Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБФОРУМУ З ПИТАНЬ СТВОРЕННЯ ВЕБЗАСТОСУНКІВ

DEVELOPMENT OF A WEB FORUM ON WEB APPLICATIONS CREATION

Виконав: студент 4 курсу, групи КНТ-129 сп
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Кущ С.Р.
(прізвище та ініціали)

Керівник Степаненко О.О.
(прізвище та ініціали)

Рецензент Поляков М.О.
(прізвище та ініціали)

2022

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

_____С.О. Субботін

"____" _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Куща Сергія Руслановича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка вебфоруму з питань створення вебзастосунків. Development of a Web Forum on Web Applications Creation

керівник проекту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент ,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від " 27 " квітня 2022 року № 102

2. Строк подання студентом проекту (роботи) 7 червня 2022 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат. Перелік скорочень та умовних познак. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проектується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Висновки. Перелік джерел посилання. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Степаненко О.О., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 01 ” лютого 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи3,4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент _____ Кущ С.Р.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Степаненко О.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 117 сторінок, 27 рисунків, 7 таблиць, 4 додатки, 20 джерел.

ВЕБЗАСТОСУНОК, ВЕБФОРУМ, ПОВІДОМЛЕННЯ, ТЕМА, EXPRESS, JWT, NODE.JS, QUASAR, THREAD, VUE.JS

Об’єкт дослідження – вебфорум з питань створення вебзастосунків.

Предмет дослідження – методи побудови вебзастосунку для створення вебфоруму.

Мета роботи – розробка вебфоруму для обговорення користувачами питань, пов’язаних зі створенням вебзастосунків.

Матеріали, методи та технічні засоби: Visual Studio Code, Vue.js, Quasar, Express.js, Node.js

Актуальність створення такого вебфоруму пояснюється затребуваністю подібних рішень в глобальній мережі, зокрема у спільноті розробників вебзастосунків, мала кількість форумів, локалізованих для україномовних користувачів.

В результаті роботи було розроблено вебфорум з питань створення вебзастосунків, виправлено більшість недоліків аналогічних рішень при збереженні ряду їх переваг.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 117 pages, 27 figures, 7 tables, 4 appendixes, 20 sources.

EXPRESS, JWT, MESSAGE, NODE.JS, QUASAR, THEME, THREAD, VUE.JS, WEB APPLICATION, WEB FORUM

The object of research is a web forum on web applications creation.

The subject of research – methods of building a web application for creating a web forum.

The purpose of the work is to develop a web forum for users to discuss issues related to creating web applications.

Materials, methods and technical means: Visual Studio Code, Vue.js, Quasar, Express.js, Node.js.

The urgency of creating such a web forum is due to the demand for such solutions in the global network, in particular in the community of web application developers, a small number of forums localized for Ukrainian-speaking users.

As a result, a web forum on web applicationcreation was developed, and most of the shortcomings of similar solutions were corrected while maintaining a number of their advantages.

ЗМІСТ

Реферат 4
Перелік скорочень та умовних познак 8
Вступ 9
1 Аналіз предметної області 11
1.1 Вебфорум 11
1.2 Різновиди вебфорумів та їх особливості 12
1.3 Тематичні вебфоруми з питань розроюки програмного забезпечення . 14
1.4 Огляд існуючих рішень 15
1.5 Постанова завдання 24
1.6 Висновки за розділом 1 28
2 Вибір та обґрунтування структури системи, яка проєктується 29
2.1 Вибір інструментарію розробки програмного продукту 29
2.2 Архітектура застосунку та використані технології 37
2.3 Висновки за розділом 2 41
3 Основні рішення щодо реалізації компонентів системи 42
3.1 Використані моделі даних 42
3.2 Алгоритм введення даниз у застосунок 44
3.3 Механізм авторизаціх користувачів 46
3.4 Висновки за розділом 3 48
4 Керівництво програміста 49
4.1 Призначення й умови використання 49
4.2 Характеристики програмного продукту 49
4.3 Звернення до програмного продукту 52
4.4 Вхідні і вихідні дані 53
4.5 Висновки за розділом 4 54
5 Керівництво оператора 56
5.1 Призначення й умови виконання програмного продукту 56
5.2 Виконання програмного продукту 56

5.3 Повідомлення оператору 65
5.4 Висновки за розділом 5 65
Висновки 66
Перелік джерел посилання 68
Додаток А Технічне завдання 70
Додаток Б Опис програмного продукту 76
Додаток В Лістинги файлів роботи 85
Додаток Г Слайди презентації 110

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ВЗ	– вебзастосунок;
ПЗ	– програмний застосунок
ПП	– програмний продукт;
СКБД	– системи керування базами даних;
API	– Application Programming Interface;
IDE	– Integrated Development Environment;
JWT	– JSON Web Token
MVVM	– Model-View-ViewModel
REST	– Representational state transfer
SPA	– Single Page Application
WYSIWYG	– What You See Is What You Get

ВСТУП

У світі існує величезна кількість форм, засобів та способів спілкування, пошуку та обміну інформації. Сьогодні переважна більшість з них представлена у мережі Інтернет. Поміж служб глобальної мережі, що надають такі можливості широко поширені вебфоруми.

Форуми дуже популярні в середовищі розробників програмного забезпечення, тут спілкуються як програмісти-новачки, так і фахівці з десятками років досвіду. Розробка вебзастосунків (ВЗ) сьогодні дуже стрімко розвивається, програмістам доводиться працювати з величезною кількістю технологій, застосунків та модулів. Тематичні вебфоруми є одним із найкращих засобів пошуку допомоги з вирішенням проблем та обміну досвідом.

Метою роботи є розробка вебфоруму з питань створення ВЗ, аналіз проблеми, огляд існуючих аналогів, виділення їх переваг та недоліків, проектування архітектури застосунку, інформаційних моделей та програмна реалізація проекту.

Актуальність створення такого вебфоруму пояснюється затребуваністю подібних рішень в глобальній мережі, зокрема у спільноті розробників ВЗ, мала кількість форумів, локалізованих для україномовних користувачів.

Існують сотні, якщо не тисячі форумів в мережі Інтернет, значна частина яких функціонує вже протягом багатьох років. Позитивний наслідок цього – величезний об’єм накопичених обговорень та відповідей, у яких можна здійснювати пошук, негативний – застарілість використаних технологій.

Більшість існуючих рішень побудовані з використанням спеціальних рушіїв, призначених для створення форумів, які створені вже давно і активно не оновлюються. Такі форуми мають широкий функціонал, що допрацьовувався багато років, але застарілий, перевантажений та

недружелюбний до нових користувачів інтерфейс, що зазвичай некоректно функціонує на мобільних пристроях.

Для усунення деяких недоліків існуючих продуктів, перейнявши їх функціональні переваги, було вирішено розробити новий продукт.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перший розділ дипломної роботи присвячено огляду особливостей функціонування форумів у мережі Інтернет, зокрема особливостям тематичних форумів з програмування. В розділі буде проаналізована предметна область, особливості використання форумів програмістами, буде здійснено огляд та порівняння існуючих рішень, виявлено їх переваги та недоліки, визначені межі проекту, функціональність створюваної системи, вимоги до безпеки та інтерфейсу.

1.1 Вебфорум

Глобальна мережа Інтернет широко використовується для спілкування, обговорень, дискусій. Тут набагато легше відшукати людину, яка буде розбиратися в питанні, що тебе цікавить, або просто цікавого співрозмовника з подібними твоїм інтересами. Більшість людей відчувають себе більш розкуто при спілкуванні через мережу. Одним з різновидів інтернет-ресурсів, що використовуються для спілкування, є вебфорум.

Вебфорум, інтернет-форум, або просто форум – це інтернет-сервіс для проведення онлайн-дискусій, на якому люди спілкуються шляхом створення тем для обговорення, з наступним розміщенням у них повідомлень[1]-[3].

Суть форумів в Інтернеті – це обмін думками та досвідом, спілкування з однодумцями, безкорислива допомога. Часто на форумах можна знайти інформацію, яку не відшукати ні в книгах, ні в інтернет-джерелах, бо користувачі форумів діляться, своїм, реальним досвідом [1]-[3].

На відміну від чатів і месенджерів, де спілкування відбувається синхронно, з майже миттєвою відповіддю, на форумах спілкування між користувачами відбувається асинхронно, тобто очікувати миттєвої відповіді на своє повідомлення не варто. Перевагою асинхронного спілкування є те,

що відповідь на ваше повідомлення у більшості випадків буде розгорнутою й вичерпною [1]-[3].

Від звичайних вебсайтів форуми відрізняються тим, що наповненням їх інформацією займається безпосередньо аудиторія користувачів, тому важливо мати перелік правил форуму для контролю за його наповненням.

Вебфоруми майже завжди мають відповідальних осіб та супроводжувальний персонал – модераторів та адміністраторів. Модератори мають більше прав, ніж рядовий користувач, контролюють наявність небажаного контенту у наповнені форуму та можуть маніпулювати ним у разі порушення правил. Адміністратор зазвичай має всі права модератора а також контролює технічну частину форуму, може маніпулювати базою даних та змінювати права доступу користувачів [1]-[3].

Інтернет-форуми мають ієрархічну (деревоподібну) структуру. Форум ділиться на розділи, підрозділи з певною тематикою, в яких користувачі створюють теми для обговорення шляхом розміщення повідомлень у цих темах. Повідомлення також зазвичай мають ієрархічну структуру, тобто користувачі можуть відповідати на повідомлення інших користувачів [1]-[3].

Вебфоруми зазвичай мають специфічний набір жаргонізмів, які можуть різнитися залежно від тематики форуму та звичаїв його постійних користувачів. Наприклад, окрема тема з повідомленнями у ній називається «thread», повідомлення, яке не відповідає темі, що обговорюється – «off-topic» [1]-[3].

1.2 Різновиди вебфорумів та їх особливості

Існує велике різноманіття вебфорумів, що відрізняються підходом до структури тем та повідомлень, розмежуванням прав доступу та функціоналом. Розглянемо найбільш поширені види для визначення оптимального підходу до функціонування застосунку, що створюється.

Вебфоруми можуть мати різний підхід до розмежування прав доступу. Так, деякі форуми можуть навіть не потребувати реєстрації та дозволяють переглядати теми, повідомлення, створювати власні будь-яким випадковим відвідувачам. Такі форуми зазвичай є анонімними і не зберігають інформації про користувача, що залишив на ньому повідомлення. Найбільш поширеним варіантом є форуми, які потребують попередньої реєстрації для створення тем та повідомлень, але дозволяють перегляд існуючих без створення профілю користувача [1]-[3].

Форуми зазвичай відкриті для пересічного користувача в Інтернеті, але існують і закриті форуми, отримати доступ до яких можна лише за персональним запрошенням від адміністратора. Існують також суміжні варіанти, коли більшість розділів форуму відкрита для пересічних користувачів, а деякий набір розділів для використання потребує дозволу від адміністрації [1]-[3].

За способом визначення тем, що обговорюються існують форуми з динамічним і статичним набором тем. На форумах з динамічним набором тем нові теми створюються користувачами форуму в рамках установлених правил. На форумах зі статичним набором тем список тем визначається адміністрацією форуму [1]-[3].

Форуми можуть бути тематичними – на яких обговорюється вузький набір тем з якоїсь галузі, або можуть мати широкий набір тем і ділитися на велику кількість підфорумів.

Широке розповсюдження мають такі типи форумів, як Textboard та Imageboard. Textboard – це найпростіший вид вебфорумів, спілкування на якому не потребує реєстрації і проводиться виключно за допомогою тексту, без включень. Такі форуми, як правило, набагато простіші, ніж традиційні форуми, створені з використанням спеціальних рушіїв. Imageboard – це розширення концепції Textboard, що дозволяє розміщувати зображення в тексті, розширюючи можливості спілкування, зміщуючи увагу на візуальну подачу інформації [1]-[3].

Більшість відносно сучасних вебфорумів для створення і форматування змісту тем та повідомлень використовують спеціальні текстові редактори з розширеним функціоналом. Переважна більшість таких редакторів називається WYSIWYG-редакторами (що бачу, те і отримаю) і дозволяють одразу при редагуванні тексту бачити, як він у результаті виглядатиме на вебсторінці. Такі редактори дозволяють форматувати набраний текст, додавати до нього зображення, посилання, створювати таблиці, списки та багато іншого.

ВЗ, що створюється буде тематичним, зосередженим, на обговоренні питань розробки вебзастосункі, список розділів та підрозділів визначатиметься і редагуватиметься адміністрацією форуму. Випадкові відвідувачі зможуть переглядати всі розділи та теми, без виключення, але для створення нових необхідно буде зареєструватися.

1.3 Тематичні вебфоруми з питань розробки програмного забезпечення

Коли програмісти створюють нове програмне забезпечення (ПЗ), або додають функціонал до вже існуючого, вони постійно стикаються з проблемами використання нових, незнайомих бібліотек, засобів розробки та алгоритмів. Вирішення цих питань вони зазвичай шукають в мережі Інтернет, за допомогою пошукових систем, таких як Google, та на спеціалізованих ресурсах, таких як тематичні вебфоруми [4]-[5].

Пошукові запити можна поділити на запити високого рівня – для пошуку загальних керівництв та низького – для пошуку більш конкретних фрагментів програмного коду, прикладів використання певних бібліотек та функцій [4]-[5].

Вебфоруми накопичують величезну кількість відповідей на питання та фрагментів програмного коду. Розвинуті технології пошуку дозволяють програмістам швидко знайти вирішення проблеми, що їх бентежить, або

спитати про неї в інших користувачів, якщо ця проблема ще не обговорювалася. Зазвичай більшість проблем, з якими стикаються програмісти, вже кимось вирішувалася і рішення може бути швидко знайденим [4]-[5].

Негативним наслідком того, що вебформами для програмістів користуються як досвідчені спеціалісти, так і новачки, є те, що рішення проблеми чи фрагменти програмного коду можуть містити масу помилок. Багато програмістів, а особливо новачки, часто можуть бездумно копіювати такі фрагменти коду у свій проект, що може спричинити багато проблем, помилок і дірок у безпеці [4]-[5].

Форуми для програмістів мають деякі функціональні відмінності від форумів з іншою тематикою. Зазвичай вони дозволяють публікувати фрагменти програмного коду, форматовано відображають його та підсвічують загальний синтаксис. Деякі форуми, з використанням сторонніх ресурсів, можуть навіть інтерпретувати програмний код, дозволяючи редагувати його та відображати результат його виконання в реальному часі. Корисною також є можливість завантаження файлів, з прикріпленням їх до тем та повідомлень, що дозволить більш зручно поширювати великі фрагменти коду, архіви з бібліотеками та інші спеціалізовані файли.

1.4 Огляд існуючих рішень

Вебформи для програмістів створюються для проведення онлайн-дискусій, пошуку допомоги з проблемами, матеріалів для навчання. Існує багато ВЗ, що вирішують цю задачу.

Існуючі рішення мають свої переваги й недоліки, тому для створення конкурентоспроможного застосунку необхідно на початку проаналізувати існуючі аналоги, виділити переваги, які можна унаслідувати й покращити та недоліки, які необхідно виправити.

Для аналізу вирішено використати сервіси «Replace.org.ua», «Stack Overflow» та «Dream.In.Code».

1.4.1 «Replace.org.ua»

«REPLACE» – один з небагатьох україномовних форумів для програмістів, функціонує з 2012 року. Мовою спілкування на сайті є виключно українська, іншомовний контент не вітається, перегляд такого наповнення повинен приховуватись функціоналом сайту, з вказанням використаної мови. Виключенням з правила є англійська мова [6].

Форум не має вузької направленості, розділи форуму охоплюють більшість напрямків програмування. На головній сторінці розміщені розділи та основні теми для обговорення (рис. 1.1).

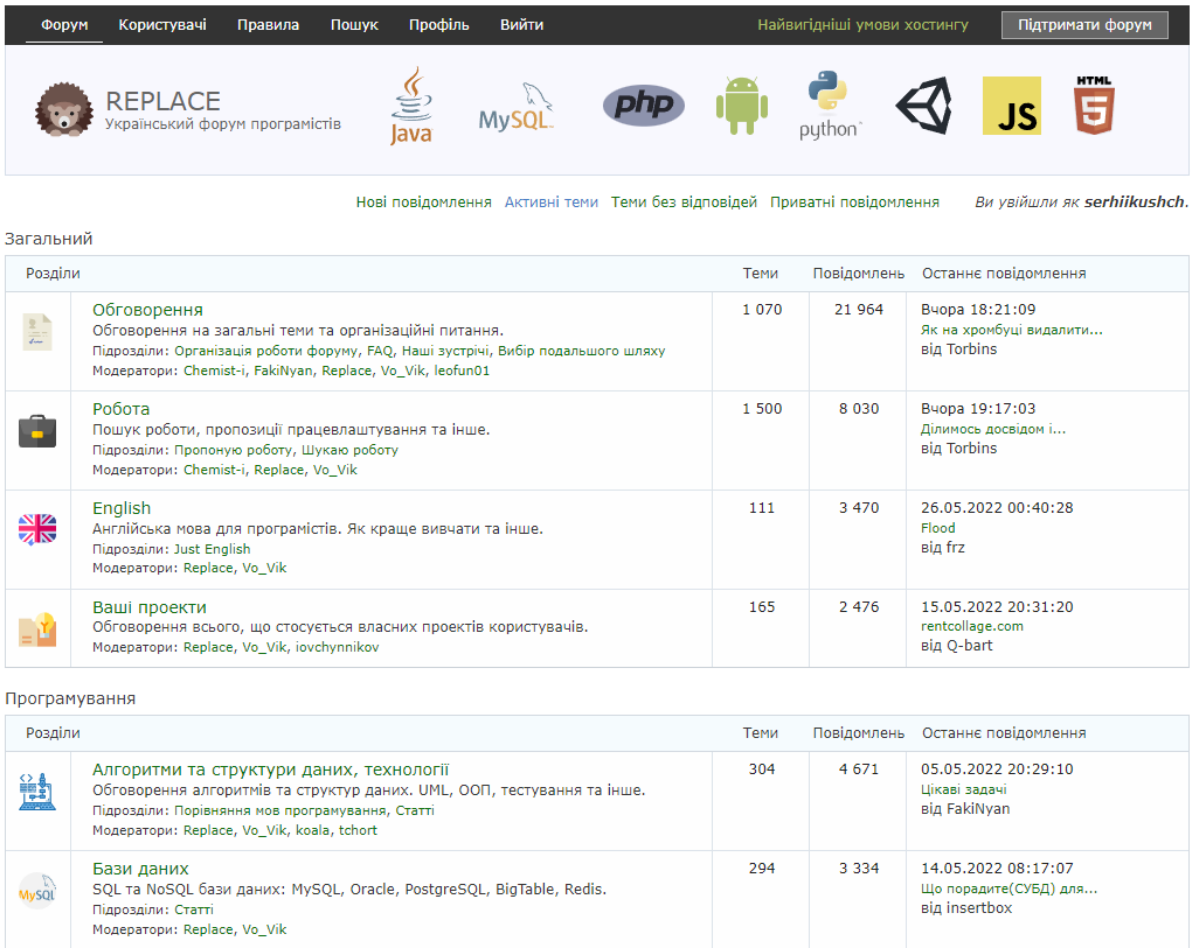

Рисунок 1.1 – Головна сторінка форуму «REPLACE»

Пошук по темам працює чудово, але для пошуку необхідно перейти на окрему сторінку.

Сайт виглядає досить сучасно, має зручну панель навігації, на якій розміщені всі основні посилання, але містить ненав'язливу рекламу хостингу та посилання для підтримки форуму.

Реєстрація зручна, з підтвердженням через посилання у email-листі.

Перелік тем містить інформацію про кількість відповідей та переглядів, дату створення. Функціонал оцінювання тем та їх сортування за наданими даними – відсутній (рис. 1.2).

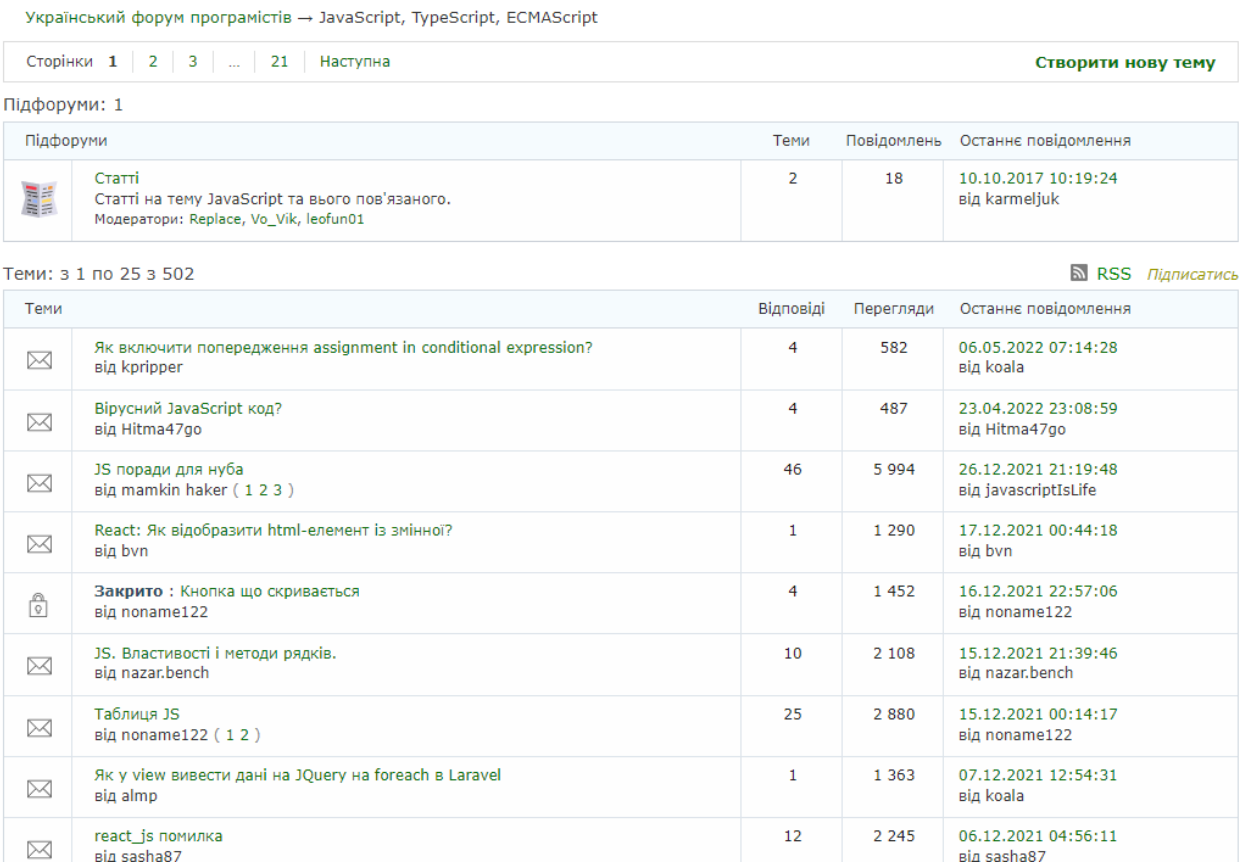

Рисунок 1.2 – Перелік тем розділу JavaScript

При створенні тем та повідомлень використовується відносно зручний редактор, є можливість прикріплювати зображення, підсвічувати синтаксис програмного коду. Обмін файлами, оцінювання відповідей та зручне форматування тексту не реалізовано.

Усі розділи та теми є загальнодоступними для всіх користувачів, для створення нових необхідно зареєструватися [6].

Отже, вебсайт має відносно сучасний дизайн, зручну навігацію і повністю відкритий для нових користувачів, але йому не вистачає функціоналу. Сайт функціонує вже 10 років і має досить багато накопичених обговорень, але користувачів у мережі дуже мало, сайту слід приваблювати нових. Основним недоліком є те, що застосунок не оптимізований для роботи на мобільних пристроях.

1.4.2 «Stack Overflow»

«Stack Overflow» – один з, якщо не найбільший форум для розробників програмного забезпечення, частина мережі Q&A сайтів «Stack Exchange», функціонує з 2008 року, постійно оновлюється і містить величезну кількість накопичених обговорень і рішень проблем [7].

Форум не має явно виражених розділів, теми фільтруються за набором визначених тегів, що дуже зручно (рис. 1.3).

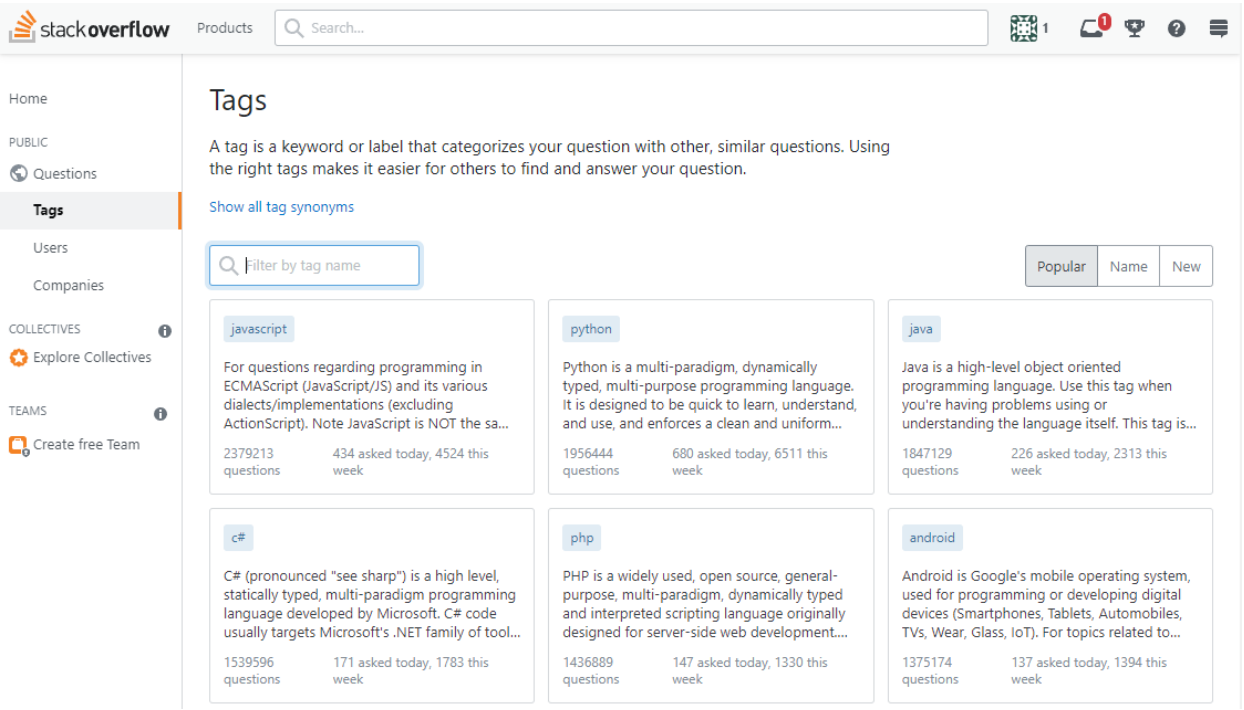

Рисунок 1.3 – Перелік тегів для фільтрування тем

Реєстрація зручна, з підтвердження через email-лист.

Кількість користувачів у мережі завжди дуже висока і відповідь на питання зазвичай довго чекати не доводиться. Сайт виглядає сучасно, добре адаптований для використання на мобільних пристроях.

Пошук працює чудово, розміщений прямо на панелі навігації, що дуже зручно.

Теми підтримують фільтрування по тегам, сортування по популярності за період часу, по переглядам , оцінкам користувачів. Перелік тем містить інформацію про кількість переглядів, відповідей та оцінку виставлену користувачами (рис. 1.4).

