Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-РЕСУРСУ «ОНЛАЙН ФОРУМ»

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Попович Ю. Ю
(прізвище та ініціали)
Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: к.т.н., доцент каф. АІІТ
Гармаш В. В.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А. А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Поповичу Юрію Юрійовичу

1. Тема роботи: Розробка WEB-ресурсу «Онлайн форум»
керівник роботи: Крилик Л. В., к.т.н., доц.
затверджені наказом вищого навчального закладу «20» 03.2025 року № 97
2. Термін подання студентом роботи 30 травня 2025р.
3. Вихідні дані до роботи: мова програмування – об'єктно-орієнтована, функціональна; кількість вкладеності тем – без обмежень; кількість підтримуваних платформ не менше 2; система має підтримувати багаторівневу структуру з щонайменше 3 основними модулями: аутентифікація, сповіщення та система лайків; підтримка щонайменше 2 ролей користувачів з різними рівнями доступу до функціоналу системи; інтуїтивно зрозумілий інтерфейс.
4. Зміст текстової частини (перелік питань, які потрібно розробити): вступ; обґрунтування доцільності розробки WEB-ресурсу «Онлайн форум»; проектування WEB-ресурсу «Онлайн форум»; програмна реалізація WEB-ресурсу «Онлайн форум»; висновки; список використаних джерел; додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): загальна структурна схема функціонування системи; загальна структурна схема компонентів програмного модуля «Онлайн форум»; UML-діаграма класів модуля роботи з аутентифікацією серверної частини; UML-діаграма класів модуля роботи з профілем серверної частини; UML-діаграма класів модуля роботи із сповіщеннями серверної частини; фрагмент ER-діаграми, що демонструє зв'язки між екземплярами сутностей предметної області «Онлайн форум»; ER-модель предметної області «Онлайн форум»; схема алгоритму роботи програмного модуля WEB-ресурсу «Онлайн форум» діаграма діяльності для методу login сервісу AuthService; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Крилик Л. В., к.т.н., доц. каф. КН		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Вступ. Обґрунтування доцільності розробки WEB-ресурсу «Онлайн форум»	05.05.25	07.05.25	
2	Проектування WEB-ресурсу «Онлайн форум»	08.05.25	11.05.25	
3	Програмна реалізація WEB-ресурсу «Онлайн форум»	12.05.25	15.05.25	
4	Висновки та аналіз отриманих результатів	16.05.25	26.05.25	
5	Оформлення додатків	27.05.25	29.05.25	
6	Розробка інструкції користувача	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ **Попович Ю. Ю**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Крилик Л. В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 108 сторінок формату А4, які включають 28 рисунків і 8 таблиць. У списку використаних джерел зазначено 31 найменування.

Метою роботи є розширення функціональних можливостей WEB-ресурсу «Онлайн форум».

У загальній частині роботи на основі аналізу існуючих технічних рішень, методів та технологій доведена доцільність розробки WEB-ресурсу «Онлайн форму». Розроблено алгоритм роботи та UML діаграми класів серверної частин програмного модуля «Онлайн форум», які спрощують розробку та розуміння структури програмного забезпечення. Програмний продукт розроблений в інтегрованому середовищі Visual Studio Code.

Розроблено програмний продукт та проведено його тестування. Результати тестування розробки вказують на те, що основні функції WEB-ресурсу «Онлайн форум» успішно реалізовано.

Ключові слова: програмний модуль, WEB-розробка, WEB-ресурс, форум, серверний рендеринг.

‘

ABSTRACT

The bachelor's thesis consists of 108 pages of A4 format, including 28 figures and 8 tables. The list of references contains 31 sources.

The aim of the thesis is to expand the functional capabilities of the «Online Forum» web resource.

In the general part of the thesis, based on the analysis of existing technical solutions, methods, and technologies, the feasibility of developing the «Online Forum» web resource is justified. An algorithm of operation and UML class diagrams for the server-side module of the «Online Forum» have been developed, which simplify the development process and understanding of the software structure. The software product was developed using the Visual Studio Code integrated development environment.

The software product has been developed and tested. The testing results indicate that the main functions of the «Online Forum» web resource have been successfully implemented.

Keywords: software module, web development, web resource, forum, server-side rendering.

‘

ЗМІСТ

ВСТУП

Актуальність досліджень.

Нині потреба у платформах для обміну знаннями, думками та досвідом у будь-яких сферах нашого життя обумовлює актуальність розробки WEB-ресурсу «Онлайн форум». На сьогодні форуми, наряду з чатами, залишаються одним із найпопулярніших інструментів комунікації, особливо в умовах глобалізації та цифровізації. Обидва формати сприяють інтерактивній взаємодії між користувачами, але виконують різні функції та відповідають різним потребам. Форум забезпечує можливість організації спільнот за інтересами, структурованих обговорень та тривалого збереження інформації, що дозволяє повернутися до матеріалів навіть через тривалий час.

Чати, у свою чергу, орієнтовані на миттєвий обмін повідомленнями у реальному часі, що робить їх незамінними для оперативної комунікації. Проте їхній динамічний характер і відсутність структурованості можуть створювати труднощі у відстеженні довготривалих обговорень або пошуку необхідної інформації. Форум не є замінником чату, так само як чат не може повноцінно замінити форум: кожен з цих інструментів має свої унікальні переваги. Форум найбільше відповідає стилю глибоких дискусій, збереження знань і створення спільнот, тоді як чати найкраще працюють для швидкого вирішення питань і неформального спілкування. Разом вони доповнюють один одного, задовольняючи різні аспекти сучасних потреб у комунікації.

