптивність інтерфейсу календаря на різних пристроях, що забезпечило узгодженість взаємодії з користувачем. У випадку з форумом тестування засвідчило, що користувачі можуть створювати дописи, додавати фотографії та коментарі, а також ефективно керувати своїм контентом. Інтеграція з профілями

користувачів була успішно підтверджена, гарантуючи, що інформація про користувачів правильно відображається на форумі. Що стосується реєстрації та профілів користувачів, тестування підтвердило безперебійне функціонування процесу реєстрації, що дозволяє користувачам створювати облікові записи з унікальними обліковими даними. Також було перевірено, що користувачі можуть без проблем оновлювати свої профілі та керувати налаштуваннями конфіденційності.

Загалом, успішне модульне тестування цих компонентів свідчить про надійність та якість програмного забезпечення, закладаючи міцний фундамент для подальшого інтеграційного тестування та можливого розгортання.

Інтеграційне тестування фокусується на перевірці взаємодії між різними компонентами або модулями програмного забезпечення. Воно гарантує, що інтегрована система функціонує правильно в цілому і що дані безперебійно передаються між різними частинами [28].

Тестування інтеграції записів у щоденнику та календарі гарантує, що події, створені в щоденнику, будуть точно відображені в календарі і навпаки. Воно перевіряє синхронізацію нагадувань про події між двома компонентами, забезпечуючи узгодженість і надійність в управлінні розкладом і зустрічами (див. рис. 5.5).

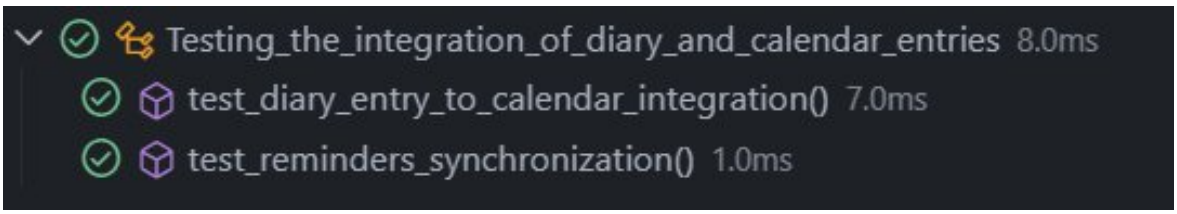

Рисунок 5.5 – Тестування інтеграції записів у щоденнику та календарі (рисунок виконаний самостійно)

Функція test_diary_entry_to_calendar_integration тестує інтеграцію між записами щоденника і календарем, гарантуючи, що події, створені в щоденнику, точно відображаються в календарі і навпаки. Функція test_reminders_synchronization перевіряє синхронізацію нагадувань про події між компонентами щоденника і календаря, забезпечуючи узгодженість і точність

сповіщення користувачів про майбутні події.

Тестування інтеграції між форумом і профілями користувачів гарантує, що інформація користувачів, наприклад, фотографії профілю та імена користувачів, правильно відображається в повідомленнях і коментарях на форумі. Воно підтверджує, що користувачі можуть безперешкодно взаємодіяти між своїми профілями і діяльністю на форумі, підтримуючи єдиний користувацький досвід у різних розділах платформи (див. рис. 5.6).

Рисунок 5.6 – Тестування інтеграції між форумом і профілями користувачів (рисунок виконаний самостійно)

Функція test_forum_profile_integration тестує інтеграцію між профілями користувачів і форумом, гарантуючи, що інформація користувача, така як фотографії профілю та імена користувачів, коректно відображається в повідомленнях і коментарях на форумі. Функція test_seamless_interaction_profiles_forum перевіряє, чи можуть користувачі безперешкодно взаємодіяти між своїми профілями та діяльністю на форумі, наприклад, публікувати повідомлення, коментувати та переглядати дискусії.

Тестування інтеграції зі сповіщеннями гарантує, що система сповіщень функціонує належним чином для різних функцій. Перевіряє, чи користувачі отримують своєчасні сповіщення про нові записи в щоденнику, майбутні події, активність на форумі або будь-які інші важливі оновлення, підвищуючи залученість користувачів та інформуючи їх про важливі події чи взаємодії (див. рис. 5.7).

Функція test_notification_integration_diary_entries тестує інтеграцію функціоналу сповіщень для записів у щоденнику, гарантуючи, що користувачі отримують своєчасні сповіщення про нові записи в щоденнику. Функція test_notification_integration_calendar_events перевіряє, чи отримують користувачі

сповіщення про майбутні події, гарантуючи, що вони поінформовані про заплановані заходи. Функція test_notification_integration_forum_activities тестує інтеграцію функціоналу сповіщень для активностей на форумі, таких як нові повідомлення або відповіді, гарантуючи, що користувачі залишаються в курсі обговорень та взаємодій.

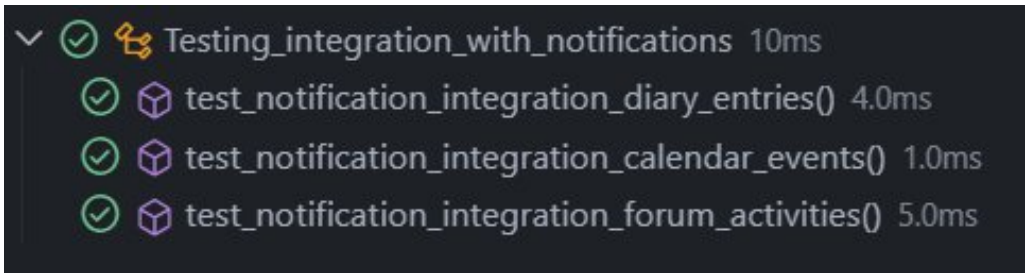

Рисунок 5.7 – Тестування інтеграції зі сповіщеннями (рисунок виконаний самостійно)

Тестування інтеграції, проведене для інтеграції між записами щоденника і календарем, інтеграції між форумом і профілями користувачів, а також інтеграції з повідомленнями, було визнано успішним. Кожен аспект інтеграції пройшов ретельну перевірку, щоб забезпечити безперебійну сумісність і функціональність всієї системи.

Інтеграція між записами в щоденнику та календарем була ретельно перевірена, підтвердивши, що події, створені в щоденнику, точно відображаються в календарі і навпаки. Синхронізація нагадувань про події між цими компонентами також була успішно протестована, завдяки чому користувачі отримують своєчасні сповіщення про заплановані заходи. Була ретельно перевірена інтеграція між форумом і профілями користувачів, щоб переконатися, що інформація користувачів, така як фотографії профілю та імена користувачів, безперешкодно інтегрується з діяльністю на форумі. Цей етап тестування підтвердив, що користувачі можуть без особливих зусиль взаємодіяти з функціями форуму, а інформація в їхніх профілях залишається незмінною і правильно відображається. Інтеграція зі сповіщеннями була ретельно протестована, щоб гарантувати, що користувачі отримують своєчасні та релевантні сповіщення про різні функціональні можливості системи, включаючи нові записи в щоденнику,

майбутні події та активності на форумі. Це тестування гарантувало, що система сповіщень функціонує надійно, підвищуючи залученість користувачів і покращуючи їхній досвід.

Таким чином, успішне інтеграційне тестування цих компонентів підкреслює надійність та злагодженість програмної системи, що відкриває шлях до її розгортання з упевненістю в її функціональності та продуктивності.