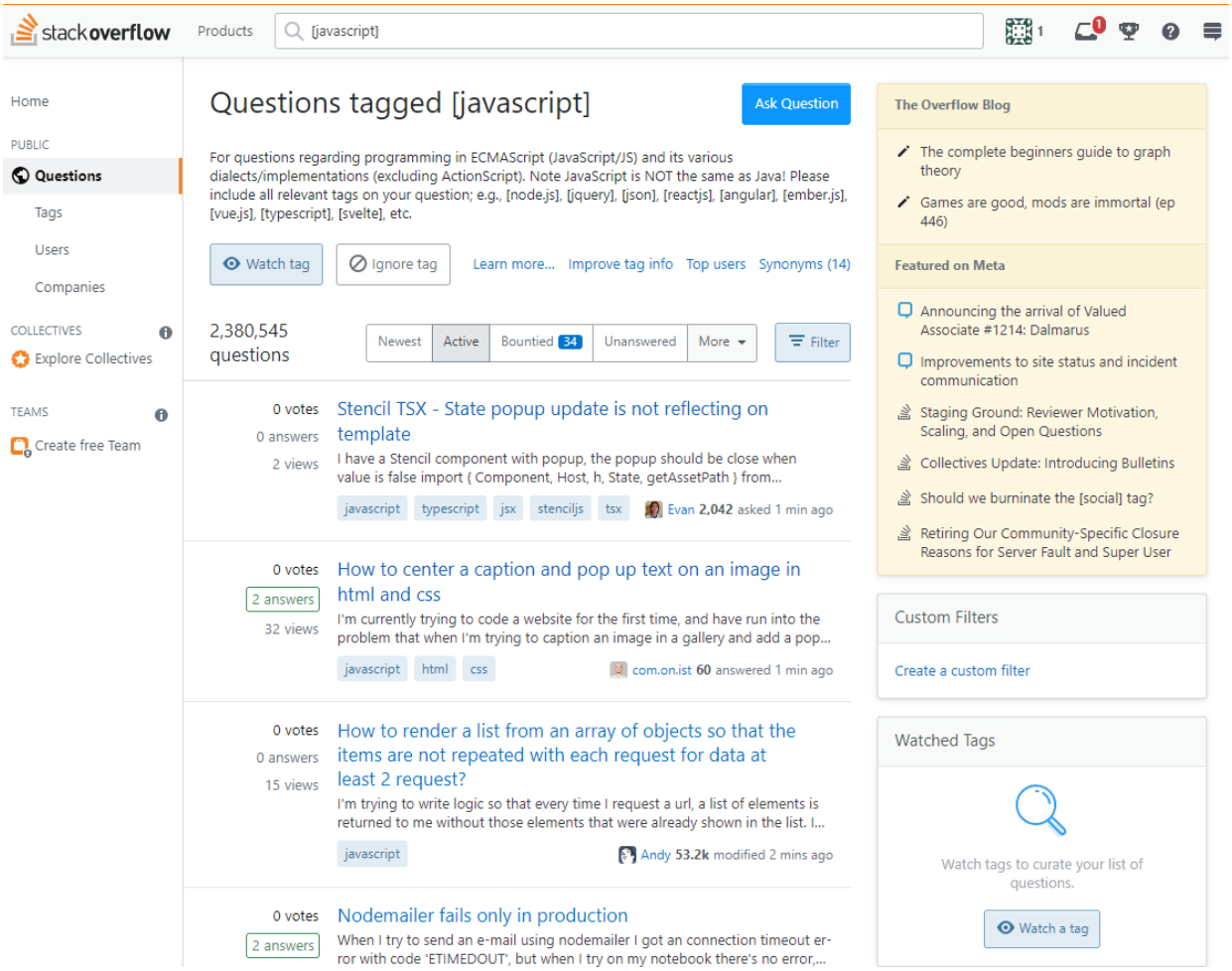

Рисунок 1.4 – Активні теми за тегом JavaScript

Доступ до тем форуму вільний, для створення нових тем та повідомлень треба зареєструватися. Реклама повністю відсутня [7].

Створення тем та повідомлень відбувається за допомогою зручного текстового редактора, реалізоване форматування тексту, є можливість прикріплювати зображення, підсвічувати синтаксис програмного коду, створювати списки, таблиці та інше. Реалізована зручна система оцінювання тем та повідомлень користувачами, є можливість сортувати за оцінкою [7].

Обмін файлами не реалізовано, що було б досить зручно для програмістів.

Загалом важко знайти технічні недоліки цього форуму. Можна виділити те, що на ресурсі не заохочується обговорення абстрактних і дискусійних питань, дублікатів вже існуючих обговорень та питань, такі теми можуть бути помічені модераторами сайту як небажані і згодом бути видаленими.

1.4.3 «Dream.in.code»

«Dream.In.Code» – популярний вебфорум для програмістів та веброзробників, працює з 2001 року, надає користувачам доступ до тисяч інструкцій, фрагментів коду та обговорень. Зарекомендував себе дружньою спільнотою експертів, захоплених поширенням свого досвіду з іншими [8].

Сайт має досить застарілий дизайн, новачкам може бути важко розібратися з використанням функціоналу сайту та навігацією. Форум зовсім не адаптований для використання на мобільних пристроях.

Форум має систему рейтингу користувачів, тем та відповідей, що дозволяє виділити найбільш корисні обговорення та експертів з поміж інших.

На форумі є реклама, яку можна відключити придбавши щомісячну підписку. Підписка також розширює ліміт особистих повідомлень, об’єму завантаження файлів, візуальні прикраси та інше [8].

Форум має великий набір підфорумів, що охоплюють широкий спектр питань розробки програмного забезпечення. Список форумів показує

кількість тем у них, список тем – кількість відповідей, переглядів, але функціонал сортування за цими та іншими критеріями відсутній (рис 1.5).

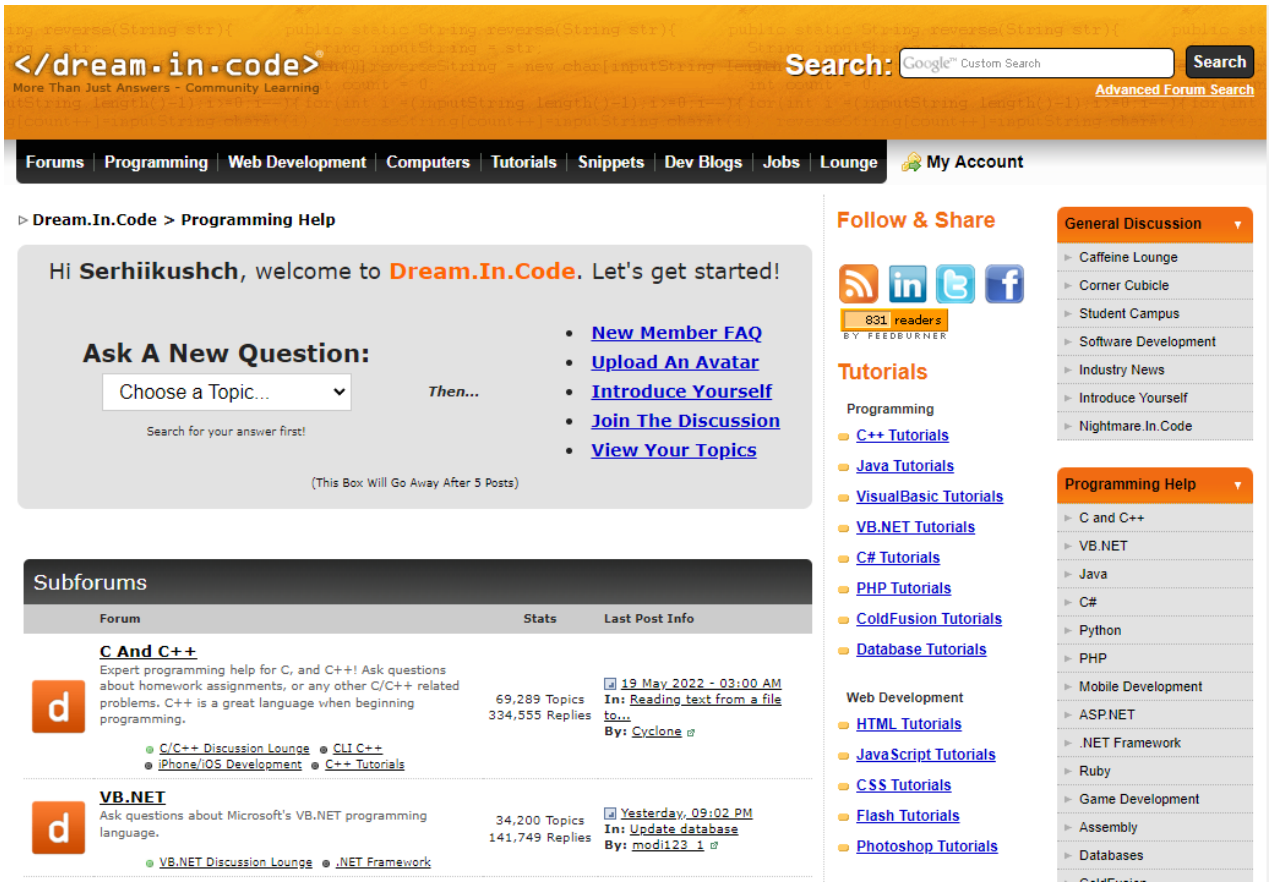

Рисунок 1.5 – Список підфорумів «Dream.In.Code»

Для створення нових тем та повідомлень у підфорумах необхідно зареєструватися. Реєстрація досить зручна, з підтвердженням через email-лист.

Створення тем та відповідей здійснюється за допомогою відносно зручного редактора, реалізовано відформатоване виведення та підсвічування синтаксису програмного коду [8].

Цей форум, на відміну від більшості подібних має можливість завантажувати файли та прикріплювати їх до повідомлень, що може бути дуже зручною функцією для програмістів [8].

Пошук відносно зручний, розміщений на панелі навігації. Також існує більш розширений варіант пошуку, для використання якого слід перейти на окрему сторінку (рис. 1.6).

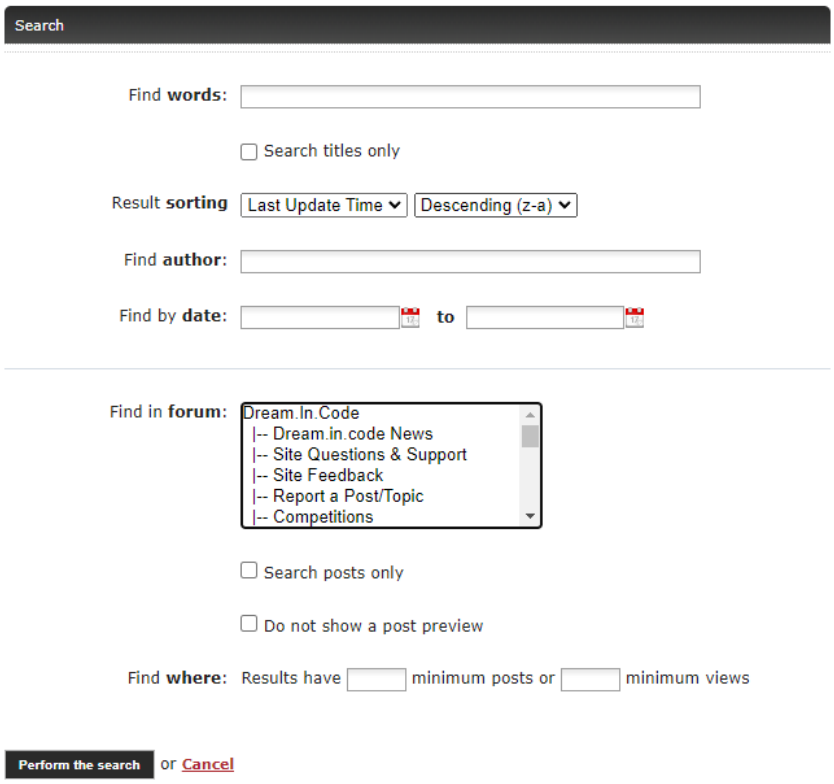

Рисунок 1.6 – Розширений пошук «Dream.In.Code»

Отже, вебсайт має величезну базу накопичених обговорень та відповідей, активну користувацьку базу. До недоліків можна віднести застарілий, частково перевантажений дизайн, відсутність адаптації для мобільних пристроїв та наявність реклами.

1.4.4 Порівняння оглянутих аналогів

Для порівняння оглянутих аналогів виділимо ряд критеріїв:

– система оцінювання тем та повідомлень – дозволяє виділити найбільш корисні з них;

– зручний дизайн та навігація – відсутність перевантаженості інтерфейсу користувача та старих, проблемних рішень;

– WYSIWYG-редактор тексту – дозволяє одразу бачити текст, так як він виглядатиме на сторінці, форматувати, створювати таблиці, списки, прикріплювати малюнки та інше;

– функціонал обміну файлами – дозволяє завантажувати файли та прикріплювати їх до тем та повідомлень;

– адаптивний дизайн – можливість зручно користуватися сайтом на мобільних пристроях, планшетах та інших пристроях.

Представимо порівняння форумів у вигляді таблиці (табл. 1.1).

Таблиця 1.1 – Порівняння оглянутих аналогів

Критерій порівняння	Replace.org.ua	Stack Overflow	Dream.in.code	Вебзастосунок, що розробляється
Система оцінювання тем та повідомлень	Відсутня	Присутня	Присутня	Присутня
Зручний дизайн та навігація	Порівняно сучасний дизайн, зручна навігація	Сучасний дизайн, зручна навігація	Застарілий дизайн, відносно зручна навігація	Сучасний дизайн, зручна навігація
WYSIWYG-редактор тексту	Відсутній	Присутній	Відсутній	Присутній
Функціонал обміну файлами	Відсутній	Відсутній	Присутній	Присутній
Адаптивний дизайн	Відсутній	Присутній	Відсутній	Присутній

Існують сотні подібних вебфорумів зі своїми особливостями, перевагами й різним функціоналом. З порівняння можна зробити висновок, що більшість з них мають низку недоліків.

Актуальною задачею є розробка ВЗ, що поєднає у собі деякі переваги існуючих систем і виправить їх недоліки. Новий форум матиме функціонал обміну файлами, зручний WYSIWYG-редактор, систему оцінювання тем та

повідомлень, вирішить проблеми з дизайном більшості старих застосунків та буде добре адаптованим для роботи на мобільних пристроях.

1.5 Постанова завдання

Метою проекту є проектування і створення вебфоруму з питань створення ВЗ, що дозволить користувачам ділитися своїм досвідом розробки, шукати відповіді на питання, фрагменти програмного коду з прикладом використання технологій і реалізацією алгоритмів.

1.5.1 Межі проекту

Основною функцією веб форумів є створення тем та повідомлень в них, тому першочергово слід реалізувати саме цей функціонал.

З допомогою JavaScript фреймворку Express буде створено серверну частину застосунку, що здійснюватиме функції зберігання й оперування даними в базі даних MongoDB. Також на сервер буде винесена частина бізнес логіки з клієнта, завантаження та скачування файлів, реалізація безпечної реєстрації/авторизації користувачів.

Сервер буде працювати на платформі Node.js, що дозволяє запускати JavaScript сценарії на сервері.

З допомогою JavaScript фреймворку Vue.js буде створено клієнтську частину застосунку у вигляді вебсайту, що функціонуватиме в середовищі браузера.

Зручний WYSIWYG-редактор буде реалізовано з використанням готового коду, у вигляді бібліотеки.

Частина елементів інтерфейсу буде реалізована з використанням фреймворку Quasar.

1.5.2 Функціональність системи

Програмний продукт, що розроблюється має реалізувати наступні функції:

– створення нових тем та повідомлень, їх редагування й видалення;

– створення відповідей на повідомлення у темах, або на відповіді інших користувачів;

– зручна навігація по розділам і темам;

– пошук по темам у підрозділах та загальний пошук по усім розділам;

– зручна система реєстрації та авторизації на сайті, активування профілю за посиланням у email-листі;

– функціонал редагування користувацького профілю та зміни фото профілю користувача;

– можливість позитивно чи негативно оцінити теми та повідомлення в них, скасувати чи змінити свою оцінку за необхідності;

– сортування тем у розділах за назвою, датою їх створення, оцінкою користувачів та кількістю повідомлень;

– сортування повідомлень у темах за датою створення та оцінкою користувачів;

– зручний WYSIWYG-редактор, з можливістю форматування тексту, додавання зображень, посилань, таблиць, списків, програмного коду з підсвічуванням синтаксису в повідомленнях та темах;

– функціонал завантаження файлів для їх прикріплення до тем та повідомлень, з подальшою можливістю їх скачування;

– функціонал для контролю наповненням форуму модераторами;

– функціонал для зміни прав доступу та статусу користувачів адміністратором;

– функціонал для створення, редагування й видалення розділів та підрозділів адміністратором.

1.5.3 Вимоги до інтерфейсу користувача

Інтерфейс повинен бути простим, зручним та інтуїтивно-зрозумілим. При виникненні помилок користувач повинен бути сповіщеним про них за допомогою спливаючих повідомлень. Макет та дизайн сайту повинен адаптуватися для використання на мобільних та інших популярних пристроях. Сайт повинен чітко ділитися на розділи і забезпечувати просту навігацію між ними. Перегляд текстової та візуальної інформація не повинен викликати незручностей у користувачів.

1.5.4 Вимоги до безпеки

Незареєстровані користувачі повинні мати доступ до перегляду усіх розділів, тем та повідомлень, для створення нових необхідна реєстрація та підтвердження через email-лист. Зареєстровані користувачі повинні мати право редагувати та видаляти лише створені ними теми та повідомлення. Доступ до додаткового функціоналу, пов’язаного з управлінням наповнення форуму, повинен бути доступним лише для користувачів з правами модератора або адміністратора. Доступ до створення розділів, підрозділів, зміни прав та статусу користувачів повинні мати лише користувачі з правами адміністратора.

1.5.5 Нефункціональні вимоги

Мінімальні системні вимоги середовища (хостингу) для коректного функціонування серверу ВЗ:

– CPU: Intel Core i5;
– RAM: 2 Gb;

– HDD/SSD: 1Gb (мінімально, для початкового запуску), але при розростанні, збільшенні кількості користувачів, тем та відповідей, для зберігання бази даних форуму може знадобитися чималий дисковий простір;

– вебсервер: Apache 2 або Nginx;

– база даних: MongoDB;

– порти: 80 (HTTP), 27018 (стандартний порт MongoDB);

– програмне забезпечення: середовище Node.js, менеджер пакетів NPM або Yarn.

Для коректного функціонування клієнтської частини розробленого ВЗ, а також у разі його доопрацювання, апаратна платформа повинна відповідати мінімальним системним вимогам браузера, в якому планується використання сайту та мати стабільне інтернет-з’єднання. Розглянемо мінімальні вимоги популярного браузера Google Chrome.

Мінімальні вимоги для комп’ютерів з операційною системною (ОС) Windows:

– Windows 7, 8, 8.1, 10 або більш пізньої версії;

– процесор Intel Pentium 4 або більш пізньої версії з підтримкою SSE 3 [9].

Мінімальні вимоги до комп’ютерів з MacOS:

– OS X El Capitan (10.11) або більш пізньої версії;

– Macbook Pro 2011, Macbook Air 2012 або більш нові версії [9].

Мінімальні вимоги для пристроїв на операційній системі Linux:

– Ubuntu 18.04 (64-розрядної версії) або новіша, Debian 10 або новіша, openSUSE 15.2 або новіша, Fedora Linux 32 або новіша;

– процесор Intel Pentium 4 або більш новий з підтримкою SSE 3 [9].

Пристрої під керуванням операційної системи Android мають мати версію Android 6.0 (Marshmallow) і вище [9].

Пристрої під керуванням операційної системи IOS мають мати версію iOS 14 і вище [9].

1.6 Висновки за розділом 1

У першому розділи дипломної роботи був здійснений аналіз предметної області, відбулося ознайомлення з питанням вебфорумів, їх типами та особливостями функціонування, зокрема з форумами для розробників програмного забезпечення. Проведення аналізу існуючих рішень, їх переваг і недоліків дозволило виділити вимоги до створюваного застосунку, що зробить його конкурентоспроможним серед аналогів. Для подальшого обґрунтування рішень щодо реалізації застосунку були виділені функціональні й нефункціональні вимоги до нього.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

У попередньому розділі відбулося ознайомлення з вебфорумами в цілому і, окремо, з тематичними форумами для програмістів та їх особливостями. Проаналізовано переваги й недоліки існуючих аналогів, на основі чого було сформовано функціональні й не функціональні вимоги до створюваного застосунку, вимоги до безпеки та інтерфейсу.

У цьому розділі, спираючись на сформовані раніше вимоги, будуть визначені технічні питання, пов’язані із вибором мови програмування, середовищем для розробки й тестування створюваного застосунку а також основні рішення щодо його загальної структури.

2.1 Вибір інструментарію розробки програмного продукту

Перед початком проектування створюваного програмного продукту вкрай важливо обрати доцільні технології для реалізації необхідного функціоналу, що не будуть обмежувати розробку своїми можливостями й, водночас, не будуть надлишковими для реалізації задуманих функцій.

Оскільки функціонування вебфоруму супроводжується зберіганням, обробкою й обміном великого об’єму даних необхідно обрати підходящу систему керування базами даних (СКБД) для зберігання та маніпуляцій з даними, вибудувати доцільну архітектуру застосунку та способи взаємодії його компонентів, обрати мову програмування та середовище для розробки.

2.1.1 Вибір мови програмування

Сьогодні існує велика кількість мов, що дозволяють створювати вебзастосунки: JavaScript, PHP, Python, Ruby, C#, Java та інші. Кожна з них має свої переваги та недоліки, але найважливішим є те, що найкращої мови

серед них немає. Вибір мови для створення ВЗ залежить від конкретної поставленої задачі, складності та масштабів проекту. Вибір мови також може обумовлюватися не стільки її можливостями, скільки створеними для цієї мови готовими рішеннями та фреймворками.

Фреймворк (каркас) – сукупність програмних рішень, що задають структуру проекту і дозволяють полегшити створення складних систем.

Розглянемо три найпопулярніші мови: JavaScript, PHP та Python.

PHP (скорочено від Hypertext Preprocessor) – скриптова мова програмування, що виконується на стороні сервера і дозволяє створювати динамічні вебсторінки [10].

PHP є повністю серверною мовою, тобто генерація вебсторінки відбувається на сервері і передається на клієнт вже у готовому вигляді. Мова є об’єктно-орієнтованою, дозволяє повторно використовувати фрагменти коду та швидко масштабувати проект, має синтаксис подібний до мови C. Мова була створена у 1994 році і постійно оновлюється, переважна кількість сторінок в мережі Інтернет створена з її допомогою. Є велика кількість готових рішень, бібліотек та фреймворків для вирішення більшості задач, що стають перед розробниками [10].

Серед переваг PHP можна виділити наступні:

– простота вивчення новачками;
– зручна й ефективна робота з базами даних;
– підтримується більшістю серверів;
– простота тестування й налагодження коду;
– більшість задач можна виконати вбудованим в мову функціоналом;
– велика кількість бібліотек, фреймворків й розробників.

Серед недоліків можна виділити наступні:

– використання фреймворків у розробці значно сповільнює швидкість виконання застосунку;
– не дозволяє модифікувати базову поведінку ВЗ.

Python – популярна скриптова мова загального призначення, може використовуватися для створення вебсайтів з серверним компонуванням сторінок та серверних частин ВЗ, написаних з використанням інших технологій [11].

Python характеризується динамічною типізацією, об’єктно-орієнтованим підходом до програмування, простим синтаксисом. Мова є популярною через простоту створення ПЗ, велику кількість фреймворків та готових рішень для вирішення задач [11].

До переваг Python можна віднести наступні:

– простота вивчення й написання коду;

– можливість запуску на більшості популярних платформ;

– велика кількість вбудованих бібліотек, готових рішень і розробників.

До недоліків можна віднести наступні:

– низька швидкість виконання, у порівнянні з іншими мовами;

– більше споживання пам’яті.

На останок розглянемо мову JavaScript. Це одна з найпопулярніших скриптових мов програмування ВЗ, є універсальною і може виконуватися як на стороні сервера, так і на клієнті.

Мова характеризується динамічною типізацією, підтримує об’єктно-орієнтований, імперативний та декларативний стилі програмування. На початку мова створювалася для додавання інтерактивності вебсторінкам, які раніше являли собою статичний HTML-код з CSS-стилями. Пізніше JavaScript значно розвинулася, розширила свої можливості і стала універсальною мовою написання скриптів, що може використовуватися для рішення найрізноманітніших задач [12].

JavaScript підтримується усіма популярними веббраузерами, може підключатися до сторінки, або вбудовуватися безпосередньо у HTML-розмітку. З використанням додаткових рішень, таких як, наприклад, фреймворки Electron та Ionic можна створювати мобільні й класичні

застосунку, що виконуються за межами браузера, виглядають як звичайні застосунки і мають доступ до використання системних функцій [12].

Середовище виконання Node.js дозволяє використовувати JavaScript для створення серверних частин ВЗ, роботи з базами даних, файлами.

Розвиток апаратної частини пристроїв, що запускають веббраузери дозволило перенести задачу компонування вебсторінок і частину бізнес логіки з сервера на клієнт. Такий підхід значно пришвидшує роботу застосунку, знімає частину навантаження з вебсервера і наділяє ВЗ поведінкою, схожою на класичні програми.

До переваг JavaScript можна віднести наступні:

– зниження навантаження сервера;
– висока швидкість виконання;
– універсальність;
– велика кількість готових бібліотек та фреймворків;
– популярність мови у розробників ВЗ.

До недоліків можна віднести наступні:

– програмний код клієнту завжди видимий;
– складність пошуку помилок та налагодження коду.

Для наочності представимо порівняння у вигляді таблиці 2.1.

Таблиця 2.1 – Порівняння оглянутих мов програмування

Критерій порівняння	PHP	Python	JavaScript
Можливість виконання на клієнті	-	-	+
Зручне налагодження	+	+	-
Кросплатформовість	+	+	+
Швидкість виконання	+	-	+
Спеціалізація на веброзробці	+	-	+
Зрозуміла та повна документація	+	+	+
Готові рішення та фреймворки	+	+	+

З таблиці видно, що усі мови мають свої особливості і виділити найкращу не можна. Оскільки застосунок, що розроблюється матиме

клієнтську та серверну частину, вирішено обрати мову JavaScript, як єдину універсальну з перелічених. Використання однієї мови і подібних технологій дозволить полегшити розробку обох частин проекту та їх взаємодію між собою. Мова має зрозумілу та повну документацію, велику кількість готових бібліотек та фреймворків, що дозволить пришвидшити процес розробки та побудови архітектури використовуючи готові рішення.

2.1.2 Вибір СКБД

У галузі технологій баз даних існує два основних напрями: SQL та NoSQL, реляційні та нереляційні БД. Відрізняються вони підходом до зберігання інформації, типами даних та загальним проектуванням.

Порівняємо реляційні та нереляційні БД для вибору більш доцільного рішення для вирішення поставленої задачі. Для зручності порівняння представимо у вигляді таблиці 2.2.