Аналіз існуючих рішень, як вітчизняних так і зарубіжних показує, що більшість платформ мають низку обмежень: складність у налаштуванні, низька продуктивність під високими навантаженнями або недостатня гнучкість у модифікації функціоналу. Ці недоліки знижують їх ефективність у вирішенні сучасних завдань користувачів.

Отже, тема «Розробка WEB-ресурсу «Онлайн форум»» є важливою та актуальною в умовах зростаючого попиту на цифрові комунікаційні платформи.

Реалізація проєкту сприятиме створенню зручного, безпечного та економічно ефективного інструменту для взаємодії користувачів.

Метою дослідження є розширення функціональних можливостей WEB-ресурсу «Онлайн форум».

Об’єктом дослідження процес онлайн-комунікації та взаємодії користувачів.

Предметом дослідження є програмні засоби та технології для реалізації WEB-ресурсу «Онлайн форум».

Задачі дослідження:

1. Обґрунтувати доцільність розробки WEB-ресурсу «Онлайн форум».
2. Спроектувати WEB-ресурс «Онлайн форум».
3. Програмно реалізувати WEB-ресурс «Онлайн форум».
4. Виконати тестування програми та провести аналіз отриманих результатів.
5. Розробити інструкцію користувача.

Апробація роботи.

Результати досліджень було апробовано на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ) м. Вінниці у 2025 р. [1].

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1] та отримано свідоцтво про реєстрацію авторського права на твір у формі комп’ютерної програми – «WEB-ресурс «Онлайн-форум» [2].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-РЕСУРСУ «ОНЛАЙН ФОРУМ»

1.1 Сучасний стан розвитку WEB-технологій

Натепер WEB-ресурси стали невід'ємною частиною нашого життя, виконуючи широкий спектр завдань, наприклад, від персональних блогів до корпоративних порталів та інтерактивних платформ. Завдяки такому різноманіттю сфер застосування, WEB-технології еволюціонували настільки, що вони дозволяють створювати складні, багатофункціональні застосунки, здатні відповідати очікуванням сучасних користувачів.

Однією з ключових характеристик сучасних WEB-ресурсів є швидкість взаємодії з користувачем. Для досягнення цієї мети використовуються різні підходи до рендерингу контенту:

- Клієнтський рендеринг (CSR) – це підхід, що використовується у багатьох сучасних WEB-застосунках, таких як SPA (Single Page Applications). Цей підхід забезпечує високу інтерактивність і гнучкість. Однак основним його недоліком є повільний час початкового завантаження, оскільки весь необхідний код передається користувачу для виконання на стороні клієнта. Також через те, що усі обчислення і виконання коду SPA виконуються на пристрої клієнта, тобто спочатку відсутня згенерована сторінка, яка б дозволила уникнути проблеми зниження продуктивності та SEO-оптимізації.

- Серверний рендеринг (SSR) – це підхід, що забезпечує швидше відображення вмісту, оскільки сервер передає готовий HTML, який браузер відображає миттєво. Це значно покращує SEO і робить контент доступним навіть для користувачів із повільним з'єднанням. Проте SSR збільшує навантаження на сервери, а для динамічних компонентів може стати складним і ресурсоємним рішенням [3].

- Статична генерація сторінок (SSG) – це підхід, що дозволяє заздалегідь згенерувати HTML-сторінки під час створення проєкту. Він забезпечує

надзвичайно швидке завантаження, але обмежений у можливостях для інтерактивних функцій [3].

Попри ці можливості, багато існуючих рішень мають значні недоліки, наприклад, застарілі системи, такі як phpBB, Simple Machines Forum або vBulletin, часто зосереджені на статичному або клієнтському рендерингу, що обмежує їхню продуктивність і зручність. Наприклад, такі системи:

- не завжди забезпечують швидку роботу на мобільних пристроях через відсутність адаптивного дизайну;
- часто неефективно працюють із пошуковими системами через недостатню оптимізацію HTML-контенту;
- мають проблеми з інтеграцією сучасних функцій, таких як миттєві оновлення чи персоналізація.

Ці недоліки відкривають можливості для вдосконалення. Наприклад, у випадку «Онлайн форуму», який поєднує серверний рендеринг і клієнтський рендеринг там, де це доцільно, можна досягти кращого балансу між продуктивністю, зручністю та гнучкістю [4].

Серверний рендеринг доцільно застосовувати для [5]:

- відображення загального вмісту, наприклад, списку тем, заголовків постів або популярних дискусій, які є критично важливими для швидкого першого завантаження та SEO;
- роботи з публічними сторінками, наприклад, сторінки з інформацією про спільноту чи правила, які мають бути доступні пошуковим системам;
- випадків, коли необхідний доступ до backend-утиліт.

Натомість клієнтський рендеринг ефективніше використовувати для:

- інтерактивних компонентів, таких як створення нового допису, реакції на коментарі чи оновлення тем у реальному часі;
- особистих кабінетів користувачів, які вимагають персоналізації та швидкої динаміки;
- випадків, коли необхідно оброблювати event-події (click, change, drag, drop тощо);

‘

- випадків, коли відбувається взаємодія з браузерним API;
- для сторінки авторизації користувача.

Існуючі підходи мають свої переваги, але не забезпечують універсального рішення, яке було б достатньо ефективним для сучасних потреб. Саме тому впровадження сучасного WEB-ресурсу «Онлайн форум» з використанням серверного рендерингу, побудованого на інструментах сучасних фреймворків (такими як Next.js, Nust, Remix тощо), дозволить вирішити існуючі проблеми та створити ефективну, зручну й масштабовану платформу.