Отже, комплексне автоматичне тестування, проведене для веб-додатку "BeautyDiary", продемонструвало його стійкість, надійність і готовність до розгортання. Завдяки ретельному автоматизованому тестуванню, що включало як модульне, так і інтеграційне тестування, кожен аспект програмного забезпечення був ретельно перевірений, щоб забезпечити його функціональність, точність та безперебійну сумісність.

Модульне тестування забезпечило детальну перевірку окремих компонентів програми, перевіряючи, що кожна функція працює за призначенням і відповідає встановленим вимогам. Такий підхід до тестування сприяв підвищенню загальної стабільності та якості програмного забезпечення.

Тестування інтеграції було зосереджене на перевірці взаємодії між різними модулями та функціями додатку, гарантуючи, що вони гармонійно працюють разом, забезпечуючи цілісний користувацький досвід. Перевіривши інтеграцію записів у щоденнику з календарем, форуму з профілями користувачів і сповіщень між різними функціями, процес тестування підтвердив безперебійний потік даних і взаємодію в додатку.

В результаті, програмне забезпечення готове до виконання свого призначення – надання користувачам безперебійної платформи для управління процедурами краси, зустрічами та взаємодією у зручний та надійний спосіб.

ВИСНОВКИ

У результаті кваліфікаційної роботи бакалавра було створено універсальний веб-додаток "Щоденник краси", що охоплює функції щоденника, календаря та форуму, враховуючи зростаючий попит на цифрові інструменти для оптимізації б’юті-процедур.

Аналіз предметної області надав цінну інформацію про вподобання користувачів, ринкові тенденції та існуючі програмні рішення. Цей аналіз став основою для формулювання цілей проекту та вимог до функціональності.

Були визначені загальні больові точки та обмеження існуючих інструментів управління красою. Ці висновки лягли в основу концептуалізації рішень для ефективного задоволення потреб користувачів.

Були сформульовані детальні вимоги до програмної системи "Щоденник краси". Ці вимоги охоплювали функціональні специфікації, рекомендації щодо дизайну інтерфейсу користувача, очікування щодо продуктивності та міркування щодо сумісності.

Структура зберігання даних була розроблена для ефективного зберігання та пошуку записів у щоденнику, подій календаря, повідомлень на форумі, профілів користувачів та сповіщень. Міркування щодо цілісності, безпеки та масштабованості даних були першочерговими в цьому процесі.

Бек-енд, що використовує передові технології, такі як Node.js, Express та MongoDB, разом з фронт-енд фреймворком React.js, є свідченням масштабованості, гнучкості та надійності веб-додатку. Завдяки ретельному аналізу була розроблена клієнт-серверна архітектура, яка забезпечує безперебійну взаємодію за допомогою HTTP-протоколів. Використання React.js на фронт-енді ще більше покращує користувацький досвід, створюючи динамічні та чуйні інтерфейси, забезпечуючи плавну навігацію та взаємодію між записами у щоденнику, подіями календаря та обговореннями на форумі.

Інтеграційне тестування забезпечило бездоганну функціональність за різних сценаріїв, а модульне тестування ще більше зміцнило впевненість у продуктивності та надійності системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Day One Journal App | Your Journal For Life / URL: https://dayoneapp.com/ (дата звернення: 13.04.2024).

2. Diarium: Cross-platform diary & journal app / URL: https://diariumapp.com/ (дата звернення: 14.04.2024).

3. Grid Diary: The simplest way to get started with keeping a diary / URL: https://griddiaryapp.com/ (дата звернення: 15.04.2024).

4. Dabble Me — Your private journal / URL: https://dabble.me/ (дата звернення: 16.04.2024).

5. My Diary - Journal, Diary, Daily Journal with Lock для Android /URL: https://my-diary-journal-diary-daily-journal-with-lock.ru.uptodown.com/android (дата звернення: 17.04.2024).

6. MSKD – Skincare Diary on the App Store / URL: https://apps.apple.com/us/app/mskd-skincare-diary/id1514439739?platform=iphone (дата звернення: 17.04.2024).

7. Miles R., Hamilton K. Learning UML 2.0: A Pragmatic Introduction to UML, 1st Edition. – Sebastopol: O'Reilly Media, 2006. – 283 p.

8. Cheesman J., Daniels J. Uml Components: A Simple Process for Specifying Component-Based Software, 1st Edition. – Boston: Addison-Wesley Professional, 2000. – 176 p.

9. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition. – Print2print, 2016. – 208 p.

10. Rappin N. Modern Front-End Development for Rails: Hotwire, Stimulus, Turbo, and React, 2nd Edition. – Pragmatic Bookshelf, 2022. – 410 p.

11. Wong C. HTTP Pocket Reference: Hypertext Transfer Protocol, 1st Edition. – Sebastopol: O'Reilly Media, 2000. – 80 p.

12. Rumbaug J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual, 2nd Edition. – Boston: Addison-Wesley Professional, 2004. – 721 p.

13. Seidl M., Scholz M., Huemer C., Kappel G. UML @ Classroom: An Introduction to Object-Oriented Modeling (Undergraduate Topics in Computer Science). – Berlin: Springer, 2015. – 218 p.

14. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd Edition. – London: Pearson, 2004. – 736 p.

15. Goodyear M. Enterprise System Architectures: Building Client Server and Web Based Systems, 1st Edition. – Boca Raton: CRC Press, 2017. – 1543 p.

16. Blokdyk G. Client Server Architecture A Complete Guide - 2020 Edition. –5STARCooks, 2020. – 308 p.

17. Telang T. Domain Name Server (DNS) Fundamentals | Exploring Traceroute, DNS Attacks and Beyond: Demystifying Domain names | DNS Performance and Security | Guide for Network Administrators & Systems Engineers. – Lets Practice Academy, 2023. – 157 p.

18. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage, 3rd Edition. – Sebastopol: O'Reilly Media, 2019. – 511 p.

19. Vohra D. Pro MongoDB Development, 1st ed. Edition. – Luxemburg: Apress, 2015. – 506 p.

20. Bassett L. Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON, 1st Edition. – Sebastopol: O'Reilly Media, 2015. – 124 p.

21. MacDonald D. Practical UI Patterns for Design Systems: Fast-Track Interaction Design for a Seamless User Experience, 1st ed. Edition. – Luxemburg: Apress, 2019. – 318 p.

22. Pereyra I. Universal Principles of UX: 100 Timeless Strategies to Create Positive Interactions between People and Technology (Volume 4) (Rockport Universal, 4). – Beverly: Rockport Publishers, 2023. – 224 p.

23. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps, 2nd Edition. – Sebastopol: O'Reilly Media, 2020. – 307 p.

24. Hunter II T. Distributed Systems with Node.js: Building Enterprise-Ready

Backend Services, 1st Edition. – Sebastopol: O'Reilly Media, 2020. – 377 p.

25. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack, 2nd Edition. – Sebastopol: O'Reilly Media, 2019. – 343 p.

26. Aleksendric M., Borucki A., Domingues L. Mastering MongoDB 7.0 - Fourth Edition: Achieve data excellence by unlocking the full potential of MongoDB, 4th ed. Edition. – Birmingham: Packt Publishing, 2024. – 434 p.

27. Riedel H. DSAT Prep Reading and Writing Advanced Module Practice Tests: Volume 1. – Independently published, 2024. – 148 p.