Таблиця 2.2 – Порівняння реляційних та нереляційних БД [13]-[14]

Критерій порівняння	SQL БД	NoSQL БД
Модель зберігання даних	Таблиці з фіксованим набором строк і стовбців	JSON-документи, пари ключ-значення, таблиці з динамічними стовбцями
Схема даних	Жорсткий, фіксований	Гнучкий
Необхідність операції об’єднання	Зазвичай необхідна	Зазвичай не потребується
Основне призначення	Загальне призначення	Документи: загальне призначення, ключ-значення: великі об’єми даних з простими запитами пошуку, динамічні стовбці: великі об’єми даних з передбачуваними запитами
Масштабування	Вертикальне, потребує збільшення сервера	Горизонтальне, розширюється додаванням додаткових серверів

Продовження таблиці 2.2

Критерій порівняння	SQL БД	NoSQL БД
Перетворення даних в об’єкт	Потребується ORM (об’єктно-реляційна проекція)	Більшість БД не потребують ORM, в MongoDB документи перетворюються в структури даних більшості популярних мов програмування без посередників
Історія виникнення	Розроблені у 1970-х роках з фокусом на зменшенні повторення даних	Розроблені наприкінці 2000-х років з фокусом на масштабування і швидкій перебудові застосунків

Оскільки для функціонування вебфоруму не так важлива швидкість оновлення даних в БД, як швидкість їх отримання й відображення на сторінці, то нереляційний підхід краще підходить для використання. Документо-орієнтована БД дозволить зберігати більшість сутностей в одному документі для кожної, без необхідності операцій об’єднання, що значно прискорить отримання даних й відправку їх на клієнт. Для функціонування вебфоруму необхідний значний об’єм пам’яті для зберігання інформації, що постійно додається, і дуже зручною є можливість горизонтального масштабування, з можливістю розподілу бази на кілька серверів.

Проаналізувавши особливості оглянутих БД, для використання в створюваному ВЗ було вирішено обрати нереляційний підхід, а конкретно СКБД MongoDB, як одну з найпопулярніших і найрозвинутіших представників документо-орієнтованих БД.

2.1.3 Вибір середовища розробки

Для написання програмного коду, його тестування, форматування, налагодження слід обрати зручне середовище для розробки. Існує велика кількість IDE та редакторів коду, що чудово підходять для розробки на

JavaScript, наприклад Visual Studio Code, Atom, WebStorm, Sublime Text, Brackets. Розглянемо особливості найбільш розвинених та популярних у розробників – Visual Studio Code (VS Code) та WebStorm.

VS Code – легкий, швидкий, багатомовний та глибоко налаштовуємий безкоштовний редактор з відкритим вихідним кодом, розроблений компанією Microsoft. Редактор доступний для операційних систем Windows, Linux та OSX, а також має вебверсію, що можна відкрити у будь-якому браузері. Редактор універсальний і підтримує більшість популярних мов програмування [15].

Базова версія редактора не настільки багата функціоналом, як повноцінні IDE, але потрібний функціонал можна легко додати, інсталювавши потрібний модуль. VS Code функціонує вже 7 років, має велику спільноту ентузіастів, що створюють розширення, тому необхідний функціонал зазвичай знайти та додати досить легко [15].

Редактор надає функціонал для налагодження коду з точками зупинки та стеками виклику, інтелектуальне автодоповнення [15].

У редактор інтегровано зручний інструментарій для роботи з Git, є можливість перегляду здійснених змін, пошуку конфліктів при злитті гілок. Усі необхідні операції з Git-репозиторієм можна виконати з використанням візуального інтерфейсу, що надається [15].

До недоліків можна віднести необхідність налаштування багатьох розширень при додаванні функціоналу. Через те, що розширення можуть вільно додаватися будь-якими користувачами, вони можуть іноді містити помилки та функціонувати не так, як очікується.

WebStorm – інтуїтивне, платне середовище для розробки (IDE), спеціалізоване на мові JavaScript, ліцензоване компанією JetBrains. Середовище доступне для операційних систем. Середовище доступне для операційних систем Windows, Linux та OSX [16].

Середовище, на відміну від VS Code має багато функцій, вбудованих в базову версію. Функціонал можна також доповнити, інсталювавши плагіни, що надаються IDE [16].

IDE дуже зручно використовувати для розробки під Node.js та використання різноманітних JavaScript-фреймворків. Середовище дозволяє генерувати базовий каркас проекту і надає поради відповідно до використаної технології. Також WebStorm надає функціонал для роботи з базами даних [16].

WebStorm кращий у рефакторингу JavaScript-коду, він дозволяє переносити функції та змінні з одного файлу до іншого. Середовище проводить статичний аналіз коду й надає корисні поради щодо його покращення, наприклад, позначає повторення фрагментів коду, невикористаний код, дає поради щодо стилю написання коду [16].

У WebStorm також інтегровано інструменти для зручної роботи з Git, відображення змін у файлах. Редактор надає більш зручний функціонал злиття гілок та пошуку конфліктів [16].

До недоліків можна віднести необхідність платної підписки для використання IDE, що може не сподобатися великій кількості користувачів. Існують місячний та річний тарифи оплати, а також надається 30 днів безкоштовного пробного періоду.

Для наочності зведемо порівняння у таблиці 2.3.

Таблиця 2.3 – Порівняння оглянутих середовищ розробки

Критерій порівняння	WebStorm	VS Code
Необхідність платної підписки	+	-
Інтелектуальне доповнення	+	+
Кросплатформовість	+	+
Зручний рефакторинг коду	+	+
Інтегрований Git-інструментарій	+	+
Можливість встановлення сторонніх розширень	-	+

Проаналізувавши особливості оглянутих середовищ для розробки програмного продукту було обрано редактор коду Visual Studio Code, в більшою мірою через можливість безкоштовного використання, легковісність та варіативність розширень, що можна інсталювати.

2.2 Архітектура застосунку та використані технології

Використання мови JavaScript дає широкий вибір підходів до створення ВЗ, серед яких популярним є створення односторінкових застосунків (SPA). Такий тип ВЗ дозволяє перенести задачу побудови інтерфейсу користувача на клієнтську частину, поведінка таких застосунків більше схожа на класичні програмні рішення, ніж на вебсайти [17].

До переваг односторінкових ВЗ можна віднести:

– динамічна і миттєва зміна елементів сторінки, як у десктопних застосунках;

– відділений від сервера рівень представлення;

– зняття навантаження по відображенню з серверів;

– зменшення часу очікування;

– простіше обслуговування коду [17].

Проаналізувавши переваги було вирішено проектувати створюваний проект, як односторінковий ВЗ.

Існує велика кількість JavaScript-фреймворків для побудови клієнтської частини SPA-застосунків. Серед найпопулярніших можна виділити React, Vue, Angular, BackBone, Meteor. Усі ці фреймворки досить схожі один на одного, але кожен має свої переваги і недоліки. Для розробки застосунку було обрано фреймворк Vue.js, серед його переваг можна виділити наступні:

– використання архітектури MVVM дозволяє двосторонню комунікацію, спрощуючи обробку блоків HTML;

– використання віртуальної копії DOM сторінки, що значно пришвидшує і оптимізує її перебудову;

– використання двосторонньої прив’язки даних спрощує їх оновлення;

– постійне вдосконалення і велика спільнота розробників;

– невеликий розмір самої бібліотеки;

– докладна і зрозуміла документація [18].

Для створення локального сховища на клієнті, що буде доступним із усіх його компонентів було використано бібліотеку Pinia.

Для швидкої побудови адаптивного інтерфейсу, в більшості його елементів використано Vue-фреймворк Quasar.

Для реалізації WYSIWYG-редактора тексту використано JavaScript-бібліотеку TinyMCE.

Для підсвічування синтаксису програмного коду у темах та повідомленнях використано JavaScript-бібліотеку Prism.js.

Для зв’язку з базою даних MongoDB, побудови схем об’єктів, виконання запитів використано бібліотеку JavaScript-бібліотеку mongoose.

Для роботи SPA-застосунку з БД необхідна серверна частина. Середовище виконання Node.js дозволяє створювати серверну частину за допомогою мови JavaScript. Для побудови серверної частини було обрано Node.js-фреймворк Express.

Express – один з найбільш підтримуваних і найчастіше використовуваних Node.js-фреймворків. До його переваг зв’язки Node.js-Express можна віднести наступні:

– швидка розробка застосунку;

– мінімізує витрати на розробку застосунків через використання JavaScript як на клієнті так і на сервері;

– чудова продуктивність;

– зручне оброблення запитів шляхом побудови ланцюжків проміжних обробників (middleware);

– доступ до використання JavaScript-бібліотек [19].

Серверна частина побудована згідно архітектури REST. Обмін даними між клієнтом і сервером відбувається за допомогою HTTP-запитів у вигляді JSON-об’єктів для текстової інформації та бінарних файлів для обміну файлами. Кожній окремій одиниці даних відповідає конкретне URL-посилання. Дії над даними визначаються методом HTTP запиту (GET, POST, PUT, DELETE та інші).

Загальну архітектуру клієнт-серверного застосунку, зв’язки зі сховищами та БД, комунікацію між клієнтом і сервером представлено на рисунку 2.1.

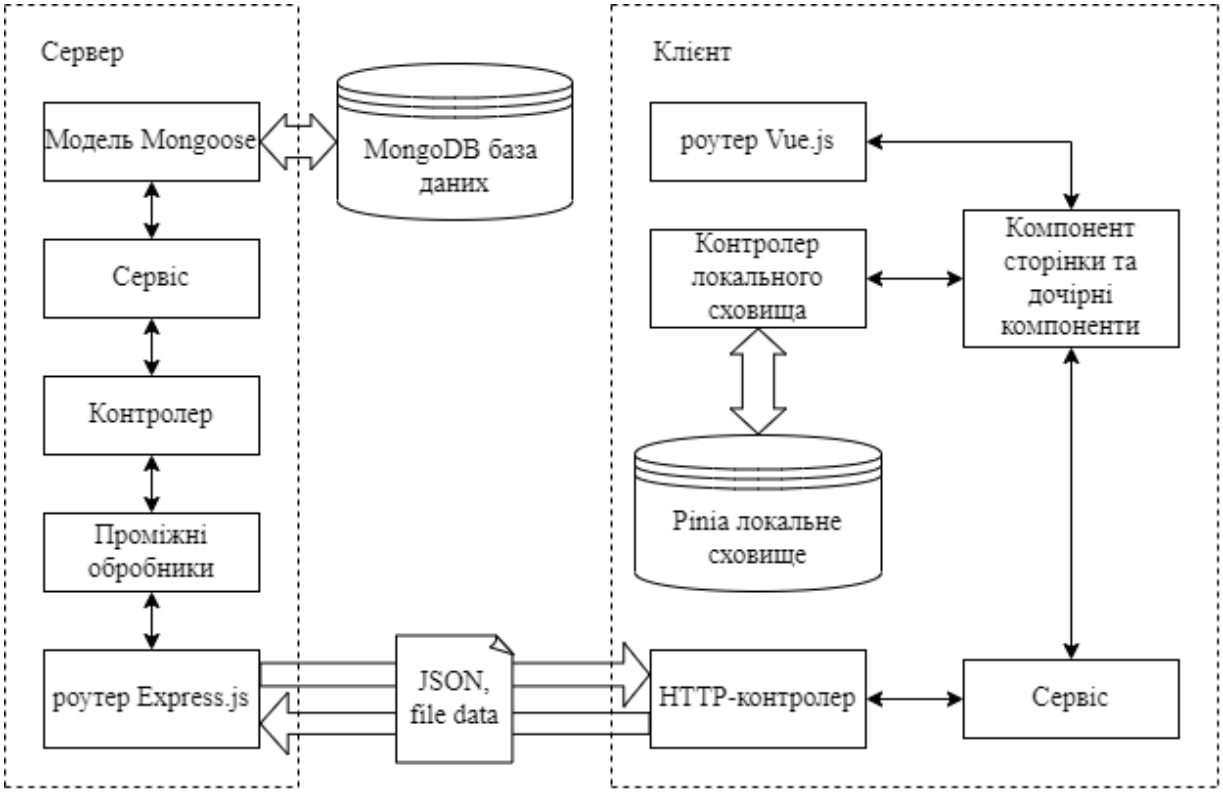

Рисунок 2.1 – Архітектура застосунку

ВЗ, що розроблюється, матиме чотири категорії користувачів: гість (незареєстрований користувач), зареєстрований користувач, модератор та адміністратор.

Незареєстрований користувач матиме право переглядати усі розділи та підрозділи, теми, відповіді у них, здійснювати сортування та скачувати прикріплені файли. У зареєстрованого користувача з'являється можливість створювати нові теми, повідомлення, відповіді на повідомлення, оцінювати теми та повідомлення, прикріплювати до них файли, редагувати свій профіль користувача.

Користувачі з правами модератора можуть редагувати та видаляти будь які теми та повідомлення в цілях модерації наповнення форуму.

Адміністратор має можливість редагувати список розділів та підрозділів, змінювати права доступу та статус користувачів.

Для наглядності структурна схема проекту, що розроблюється наведена на рисунку 2.2.

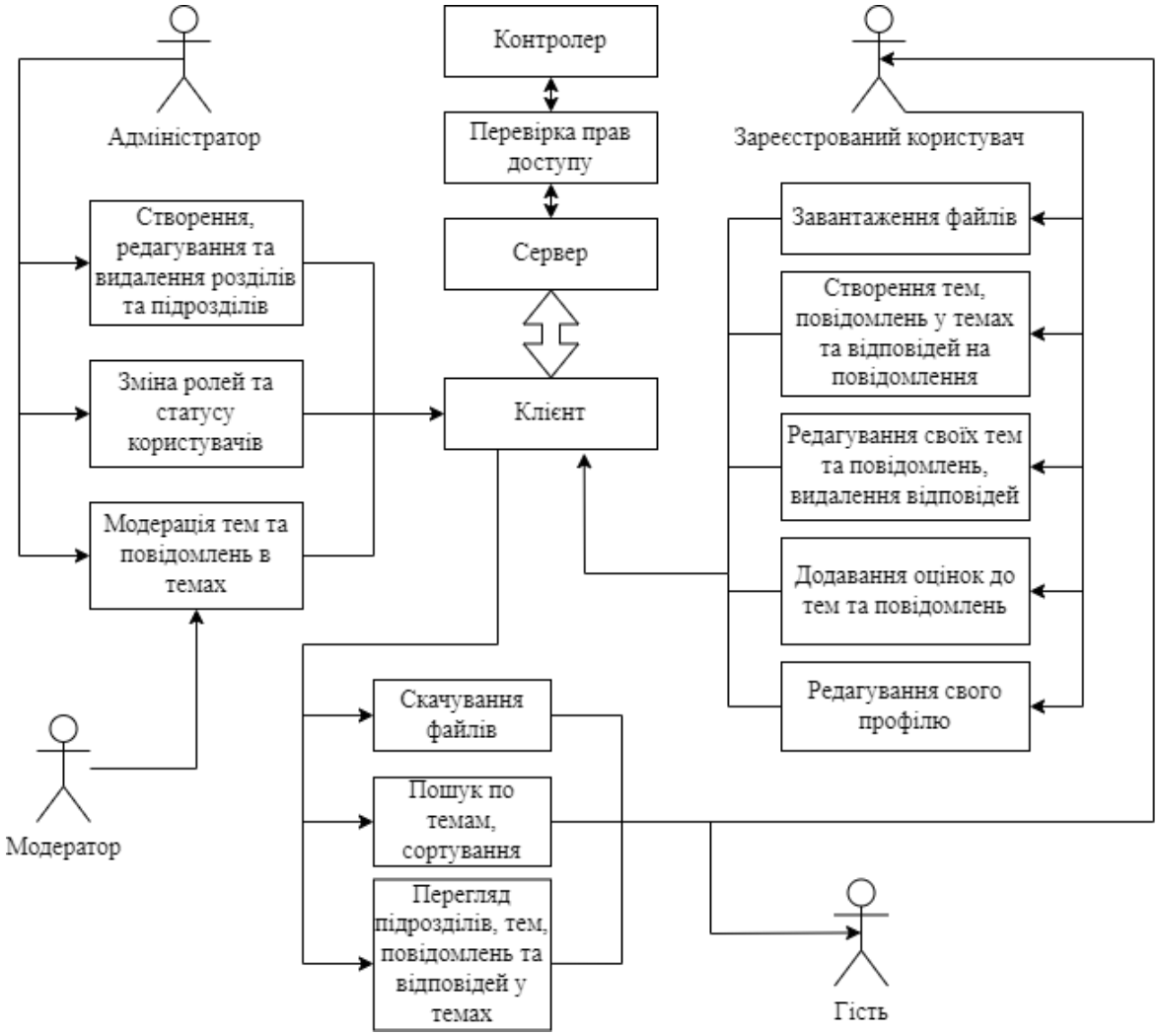

Рисунок 2.2 –Структурна схема застосунку

2.3 Висновки за розділом 2

У другому розділі дипломної роботи були досліджені технічні аспекти розробки ВЗ. В якості мови програмування для розробки було обрано JavaScript через її універсальність та можливість розробки SPA-додатків. В якості СКБД обрано MongoDB, через швидкий пошук інформації та можливість горизонтального масштабування. В якості середи розробки обрано редактор VS Code. Для полегшення й прискорення розробки клієнтської частини вирішено використовувати фреймворки Vue.js та Quasar, серверної – Express.js. Була чітко визначена архітектура застосунку, його структурна схема, розподіл прав доступу користувачів, методи обміну даних між клієнтом та сервером і додаткові програмні засоби, використані при розробці.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Основним завданням роботи є створення застосунку вебфоруму. Головною функцією форумів є перегляд, пошук та створення тем та повідомлень. Щоб вирішити цю‑задачу необхідно правильно спроектувати моделі даних, алгоритми їх обробки та продумати взаємодію з користувачем.

3.1 Використані моделі даних

Створена користувачами інформація повинна бути швидко доступною і має тривалий час зберігатися на сервісі. Тому важливо побудувати оптимальну модель бази даних, що забезпечить швидкодію отримання інформації. Для зручності спочатку побудуємо об’єктно-орієнтовану модель. Діаграму БД можна побачити на рисунку 3.1.

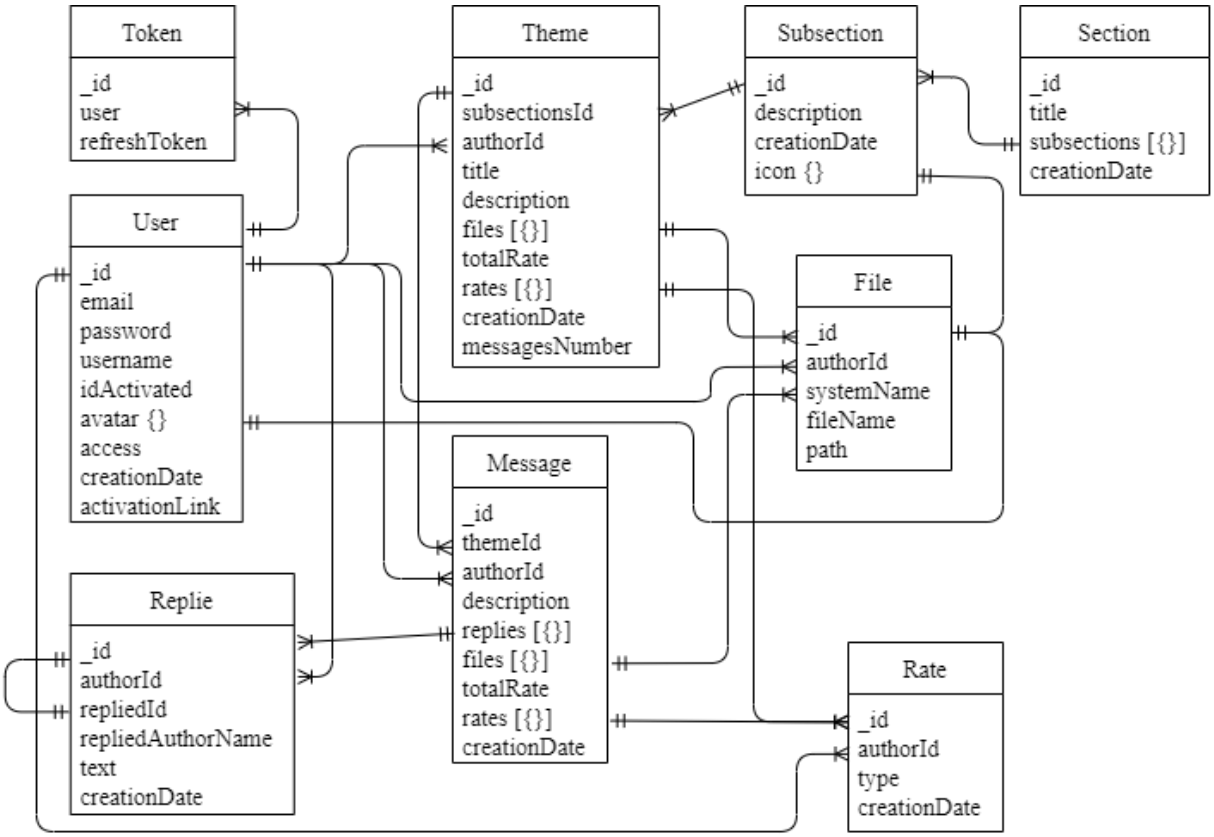

Рисунок 3.1 – Діаграма БД застосунку

Оскільки створюваний ВЗ використовує СКБД MongoDB, на основі спроектованої об’єктно-орієнтованою моделі за допомогою JavaScript-бібліотеки mongoose були описані ієрархічні схеми даних. Описання схем даних необхідне для роботи з БД, створення та маніпуляцій з колекціями документів в ній.

Загалом можна виділити наступні документи: User, Token, Section. Theme та Message. Піддокументи Subsection, Rate, Replie та File зберігаються в складі основних сутностей. Призначення документів наведене в таблиці 3.1

Таблиця 3.1 – Призначення документів БД

Назва документу / піддокументу	Призначення
User	Зберігає інформацію про зареєстрованого користувача, його рівень доступу, статус, посилання для активації, фото профілю
Token	Зберігає refresh-токени користувачів для поновлення авторизації при закінченні терміну життя access-токену
Section	Зберігає інформацію про розділи форуму та мість в собі масив підрозділів
Theme	Зберігає інформацію про створену користувачем тему, виставлені їй оцінки, прикріплені файли та кількість повідомлень
Message	Зберігає інформацію про створене користувачем повідомлення в темі, виставлені йому оцінки, прикріплені файли та вміщує в собі масив відповідей на повідомлення
Subsection	Піддокумент, зберігає інформацію про один підрозділ та його іконку
Rate	Піддокумент, зберігає інформацію про одну оцінку, її тип та автора
Replie	Піддокумент, зберігає інформацію про відповідь на повідомлення
File	Піддокумент, зберігає інформацію про один файл, шлях до нього, оригінальне й унікальне системне ім’я

Використання піддокументів дозволяє виключити процес об’єднання при виборі документу з ними, значно прискорюючи пошук та отримання клієнтом необхідних для відображення даних. Такий підхід дещо сповільнює

й ускладнює процес оновлення самих піддокументів, але для застосунку вебфоруму ключову роль грає саме швидкість вибірки даних.

3.2 Алгоритм введення даних у застосунок

Оскільки основною задачею застосунку є запис та управління даними, вкрай важливо забезпечити правильний алгоритм їх обробки та валідації.

Для збереження цілісності даних та уникнення критичних помилок в роботі застосунку перевірка даних виконується як на клієнті, так і на сервері. На клієнті це відбувається шляхом валідації форм в процесі їх заповнення даними та перед відправкою на сервер. На сервері валідація даних відбувається шляхом використанням спеціальних проміжних обробників для валідації перед відправкою даних до контролерів. Якщо це необхідно, то перевірка може бути доповненою в самому контролері. Як останній рубіж перевірки правильності надісланих даних виступає сама СКБД Mongodb, перевіряючи вміст запитів до бази згідно правил, що були прописані в процесі опису схем документів.

Створюваний ВЗ надає користувачам можливість прикріплювати файли довільного формату до тем та повідомлень. Якщо користувач обрав файл, або декілька файлів для завантаження, вони додаються до HTTP-запиту й відправляються для зберігання на сервер разом з текстовим наповненням. Зазвичай для цього використовуються POST та PUT запити з HTTP-параметром типу навантаження «multipart/form-data».

Коли сервер у запиті отримує перелік файлів, для кожного файлу за допомогою бібліотеки uuid генерується унікальне системне ім’я. Далі кожен файл перейменовується і, з новою назвою, записується у відповідний до запиту каталог на сервері. Після цього в документ створеної теми чи повідомлення додається піддокумент для кожного файлу, що містить інформацію про його оригінальну назву, автора, системне ім’я та шлях

зберігання на сервері. Ці дані в подальшому використовуються для скачування файлів зі збереженням оригінальної назви.

Блок схему алгоритму створення нової теми можна побачити на рисунку 3.2. Алгоритм створення повідомлень також практично ідентичний нижче наведеному.

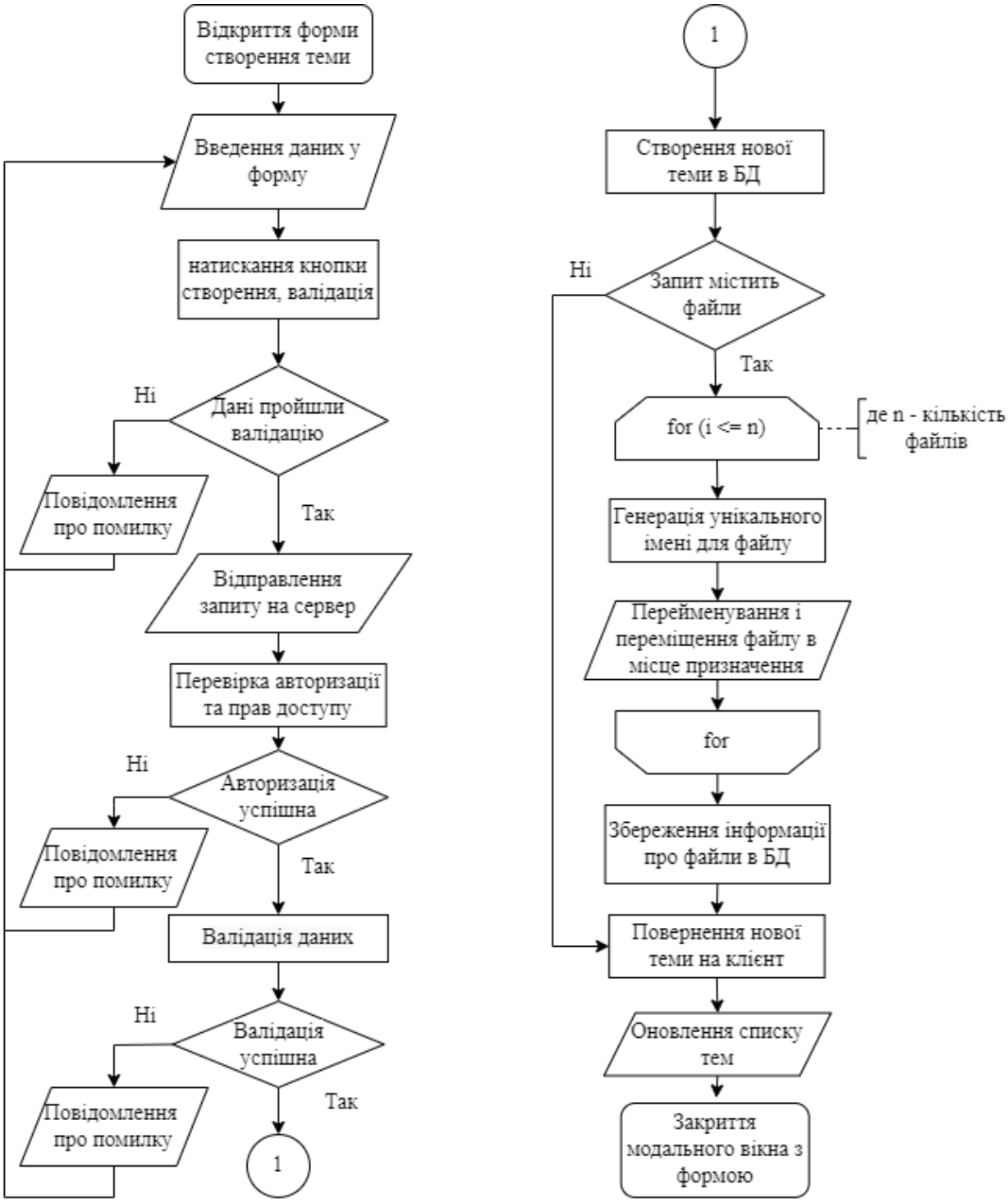

Рисунок 3.2 – Блок-схема алгоритму створення нової теми

3.3 Механізм авторизації користувачів

Створюваний ВЗ повинен реалізувати поділ прав доступу між різними категоріями користувачів та забезпечувати механізми реєстрації та авторизації. Для забезпечення високої безпеки в цих процесах було вирішено використати підхід з парою access і refresh JWT-токенів.