1.2 Значення форумів у сучасному світі

Форуми є популярним інструментом комунікації, що об’єднує людей за спільними інтересами, сприяючи обміну знаннями та досвідом. У бізнесі вони допомагають залучати клієнтів, надавати підтримку, обговорювати продукти та збирати зворотний зв’язок, а також формувати експертні спільноти. В освіті форуми, наприклад Coursera, сприяють обміну знаннями та вирішенню проблем. Тематичні форуми, як Stack Overflow для програмістів, слугують для обговорення технічних питань і створення архівів знань. Серед ентузіастів форуми з відеоігор, кулінарії чи колекціонування дозволяють ділитися новинами, порадами та організовувати заходи.

Не зважаючи на досягнення у розвитку, сучасні форуми стикаються з низкою проблем, які обмежують їхній потенціал і потребують вирішення [6 − 8]:

- багато класичних форумів мають застарілий дизайн, що ускладнює використання форумів, особливо для молодшого покоління, яке звикло до інтерактивних і зручних інтерфейсів соціальних мереж.
- відсутність чи обмеженість функцій обміну інформацією у реальному часі робить форуми менш привабливими, оскільки в умовах конкуренції з соціальними мережами це стає вагомим недоліком.

- відсутність адаптивності, тобто багато платформ не адаптовані для використання на мобільних пристроях, що є критичним у світі, де більшість інтернет-користувачів взаємодіють із контентом через смартфони.

- форуми, у звязку зі своєю важкою логікою та масивам інформації, стають мішенню для спаму чи хакерських атак, хоча сьогодні все більше і більше розробників використовують сервіси для боротьби зі спам атаками.

Онлайн форуми залишаються важливим елементом цифрового середовища, але їхній розвиток вимагає адаптації до сучасних тенденцій. Впровадження адаптивного дизайну, інтеграція інноваційних технологій і поліпшення безпеки є основними напрямками для підвищення ефективності цих платформ.

1.3 Огляд систем-аналогів

Для того, щоб створити ефективний ресурс, важливо провести аналіз існуючих аналогів, визначивши їх переваги, недоліки та можливості для вдосконалення.

Розглянемо сервіси, такі як «UKRAINE GTA», «Політичний форум» та «Red Forum», які забезпечують базовий функціонал взаємодії користувачів.

«UKRAINE GTA» – це форум, що створений для взаємодії гравців комп’ютерної гри GTA [9]. Такий ресурс застосовують гравці для обговорення ігрового процесу, подій і механік, а також для спілкування з іншими учасниками. Тут можна подати заявку на вступ до фракцій, таких як поліція, армія чи мафії, або ж оформити запит на отримання майна. Гравці також використовують його для апеляцій банів, де пояснюють обставини і просять про розблокування. Крім цього, форум слугує місцем для подання заявок на лідерство, адміністрацію або інші офіційні позиції на сервері. Зовнішній вигляд форуму наведений на рисунку 1.1.

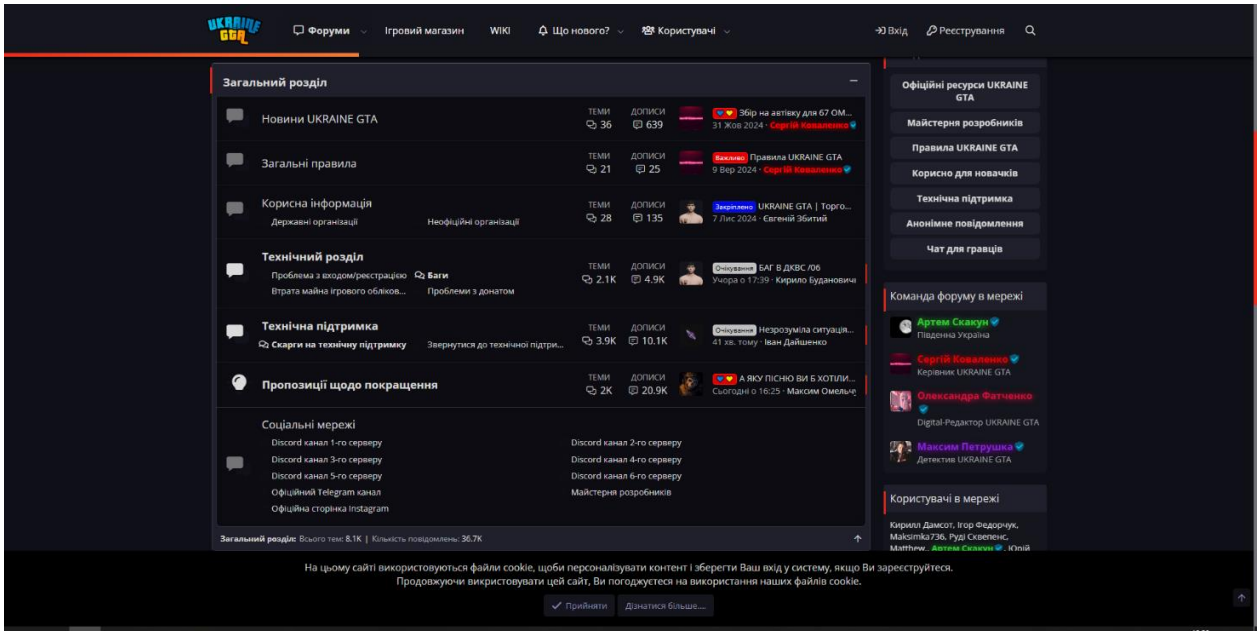

Рисунок 1.1 − Загальний вигляд інтерфейсного вікна форуму «UKRAINE GTA»

Переваги форуму «UKRAINE GTA» такі:

- форум спеціалізується на конкретній темі, що забезпечує концентрацію цільової аудиторії;
- має чітку структуру розділів, що дозволяє швидко знайти потрібну інформацію;
- усі обговорення після завершення дискусій переміщуються у архів, що дозволяє переглядати їх у майбутньому.