28. Sambamurthy M. Test Automation Engineering Handbook: Learn and implement techniques for building robust test automation frameworks, 1st Edition. – Birmingham: Packt Publishing, 2023. – 276 p.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту

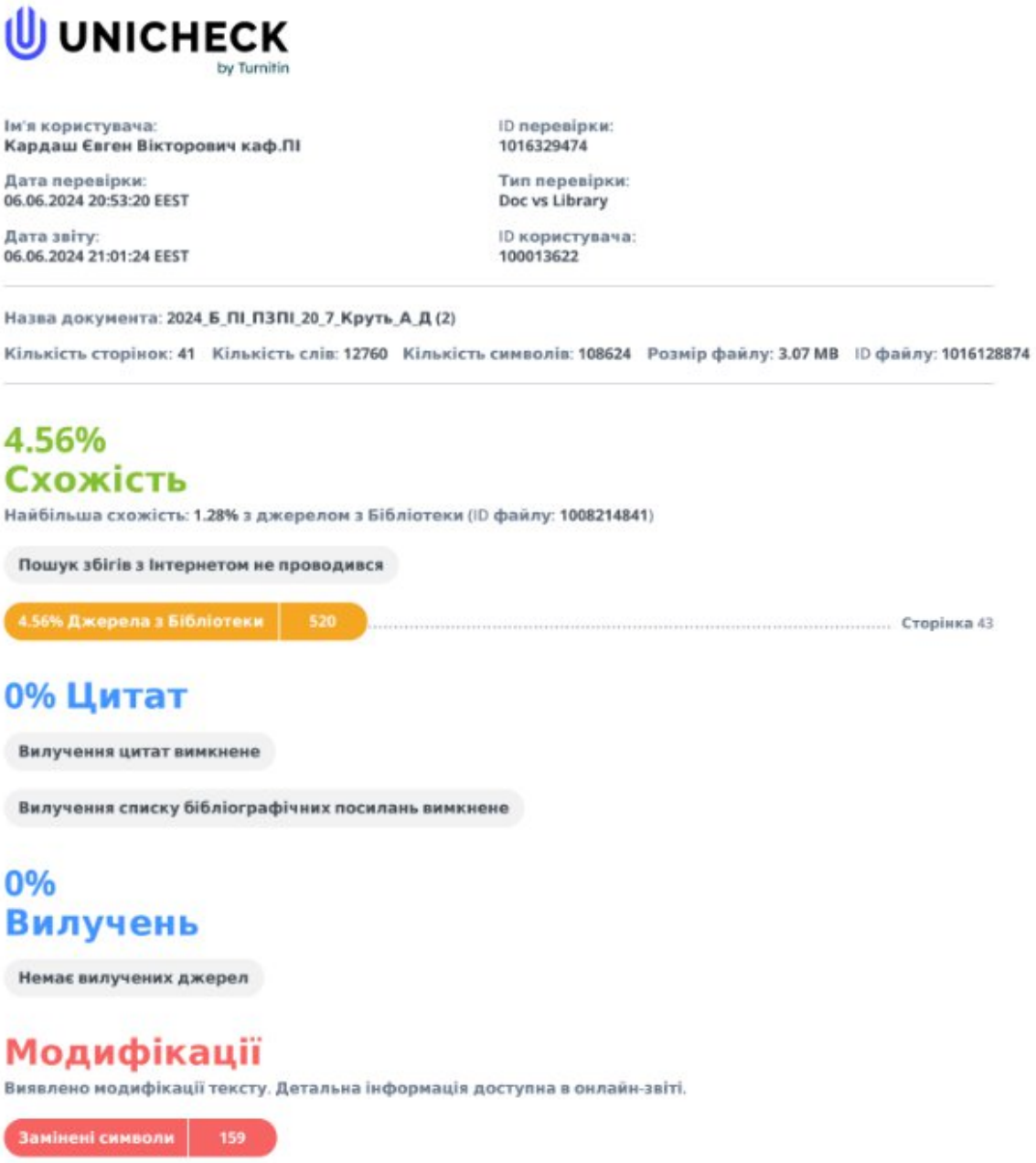

Рисунок А.1 - Звіт результатів перевірки на унікальність тексту (рисунок виконаний самостійно)

ДОДАТОК Б

Слайди презентації

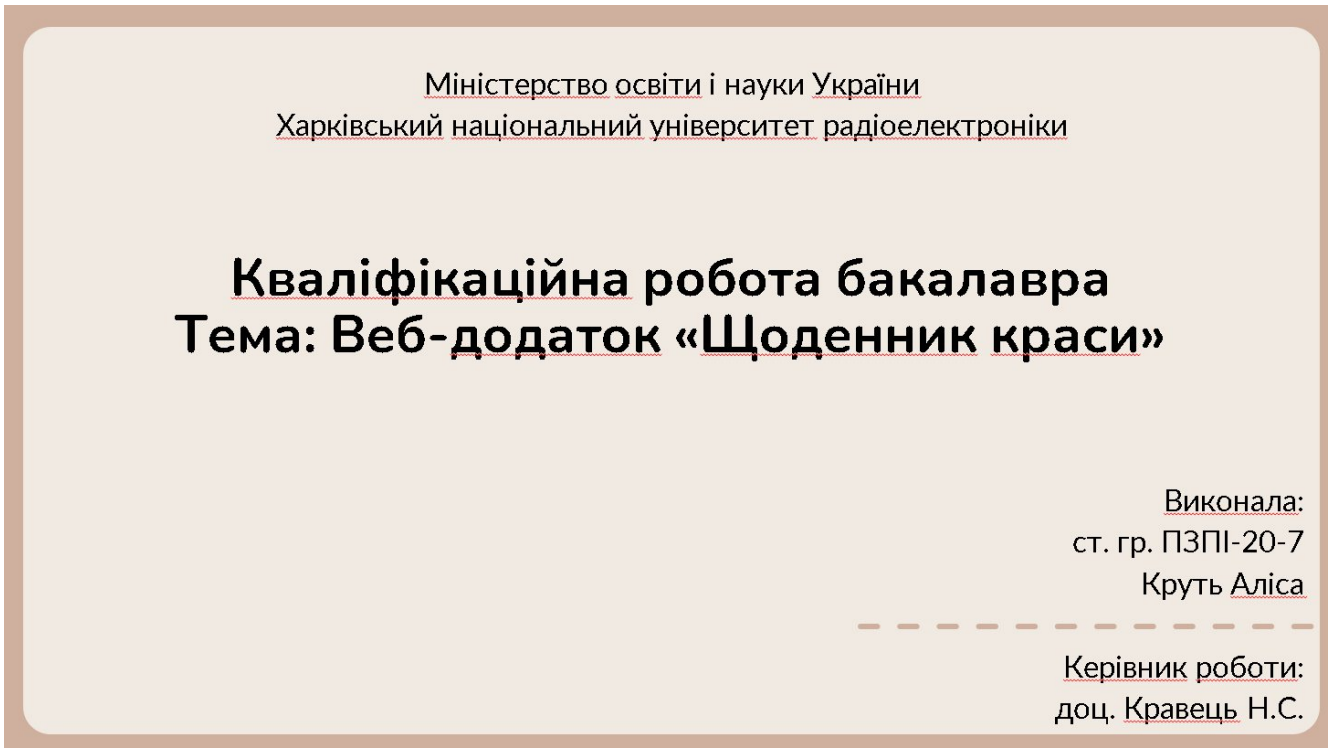

Рисунок Б.1 – Слайд презентації 1 (рисунок виконаний самостійно)