Блок схему алгоритму реєстрації можна побачити на рисунку 3.3

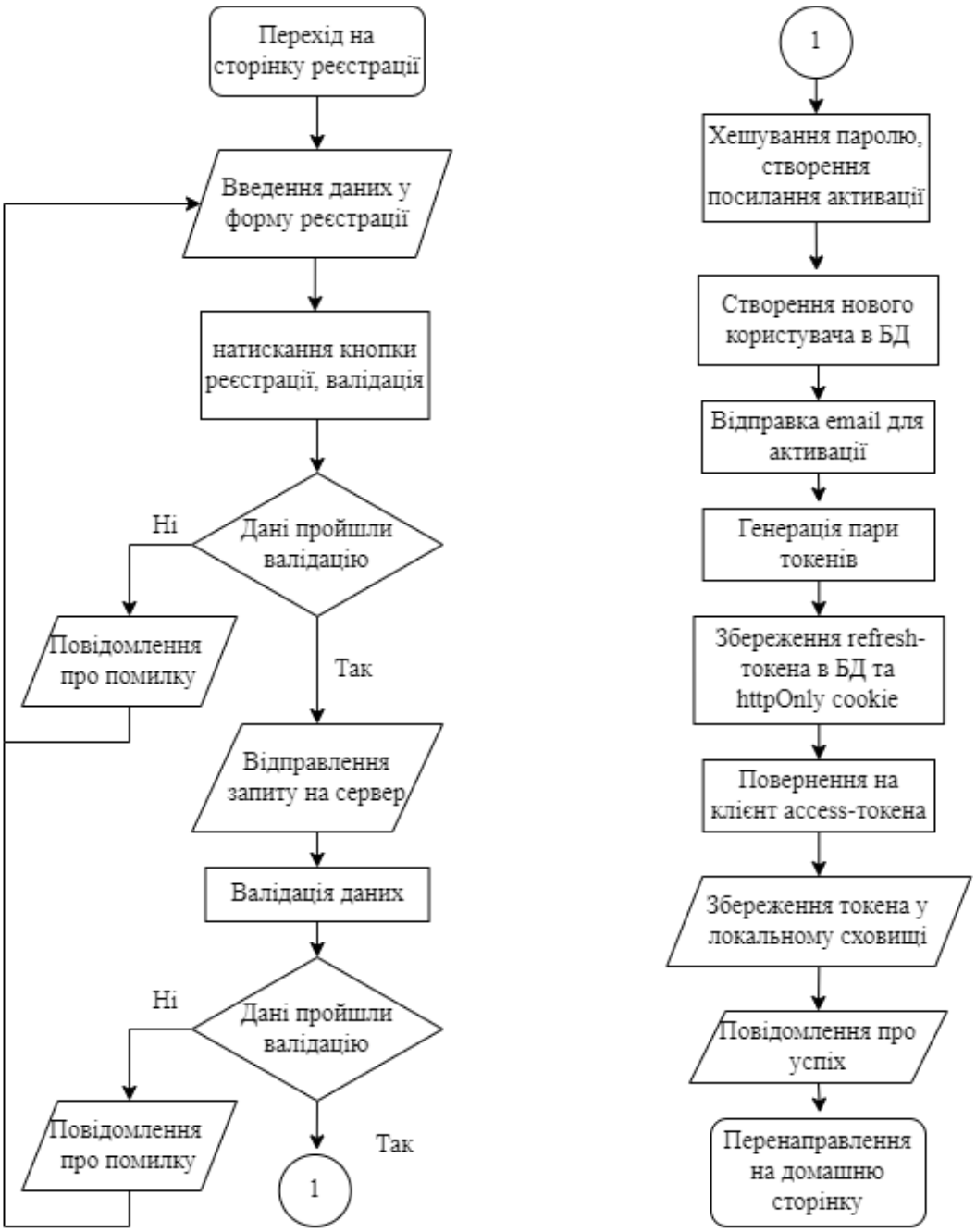

Рисунок 3.3 – Блок-схема алгоритму реєстрації

Під JWT-токеном мають на увазі деяку зашифровану строку формату «header.payload.signature», де header – це заголовок, що містить дані про алгоритм хешування, payload – це зашифрований JSON документ, що містить корисні дані, а signature – підпис, що вираховується з заголовку, даних і вказаного секретного ключа. При створенні токенів зазначається час їх життя, після проходження якого вони вважаються не дійсними [20].

При авторизації з використанням JWT-токенів зазвичай застосовується пара access і refresh токенів [20].

Access-токен зазвичай має невеликий період життя, зберігається на клієнті і в корисному навантаженні містить інформацію про користувача. Він відправляється разом з запитами на сервер і, за необхідності, може використовуватися для перевірки авторизації користувача, або для отримання даних про нього без додаткового звертання в БД [20].

Refresh-токен зазвичай має тривалий період життя, зберігається в БД і у httpOnly-куках, які неможливо змінити з клієнтської сторони. Використовується він для оновлення access-токенів після закінчення періоду їх життя. При оновленні access-токену перезаписується також і refresh-токен.

Інколи використовується декілька access-токенів, прив’язаних до різних пристроїв користувача. Також існують рішення з декількома refresh-токенами, щоб запобігти необхідності авторизації користувача після закінчення терміну його життя [20].

У додатку, що розроблюється використовується одна пара access-refresh токенів. При реєстрації чи авторизації створюється пара токенів, refresh пишеться в БД та httpOnly-куки, access – відправляється на клієнт

Якщо на сервері виникає помилка при валідації access-токена, клієнт відправляє запит на оновлення з refresh-токеном із httpOnly-кук. На сервері refresh-токен перевіряється на валідність і порівнюється зі збереженим в БД. Якщо проблем немає, то на генерується нова пара токенів й замінює старі. У іншому випадку – клієнт вважається неавторизованим і потребується повторний вхід в систему.

Блок-схему алгоритму авторизації можна побачити на рисунку 3.4

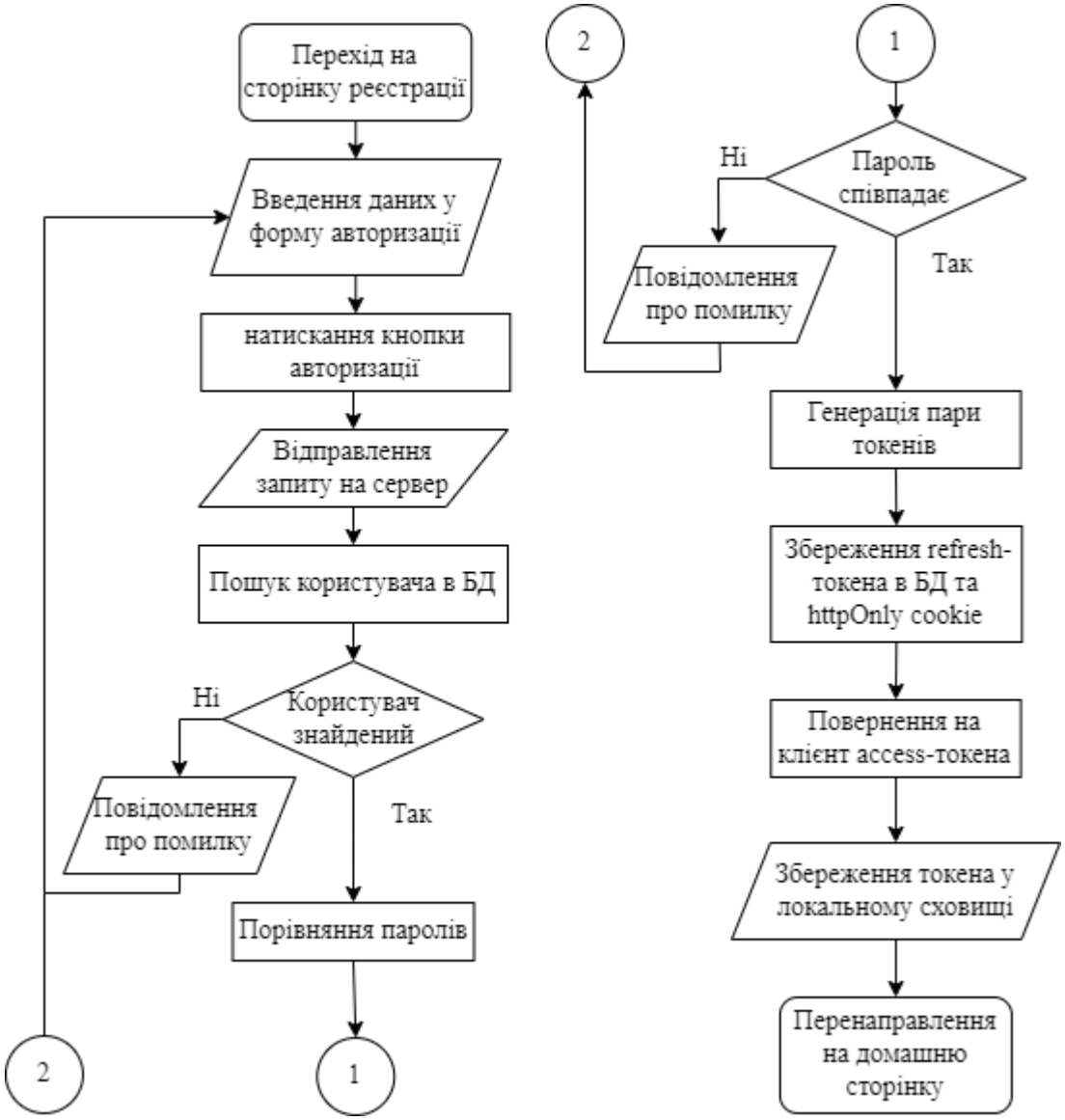

Рисунок 3.4 – Блок-схема алгоритму авторизації

3.4 Висновки за розділом 3

У цьому розділі було розроблено об’єктно-орієнтовану модель БД, на основі якої було описано схеми для роботи з документами. Був розглянутий загальний алгоритм введення даних в систему та їх валідації. Описаний підхід до авторизації користувачів з використанням access та refresh токенів. Наведені блок-схеми алгоритмів створення нової теми, реєстрації та авторизації.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання

Розроблений клієнт-серверний ВЗ призначений для обговорення різних тем, пов’язаних з розробкою вебзастосунків, пошуку рішення проблем, прикладів використання бібліотек та фреймворків, фрагментів коду.

Мінімальні системні вимоги до середовища для розгортання і доопрацювання серверної частини ВЗ:

– CPU: Intel Core i5;
– RAM: 4 Gb;
– HDD/SSD: 1Gb;
– вебсервер: Apache 2 або Nginx;
– СКБД MongoDB;
– порти: 80 (HTTP), 27018 (стандартний порт MongoDB);
– програмне забезпечення: середовище Node.js, менеджер пакетів NPM або Yarn.

Мінімальні системні вимоги до середовища для розгортання і доопрацювання клієнтської частини ВЗ:

– CPU: двоядерний процесор, з підтримкою SSE3-інструкцій;
– RAM: 4 Gb;
– HDD/SSD: 1Gb;
– Операційна система Windows 7, OS X El Capitan, Linux;
– програмне забезпечення: браузер, менеджер пакетів NPM або Yarn.

4.2 Характеристики програмного продукту

Розроблений ВЗ складається із клієнтської частини, що відповідає за побудову інтерфейсу та взаємодію з користувачем і серверної частини, відповідальної за маніпуляції з даними у БД та авторизацію користувачів.

Клієнтська частина являє собою SPA-додаток, розроблений з використанням фреймворку Vue.js, для побудови деяких елементів інтерфейсу використовується фреймворк Quasar. Для створення локального сховища даних використано бібліотеку Pinia. Файлова структура клієнтської частини представлена на рисунку 4.1. Опис вмісту і призначення каталогів клієнтської частини наведено в таблиці 4.1.

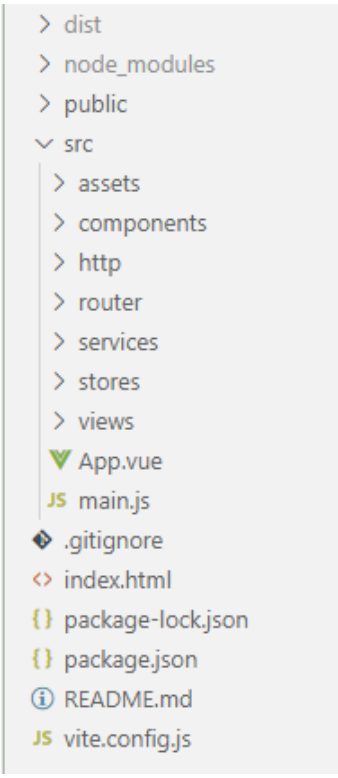

Рисунок 4.1 – Файлова структура клієнтської частини застосунку

Таблиця 4.1 – Вміст і призначення каталогів клієнту

Назва каталогу	Вміст і призначення
Кореневий	Містить файли з налаштуваннями та головний html-файл, який являється точкою входу
dist	Містить зібраний та спакований застосунок, готовий до розгортання на сервері
node_modules	Вміщує в собі всі файли бібліотек, підключених через менеджер пакунків npm
public	Містить загальнодоступні файли
assets	Містить css файли і зображення, необхідні для роботи застосунку

Продовження таблиці 4.1

Назва каталогу	Вміст і призначення
components	Містить файли формату «.vue» з компонентами, що використовуються в проекті
http	Містить налаштований сервіс для відправки HTTP-запитів
router	Містить опис URL-маршрутів клієнту і відповідність їм компонентів сторінок
services	Містить файли сервісів, у яких здійснюється зв’язок клієнту з сервером
stores	Містить імплементацію локального сховища даних
views	Містить головні компоненти сторінок додатку

Серверна частина додатку функціонує в середовищі виконання JavaScript-коду Node.js, розроблена з використанням фреймворка Express.js, за допомогою бібліотеки mongoose здійснює маніпуляції з даними в СКБД MongoDB. Файлова структура серверної частини представлена на рисунку 4.2. Опис вмісту і призначення каталогів серверної частини наведено в таблиці 4.2.

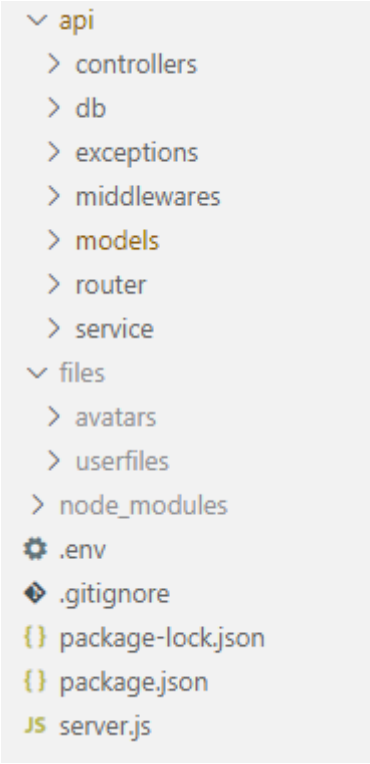

Рисунок 4.2 – Файлова структура серверної частини застосунку

Таблиця 4.2 – Вміст і призначення каталогів серверної частини

Назва каталогу	Вміст і призначення
Кореневий	Містить файли з налаштуваннями та головну точку входу
avatars	Містить файли фото профілю користувачів та іконки підрозлілів
userfiles	Містить файли, які користувачі прикріпили до тем та повідомлень
node_modules	Містить в собі всі файли бібліотек, підключених через менеджер пакунків npm
router	Містить опис URL-маршрутів додатку та призначених для їх опрацювання контролерів та проміжних обробників
models	Містить опис схем документів для роботи з БД
db	Містить файл з реалізацією підключення до БД
exceptions	Містить власну реалізацію помилок для відправки на клієнт
middlewares	Містить проміжні обробники для опрацювання запитів
controllers	Містить контролери для обробки запитів
services	Містить сервіси, що виконують маніпуляції з даними в БД

Серверна частина побудована згідно архітектури REST. Обмін даними між клієнтом і сервером відбувається за допомогою HTTP-запитів у вигляді JSON-об’єктів для текстової інформації та бінарних файлів для обміну файлами.

4.3 Звернення до програмного продукту

Актуальний код продукту можна склонувати використовуючи систему контролю версій Git за посиланнями: «https://gitlab.com/skusch/forum-server.git» – для серверної частини і «https://gitlab.com/skusch/forum-client.git» – для клієнтської відповідно. Далі необхідно відкрити системну консоль, перейти в корінь склонованих каталогів і підтягнути всі необхідні для роботи застосунків бібліотеки консольною командою «npm install». Після завершення інсталювання бібліотек можна запустити як клієнтську, так і серверну частину для локального виконання консольною командою «npm run start» в обох

каталогах. Після чого отримати доступ до клієнтської частини можна в браузері за посиланням «http://localhost:3000/». Запити до сервера необхідно відсилати за адресою «http://localhost:3090/api/». Зупинити локальне виконання застосунків можна комбінацією клавіш «Ctrl+C» в консолі.

4.4 Вхідні та вихідні дані

Вхідними даними розробленого ВЗ є текстові дані, які користувачі вписують у форми при створенні тем, повідомлень та відповідей, а також файли довільного формату, які користувачі можуть завантажувати для прикріплення до тем та повідомлень. Приклад заповнення форми створення теми з використанням WYSIWYG-редактора тексту і завантаження файлу наведено на рисунку 4.3.

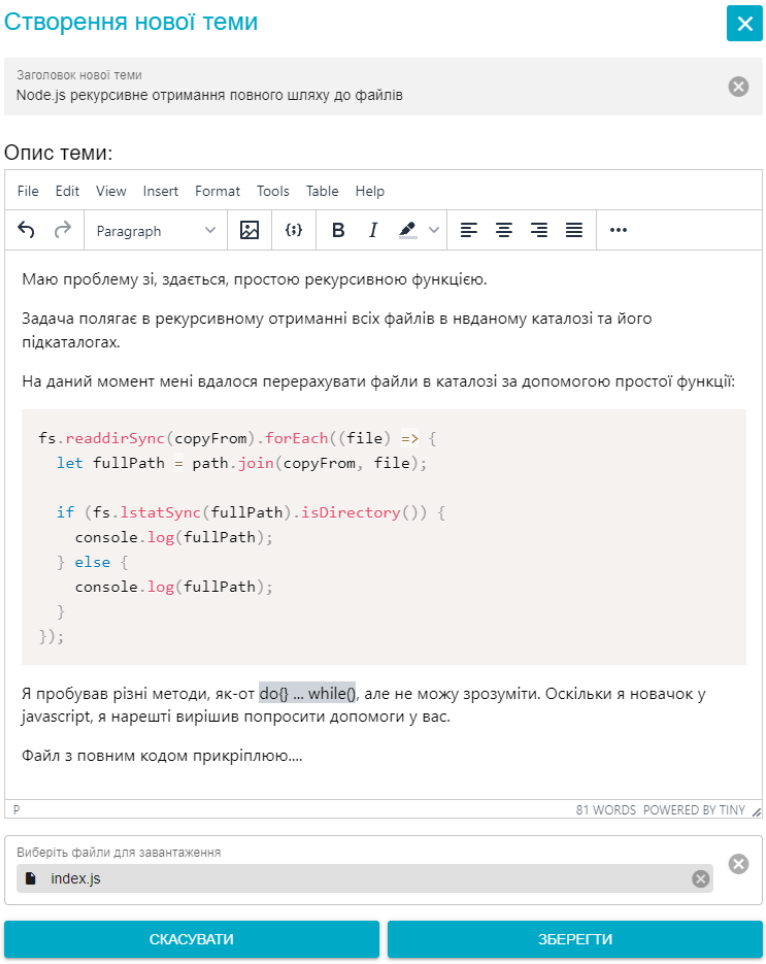

Рисунок 4.3 – Приклад вхідних даних для при створенні теми

Вихідними даними є відображення на сторінках збереженої в БД текстової інформації, зображень, посилань, а також доступ до скачування файлів, завантажених користувачами, за посиланнями. Приклад відображення відповіді та посилання на скачування файлів, прикріплених до теми наведено на рисунку 4.4.

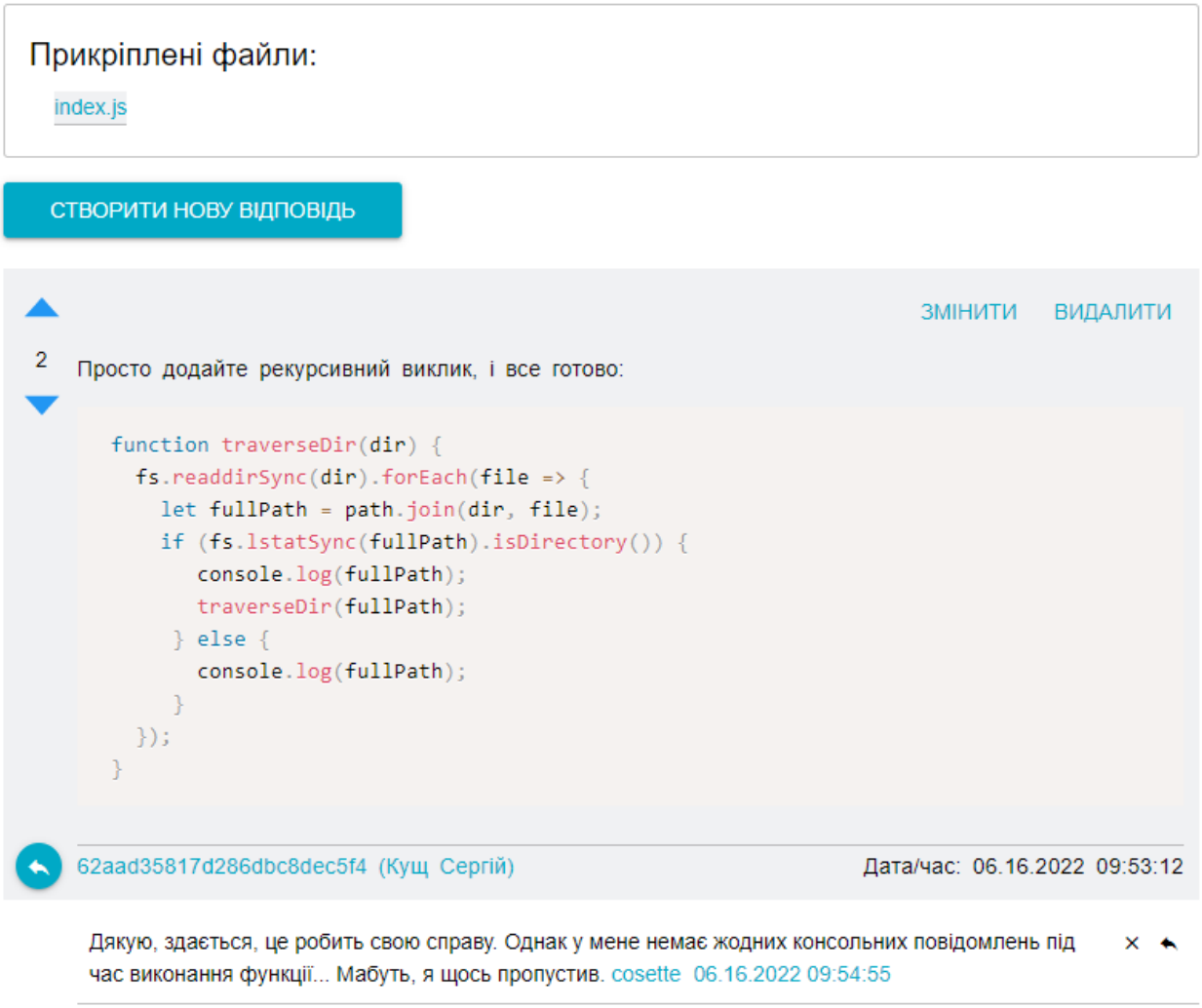

Рисунок 4.4 – Приклад вихідних даних

4.5 Висновки за розділом 4

У цьому розділі було визначене призначення розробленої системи. Був сформований перелік мінімальних системних вимог для роботи й доопрацювання серверної і клієнтської частини окремо. Складено інструкцію

з розгортання ВЗ локально для доопрацювання. Було описано файлову структуру застосунку, його архітектуру й основні використані при розробці технології, визначено вхідні й вихідні дані з прикладами.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення й умови виконання програмного продукту

Розроблений клієнт-серверний ВЗ призначений для обговорення різних тем, пов’язаних з розробкою вебзастосунків, пошуку рішення проблем, прикладів використання бібліотек та фреймворків, фрагментів коду.

Для використання клієнтами розробленого ВЗ достатньо мати пристрій з відносно новою версією одного з популярних веббраузерів та стабільне з’єднання з мережею Інтернет.

5.2 Виконання програмного продукту

Коли клієнтська частина ВЗ завантажена, ми можемо бачити інтерфейс головної сторінки, її зображення наведене на рисунку 5.1.

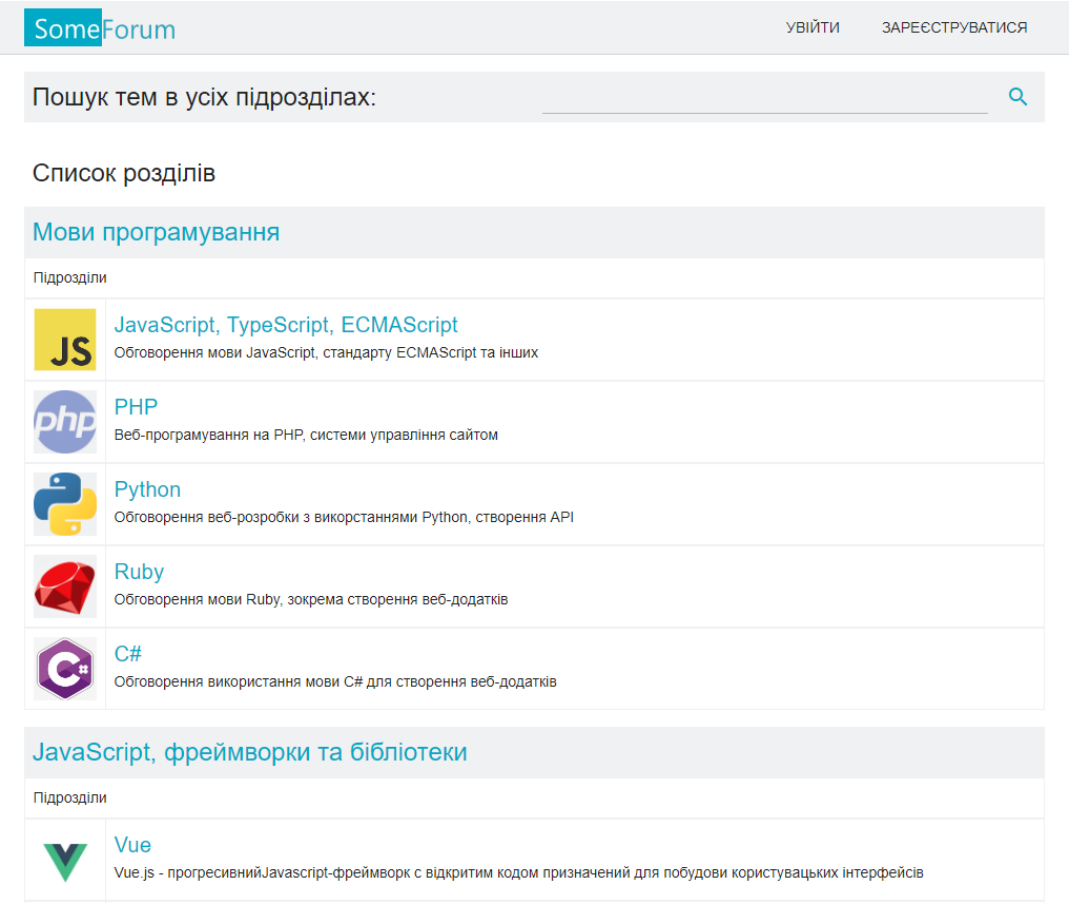

Рисунок 5.1 – Головна сторінка форуму

На цій сторінці ми маємо можливість глобального пошуку тем по всім підрозділам форуму. Пошук відбувається у назвах тем та їх описі, також можна здійснювати пошук по id теми. Щоб здійснити глобальний пошук треба вписати запит у поле пошуку і натиснути на копку поряд, після чого відбудеться перехід на сторінку з результатами.