Однак існують і недоліки:

- вузька спеціалізація, оскільки він застосовується у переважній більшості лише для офіційної комунікації між гравцями та адміністрацією;
- можливе повільне завантаження сторінок, оскільки весь програмний код виконується на стороні клієнта;
- відсутність інтеграції форуму з ігровим акаунтом ускладнює захист від фейкових профілів і спаму;
- має проблеми при роботі з мобільними пристроями.

«Політичний форум» – це форум, який підтримується власниками WEB-сайту Українська правда і призначений для обговорення актуальних політичних питань, суспільних подій, економічних тем та інших важливих аспектів життя
‘

країни. Користувачі можуть ділитися своїми думками щодо поточних подій, обговорювати політичні рішення, закони чи ініціативи, спілкуватися з іншими учасниками, обмінюючись аргументами та поглядами (рис. 1.2) [10].

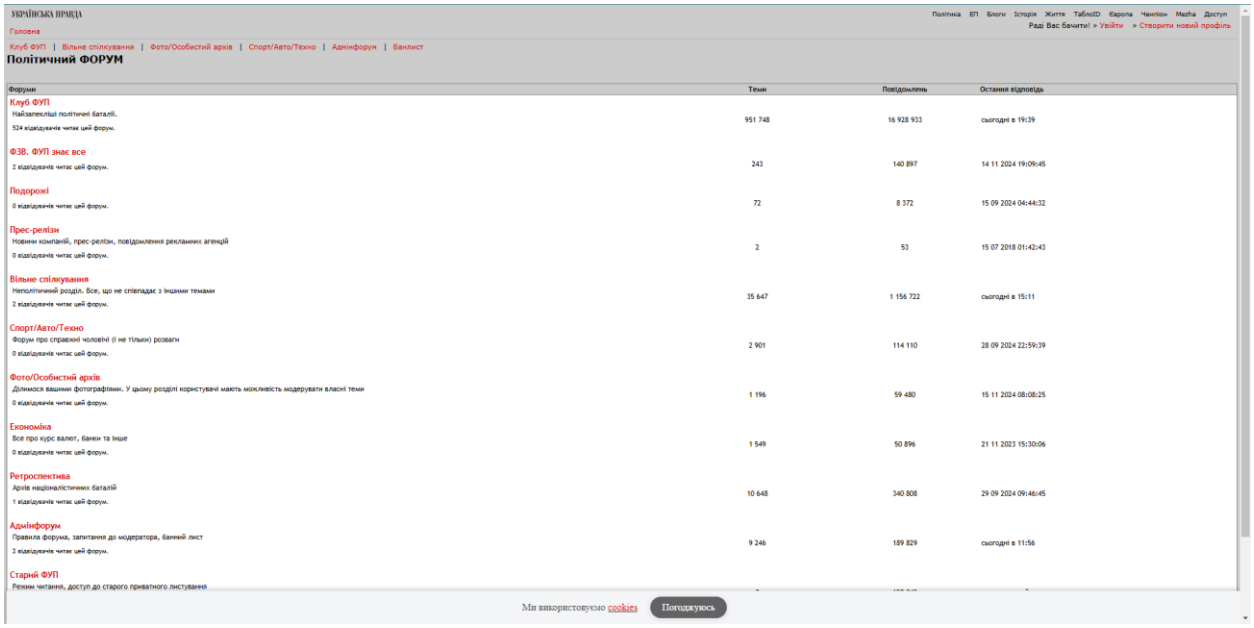

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна форуму «Політичний форум»

Переваги форуму «Політичний форум»:

- наявність великої кількості підрозділів;
- форум є публічним і не містить закритих розділів;
- містить надзвичайно велику кількість різноманітної інформації.

Недоліки:

- застарілий інтерфейс;
- досить погана підтримка користувачів, що використовують screen reader;
- погана SEO оптимізація;
- інтерфейс неінтуїтивний.

Форум «RED forum» – персональний некомерційний проект, який переважно орієнтований на фахівців і любителів усіх галузей телекомунікацій. Уміння дискутувати, наявність почуття гумору та проява грамотності є обов'язковими рисами для кожного з його учасників. Форум об’єднує учасників,

які цікавляться новинками в технологічній сфері, обмінюються досвідом, діляться порадами щодо використання техніки чи послуг, а також обговорюють інші загальні теми, які цікавлять спільноту. Це місце для консультацій, вирішення технічних питань і неформального спілкування між учасниками, об’єднаними спільними інтересами (рис. 1.3) [11].