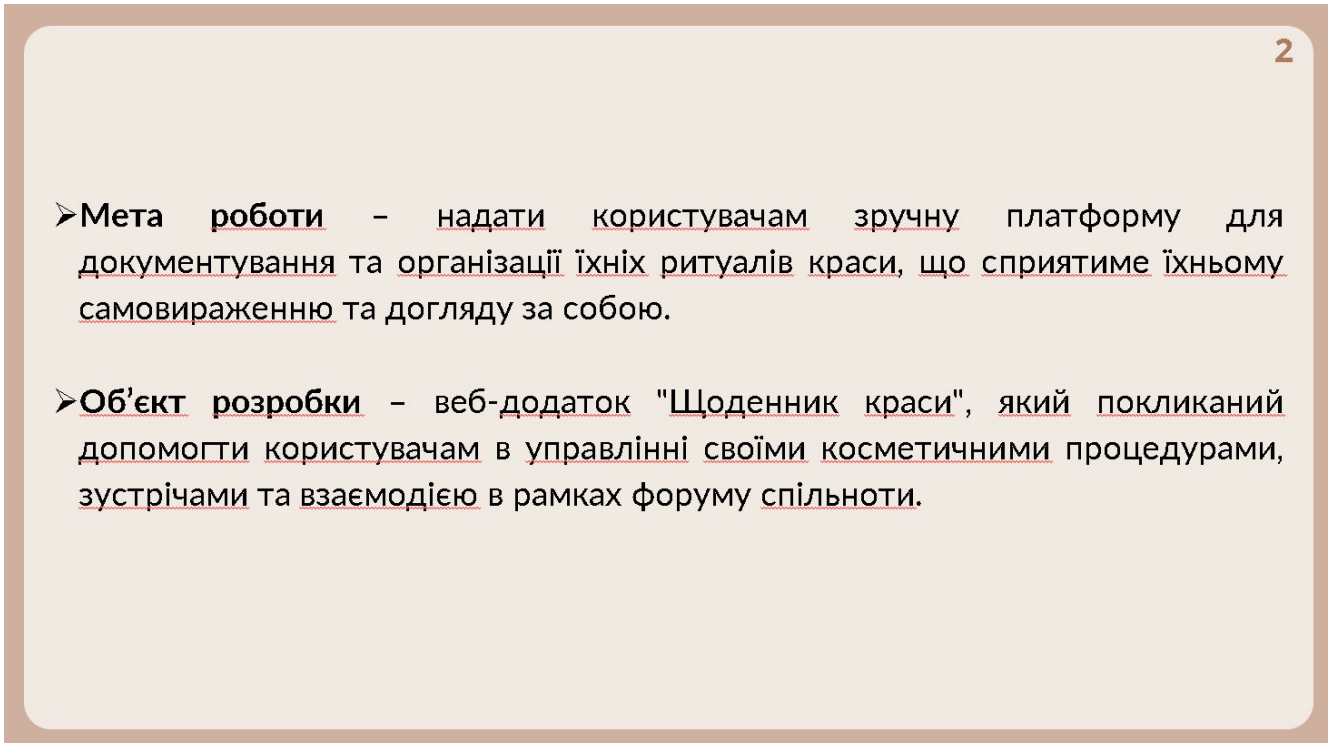

Рисунок Б.2 – Слайд презентації 2 (рисунок виконаний самостійно)

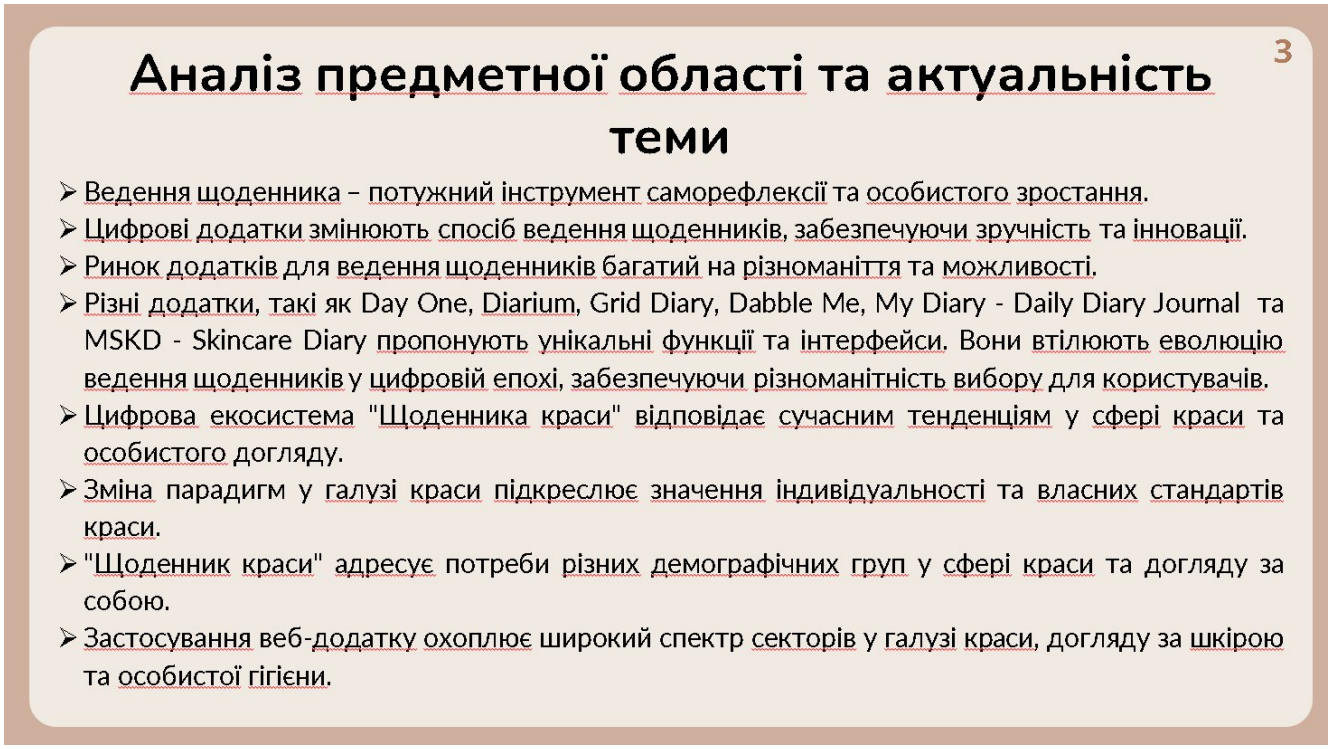

Рисунок Б.3 – Слайд презентації 3 (рисунок виконаний самостійно)