На головній сторінці розміщено список тематичних розділів та підрозділів форуму. Якщо натиснути на назву підрозділу, або його іконку, то відбудеться перехід на сторінку зі списком тем обраного підрозділу. Зображення такої сторінки можна побачити на рисунку 5.2.

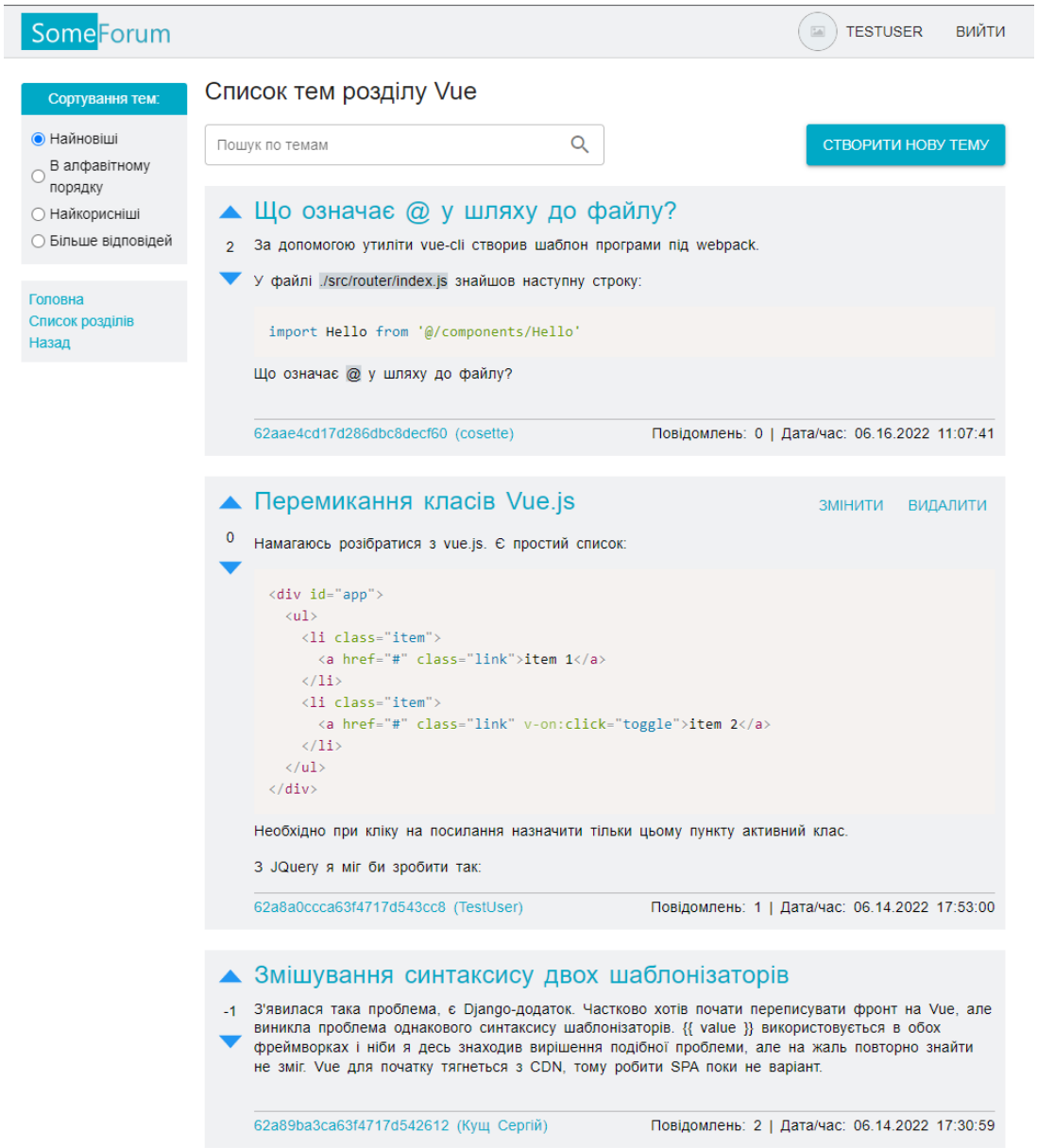

Рисунок 5.2 – Типова сторінка зі списком тем обраного підрозділу

Для створення нової теми необхідно зареєструватися на форумі. Перейти до сторінок реєстрації та авторизації можна натиснувши відповідні кнопки на панелі навігації. Форми для авторизації і реєстрації зображено на рисунку 5.3.

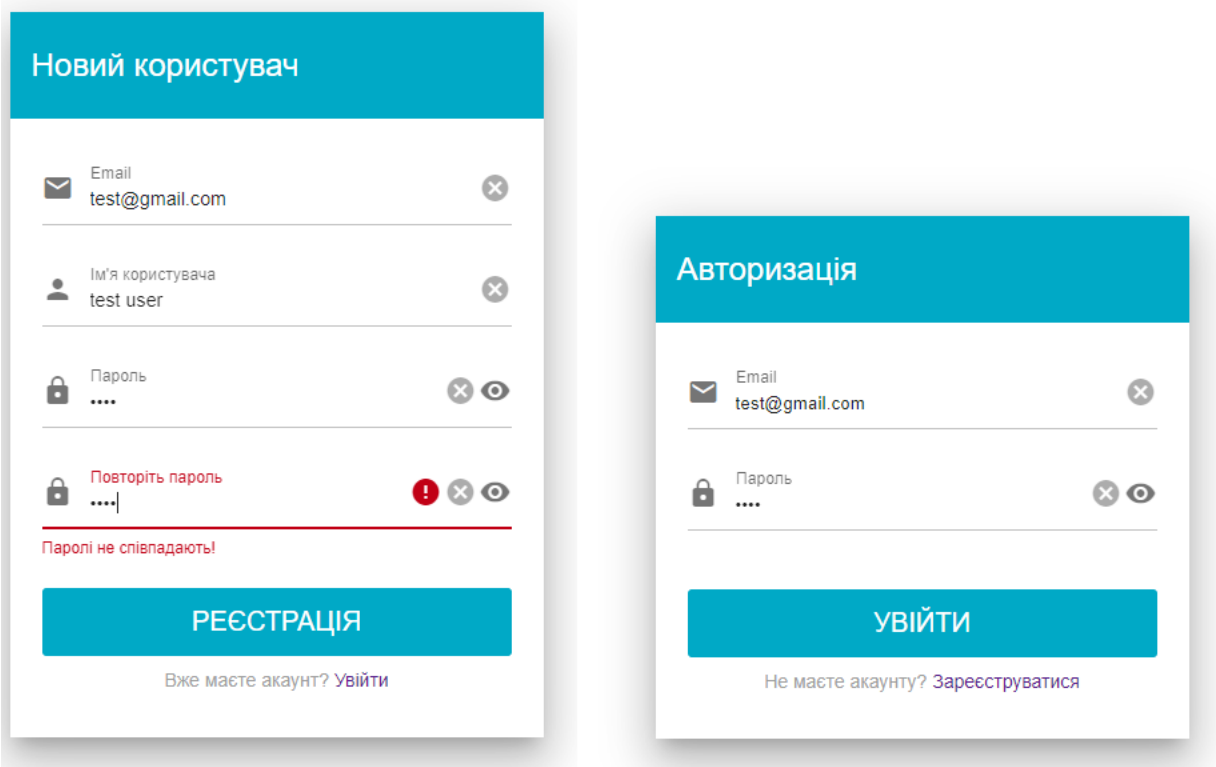

Рисунок 5.3 – Форми для реєстрації та авторизації

На сторінці зі списком тем підрозділу ми можемо здійснювати пошук по темам підрозділу. Для цього необхідно вписати запит в поле для пошуку. Запит відбувається автоматично, через 0,5 секунди після набору символу.

Теми завантажуються не усі разом, а групами. Завантаження нової групи тем відбувається при вертикальному прокручуванню сторінки донизу.

Також тут ми можемо здійснювати сортування тем за датою створення, назвою, користувацькою оцінкою та кількістю повідомлень перемикаючи відповідні елементи у лівому меню.

Для позитивної чи негативної оцінки теми необхідно натиснути на кнопку зі стрілочкою угору, або униз відповідно. Щоб прибрати оцінку треба навести курсор на оцінку теми і замість неї з’явиться необхідна кнопка.

Блок з інформацією про окрему тему в списку містить наступні дані:

– заголовок теми;
– короткий опис теми;
– id теми;
– ім’я автора;
– кількість повідомлень в темі;
– дата та час створення.

Якщо тему створили ви, або ви маєте права модератора чи адміністратора, то у правому кутку блоку теми з’являються кнопки для її редагування та видалення.

Форму оновлення теми можна побачити на рисунку 5.4

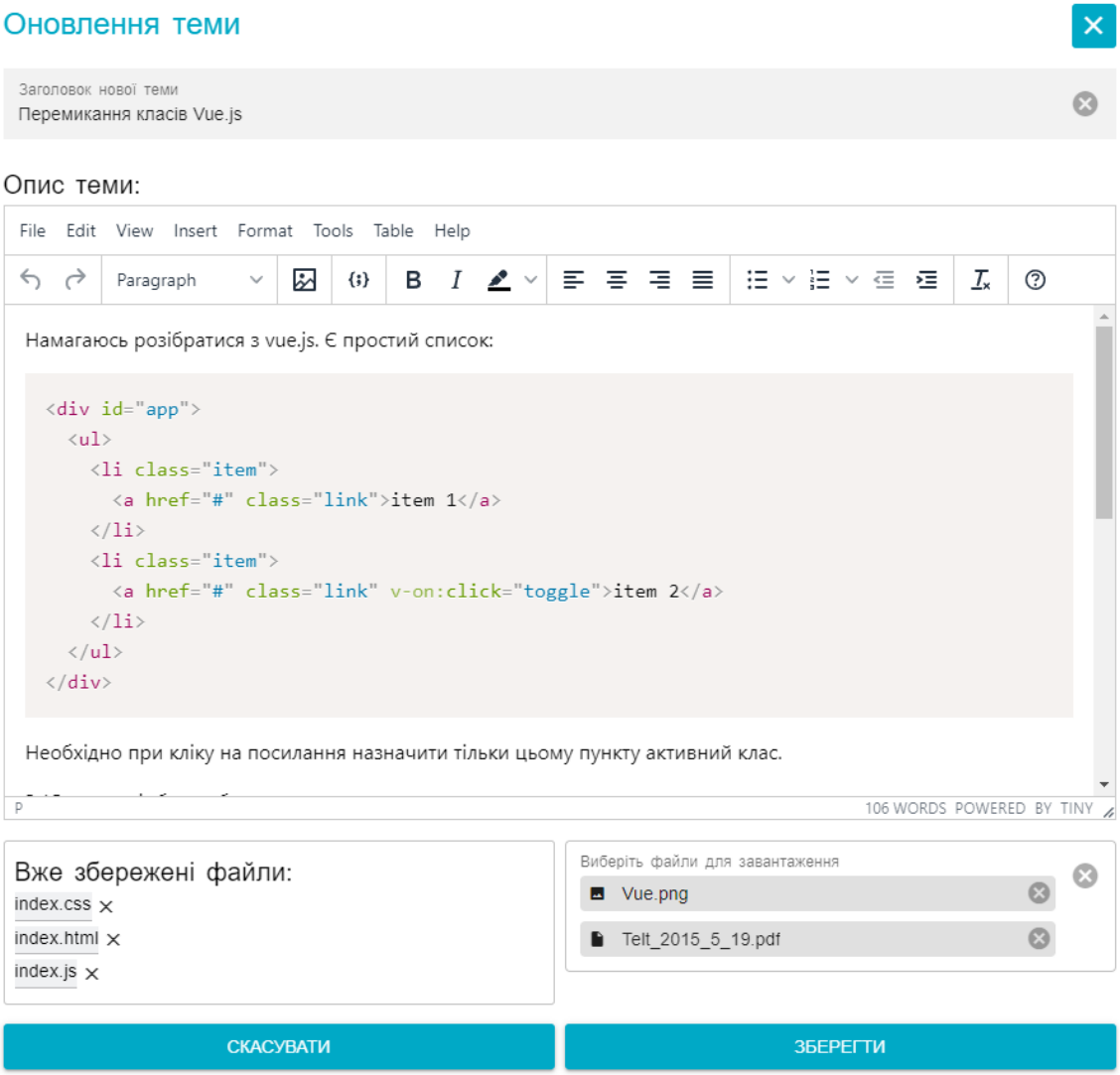

Рисунок 5.4 – Форми для оновлення теми

Для створення нової теми слід натиснути на кнопку створити нову тему, тоді з’явиться модальне вікно з формою для створення нової теми. Форми створення і оновлення теми майже ідентичні, за виключенням того, що форма оновлення теми містить функціонал для видалення вже завантажених файлів.

На формі оновлення теми можна побачити поле для редагування назви теми, WYSIWYG-редактор тексту для редагування опису теми, список вже прикріплених файлів, з можливістю їх видалення та поле для завантаження нових файлів.

Для того щоб відкрити сторінку окремої теми необхідно натиснути на її назву, після чого відбудеться перехід. На рисунку 5.5 можемо побачити типову сторінку теми, з повідомленнями й прикріпленими файлами.

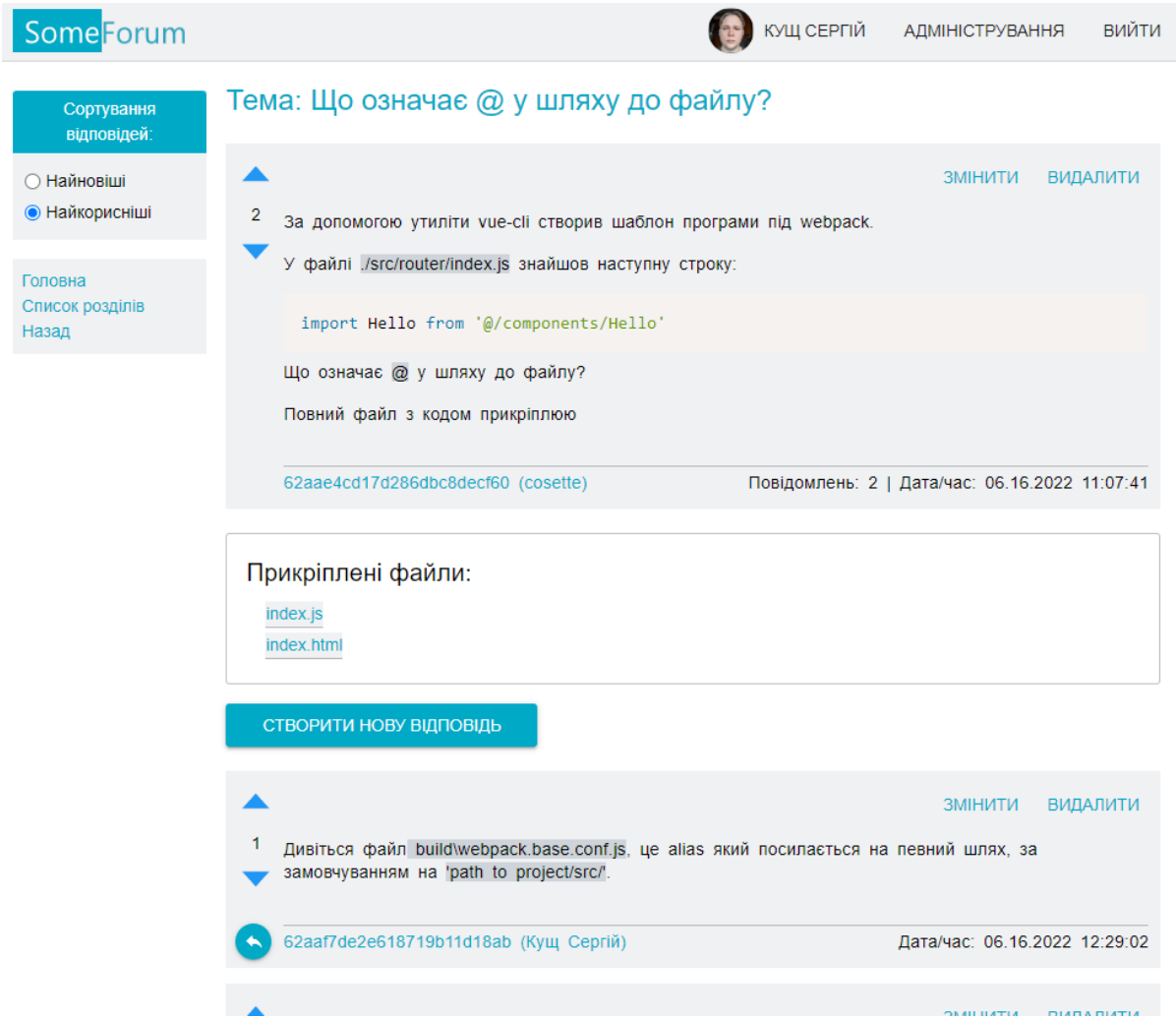

Рисунок 5.5 – Типові сторінка теми з повідомленнями

Сторінка теми містить повний опис теми, список прикріплених файлів з посиланнями для їх завантаження, список повідомлень у темі та відповідей на повідомлення.

Для завантаження прикріплених файлів необхідно натиснути на назву необхідного файлу.

Створити нове повідомлення можна натиснувши на однойменну кнопку. Форма створення й оновлення повідомлень ідентична такій для тем, за виключенням відсутності поля для вводу назви.

На сторінці тем ми можемо сортувати повідомлення за датою створення і оцінкою користувача, подібно до сторінки зі списком тем. Також, за аналогією до тем, ми можемо оцінювати повідомлення, змінювати та видаляти їх.

Для повідомлень існує можливість залишити відповідь на них, або на відповідь інших користувачів. Приклад повідомлення з прикріпленими файлами та відповідями можна побачити на рисунку 5.6.

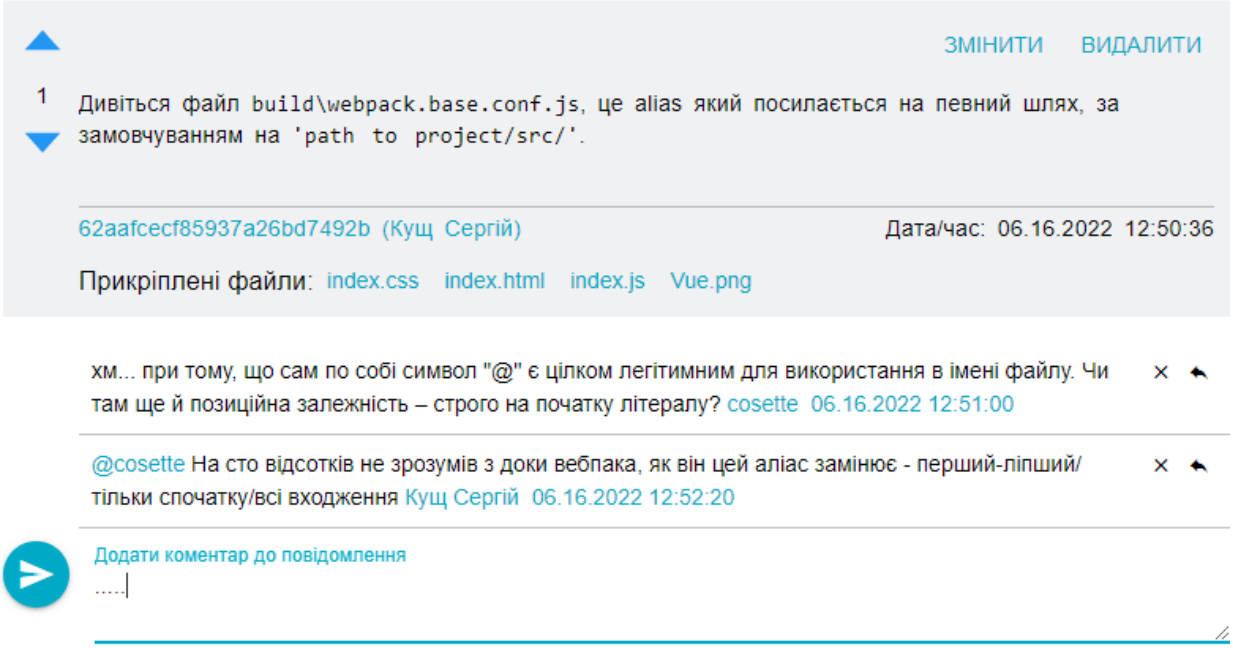

Рисунок 5.6 – Приклад повідомлення з прикріпленими файлами та відповідями

Для переходу на сторінку редагування профілю користувача необхідно натиснути на своє ім’я на панелі навігації. На цій сторінці ми маємо можливість оновити ім’я користувача, оновити або видалити фото профілю. Інтерфейс сторінки зображено на рисунку 5.7.

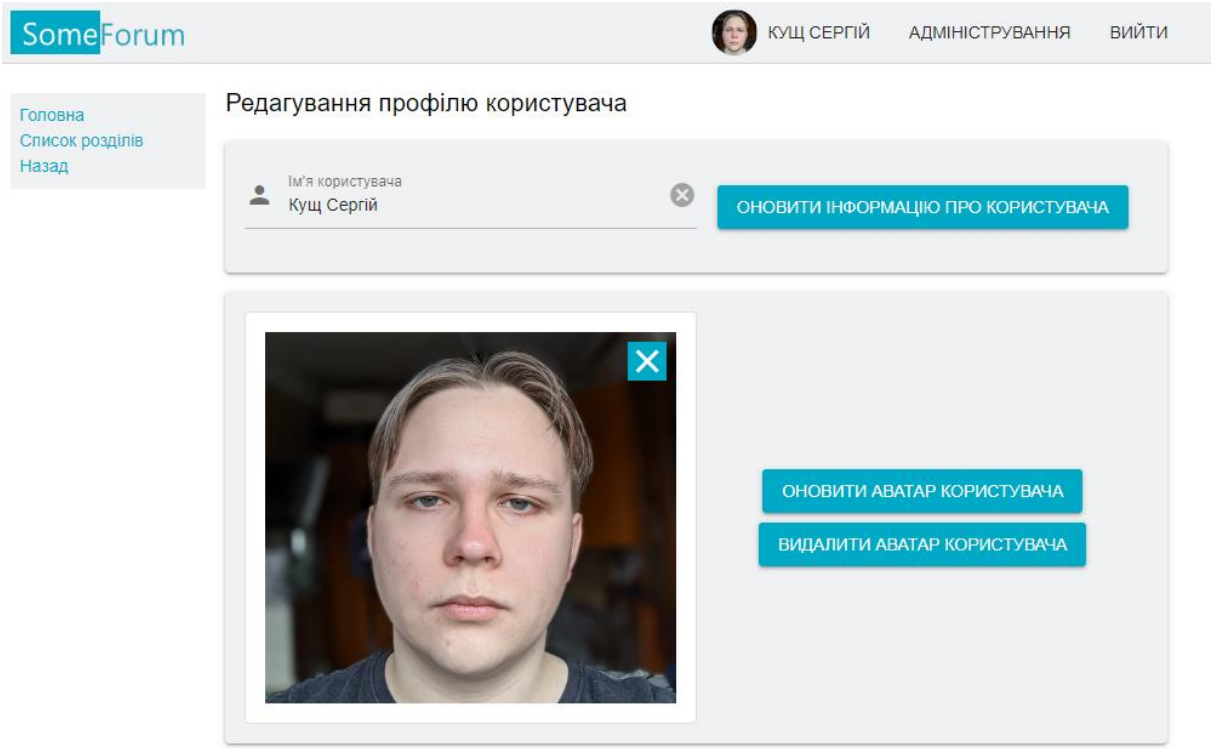

Рисунок 5.7 – Сторінка редагування профілю користувача

Якщо ви маєте права доступу адміністратора або модератора, то на панелі навігації з’являється кнопка «Адміністрування», натиснувши на яку відбудеться перехід до панелі адміністрування.

Панель адміністрування має чотири вкладки: користувачі, розділи, теми та повідомлення. Адміністратори мають доступ до усіх вкладок, модератори – лише до вкладок «Теми» та «Повідомлення».

Вкладки «Теми» та «Повідомлення» містять список тем і повідомлень відповідно, дозволяють здійснювати глобальний пошук по всім темам і повідомленням, редагувати їх та видаляти. Цього функціоналу достатньо для зручної модерації наповнення форуму.

Вкладка «Розділи» містить список розділів і підрозділів форуму, дозволяє створювати їх, редагувати, видаляти та оновлювати іконки підрозділів. Здійснення цих операцій запускається натисканням на кнопки з відповідними діям іконками. Інтерфейс вкладки «Розділи» наведено на рисунку 5.8.

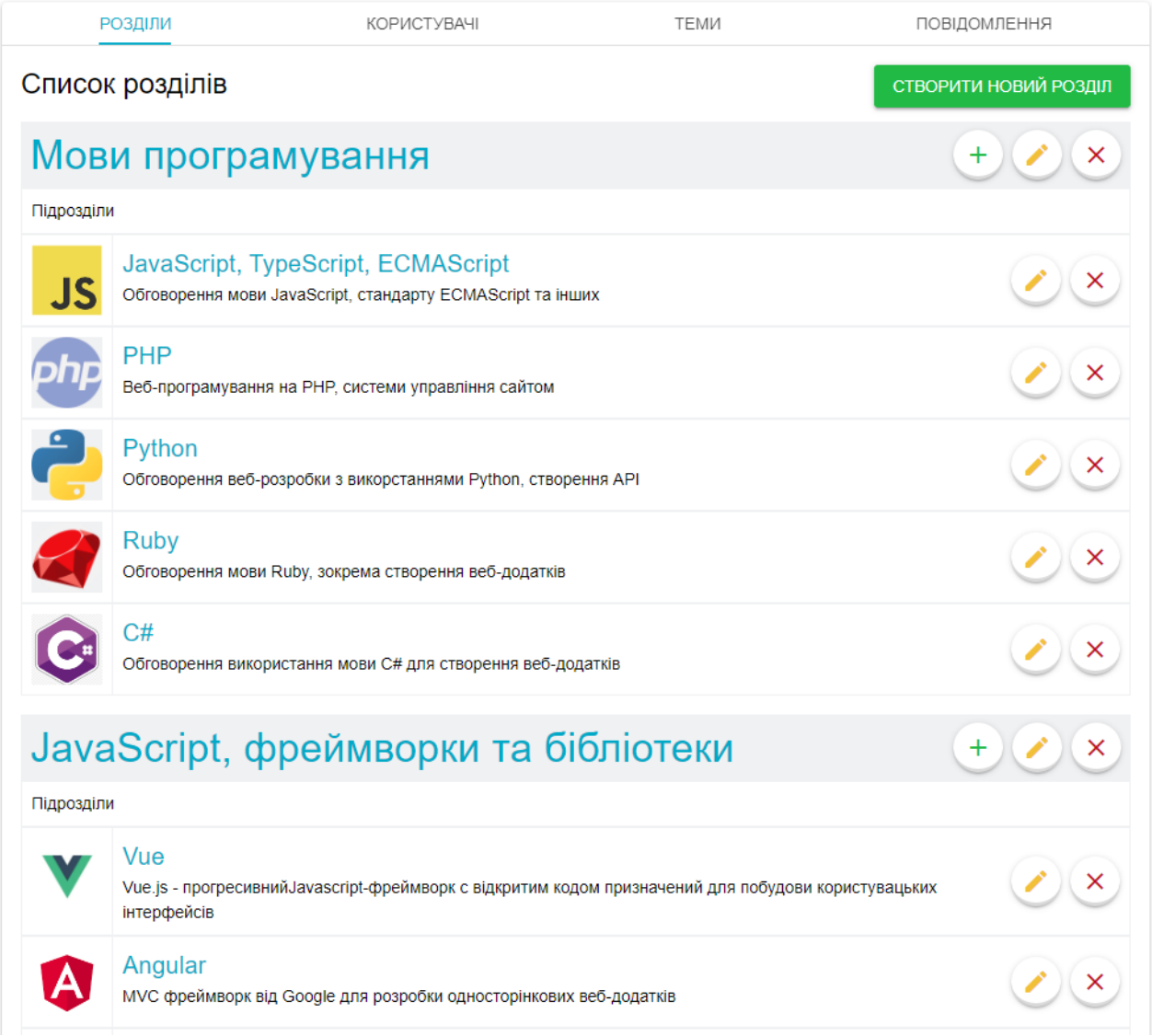

Рисунок 5.8 – Вкладка «Розділи» панелі адміністрування

Вкладка «Користувачі» містить список користувачів, з інформацією про кожного, функціонал пошуку по користувачам. Використовуючи цю сторінку адміністратор може змінювати права доступу користувачів,

активувати чи деактивувати їх профіль натисканням на однойменні кнопки. Інтерфейс вкладки «Користувачі» наведено на рисунку 5.9.