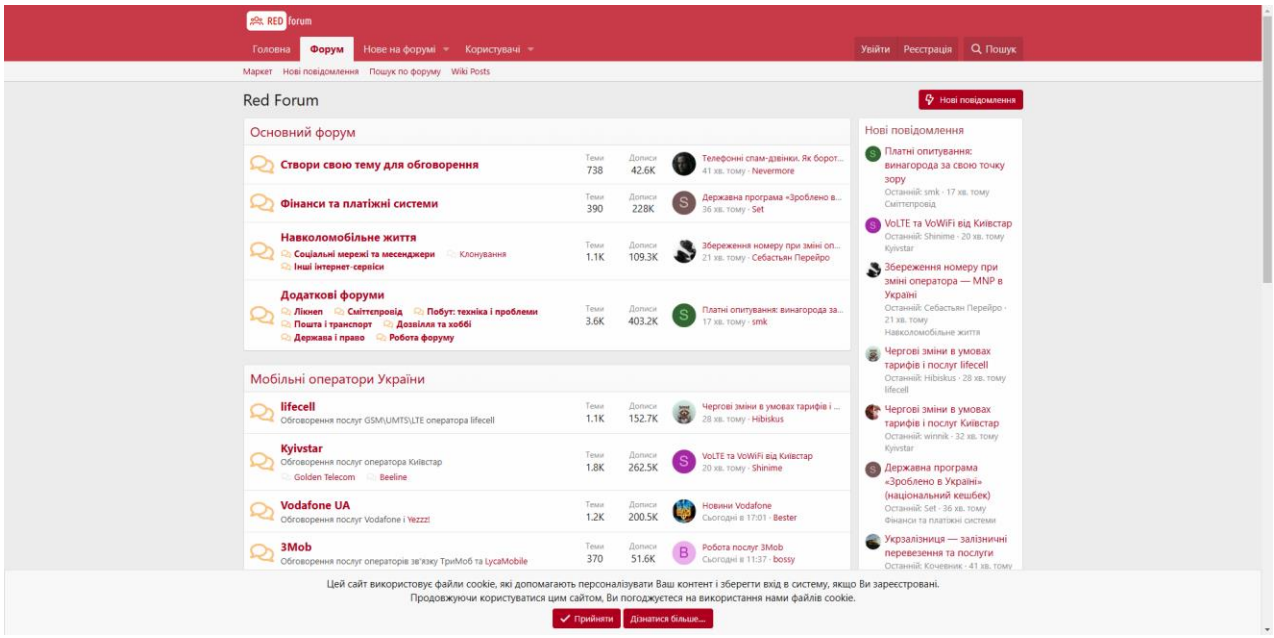

Рисунок 1.3 − Загальний вигляд інтерфейсного вікна форуму «Red Forum»

Переваги форуму «Red Forum»:

- як і форум UKRAINE GTA, він має досить схожу структуру;
- має чітку структуру розділів;
- немає проблем із роботою на мобільних пристроях.

Недоліки:

- можливе повільне завантаження сторінок, оскільки весь програмний код виконується на стороні клієнта;
- побудований з використанням XenForo, є комірційним продуктом, тому необхідно постійно поновлювати платну ліцензію.

‘

1.4 Формулювання вимог та постановка задачі

Зі зростанням популярності форумів як платформ для обміну думками виникає потреба в зручному та ефективному онлайн-форумі для широкої аудиторії. Багато існуючих платформ обмежені вузькою спеціалізацією, повільним завантаженням або слабкою безпекою. Наприклад, «Політичний форум» має багато розділів, але застарілий та має неінтуїтивний інтерфейс, який ускладнює використання.

Необхідно створити WEB-ресурс «Онлайн форум» із зручним, інтуїтивним інтерфейсом, швидкою роботою та високим рівнем безпеки. Дизайн має бути зручним та зрозумілим для усіх користувачів, із легким введенням даних, чіткими результатами пошуку та обговорень.

Завдання для розробки цього онлайн форуму включають:

- реалізація модуля реєстрації та авторизації користувачів.
- розробка системи взаємодії з секціями, темами, постами та коментарями.
- створення інтуїтивно зрозумілої системи коментування та відповідей.
- реалізація модераційних інструментів для підтримки порядку на форумі.
- проведення тестування та оптимізація роботи ресурсу.

Ключовим аспектом є зручність використання ресурсу для різних категорій користувачів, що забезпечить популярність та ефективність онлайн форуму.

1.5 Висновок

В цьому розділі було проведено детальний аналіз існуючих платформ для реалізації онлайн форумів, а також розглянуті їхні переваги та недоліки. Враховуючи швидкий розвиток WEB-технологій і сучасні вимоги до WEB-ресурсів, було прийнято рішення розробити універсальний WEB-ресурс для онлайн форуму. Такий підхід дозволить забезпечити високу функціональність, адаптивність та зручність для користувачів.

Аналіз існуючих форумів, таких як «UKRAINE GTA», «Політичний форум» і «Red Forum», показав, що більшість з них мають обмеження, пов'язані з вузькою спеціалізацією, повільним завантаженням сторінок та проблемами з SEO-оптимізацією. Сучасні підходи до рендерингу контенту, такі як клієнтський рендеринг, серверний рендеринг та статична генерація сторінок, мають свої переваги, але не забезпечують універсального рішення для всіх задач. Тому для онлайн форуму доцільно застосовувати комбінований підхід, використовуючи серверний рендеринг для швидкого завантаження контенту та SEO, а клієнтський рендеринг для інтерактивних компонентів та персоналізації.

Створення онлайн форуму з інтеграцією сучасних технологій дозволить усунути існуючі проблеми, такі як застарілий дизайн і недостатня інтерактивність, що є основними недоліками сучасних форумів. Онлайн форум, який буде враховувати ці аспекти, стане зручним і ефективним інструментом для взаємодії користувачів, обміну знаннями та створення спільнот. Впровадження використання сучасних фреймворків дозволить створити зручну, швидку та безпечну платформу.

Отже, розробка онлайн форуму є актуальним і необхідним кроком для покращення досвіду користувачів, а також для забезпечення ефективної комунікації в мережі.