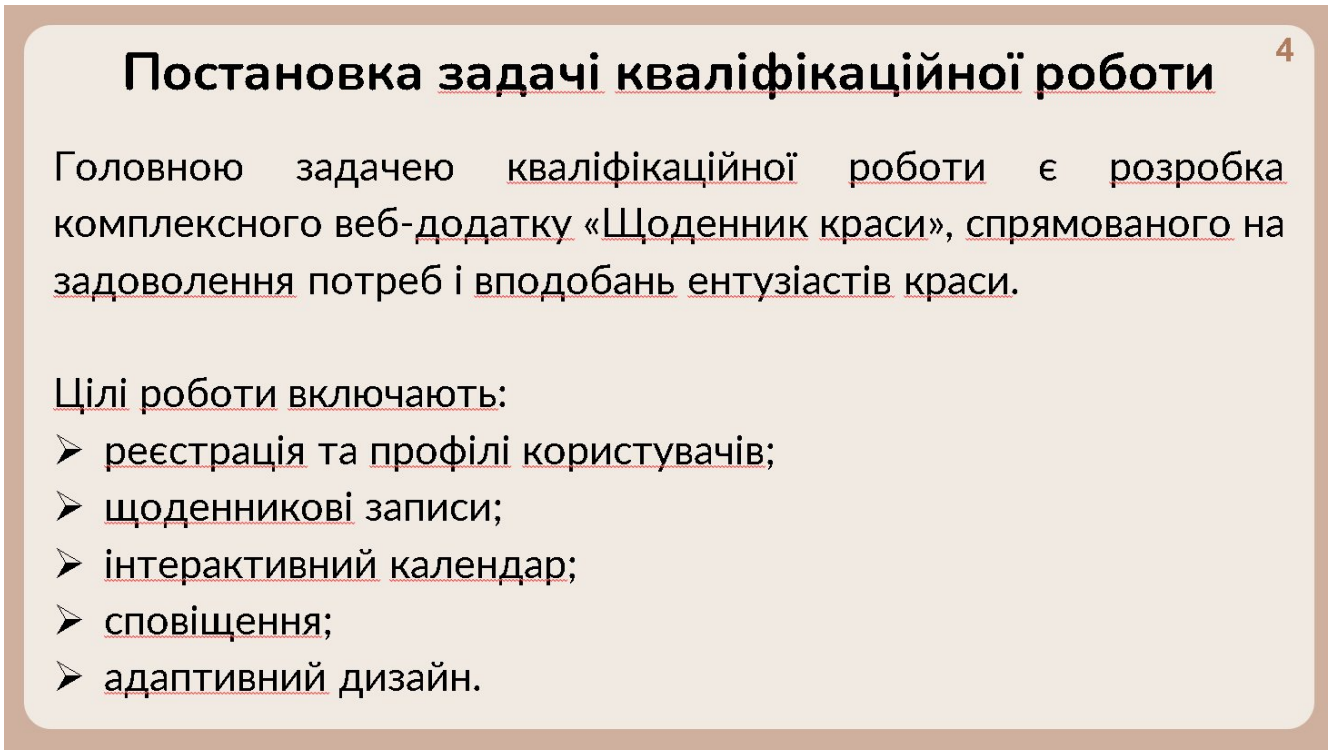

Рисунок Б.4 – Слайд презентації 4 (рисунок виконаний самостійно)

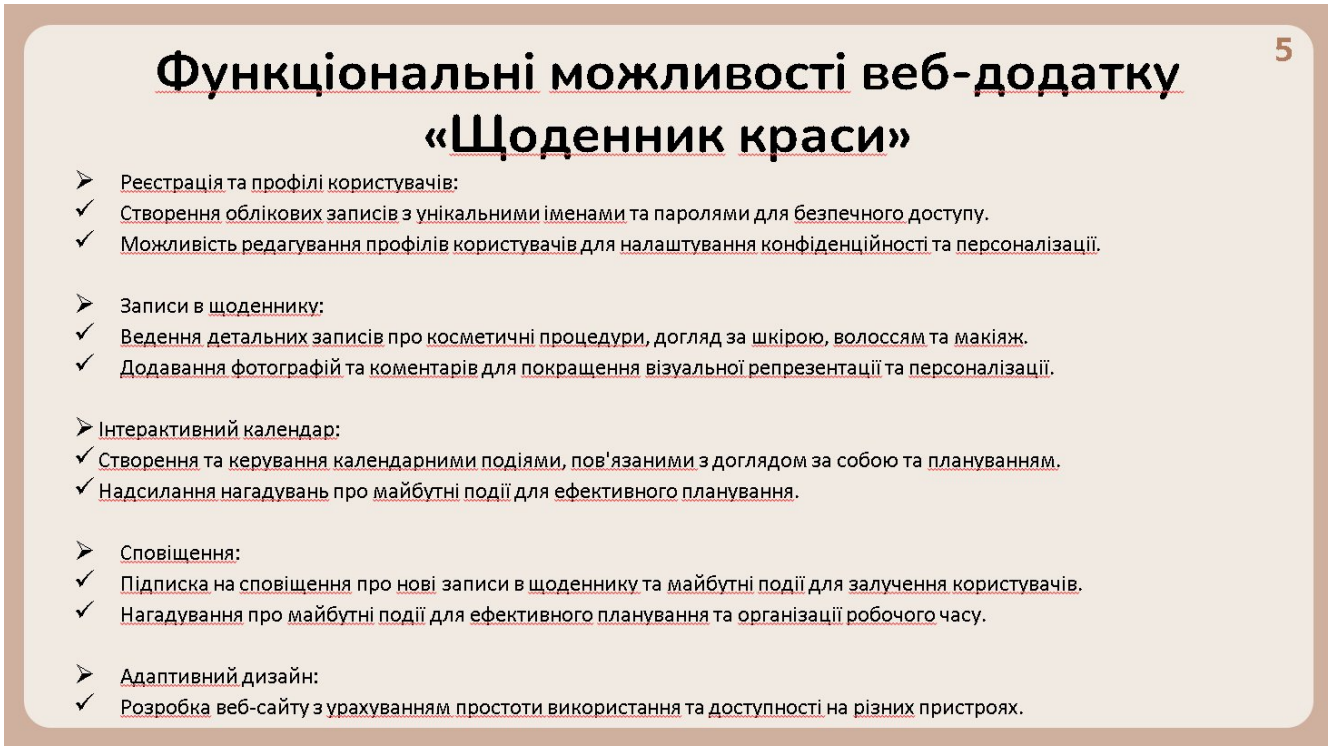

Рисунок Б.5 – Слайд презентації 5 (рисунок виконаний самостійно)