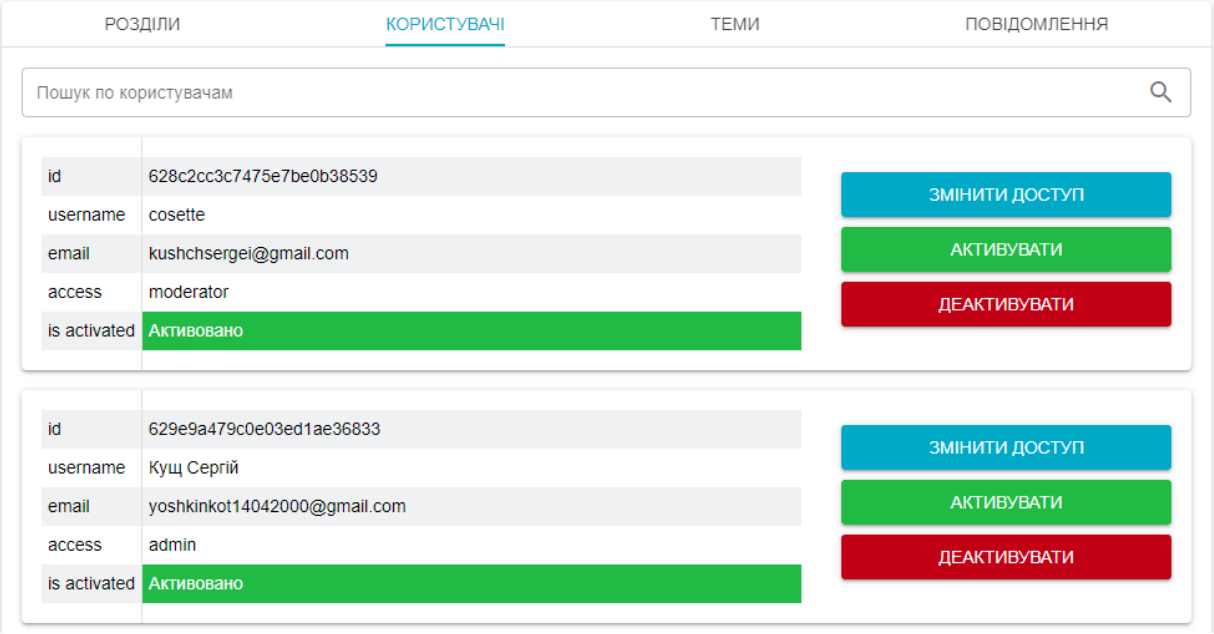

Рисунок 5.9 – Вкладка «Користувачі» панелі адміністрування

Інтерфейс застосунку чудово динамічно адаптується для використання на мобільних пристроях. Приклад адаптивної поведінки наведено на рисунку 5.10

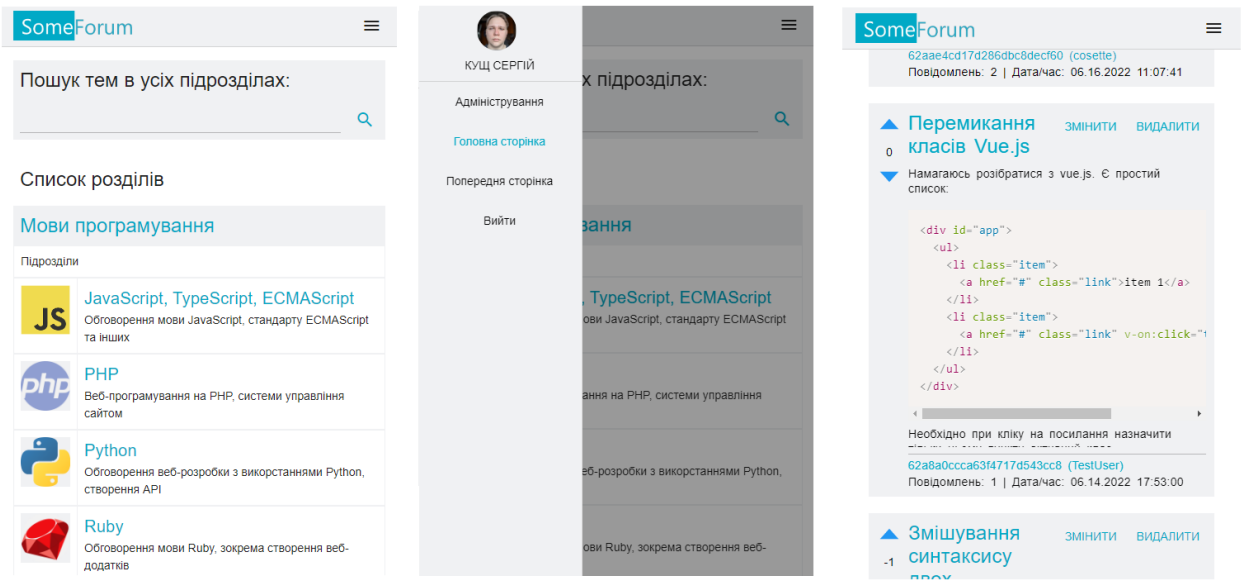

Рисунок 5.10 – Адаптація інтерфейсу під мобільні пристрої

5.3 Повідомлення оператору

Під час роботи розроблений ВЗ реагує на виникнення помилок в процесах, або успішного завершення якоїсь дії відображенням спливаючих повідомлень. Різноманіття повідомлень дуже велике, але зазвичай вони мають позитивний чи негативний характер. Приклад типових повідомлень можна побачити на рисунку 5.11.

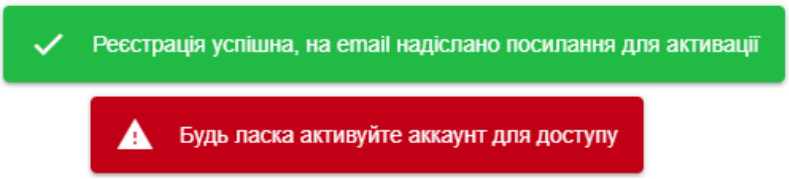

Рисунок 5.11 – Приклад типових спливаючих повідомлень

5.4 Висновки за розділом 5

У цьому розділі булі сформовані вимоги до апаратного забезпечення для роботи користувачів з клієнтом застосунку, описані сторінки, елементи управління, загальні принципи взаємодії користувача з системою, розглянуто типові повідомлення від системи.

ВИСНОВКИ

В ході виконання роботи був здійснений аналіз предметної області, відбулося ознайомлення з питанням вебфорумів, їх типами та особливостями функціонування, зокрема з форумами для розробників програмного забезпечення.

Проведення аналізу існуючих рішень, їх переваг і недоліків дозволило виділити вимоги до створюваного застосунку, що зробило його конкурентоспроможним серед аналогів, а його розробку актуальною задачею.

Також в процесі були досліджені технічні аспекти розробки ВЗ. В якості мови програмування для розробки було обрано JavaScript через її універсальність та можливість розробки SPA-додатків. В якості СКБД обрано MongoDB, через швидкий пошук інформації та можливість горизонтального масштабування. В якості середи розробки обрано редактор VS Code.

Була побудована структурна схема застосунку, чітко визначений розподіл прав доступу користувачів, методи обміну даних між клієнтом та сервером.

Клієнтську частину ВЗ було розроблено як SPA-додаток, з використанням фреймворку Vue.js. Для зручної побудови адаптивного інтерфейсу був використаний фреймворк Quasar. Для побудови локального сховища даних використана бібліотека Pinia.

Серверна частина додатку функціонує в середовищі виконання JavaScript-коду Node.js, розроблена з використанням фреймворка Express.js, за допомогою бібліотеки mongoose здійснює маніпуляції з даними в СКБД MongoDB. Вона побудована згідно архітектури REST. Обмін даними між клієнтом і сервером відбувається за допомогою HTTP-запитів у вигляді JSON-об’єктів для текстової інформації та бінарних файлів для обміну файлами.

В результаті виконання роботи було розроблено клієнт-серверний ВЗ, що виконує функції вебфоруму, призначений для обговорення різних тем, пов’язаних з розробкою вебзастосунків, пошуку рішення проблем, прикладів використання бібліотек та фреймворків, фрагментів коду.

Створений програмний продукт відповідає всім вимогам, визначеним на початку розробки і зазначеним у технічному завданні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Холявкіна Т. В Створення вебфоруму при використанні сучасних технологій веброзробки. [Текст] / Т. В. Холявкіна, Я. О. Резаєв, О. О. Харченко. // Проблеми інформатизації та управління. – 2020. – №2.64. – С. 69-76.
2. Пелещишин А. М. Оптимізація форумів та інших форм спільнот користувачів WWW. [Текст] / А. М. Пелещишин // Вісник Національного університету «Львівська політехніка». Інформаційні системи та мережі. – 2005. – №549. – С. 138-148.
3. Edeh Michael Online discussion forum as a tool for interactive learning and communication. [Text] / Onyema, Edeh Michael // International Journal of Recent Technology and Engineering. – 2019. – №8.4. – С. 4852-4859.
4. Smrithi M. Understanding the usage of online forums as learning platforms. [Text] / Rekha, V. Smrithi, S. Venkatapathy. // Procedia Computer Science. – 2015. – №46. – С. 499-506.
5. Bai We A qualitative investigation of insecure code propagation from online forums. [Text] / Bai Wei, Omer Akgul, and Michelle L. Mazurek // Cybersecurity Development (SecDev) IEEE. – 2019. – №1. – С. 34-47.
6. Replace.org.ua [Електрон. ресурс]. – Режим доступу: https://replace.org.ua/rules/
7. Tour - Stack Overflow [Electronic resource]. – Access mode: https://stackoverflow.com/tour
8. Dream.in.code – Rules [Electronic resource]. – Access mode: https://www.dreamincode.net/forums/index.php?showannouncement=22
9. Браузер Chrome: требования к системе [Электрон. ресурс]. – Режим доступа: https://support.google.com/chrome/a/answer/7100626
10. PHP Manual [Electronic resource]. – Access mode: https://www.php.net/manual/en/index.php

11. General Python FAQ [Electronic resource]. – Access mode: https://docs.python.org/3/faq/general.html

12. JavaScript [Electronic resource]. – Access mode: https://developer.mozilla.org/en-US/docs/Web/JavaScript

13. Sharma V. Sql and nosql databases. [Text] / V. Sharma, M. Dave // International Journal of Advanced Research in Computer Science and Software Engineering. – 2012. – Т.2. – №. 8. – С. 20-27.

14. NoSQL vs SQL Databases [Electronic resource]. – Access mode: https://www.mongodb.com/nosql-explained/nosql-vs-sql

15. Visual Studio Code – Code Editing. Redefined [Electronic resource]. – Access mode: https://code.visualstudio.com

16. WebStorm: The smartest JavaScript IDE [Electronic resource]. – Access mode: https://www.jetbrains.com/webstorm/

17. Scott Jr E. A. SPA Design and Architecture: Understanding single-page web applications. [Text]: all in one / Simon and Schuster. – Manning, ShelterIsland, New York, 2016. – 287 p.

18. Vue.js The Progressive JavaScript Framework [Electronic resource]. – Access mode: https://vuejs.org/

19. Express – Node.js web application framework [Electronic resource]. – Access mode: Режим доступу: https://expressjs.com/en/

20. JSON Web Token Introduction [Electronic resource]. – Access mode: https://jwt.io/introduction

ДОДАТОК А

Технічне завдання

Вступ

Розроблений клієнт-серверний ВЗ на тему «Розробка вебфоруму з питань створення вебзастосунків» призначений для обговорення різних тем, пов’язаних з розробкою ВЗ, пошуку рішення проблем, прикладів використання бібліотек та фреймворків, фрагментів коду.

А.1 Мета розробки і її призначення

Об’єкт дослідження – вебфорум з питань створення вебзастосунків.

Предмет дослідження – методи побудови вебзастосунку для створення вебфоруму.

Мета роботи – розробка вебфоруму для обговорення користувачами питань, пов’язаних зі створенням вебзастосунків.

Актуальність розробки такого вебфоруму пояснюється затребуваністю подібних рішень в мережі, зокрема у спільноті розробників ВЗ, мала кількість форумів, локалізованих для україномовних користувачів.

Більшість існуючих рішень побудовані з використанням спеціальних рушіїв, призначених для створення форумів, які створені вже давно і активно не оновлюються. Такі форуми мають застарілий, перевантажений та недружелюбний до нових користувачів інтерфейс, що зазвичай некоректно функціонує на мобільних пристроях.

Для усунення деяких недоліків існуючих продуктів, перейнявши їх функціональні переваги, було вирішено розробити новий продукт.

А.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

Вебфорум – це інтернет-сервіс для проведення онлайн-дискусій.

На форумах люди спілкуються шляхом створення тем для обговорення, з наступним розміщенням у них повідомлень.

Суть форумів в Інтернеті – це обмін думками та досвідом, спілкування з однодумцями, безкорислива допомога. Часто на форумах можна знайти інформацію, яку не відшукати ні в книгах, ні в інтернет-джерелах, бо користувачі форумів діляться, своїм, реальним досвідом.

Форуми можуть бути тематичними – на яких обговорюється вузький набір тем з якоїсь галузі, або можуть мати широкий набір тем і ділитися на велику кількість підфорумів.

Коли програмісти створюють нове програмне забезпечення (ПЗ), або додають функціонал до вже існуючого, вони постійно стикаються з проблемами використання нових, незнайомих бібліотек, засобів розробки та алгоритмів. Вирішення цих питань вони зазвичай шукають в мережі Інтернет, за допомогою пошукових систем, таких як Google, та на спеціалізованих ресурсах, таких як тематичні вебфоруми.

А.3 Підстави для розробки

Розробка виконується на підставі завдання на випускну дипломну роботу бакалавра, яке було визначене за темою «Розробка вебфоруму з питань створення вебзастосунків», затверджене наказом по НУ «Запорізька політехніка» № 102 від 27 квітня 2022 року.

А.4 Основні вимоги до програми, що розробляється, початковим даним і результату

А.4.1 Вимоги до функціональних характеристик

Програмний продукт, що розроблюється має реалізувати наступні функції:

– створення нових тем та повідомлень, їх редагування й видалення;

– створення відповідей на повідомлення у темах, або на відповіді інших користувачів;

– зручна навігація по розділам і темам;

– пошук по темам у підрозділах та загальний пошук по усім розділам;

– зручна система реєстрації та авторизації на сайті, активування профілю за посиланням у email-листі;

– функціонал редагування користувацького профілю та зміни фото профілю користувача;

– можливість позитивно чи негативно оцінити теми та повідомлення в них, скасувати чи змінити свою оцінку за необхідності;

– сортування тем у розділах за назвою, датою їх створення, оцінкою користувачів та кількістю повідомлень;

– сортування повідомлень у темах за датою створення та оцінкою користувачів;

– зручний WYSIWYG-редактор, з можливістю форматування тексту, додавання зображень, посилань, таблиць, списків, програмного коду з підсвічуванням синтаксису в повідомленнях та темах;

– функціонал завантаження файлів для їх прикріплення до тем та повідомлень, з подальшою можливістю їх скачування;

– функціонал для контролю наповненням форуму модераторами;

– функціонал для зміни прав доступу та статусу користувачів адміністратором;

– функціонал для створення, редагування й видалення розділів та підрозділів адміністратором.

А.4.2 Вимоги до надійності

Для збереження цілісності даних та уникнення критичних помилок в роботі застосунку перевірка даних виконується як на клієнті, так і на сервері.

Перебої у роботі системи не повинні спричиняти втрату інформації.

Функціонал адміністрування та модерації повинен бути ретельно захищеним від несанкціонованих спроб використання.

Функціонал створення тем та повідомлень повинен бути недоступним для використання незареєстрованими користувачами.

Для забезпечення високого рівня безпеки при авторизації користувачів повинні використовуватися access та refresh JWT-токени.

А.4.3 Умови експлуатації

Для використання клієнтами розробленого ВЗ достатньо мати пристрій з відносно новою версією одного з популярних веббраузерів та стабільне з’єднання з мережею Інтернет. Для створення нових тем та повідомлень користувачу необхідно зареєструватися

А.4.4 Вимоги до складу та параметрів технічних засобів

Мінімальні системні вимоги до середовища для розгортання і доопрацювання серверної частини ВЗ:

– CPU: Intel Core i5;
– RAM: 4 Gb;
– HDD/SSD: 1Gb;
– вебсервер: Apache 2 або Nginx;
– СКБД MongoDB;
– порти: 80 (HTTP), 27018 (стандартний порт MongoDB);
– програмне забезпечення: середовище Node.js, менеджер пакетів NPM або Yarn.

Мінімальні системні вимоги до середовища для розгортання і доопрацювання клієнтської частини ВЗ:

– CPU: двоядерний процесор, з підтримкою SSE3-інструкцій;

– RAM: 4 Gb;
– HDD/SSD: 1Gb;
– Операційна система Windows 7, OS X El Capitan, Linux;
– програмне забезпечення: браузер, менеджер пакетів NPM або Yarn.

А.4.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує маркування та упакуванні.

А.4.6 Вимоги до траспортування та збереження

Код створеного продукту розміщено у репозиторії Gitlab, тому він не потребує ніяких особливих вимог до транспортування та збереження.

А.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:
– технічне завдання;
– опис програми;
– керівництво програміста;
– керівництво оператора.

А.6 Основні обмеження на установку і використання програмного продукту

Розроблений ВЗ не потребує інсталяції, увесь необхідний код завантажується при відкритті застосунку в браузері. Для комфортної експлуатації пристрій, на я кому запускається продукт повинен відповідати встановленим вимогам.

ДОДАТОК Б

Опис програмного продукту

Б.1 Загальні відомості

В результаті роботи було розроблено клієнт-серверний ВЗ вебфоруму з питань створення вебзастосунків.

Розроблений клієнт-серверний ВЗ призначений для обговорення різних тем, пов’язаних з розробкою вебзастосунків, пошуку рішення проблем, прикладів використання бібліотек та фреймворків, фрагментів коду.

Мінімальні системні вимоги до середовища для розгортання і доопрацювання серверної частини ВЗ:

– CPU: Intel Core i5;
– RAM: 4 Gb;
– HDD/SSD: 1Gb;
– вебсервер: Apache 2 або Nginx;
– СКБД MongoDB;
– порти: 80 (HTTP), 27018 (стандартний порт MongoDB);
– програмне забезпечення: середовище Node.js, менеджер пакетів NPM або Yarn.

Мінімальні системні вимоги до середовища для розгортання і доопрацювання клієнтської частини ВЗ:

– CPU: двоядерний процесор, з підтримкою SSE3-інструкцій;
– RAM: 4 Gb;
– HDD/SSD: 1Gb;
– Операційна система Windows 7, OS X El Capitan, Linux;
– програмне забезпечення: браузер, менеджер пакетів NPM.

Б.2 Функціональне призначення

Програмний продукт, що розроблюється має реалізувати наступні функції:

– створення нових тем та повідомлень, їх редагування й видалення;

– створення відповідей на повідомлення у темах, або на відповіді інших користувачів;

– зручна навігація по розділам і темам;

– пошук по темам у підрозділах та загальний пошук по усім розділам;

– зручна система реєстрації та авторизації на сайті, активування профілю за посиланням у email-листі;

– функціонал редагування користувацького профілю та зміни фото профілю користувача;

– можливість позитивно чи негативно оцінити теми та повідомлення в них, скасувати чи змінити свою оцінку за необхідності;

– сортування тем у розділах за назвою, датою їх створення, оцінкою користувачів та кількістю повідомлень;

– сортування повідомлень у темах за датою створення та оцінкою користувачів;

– зручний WYSIWYG-редактор, з можливістю форматування тексту, додавання зображень, посилань, таблиць, списків, програмного коду з підсвічуванням синтаксису в повідомленнях та темах;

– функціонал завантаження файлів для їх прикріплення до тем та повідомлень, з подальшою можливістю їх скачування;

– функціонал для контролю наповненням форуму модераторами;

– функціонал для зміни прав доступу та статусу користувачів адміністратором;

– функціонал для створення, редагування й видалення розділів та підрозділів адміністратором.

Б.3 Опис логічної структури

Загальну архітектуру клієнт-серверного застосунку, зв’язки зі сховищами та БД, комунікацію між клієнтом і сервером представлено на рисунку Б.1.

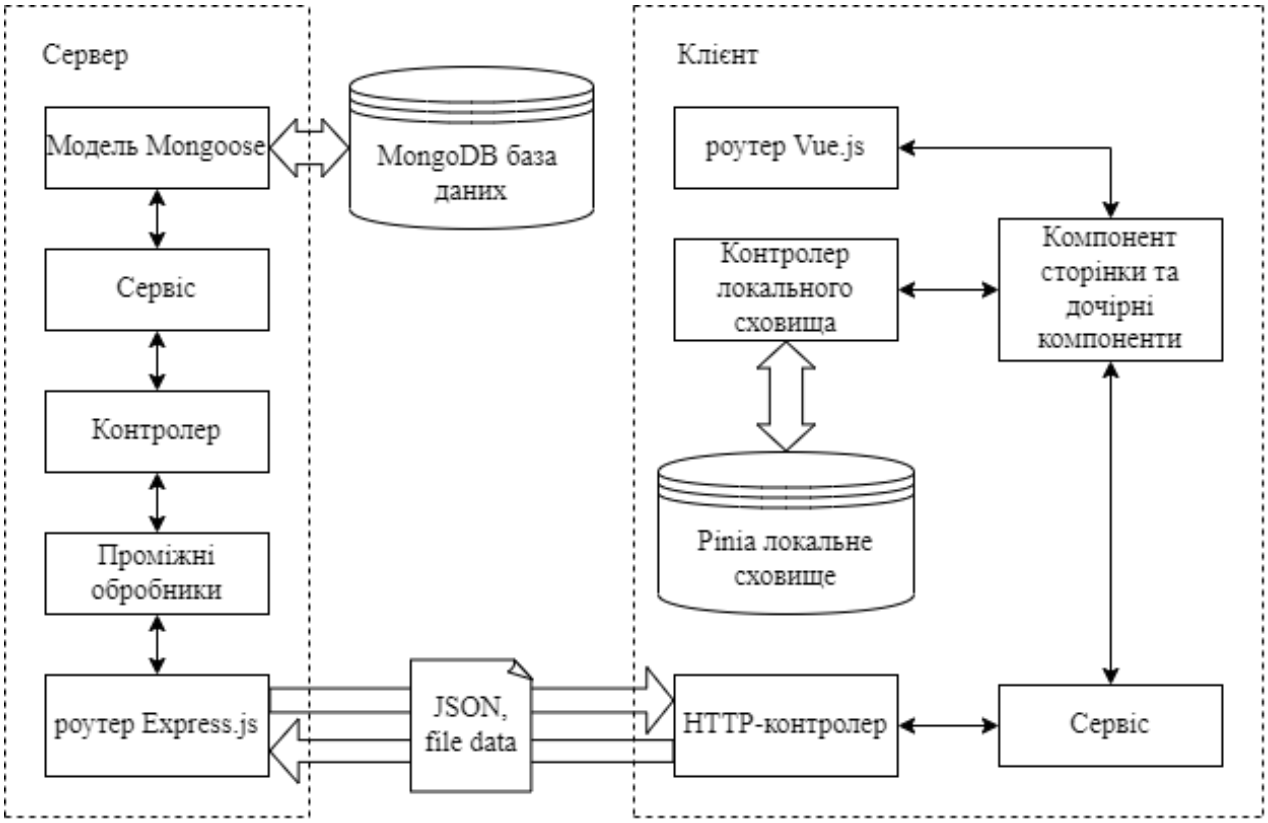

Рисунок Б.1 – Архітектура застосунку

ВЗ, що розроблюється, матиме чотири категорії користувачів: гість (незареєстрований користувач), зареєстрований користувач, модератор та адміністратор.

Незареєстрований користувач матиме право переглядати усі розділи та підрозділи, теми, відповіді у них, здійснювати сортування та скачувати прикріплені файли.

У зареєстрованого користувача з’являється можливість створювати нові теми, повідомлення, відповіді на повідомлення, оцінювати теми та

повідомлення, прикріплювати до них файли, редагувати свій профіль користувача.

Користувачі з правами модератора можуть редагувати та видаляти будь які теми та повідомлення в цілях модерації наповнення форуму.

Адміністратор має можливість редагувати список розділів та підрозділів, змінювати права доступу та статус користувачів.

Для наглядності структурна схема проекту, що розроблюється наведена на рисунку Б. 2.