2 ПРОЕКТУВАННЯ WEB-РЕСУРСУ «ОНЛАЙН ФОРУМ»

2.1 Розробка структури програмного модуля WEB-ресурсу «Онлайн форум»

WEB-ресурс «Онлайн форум» є клієнт-серверною системою, що функціонує через браузер. Його архітектура розділена на дві основні частини: клієнтську частину, яка реалізовуватиметься з використанням Next.js, та серверну частину, що буде побудована на базі NestJS. Кожен з цих компонентів буде мати власну структуру та відповідатиме за окремі функціональні задачі.

Клієнтська частина відповідатиме за відображення інтерфейсу та взаємодію з ним, а серверна – за логіку, роботу з базою даних і авторизацію. У Next.js застосовуватимуться різні стратегії рендерингу для продуктивності, SEO та зручного користувацького досвіду. Сторінки секцій, тем і профілів використовуватимуть інкрементальний статичний рендеринг із кешем, що періодично оновлюватиметься, а при мутаціях кеш оновлюватиметься вручну для підтримки актуальності контенту. Сторінки постів використовуватимуть серверний рендеринг для динамічного завантаження даних, покращення SEO та швидкого відображення. Коментарі до постів завантажуватимуться на клієнті через клієнтський рендеринг, зменшуючи навантаження на сервер і забезпечуючи гнучке оновлення інтерфейсу в реальному часі для коментарів, лайків і сповіщень [12].

З боку клієнта маршрутизація реалізовуватиметься за допомогою Next.js App Router v.15, який забезпечує автоматичне розділення логіки за шляхами сторінок та дозволяє гнучко налаштовувати SSR, ISR чи SSG для окремих маршрутів, що сприяє швидкості роботи застосунку та зменшенню кількості запитів до сервера.

Для оптимізації SEO на сторінках, що використовують серверний рендеринг, генеруються динамічні мета-теги, наприклад, title, description, keywords, які формуються на основі вмісту сторінки.

Функціонально, WEB-ресурс «Онлайн форум» підтримуватиме:

- створення та редагування секцій і тем з довільною вкладеністю;
- створення постів та коментарів із можливістю редагування, включаючи завантаження зображень до коментарів;
- систему вподобань та сповіщень для інформування користувачів про взаємодії, зокрема сповіщення про лайки до коментарів та сповіщень власнику про нові коментарі;
- ролі користувачів із розширеними повноваженнями з можливістю редагування та видалення не лише власних матеріалів.

Система сповіщень наразі включатиме сповіщення про лайки до коментарів користувача та сповіщення власникам постів про нові коментарі до їхніх публікацій. У майбутньому планується розширення цієї функціональності для охоплення інших подій, таких як згадки користувачів у коментарях чи модераційні дії.

Для зручності реалізації цих функціональних можливостей та для забезпечення масштабованості та підтримку застосунку, структура програмного коду розділена на окремі функціональні модулі. Такий підхід дозволить ефективно організовувати код, ізолювати логіку окремих частин системи та прискорити процес розробки. Кожен модуль реалізовуватиме окрему функціональність, що дозволить уникнути дублюванню коду.

Клієнтську частину можна розділити на декілька ключових модулів, кожен із яких відповідатиме за окрему функціональність і включатиме відповідні сторінки та допоміжні ресурси. Будуть розроблені такі модулі:

- модуль аутентифікації та управління користувачами;
- модуль роботи з постами;
- модуль роботи з темами;
- модуль роботи з секціями;
- модуль роботи з профілем.

Модуль аутентифікації та управління користувачами забезпечуватиме обробку авторизації, реєстрації, відновлення пароля та підтвердження пошти.

Цей модуль складатиметься зі сторінок авторизації, реєстрації, відновлення паролю та підтвердження пошти. Дані, що наводитиме користувач будуть використані у хуках, що дозволять взаємодіяти з серверним API.

Модуль роботи з постами відповідатиме за перегляд, створення та редагування постів. Він міститиме сторінки перегляду поста та створення нового поста.

Модуль роботи з темами забезпечуватиме відображення та управління темами. Включатиме сторінку списку тем, з якою можливо взаємодіяти завдяки використання хуків і вбудованим можливостями Next.js.

Модуль роботи з секціями відповідатиме за перегляд і управління секціями. Він містить сторінку секції та допоміжні компоненти, що забезпечать набір функціональних можливостей для роботи з секціями.

Модуль роботи з профілем відповідатиме за відображення, редагування та видалення профіля користувача. Він включає сторінку профіля з можливістю редагування особистих даних, а також компоненти для відображення інформації про користувача та форми оновлення профілю, забезпечуючи взаємодію з сервером через відповідні API-запити.

Зі сторони сервера буде реалізовано обробку запитів завдяки використанню NextJS.

Серверна частина складається з таких модулів:

- модуль роботи з аутентифікацією;
- модуль роботи з профілем;
- модуль роботи із сповіщеннями;
- модуль роботи з постами;
- модуль роботи з коментарями;
- модуль роботи з темами;
- модуль роботи з секціями.

Модуль аутентифікації відповідатиме за вхід, реєстрацію, авторизацію через сторонні сервіси, вихід, отримання нових токенів, підтвердження пошти, відновлення та скидання пароля. Він оброблятиме запити, перевірятиме облікові

дані, генеруватиме JWT-токени та інтегруватиметься з модулями для надсилання листів підтвердження та управління токенами в HttpOnly cookies, забезпечуючи безпечну взаємодію.