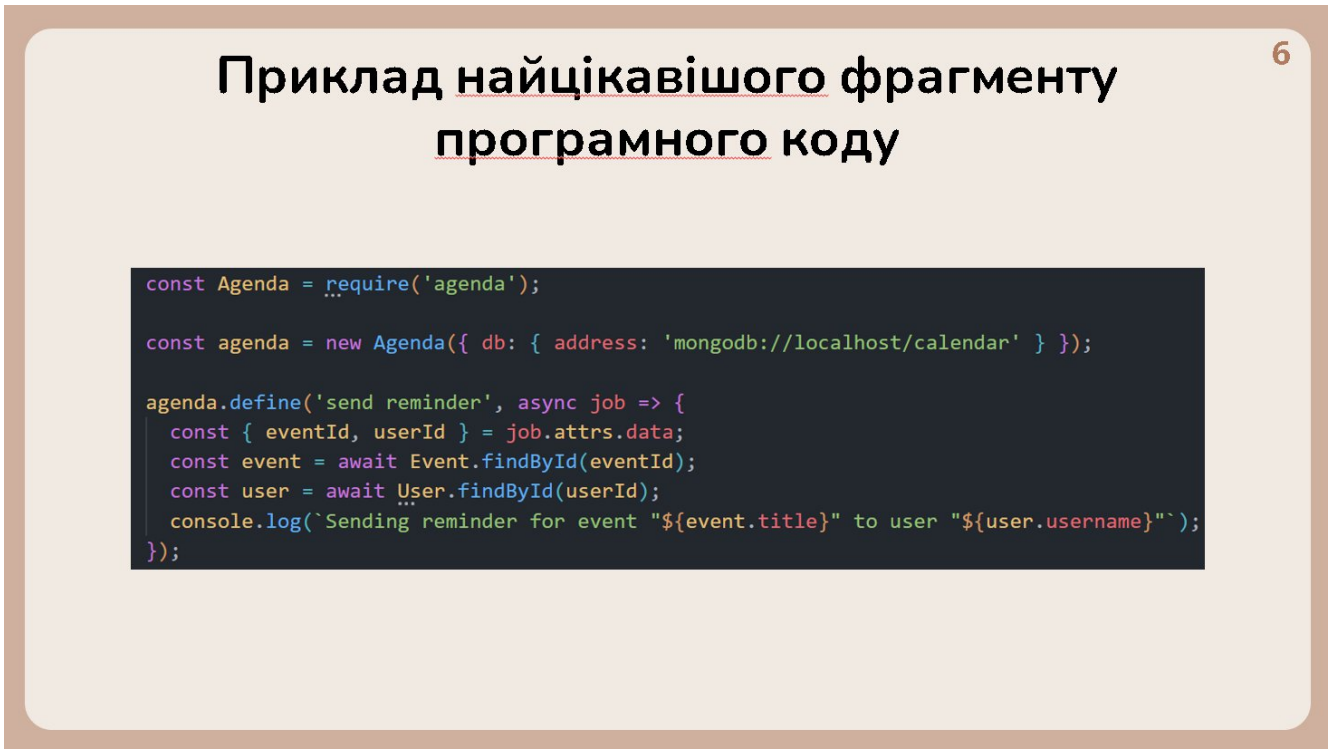

Рисунок Б.6 – Слайд презентації 6 (рисунок виконаний самостійно)

Рисунок Б.7 – Слайд презентації 7 (рисунок виконаний самостійно)

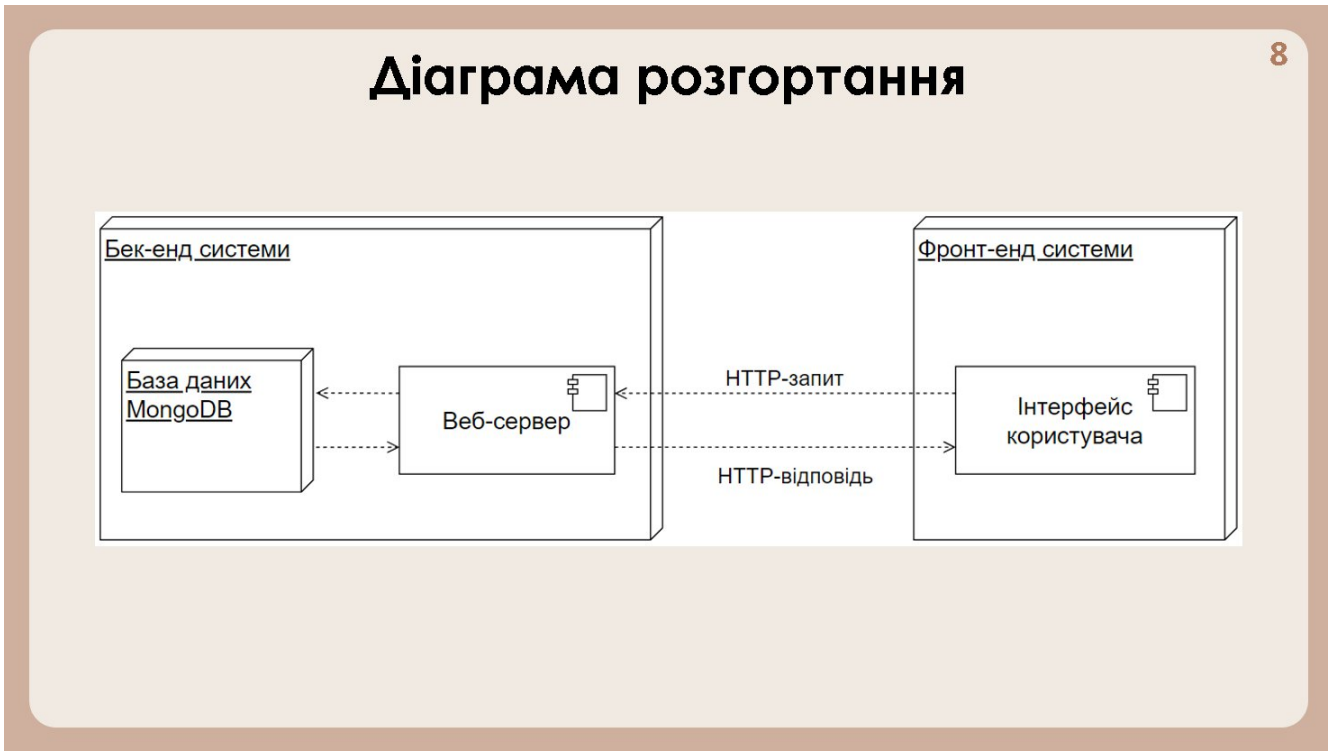

Рисунок Б.8 – Слайд презентації 8 (рисунок виконаний самостійно)

Рисунок Б.9 – Слайд презентації 9 (рисунок виконаний самостійно)