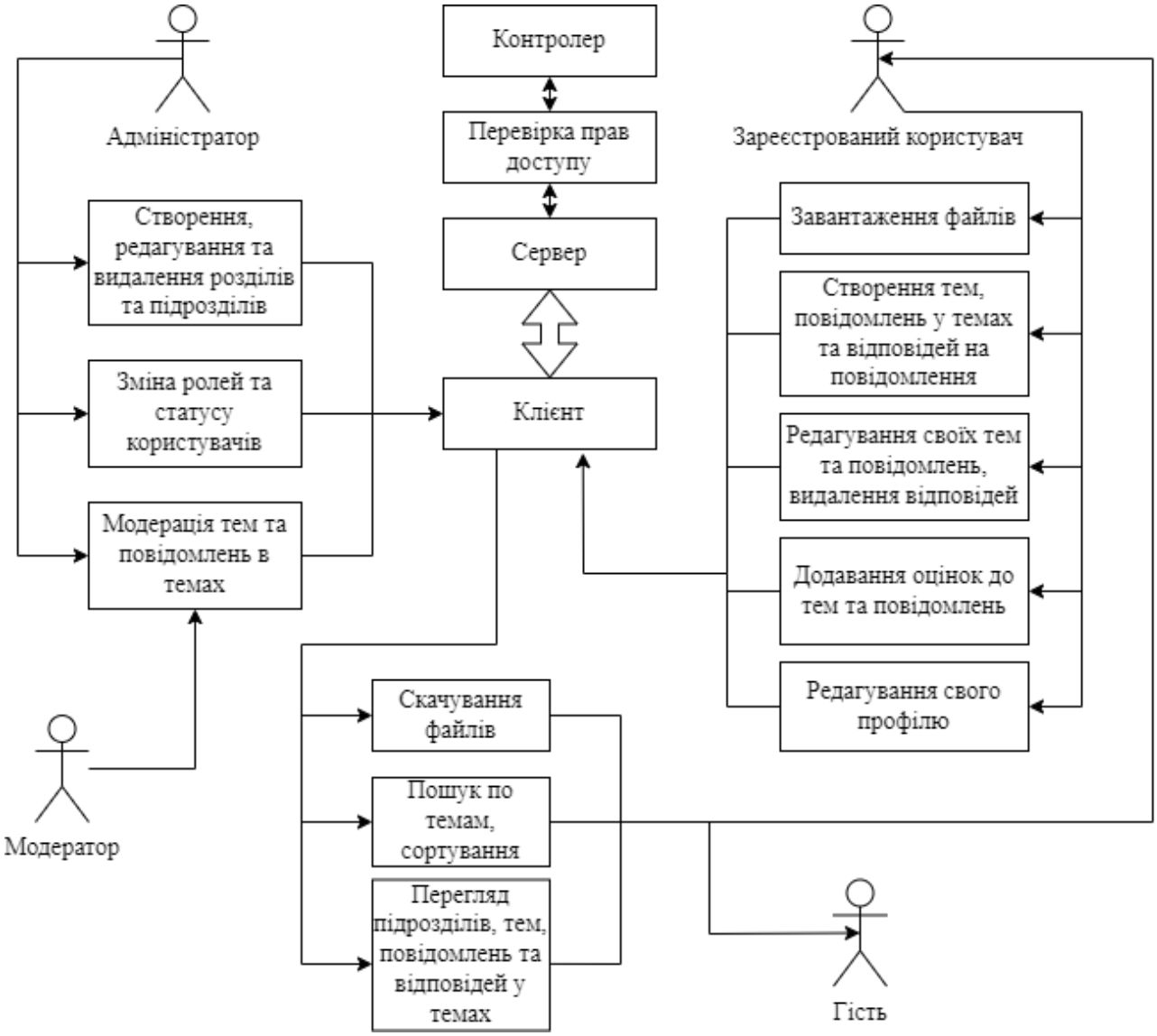

Рисунок Б.2 – Структурна схема застосунку

Б.4 Вибір інструментарію розробки програмного продукту

В якості мови програмування для розробки було обрано JavaScript через її універсальність та можливість розробки SPA-додатків. В якості СКБД обрано MongoDB, через швидкий пошук інформації та можливість горизонтального масштабування. В якості середи розробки обрано редактор VS Code.

Клієнтську частину ВЗ було розроблено як SPA-додаток, з використанням фреймворку Vue.js. Для зручної побудови адаптивного інтерфейсу був використаний фреймворк Quasar. Для побудови локального сховища даних використана бібліотека Pinia.

Серверна частина додатку функціонує в середовищі виконання JavaScript-коду Node.js, розроблена з використанням фреймворка Express.js, за допомогою бібліотеки mongoose здійснює маніпуляції з даними в СКБД MongoDB. Вона побудована згідно архітектури REST. Обмін даними між клієнтом і сервером відбувається за допомогою HTTP-запитів у вигляді JSON-об’єктів для текстової інформації та бінарних файлів для обміну файлами.

Б.5 Виклик і завантаження

Актуальний код продукту можна склонувати використовуючи систему контролю версій Git за посиланнями: «https://gitlab.com/skusch/forum-server.git» – для серверної частини і «https://gitlab.com/skusch/forum-client.git» – для клієнтської відповідно. Далі необхідно відкрити системну консоль, перейти в корінь склонованих каталогів і підтягнути всі необхідні для роботи застосунків бібліотеки консольною командою «npm install». Після завершення інсталювання бібліотек можна запустити як клієнтську, так і серверну частину для локального виконання консольною командою «npm run start» в обох

каталогах. Після чого отримати доступ до клієнтської частини можна в браузері за посиланням «http://localhost:3000/». Запити до сервера необхідно відсилати за адресою «http://localhost:3090/api/». Зупинити локальне виконання застосунків можна комбінацією клавіш «Ctrl+C» в консолі.

Б.6 Приклади реалізації

Оскільки основною задачею застосунку є запис та управління даними, вкрай важливо забезпечити правильний алгоритм їх обробки та валідації.

Блок схему алгоритму створення нової теми можна побачити на рисунку Б.3. Алгоритм створення повідомлень також практично ідентичний нижче наведеному.

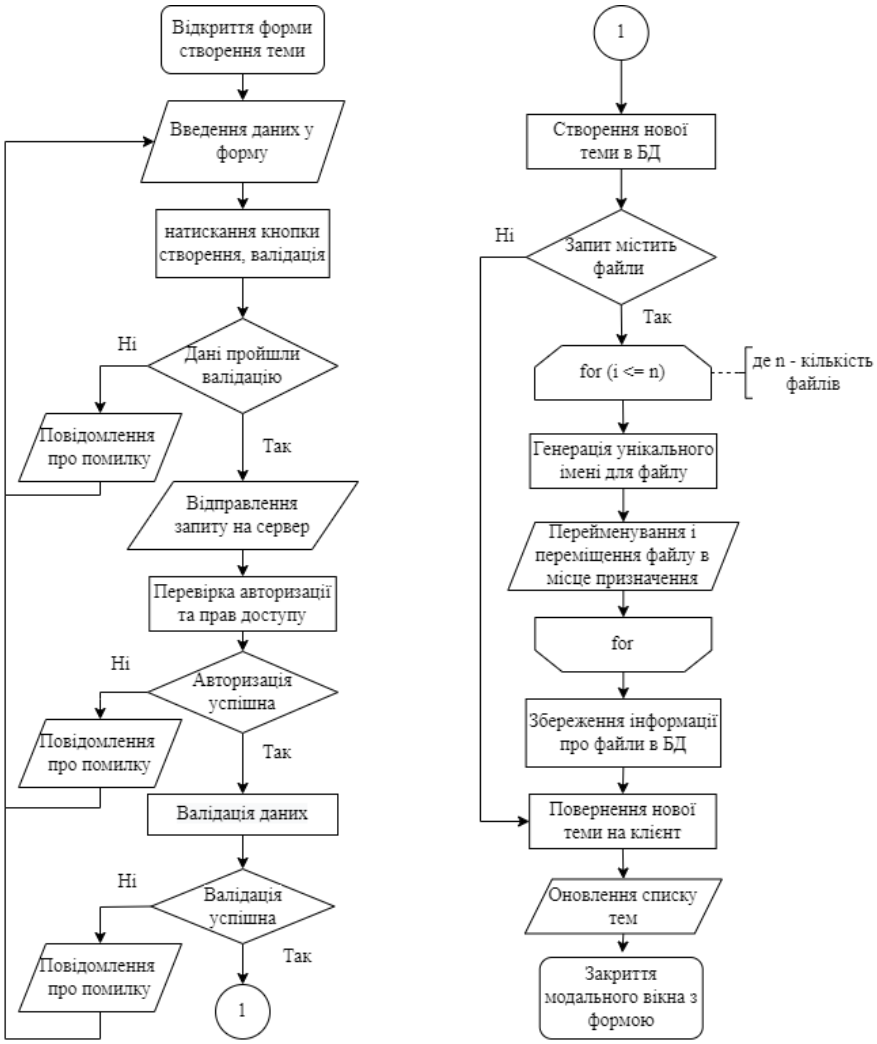

Рисунок Б.3 – Блок-схема алгоритму створення нової теми

Для збереження цілісності даних та уникнення критичних помилок в роботі застосунку перевірка даних виконується як на клієнті, так і на сервері. На клієнті це відбувається шляхом валідації форм в процесі їх заповнення даними та перед відправкою на сервер. На сервері валідація даних відбувається шляхом використанням спеціальних проміжних обробників для валідації перед відправкою даних до контролерів. Якщо це необхідно, то перевірка може бути доповненою в самому контролері. Як останній рубіж перевірки правильності надісланих даних виступає сама СКБД Mongodb, перевіряючи вміст запитів до бази згідно правил, що були прописані в процесі опису схем документів.

Б.7 Вхідні та вихідні дані

Вхідними даними розробленого ВЗ є текстові дані, які користувачі вписують у форми при створенні тем, повідомлень та відповідей, а також файли довільного формату, які користувачі можуть завантажувати для прикріплення до тем та повідомлень. Приклад заповнення форми створення теми з використанням WYSIWYG-редактора тексту і завантаження файлу наведено на рисунку Б.4.

Вихідними даними є відображення на сторінках збереженої в БД текстової інформації, зображень, посилань, а також доступ до скачування файлів, завантажених користувачами, за посиланнями. Приклад відображення відповіді та посилання на скачування файлів, прикріплених до теми наведено на рисунку Б.5.

Створення нової теми

Заголовок нової теми
Node.js рекурсивне отримання повного шляху до файлів

Опис теми:

File Edit View Insert Format Tools Table Help

Paragraph

Маю проблему зі, здається, простою рекурсивною функцією.

Задача полягає в рекурсивному отриманні всіх файлів в нвданому каталозі та його підкаталогах.

На даний момент мені вдалося перерахувати файли в каталозі за допомогою простої функції:

```
fs.readdirSync(copyFrom).forEach((file) => {
  let fullPath = path.join(copyFrom, file);

  if (fs.lstatSync(fullPath).isDirectory()) {
    console.log(fullPath);
  } else {
    console.log(fullPath);
  }
});
```

Я пробував різні методи, як-от do{} ... while(), але не можу зрозуміти. Оскільки я новачок у javascript, я нарешті вирішив попросити допомоги у вас.

Файл з повним кодом прикріплюю....

P 81 WORDS POWERED BY TINY

Виберіть файли для завантаження
index.js

СКАСУВАТИ ЗБЕРЕГТИ

Рисунок Б.4 – Приклад вхідних даних для при створенні теми

Прикріплені файли:
index.js

СТВОРИТИ НОВУ ВІДПОВІДЬ

ЗМІНИТИ ВИДАЛИТИ

2 Просто додайте рекурсивний виклик, і все готово:

```
function traverseDir(dir) {
  fs.readdirSync(dir).forEach(file => {
    let fullPath = path.join(dir, file);
    if (fs.lstatSync(fullPath).isDirectory()) {
      console.log(fullPath);
      traverseDir(fullPath);
    } else {
      console.log(fullPath);
    }
  });
}
```

62aad35817d286dbc8dec5f4 (Кущ Сергій) Дата/час: 06.16.2022 09:53:12

Дякую, здається, це робить свою справу. Однак у мене немає жодних консольних повідомлень під час виконання функції... Мабуть, я щось пропустив. cosette 06.16.2022 09:54:55

Рисунок Б.5 – Приклад вихідних даних

ДОДАТОК В

Лістинги файлів роботи

В.1 Файл AdminPage.vue

```
<template>
  <q-page class="bg-primary">
    <div class="container">
      <div class="q-pa-md">
        <div class="q-gutter-y-md">
          <q-card>
            <q-tabs
              align="justify"
              v-model="tab"
              class="text-grey-10"
              active-color="accent"
              indicator-color="accent"
              narrow-indicator
              dense
            >
              <q-tab name="sections" label="Розділи" />
              <q-tab name="users" label="Користувачі" />
              <q-tab name="themes" label="Теми" />
              <q-tab name="messages" label="Повідомлення" />
            </q-tabs>
            <q-separator />
            <q-tab-panels v-model="tab" animated>
              <q-tab-panel name="sections">
                <AdminSectionsVue />
              </q-tab-panel>

              <q-tab-panel name="users">
                <AdminUsersVue />
              </q-tab-panel>
              <q-tab-panel name="themes">
                <AdminThemesVue />
              </q-tab-panel>

              <q-tab-panel name="messages">
                <AdminMessagesVue />
```

```
              </q-tab-panel>
            </q-tab-panels>
          </q-card>
        </div>
      </div>
    </div>
  </q-page>
</template>

<script>
import AdminSectionsVue from "../components/AdminSections.vue";
import AdminUsersVue from "../components/AdminUsers.vue";
import AdminThemesVue from "../components/AdminThemes.vue";
import AdminMessagesVue from "../components/AdminMessages.vue";
import { mapStores } from "pinia";
import { useUserStore } from "../stores/user.js";

export default {
  data: function () {
    return {
      tab: "sections",
    };
  },
  mounted() {
    this.tab =
    this.userStore.user.access == "moderator" ? "messages" : "sections";
  },
  components: {
    AdminSectionsVue,
    AdminUsersVue,
    AdminThemesVue,
    AdminMessagesVue,
  },
  computed: {
    ...mapStores(useUserStore),
  },
};
</script>
```

В.2 Файл Authentication.vue

```
<template>
  <q-page class="window-height row justify-center items-center bg-
primary">
    <div class="column q-pa-lg">
      <div class="row">
        <q-card square class="shadow-20" style="width: 400px">
          <q-card-section class="bg-accent">
            <h4 class="text-h5 text-white q-my-sm">{{ title }}</h4>
          </q-card-section>
          <q-card-section>
            <q-form class="q-px-sm q-pt-sm">
              <q-input
                ref="email"
                square
                clearable
                v-model="email"
                type="email"
                label="Email"
                lazy-rules
              :rules="[this.isEmpty, this.isValidEmail, this.isTooshort]"
              >
                <template v-slot:prepend>
                  <q-icon name="email" />
                </template>
              </q-input>
              <q-input
                ref="username"
                v-if="isRegister"
                square
                clearable
                v-model="username"
                type="username"
                label="Ім'я користувача"
                lazy-rules
                :rules="[this.isEmpty, this.isTooshort]"
              >
                <template v-slot:prepend>
```

```
        <q-icon name="person" />
      </template>
    </q-input>
    <q-input
      ref="password"
      square
      clearable
      v-model="password"
      :type="passwordFieldType"
      lazy-rules
      :rules="[this.isEmpty, this.isTooshort]"
      label="Пароль"
    >
      <template v-slot:prepend>
        <q-icon name="lock" />
      </template>
      <template v-slot:append>
        <q-icon
          :name="visibilityIcon"
          @click="switchPasswordVisibility"
          class="cursor-pointer"
        />
      </template>
    </q-input>
    <q-input
      ref="password2"
      v-if="isRegister"
      square
      clearable
      v-model="password2"
      :type="passwordFieldType"
      lazy-rules
  :rules="[this.isEmpty, this.isTooshort, this.isPasswordSame]"
      label="Повторіть пароль"
    >
      <template v-slot:prepend>
        <q-icon name="lock" />
      </template>
      <template v-slot:append>
```

```
                <q-icon
                  :name="visibilityIcon"
                  @click="switchPasswordVisibility"
                  class="cursor-pointer"
                />
              </template>
            </q-input>
          </q-form>
        </q-card-section>
    <q-card-actions class="q-px-lg q-pb-md justify-center items-center">
          <q-btn
            unelevated
            size="lg"
            color="accent"
            @click="submit"
            class="full-width text-white"
            :label="btnLabel"
          />
          <p class="text-grey-6 q-mt-sm">
           {{ isRegister ? "Вже маєте акаунт?" : "Не маєте акаунту?" }}
            <router-link
              style="text-decoration: none"
              :to="isRegister ? '/log-in' : '/sign-up'"
            >
              {{ isRegister ? "Увійти" : "Зареєструватися" }}
            </router-link>
          </p>
        </q-card-actions>
      </q-card>
    </div>
  </div>
</q-page>
</template>
<script>
import router from "../router";
import { mapStores } from "pinia";
import { useUserStore } from "../stores/user";

export default {
```

```
  props: ["register"],
  data: function () {
    return {
      title: "Авторизація",
      email: "",
      username: "",
      password: "",
      password2: "",
      isRegister: false,
      passwordFieldType: "password",
      btnLabel: "Увійти",
      visibility: false,
      visibilityIcon: "visibility",
    };
  },
  mounted() {
    this.isRegister = this.register;
    this.setFormType();
  },
  computed: {
    ...mapStores(useUserStore),
  },
  methods: {
    isEmpty(val) {
      return (val && val.length > 0) || "Поле повинно бути заповнене";
    },
    isPasswordSame(val) {
      const val2 = this.password;
      return (val && val === val2) || "Паролі не співпадають!";
    },
    isTooshort(val) {
      return (val && val.length > 3) || "Занадто коротко";
    },
    isValidEmail(val) {
      return (
        emailRegExp.test(val) ||
        "Будь ласка введіть email в правильному форматі"
      );
    },
```

```
async submit() {
  if (this.isRegister) {
    this.$refs.email.validate();
    this.$refs.username.validate();
    this.$refs.password.validate();
    this.$refs.password2.validate();
  } else {
    this.$refs.email.validate();
    this.$refs.password.validate();
  }
  if (!this.isRegister) {
    if (!this.$refs.email.hasError && !this.$refs.password.hasError){
      const loginData = {
        email: this.email,
        password: this.password,
      };
      try {
        await this.userStore.login(loginData);
        this.$q.notify({
          icon: "done",
          color: "positive",
          message: "Вхід успішний",
        });
        router.push({ name: "home" });
      } catch (e) {
        this.$q.notify({
          icon: "warning",
          color: "negative",
          message: e.response.data.message,
        });
      } finally {
      }
    }
  } else {
    if (
      !this.$refs.email.hasError &&
      !this.$refs.username.hasError &&
      !this.$refs.password.hasError &&
      !this.$refs.password2.hasError
```

```
                ) {
                  const registrationData = {
                    username: this.username,
                    email: this.email,
                    password: this.password,
                  };
                  try {
                    await this.userStore.registration(registrationData);

                    this.$q.notify({
                      icon: "done",
                      color: "positive",
                      message:
                "Реєстрація успішна, на email надіслано посилання для активації",
                    });
                    router.push({ name: "home" });
                  } catch (e) {
                    this.$q.notify({
                      icon: "warning",
                      color: "negative",
                      message: e.response.data.message,
                    });
                  } finally {
                  }
                }
              }
            },
            setFormType() {
              this.title = this.isRegister ? "Новий користувач" : "Авторизація";
              this.btnLabel = this.isRegister ? "Реєстрація" : "Увійти";
            },
            switchPasswordVisibility() {
              this.visibility = !this.visibility;
              this.passwordFieldType = this.visibility ? "text" : "password";
              this.visibilityIcon = this.visibility ? "visibility_off" :
"visibility";
            },
          },
        };
```

```
</script>
```

В.3 Файл HomePage.vue

```
<template>
  <q-page class="page">
    <div class="container q-px-md">
      <div class="row">
        <div class="col">
          <div class="sections-list text-grey-10">
            <GlobalSearch />
            <h4 class="q-mb-md q-px-sm text-h5">Список розділів</h4>
            <div
              class="section q-mb-md"
              v-for="section in sections"
              :key="section._id"
            >
              <div class="section-title bg-secondary">
                <h5 class="q-pa-sm q-my-none text-accent">
                  {{ section.title }}
                </h5>
              </div>
              <div class="subsections-list">
                <div class="subsections-title q-pa-sm">Підрозділи</div>
                <div
                  class="subsection"
                  v-for="subsection in section.subsections"
                  :key="subsection._id"
                >
                  <div class="subsection-icon q-pa-sm">
                    <router-link
                      :to="`/section/${section._id}/${subsection._id}`"
                    >
                      <q-avatar square size="60px">
                        <template v-if="!subsection.icon">
                       <img src="../assets/images/img-placeholder.png" />
                        </template>
                        <template v-else>
                          <q-img
```

```
                            :ratio="1"
                            class="file-link bg-secondary"
:src="`http://localhost:3090/api/files/avatars/${subsection.icon.name}`"
                          />
                        </template>
                      </q-avatar>
                    </router-link>
                  </div>
                  <div class="subsection-info q-pa-sm">
                    <div class="subsection-name">
                      <router-link
                        :to="`/section/${section._id}/${subsection._id}`"
                      >
                        <h6 class="cursor-pointer q-ma-none">
                          {{ subsection.name }}
                        </h6>
                      </router-link>
                    </div>
                    <div class="subsection-description">
                      {{ subsection.description }}
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </q-page>
</template>
<script>
import SectionsService from "../services/SectionsService";
import GlobalSearch from "../components/GlobalSearch.vue";

export default {
  data: function () {
    return {
      sections: [],
```

```
    };
  },
  components: {
    GlobalSearch,
  },
  async mounted() {
    this.sections = await SectionsService.getAllSections();
  },
};
</script>
```

В.4 Файл SubsectionPage.vue

```
<template>
  <PageWithSidebar>
    <template v-slot:sidebar-sort>
      <div class="sort-head">Сортування тем:</div>
      <div class="sort-body q-pa-sm">
        <div v-for="(option, index) in sortOptions" :key="index">
       <input name="sort" type="radio" :value="index" v-model="sort" />{{
            index
          }}
        </div>
      </div>
    </template>
    <template v-slot:page-content>
      <div v-if="subsection" class="sections-list">
        <h4 class="q-my-md text-h5 text-grey-10">
          Список тем розділу {{ subsection?.name || "..." }}
        </h4>
        <div class="row">
          <div class="col-md-6 col-xs-12">
            <q-input
              class="search"
              v-model="search"
              debounce="500"
              outlined
              clearable
              placeholder="Пошук по темам"
            >
```

```
        <template v-slot:append>
          <q-icon name="search" />
        </template>
      </q-input>
    </div>
    <div class="col-md-6 col-xs-12 flex justify-end">
      <q-btn
        class="create-theme-btn"
        @click="showNewThemeModal = true"
        color="accent"
        label="Створити нову тему"
      />
    </div>
    <NewPostModal
      :isTheme="true"
      :isActive="showNewThemeModal"
      :subsectionId="subsection._id"
      @changeModalStat="changeThemeModalStat"
    />
  </div>
</div>
<div v-else class="q-mt-lg"><GlobalSearch /></div>

<div v-if="themes.length > 0" class="themes-list">
  <q-infinite-scroll @load="onLoadMore" :offset="250">
    <template v-for="theme in themes" :key="theme._id">
      <Post
        :post="theme"
        :isTheme="true"
        @refreshPosts="refreshThemes"
        @highlightAll="highlightAll"
      />
    </template>
    <template v-slot:loading>
      <div class="row justify-center q-my-md">
        <q-spinner-dots color="primary" size="40px" />
      </div>
    </template>
  </q-infinite-scroll>
```

```
      </div>
    </template>
  </PageWithSidebar>
</template>

<script>
import { mapStores } from "pinia";
import { useUserStore } from "../stores/user.js";
import ThemesService from "../services/ThemesService";
import SectionsService from "../services/SectionsService";
import NewPostModal from "../components/NewPostModal.vue";
import GlobalSearch from "../components/GlobalSearch.vue";
import PageWithSidebar from "../components/PageWithSidebar.vue";
import Post from "../components/Post.vue";

import Prism from "prismjs";
import "prismjs/themes/prism.css";

const sortOptions = {
  Найновіші: {
    param: "creationDate",
    type: "desc",
  },
  "В алфавітному порядку": {
    param: "title",
    type: "asc",
  },
  Найкорисніші: {
    param: "totalRate",
    type: "desc",
  },
  "Більше відповідей": {
    param: "messagesNumber",
    type: "desc",
  },
};

export default {
  data: function () {
```

```
    return {
      skip: 0,
      limit: 10,
      themes: [],
      sortOptions: sortOptions,
      sort: "Найновіші",
      subsection: null,
      showNewThemeModal: false,
      search: "",
    };
  },
  props: ["globalSearch"],
  components: {
    NewPostModal,
    Post,
    GlobalSearch,
    PageWithSidebar,
  },
  watch: {
    async search() {
      await this.refreshThemes();
      Prism.highlightAll();
    },
    async sort() {
      await this.refreshThemes();
      Prism.highlightAll();
    },
  },
  async mounted() {
    if (!this.$route.params.search) {
      this.subsection = await SectionsService.getSubsection(
        this.$route.params.sectionId,
        this.$route.params.subsectionId
      );
    } else {
      this.search = this.$route.params.search;
    }
    await this.refreshThemes();
    window.Prism = window.Prism || {};
```

```
    window.Prism.manual = true;
    Prism.highlightAll();
  },
  computed: {
    ...mapStores(useUserStore),
  },
  methods: {
    async onLoadMore(index, done) {
      this.skip += this.limit;
      const newThemes = await this.getThemes();
      this.themes.push(...newThemes);
      done();
      Prism.highlightAll();
    },
    async getThemes() {
      let themes;
      if (!this.$route.params.search) {
        themes = await ThemesService.getThemesBySubsectionId(
          this.$route.params.subsectionId,
          this.search,
          this.sortOptions[this.sort].param,
          this.sortOptions[this.sort].type,
          this.skip,
          this.limit
        );
      } else {
        themes = await ThemesService.getAllThemes(
          this.search,
          this.sortOptions[this.sort].param,
          this.sortOptions[this.sort].type,
          this.skip,
          this.limit
        );
      }
      return themes;
    },
    highlightAll: () => {
      Prism.highlightAll();
    },
```

```
    async refreshThemes() {
      this.skip = 0;
      this.themes = await this.getThemes();
    },
    async changeThemeModalStat(val) {
      this.showNewThemeModal = val;
      await this.refreshThemes();
      Prism.highlightAll();
    },
  },
};
</script>
```

B.5 Файл fileService.js

```
const uuid = require("uuid");
const ThemeModel = require("../models/Themes"),
      UserModel = require("../models/Users"),
      MessageModel = require("../models/Messages"),
      SectionModel = require('../models/Sections');
const fs = require('fs');
class FilesService {
  async uploadManyUserfiles(files) {
    const results = [];
    for (var key in files) {
      const file = files[key],
        type = file.name.split(".").pop(),
        newName = `${uuid.v4()}.${type}`,
        uploadPath = __dirname + "../../../files/userfiles/" + newName;
      await file.mv(uploadPath);
      results.push({
        systemName: newName,
        fileName: files[key].name,
        path: '/files/userfiles/'
      });
    }
    return results;
  }
  deleteUserfile(file) {
```

```
    fs.unlinkSync(`${__dirname}../../..${file.path}${file.systemName}`);
  }
  async deleteAllMessagesFilesByTheme(themeId) {
    const messages = await MessageModel.find({ themeId: themeId });
    messages.forEach(message => {
      message.files.forEach(file => {
        this.deleteUserfile(file)
      })
    })
  }
  async deleteUserfileFromTheme(file, themeId) {
    const result = await ThemeModel.updateOne({_id: themeId}, {
      '$pull': {
        'files':{ '_id': file._id }
      }
    });
    this.deleteUserfile(file);
    return result;
  }
  async getFilefromTheme(themeId, fileId) {
    const theme = await ThemeModel.findById(themeId);
    const file = theme.files.filter(file => file._id.toString() ===
fileId).pop();
    return file;
  }
  async deleteFilesFromTheme(themeId) {
    const theme = await ThemeModel.findById(themeId);
    theme.files.forEach(file => {
      this.deleteUserfile(file);
    })
  }
  async getFilefromMessage(messageId, fileId) {
    const message = await MessageModel.findById(messageId);
    const file = message.files.filter(file => file._id.toString() ===
fileId).pop();
    return file;
  }
  async deleteFilesFromMessage(messageId) {
    const message = await MessageModel.findById(messageId);
```

```
  message.files.forEach(file => {
    this.deleteUserfile(file);
  })
}
async deleteUserfileFromMessage(file, messageId) {
  const result = await MessageModel.updateOne({_id: messageId}, {
    '$pull': {
      'files':{ '_id': file._id }
    }
  });
  this.deleteUserfile(file);
  return result;
}
async changeAvatar(user, file) {
  const newName = await this.uploadAvatar(file);
  if(user.avatar) {
    this.deleteUserfile(user.avatar);
  }
  const result = await UserModel.updateOne(
    { _id: user.id },
    { $set : {
      avatar: {
        systemName: newName,
        fileName: file.name,
        path: '/files/avatars/'
      }
    }}
  );
  return result;
}
async uploadAvatar(file) {
  const type = file.name.split(".").pop(),
  newName = `${uuid.v4()}.${type}`,
  uploadPath = __dirname + "../../../files/avatars/" + newName;
  await file.mv(uploadPath);
  return newName;
}
async updateSubsectionIcon(sectionId, subsectionId, file, user) {
  const newName = await this.uploadAvatar(file);
```

```
    const section = await SectionModel.findById(sectionId);
    const subsection = await section.subsections.id(subsectionId);
    if (subsection.icon) {
      this.deleteUserfile(subsection.icon);
    }
    subsection.set({
      icon: {
        authorId: user.id,
        systemName: newName,
        fileName: file.name,
        path: '/files/avatars/'
      }
    });
    const result = await section.save();
    return result;
  }
  async deleteAvatar(user) {
    await UserModel.updateOne(
      { _id: user.id },
      { $set : { avatar: null } }
    );

fs.unlinkSync(`${__dirname}../../..${user.avatar.path}${user.avatar.syste
mName}`);
  }
}
module.exports = new FilesService();
```