Модуль роботи з профілем відповідатиме за управління профілями користувачів, включаючи оновлення даних та перегляд інформації. Користувачі зможуть змінювати особисті дані, такі як ім’я чи аватар.

Модуль роботи із сповіщеннями відповідатиме за взаємодію зі сповіщеннями, наприклад, повідомлення про лайки чи нові коментарі, забезпечуючи їх доставку користувачам.

Модуль роботи з постами керуватиме створенням, редагуванням і видаленням постів, обробляючи відповідні запити від клієнта.

Модуль роботи з повідомленнями відповідатиме за роботу з коментарями, включаючи їх створення, редагування та видалення.

Модуль роботи з темами забезпечуватиме управління темами, підтримуючи їх створення, редагування та вкладену структуру. Користувачі зможуть створювати теми в межах певних секції з підтримкою створення підтем для зручної навігації.

Модуль роботи з секціями працюватиме з логікою роботи секцій, дозволяючи створювати та керувати категоріями форуму. Секції групують теми за інтересами, полегшуючи пошук матеріалів.

Важливо згадати, що Next.js має свої власні модулі, що оптимізують та покращують досвід роботи з проектом для розробника. Серед найбільш корисних можна виокремити такі: middleware та модуль логіки функціонування Next.js.

middleware у контексті Next.js – це проміжний шар, який виконується перед обробкою запиту сторінки чи API-ендпоїнта, дозволяючи реалізувати логіку перевірки авторизації, перенаправлення, обмеження доступу, локалізації тощо без необхідності змінювати код самих сторінок.

Модуль логіки функціонування Next.js – це підхід, з використанням App Router, що дозволяє реалізувати сучасну архітектуру застосунку через файлову систему, який підтримує серверні та клієнтські компоненти, а також дає змогу

використовувати різні стратегії рендерингу (SSR, SSG, ISR, CSR) залежно від потреб системи.

Структурна схема функціонування програмного модуля онлайн форуму наведена на рисунку 2.1.

Важливо згадати про те, що при створенні малих застосунків об’єднання клієнтської і серверної логіки в одному місці спрощує розробку, але для проектів, які будуть масштабуватися, і для великих систем розділення цих частин є необхідною умовою їх ефективності та гнучкості.

У малих проектах з малою командою розробників, монолітний підхід дозволяє швидко досягнути поставлених задач: від рендерингу сторінок до обробки форм і збереження даних у базі даних. Усе розміщене у одному місці, що заощаджує час на налаштування інфраструктури, розгортання чи координацію між різними частинами системи. Команда може швидко запустити продукт, а підтримка такої системи залишається простою через її компактність.