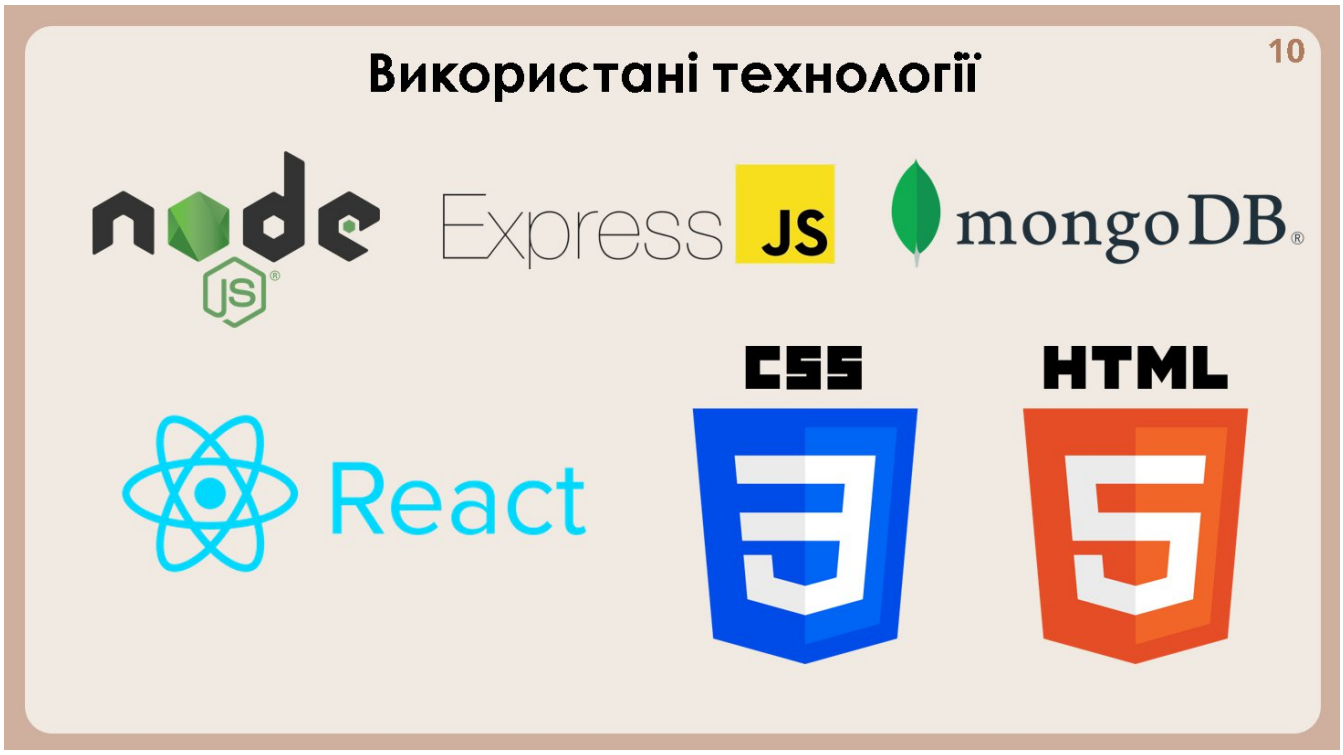

Рисунок Б.10 – Слайд презентації 10 (рисунок виконаний самостійно)

Рисунок Б.11 – Слайд презентації 11 (рисунок виконаний самостійно)

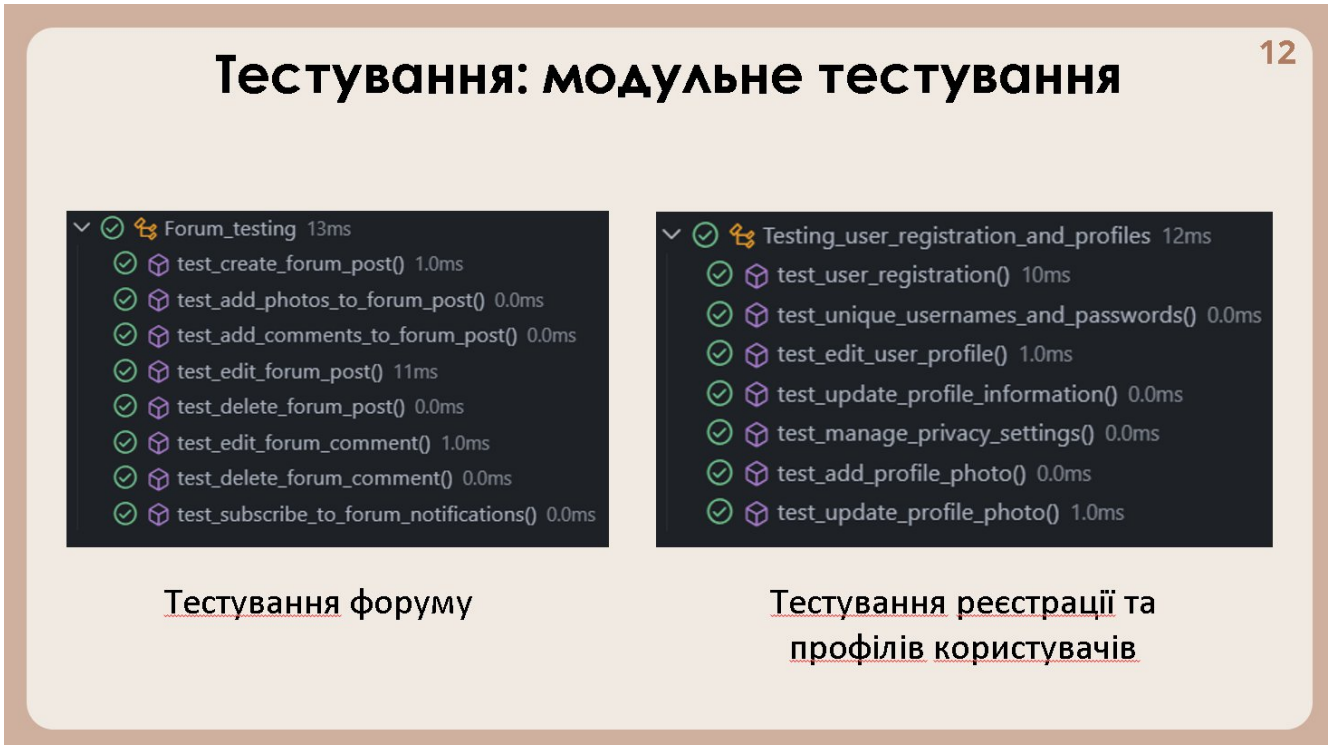

Рисунок Б.12 – Слайд презентації 13 (рисунок виконаний самостійно)

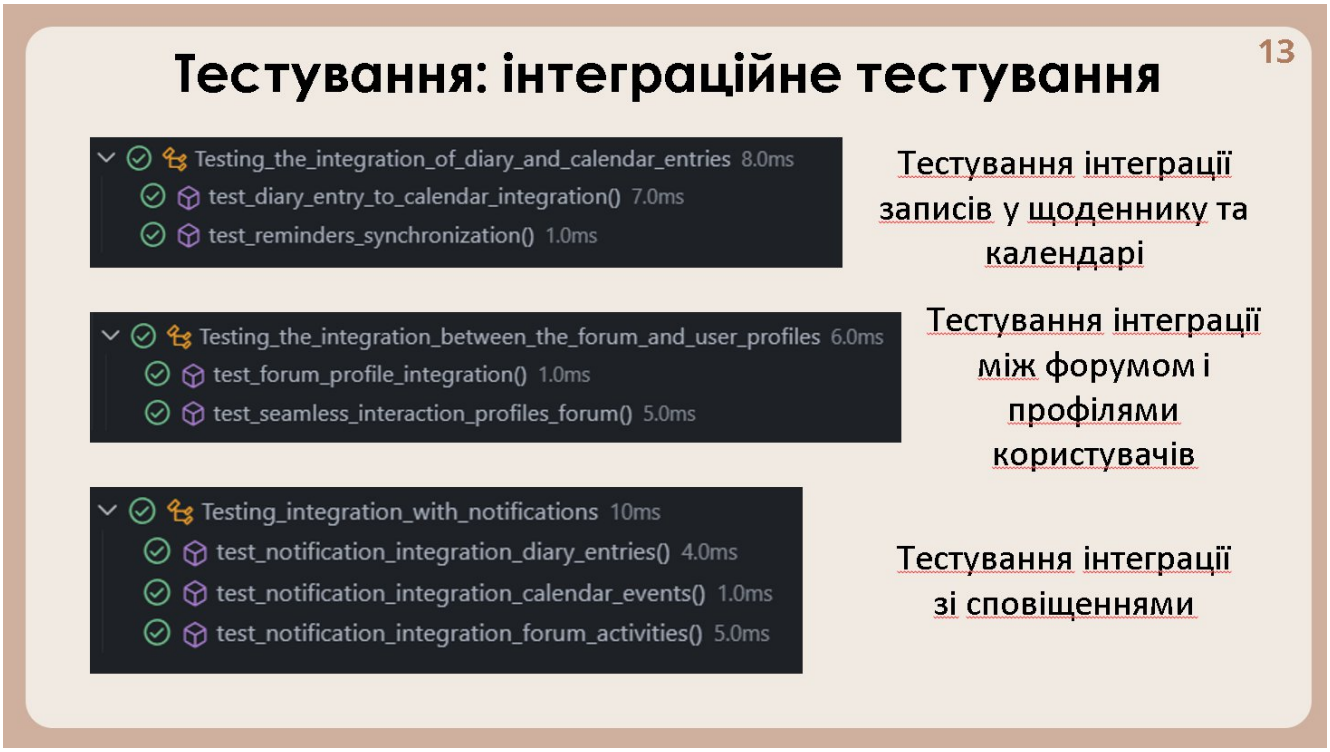

Рисунок Б.13 – Слайд презентації 13 (рисунок виконаний самостійно)