B.6 Файл messageService.js

```
const MessageModel = require('../models/Messages'),
      ThemeModel = require('../models/Themes'),
      UserModel = require('../models/Users');
const FilesService = require('./filesService');
class MessageService {
    async createMessage(themeId, description, files, authorId) {
        const message = await MessageModel.create({themeId, description,
authorId});
        await ThemeModel.updateOne(
            { _id: themeId },
```

```
            { $inc: { 'messagesNumber': 1 } }
        );
        if (files) {
       const filesToPush = await FilesService.uploadManyUserfiles(files);
            filesToPush.forEach((file) => message.files.push({
                authorId,
                systemName: file.systemName,
                fileName: file.fileName,
                path: file.path
            }));
            const result = await message.save();
            return result;
        }
        return message;
    }
    async getAllMessagesByTheme(themeId, sort, sortType) {
        const messages = await MessageModel.find({themeId: themeId})
            .populate({path: 'authorId', model: 'User', select:
'username'})
            .populate({
                path: 'replies',
                model: 'Replie',
                populate: {
                    path:  'authorId',
                    model: 'User',
                    select: 'username'
                }
            })
            .sort([[sort, sortType]]);
        return messages;
    }
    async updateMessage(messageId, files, description, authorId) {
        const message = await MessageModel.findById(messageId);
        message.description = description || message.description;
        if (files) {
       const filesToPush = await FilesService.uploadManyUserfiles(files);
            filesToPush.forEach((file) => message.files.push({
                authorId,
                systemName: file.systemName,
```

```
                fileName: file.fileName,
                path: file.path
            }));
        }
        const result = await message.save();
        return result;
    }

    async deleteMessage(messageId) {
        await FilesService.deleteFilesFromMessage(messageId);
        const message = await MessageModel.findById(messageId);
        await ThemeModel.updateOne(
            { _id: message.themeId },
            { $inc: { 'messagesNumber': -1 } }
        );
        const result = await MessageModel.deleteOne({_id: messageId});
        return result;
    }
    async addRate(messageId, authorId, type) {
        await this.removeRate(messageId, authorId);
        const result = await MessageModel.updateOne({_id: messageId}, {
            '$push': {
                'rates': {authorId, type}
            },
            '$inc' : {
            'totalRate' : type === "true" ? 1 : type === "false" ? -1 : 0
            }
        });
        return result;
    }
    async removeRate(messageId, authorId) {
        const message = await MessageModel.findById(messageId);
        const rate = message.rates.filter(rate =>
rate.authorId.toString() === authorId).pop();
        if (rate) {
            const result = await MessageModel.updateOne({_id: messageId},
            {
                '$pull': {
                    'rates':{ 'authorId': authorId }
```

```
                },
                '$inc' : {
                    'totalRate' : rate.type === true ? -1 : rate.type ===
false ? 1 : 0
                }
            });
            return result;
        }
    }
    async getMessageById(messageId) {
        const message = await MessageModel.findById(messageId);
        return message;
    }
    async getReplyById(messageId, replieId) {
        const message = await MessageModel.findById(messageId);
        const replie = await message.replies.id(replieId);
        return replie;
    }
    async createReply(messageId, text, authorId) {
        const result = await MessageModel.updateOne(
            { _id: messageId },
            { '$push': { 'replies': { text, authorId } } }
        )
        return result;
    }
    async replyToAnotherReply(messageId, repliedId, text, authorId) {
        const message = await MessageModel.findById(messageId);
        const replied = await message.replies.id(repliedId);
        const repliedAuthor = await UserModel.findById(replied.authorId);
        const result = await MessageModel.updateOne(
            { _id: messageId },
            { '$push': {
                'replies': {
                    text,
                    authorId,
                    repliedId: repliedId,
                    repliedAuthorName: repliedAuthor.username
                }
            } }
```

```
        )
        return result;
    }
    async getAllMessages(search, skip, limit) {
        let messages;
        if(!search) {
            messages = await MessageModel.find()
                .skip(skip)
                .limit(limit);
        } else {
            const regex = new RegExp(search, 'i')
            const filter = [
                { description: { $regex: regex } }
            ];
            if (new RegExp("^[0-9a-fA-F]{24}$").test(search)) {
                filter.push({ _id: search })
            }
            messages = await MessageModel.find()
                .or(filter)
                .skip(skip)
                .limit(limit);
        }
        const result = await MessageModel.populate(messages,
            {path: 'authorId', model: 'User', select: 'username'}
        );
        return result;
    }
}
module.exports = new MessageService();
```

B.7 Файл Messages.js

```
const {Schema, model} = require('mongoose');
const Rate = require('./Rates'),
      Reply = require('./Replies'),
      File = require('./Files');
const MessagesSchema = new Schema({
    themeId: {
        type: Schema.Types.ObjectId,
```

```
        ref: 'Teme',
        required: 'Message theme'
    },
    authorId: {
        type: Schema.Types.ObjectId,
        ref: 'User',
        required: 'Message author'
    },
    description: {
        type: String,
        required: 'Message text'
    },
    creationDate: {
        type: Date,
        default: Date.now
    },
    replies: {
        type: [Reply.schema],
        default: []
    },
    files: {
        type: [File.schema],
        default: []
    },
    totalRate: {
        type: Number,
        default: 0
    },
    rates: {
        type: [Rate.schema],
        default: []
    }
});

module.exports = model('Message', MessagesSchema);
```

ДОДАТОК Г

Слайди презентації

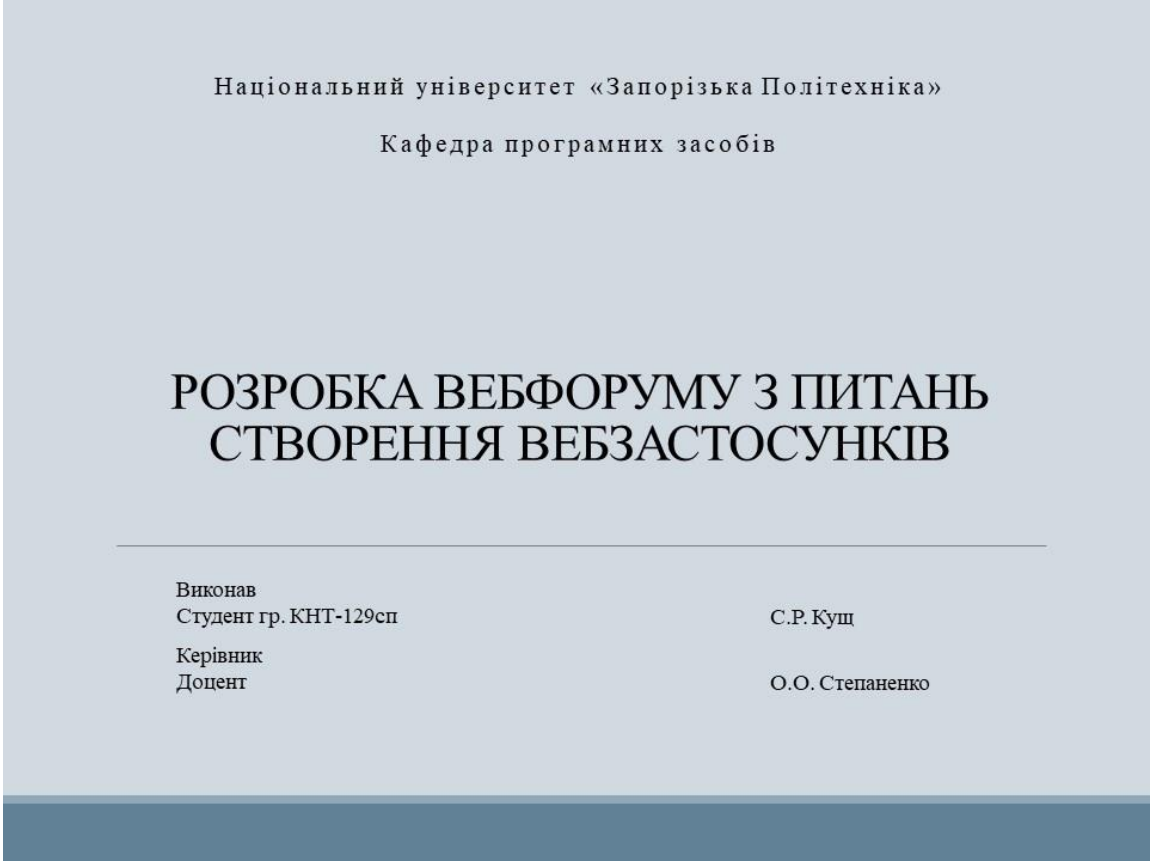

Рисунок Г.1 – Слайд № 1

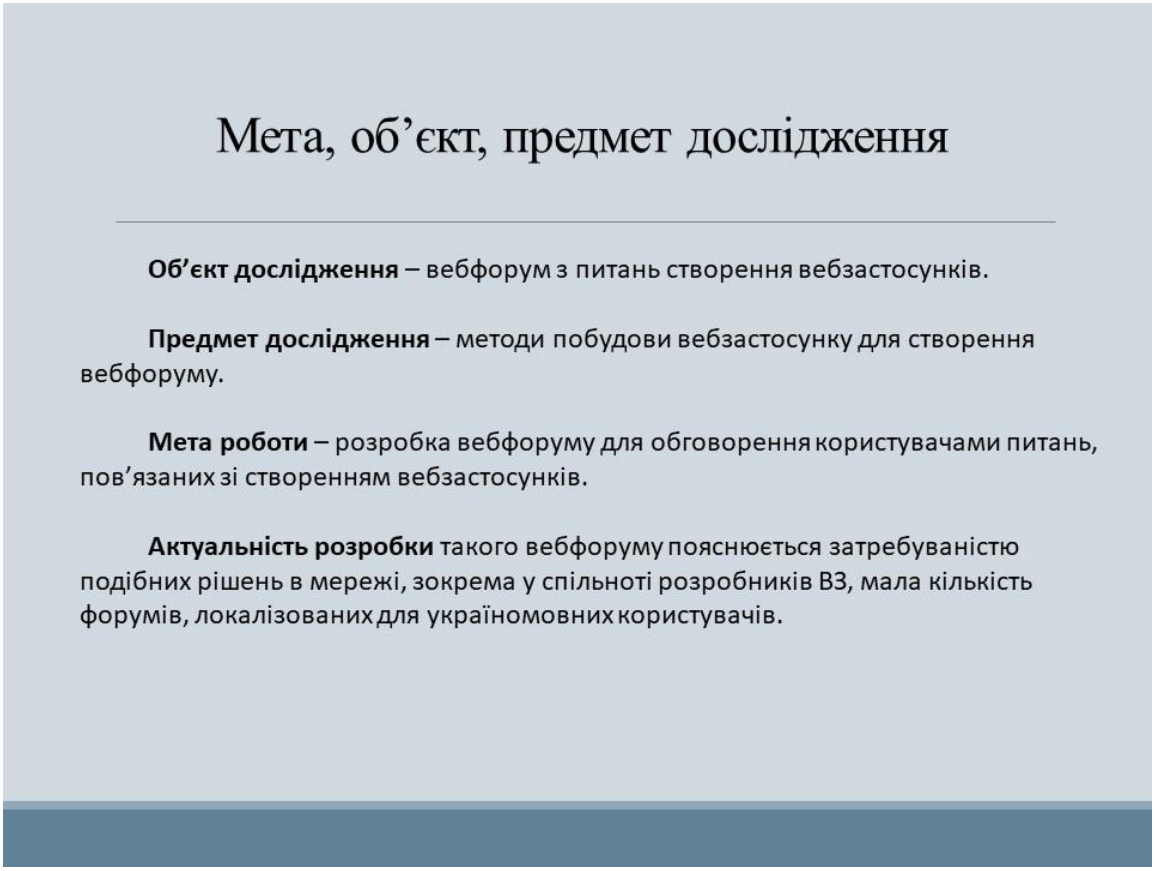

Рисунок Г.2 – Слайд № 2

Постановка завдання до роботи

Метою роботи є розробка вебфоруму з питань створення вебзастосунків, призначеного для обговорення користувачами різних тем, пов’язаних з веброзробкою, пошуку рішення проблем, прикладів використання бібліотек та фреймворків, фрагментів готового коду алгоритмів.

Рисунок Г.3 – Слайд № 3

Огляд існуючих аналогів

Критерій порівняння	Replace.org.ua	Stack Overflow	Dream.in.code	Вебзастосунок, що розробляється
Система оцінювання тем та повідомлень	Відсутня	Присутня	Присутня	Присутня
Зручний дизайн та навігація	Порівняно сучасний дизайн, зручна навігація	Сучасний дизайн, зручна навігація	Застарілий дизайн, відносно зручна навігація	Сучасний дизайн, зручна навігація
WYSIWYG-редактор тексту	Відсутній	Присутній	Відсутній	Присутній
Функціонал обміну файлами	Відсутній	Відсутній	Присутній	Присутній
Адаптивний дизайн	Відсутній	Присутній	Відсутній	Присутній

Рисунок Г.4 – Слайд № 4

Огляд мов програмування для розробки вебзастосунків

Критерій порівняння	PHP	Python	JavaScript
Можливість виконання на клієнті	-	-	+
Зручне налагодження	+	+	-
Кросплатформовість	+	+	+
Швидкість виконання	+	-	+
Спеціалізація на веброзробці	+	-	+
Зрозуміла та повна документація	+	+	+
Готові рішення та фреймворки	+	+	+

Рисунок Г.5 – Слайд №5

Огляд існуючих середовищ розробки

Критерій порівняння	WebStorm	VS Code
Необхідність платної підписки	+	-
Інтелектуальне доповнення	+	+
Кросплатформовість	+	+
Зручний рефакторинг коду	+	+
Інтегрований Git-інструментарій	+	+
Можливість встановлення сторонніх розширень	-	+

Рисунок Г.6 – Слайд № 6

Рисунок Г.7 – Слайд № 7

Рисунок Г.8 – Слайд № 8

Рисунок Г.9 – Слайд № 9

Рисунок Г.10 – Слайд № 10

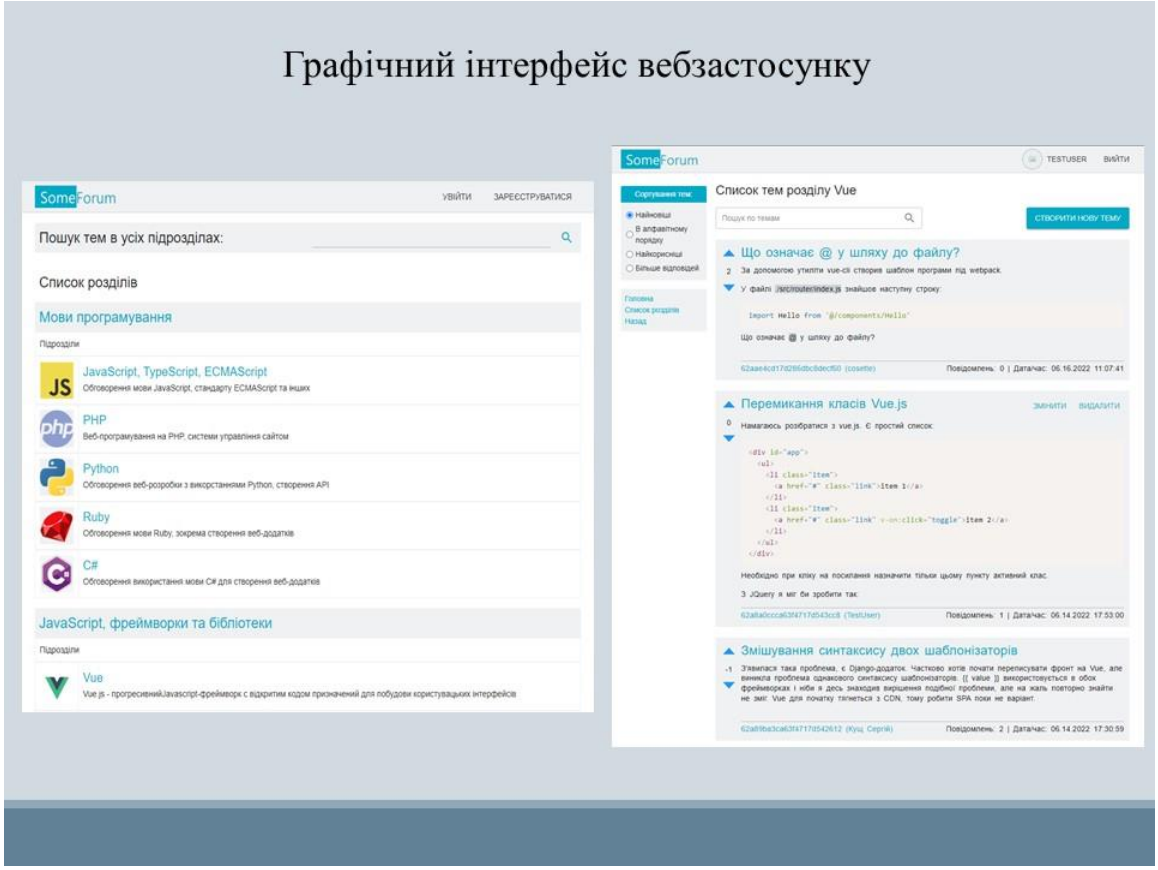

Рисунок Г.11 – Слайд № 11

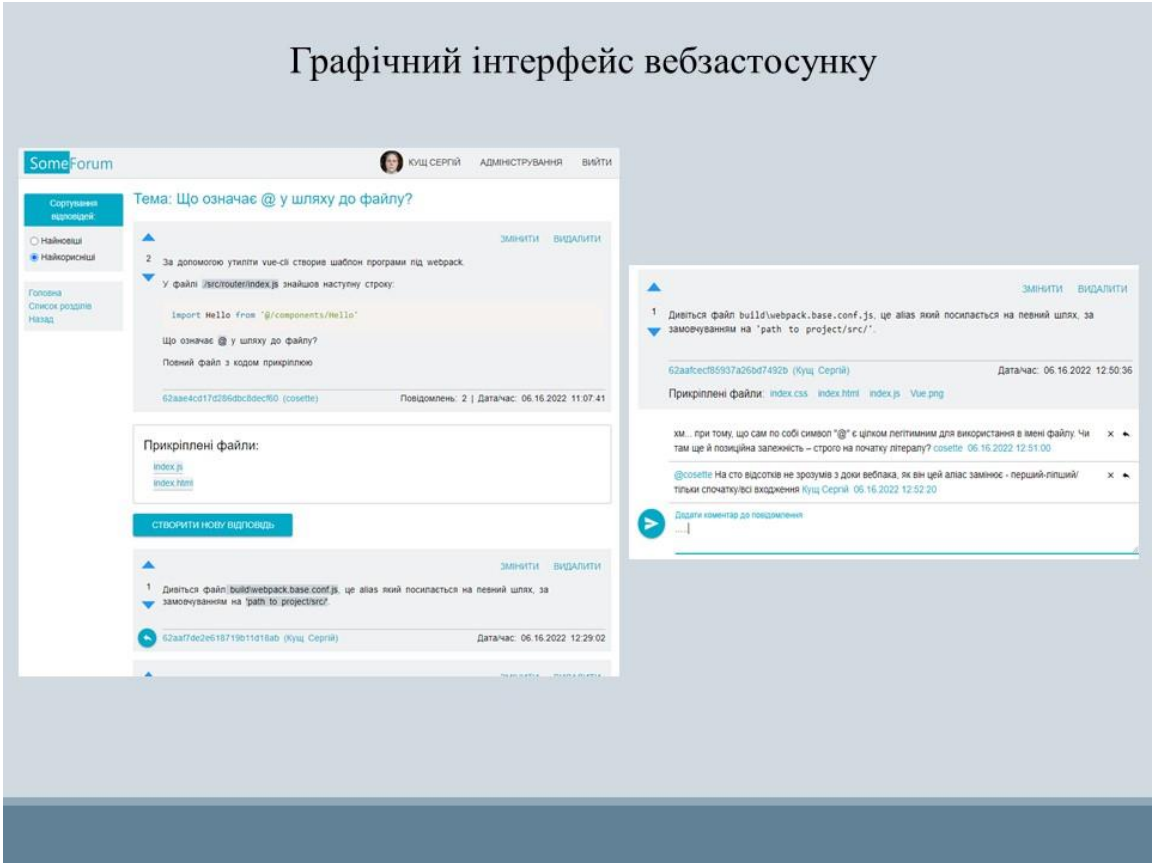

Рисунок Г.12 – Слайд № 12

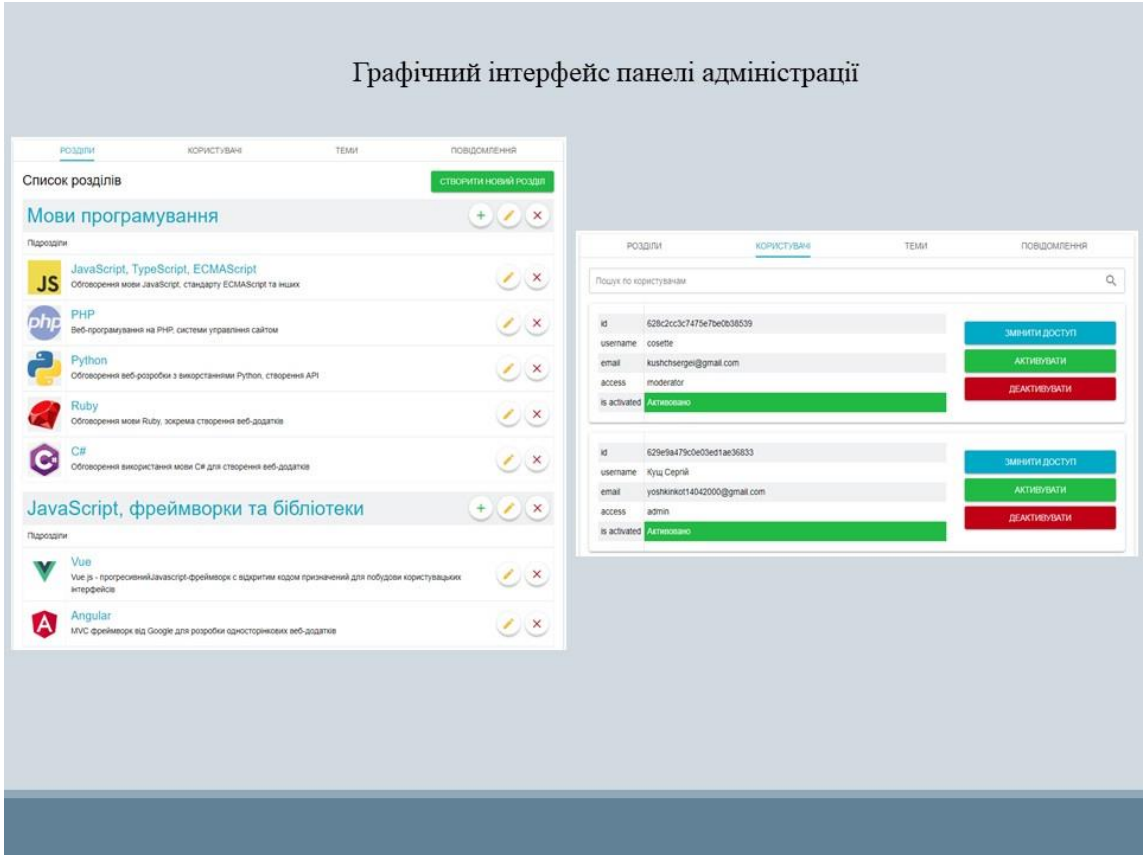

Рисунок Г.13 – Слайд № 13

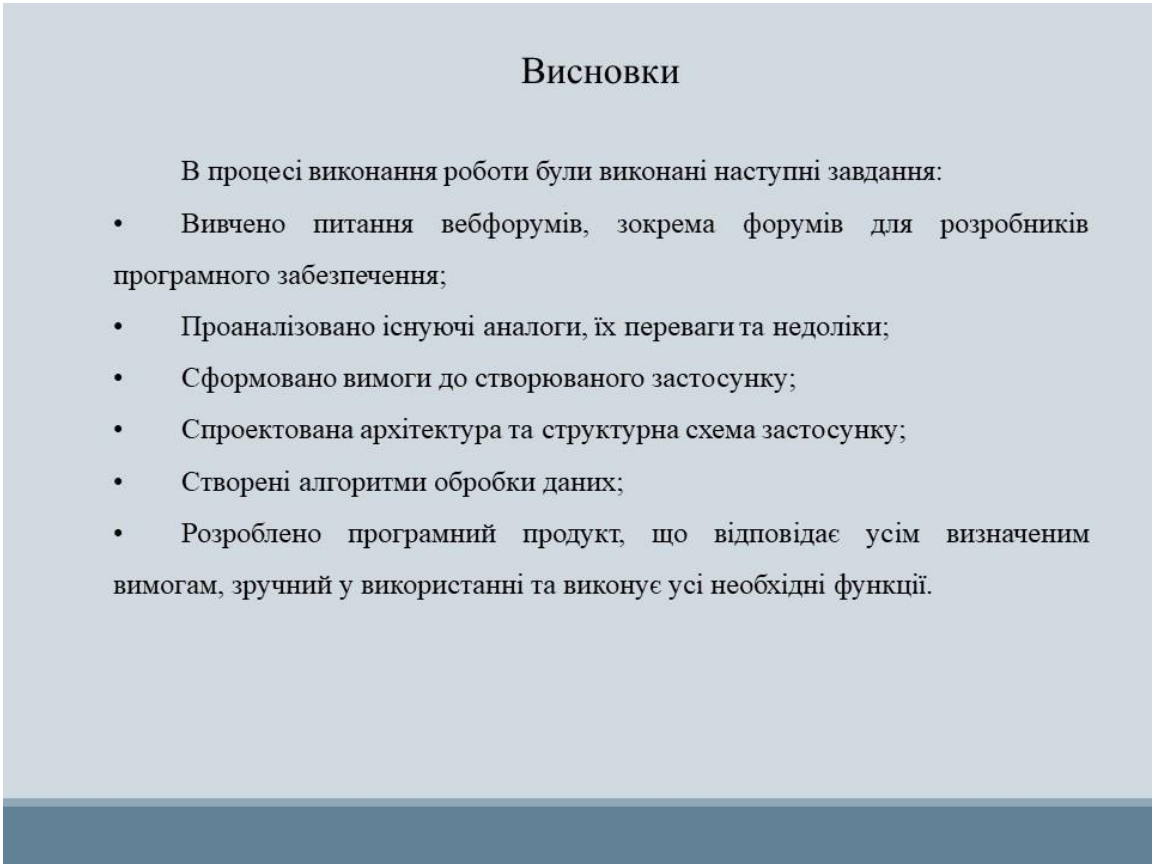

Рисунок Г.14 – Слайд № 14