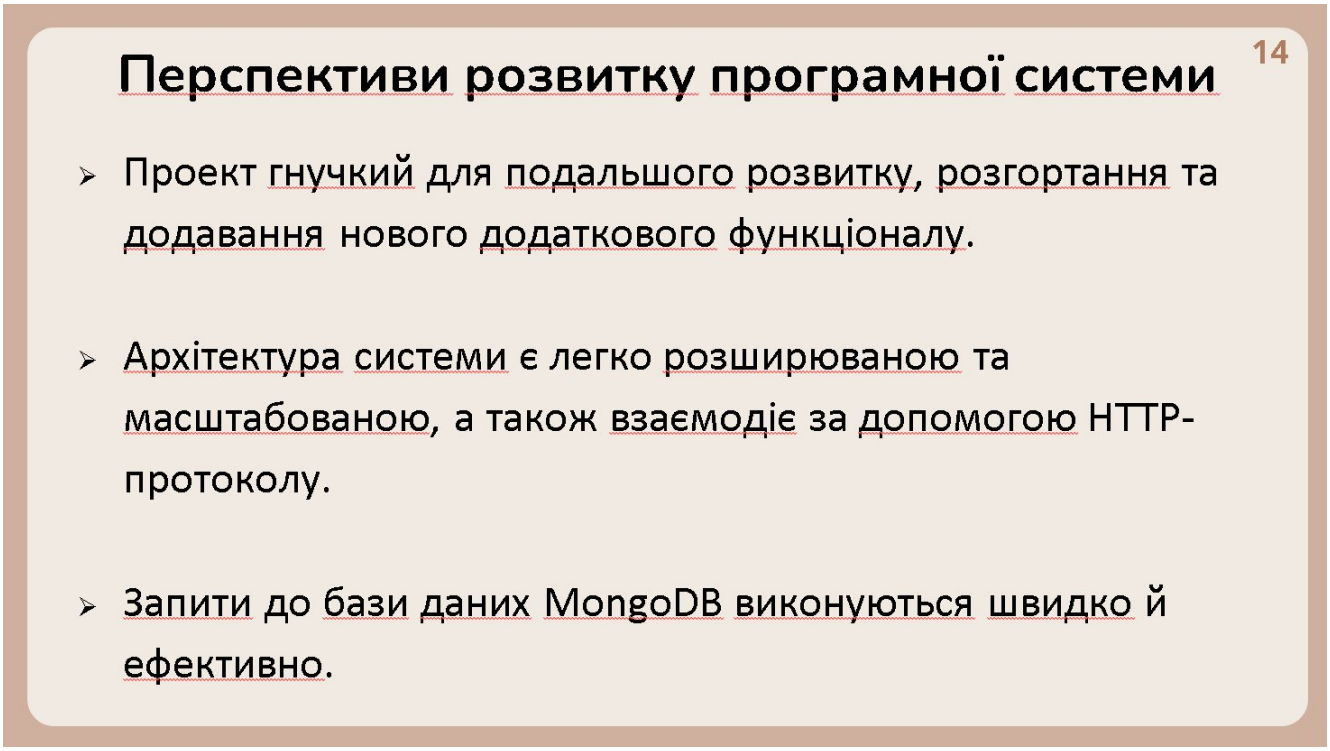

Рисунок Б.14 – Слайд презентації 14 (рисунок виконаний самостійно)

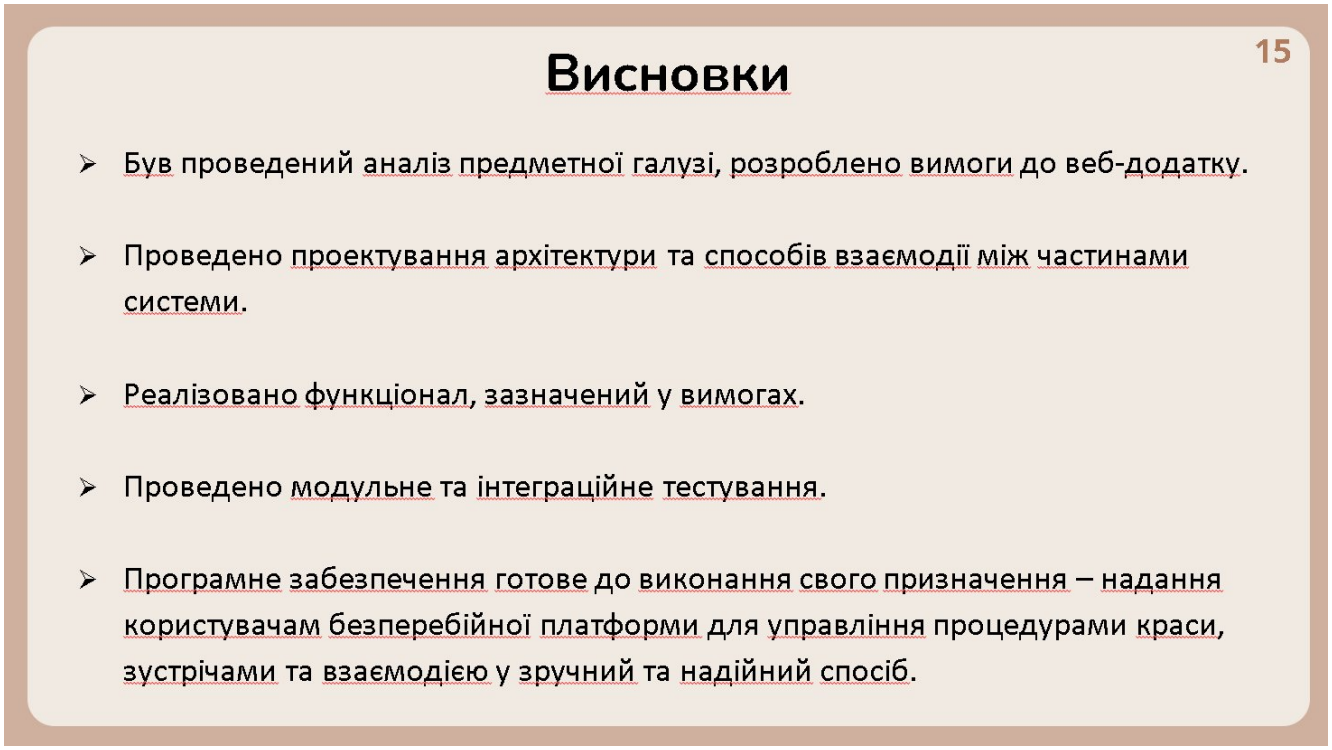

Рисунок Б.15 – Слайд презентації 15 (рисунок виконаний самостійно)