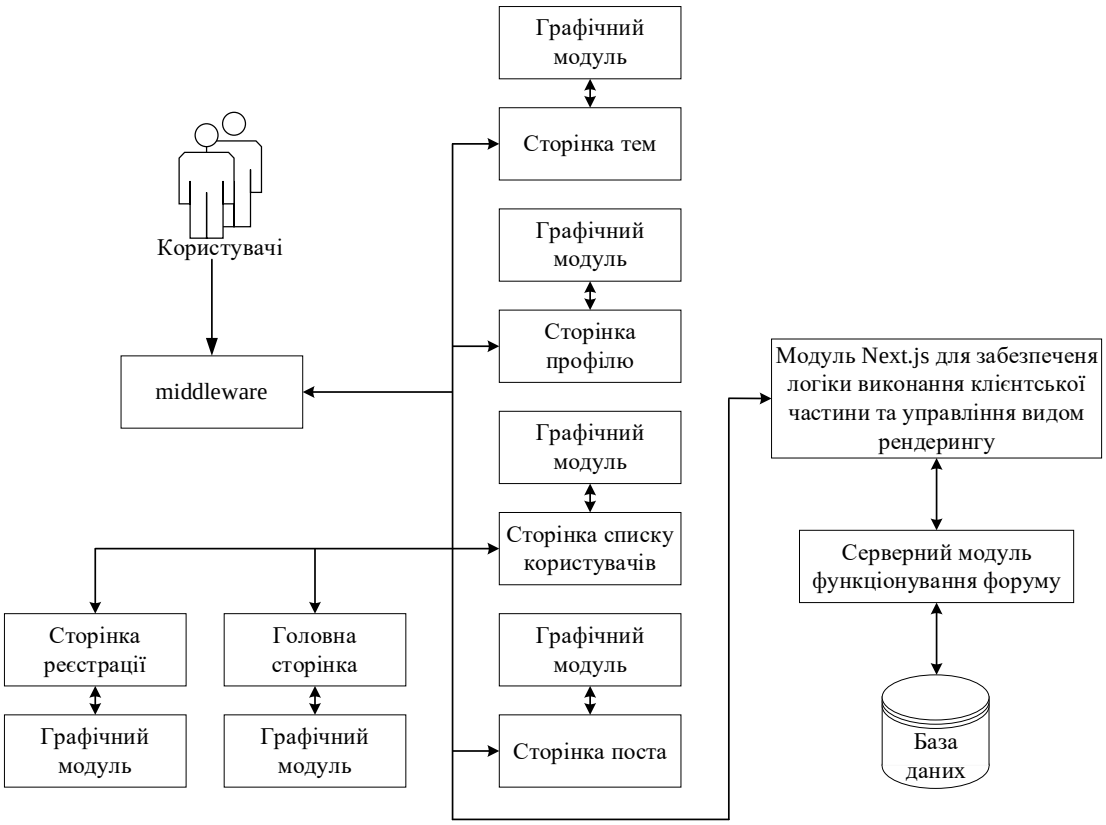

Рисунок 2.1 − Загальна структурна схема функціонування системи

Якщо навіть невеликий проєкт планується масштабувати, з одного боку, розділення клієнтської та серверної логіки створює міцну основу для зростання, об’єднана логіка ускладнює додавання нових функцій, тестування та підвищує ризик помилок, з іншого боку, розділення дає змогу чітко розмежувати відповідальність уже на початковому етапі. Тому для реалізації WEB-ресурсу «Онлайн форум» було обрано саме таку архітектуру. Клієнт відповідає за швидкість, навігацію та інтерактивність, наприклад, відображення постів і форм коментарів. Сервер забезпечує керування користувачами, збереження даних і модерацію. Такий підхід полегшує додавання нових можливостей і забезпечує гнучкість для подальшого розвитку.

Загальна структурна схема компонентів програмного модуля онлайн форуму зображена на рисунку 2.2.

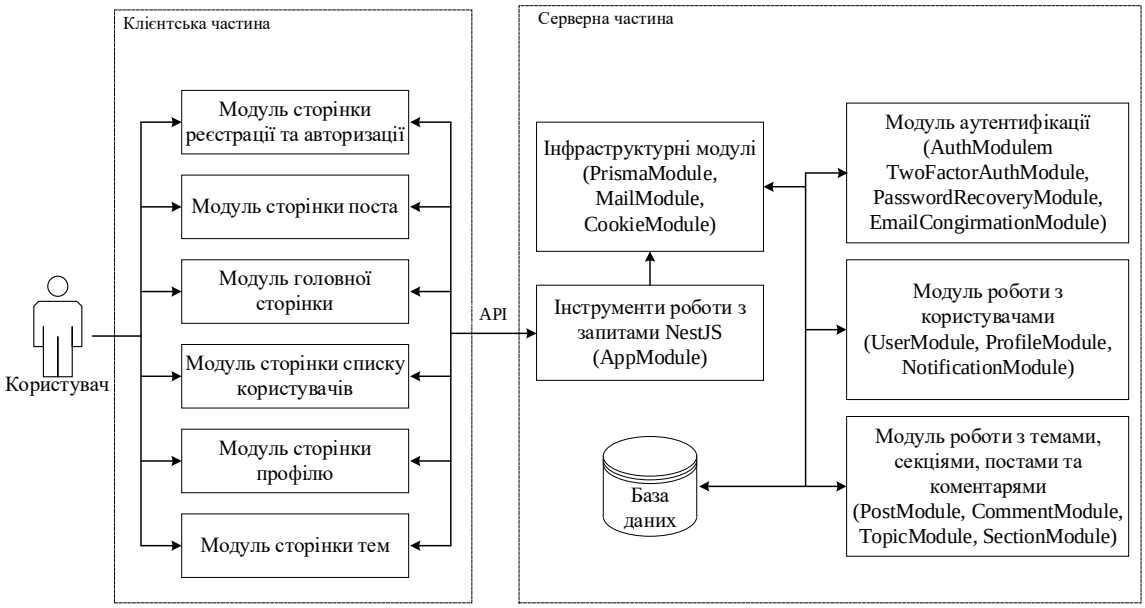

Рисунок 2.2 –Загальна структурна схема компонентів програмного модуля «Онлайн форум»

2.2 Розробка UML-діаграми класів

Для ефективного проектування архітектури системи та зручного представлення її логіки взаємодії застосовується UML – уніфікована мова

моделювання. Ця мова дозволяє створювати наочні діаграми, що описують структуру, поведінку та взаємозв'язки програмних компонентів. UML використовується на етапах проектування, аналізу та розробки, дозволяючи описувати компоненти, не звертаючи увагу на код. Завдяки цьому засобу можна легко описати складні технічні рішення, узгодити взаємодію між модулями тощо [13].

Розробки WEB-ресурсу «Онлайн форум» буде базуватись на комбінації об'єктно орієнтованого та функціонального підходів для написання програм, що дозволить створювати гнучкі, масштабовані та ефективні програмні рішення.

Об'єктний підхід використовуватиметься для серверної частини застосунку, зокрема з фреймворком NestJS, що забезпечує структуровану організацію коду та спрощує його підтримку.

Функціональний підхід буде використано для реалізації клієнтської частини з використанням фреймворку Next.js, що сприяє створенню реактивних і модульних інтерфейсів користувача, оптимізуючи продуктивність і спрощуючи розробку клієнтської логіки.

Об'єктно-орієнтоване програмування на серверній частині ґрунтується на основних принципах ООП, які забезпечують модульність, повторне використання коду та спрощують масштабування системи. До ключових принципів, що застосовуються в розробці серверної логіки, належать [14, 15]:

- абстракція – це принцип, що дозволяє звернути увагу на найважливіших характеристиках об'єкта, приховуючи другорядні деталі. Під цим розуміється створення простих і зрозумілих моделей.
- наслідування – це механізм, що дає змогу створювати нові класи, які успадковують характеристики й методи вже існуючих класів. Це дозволяє уникнути дублювання коду, підвищити гнучкість та розширюваність.
- інкапсуляція – це підхід, що полягає в об'єднанні даних і методів, які з ним працюють у єдиний фрагмент, реалізація якого приховується від зовнішнього доступу, що підвищує безпеку, стабільність тощо.

- поліморфізм – це властивість об’єктів з однаковими інтерфейсами, тобто це дозволяє використовувати методи по різному залежно від конкретного класу.

Функціональний підхід у клієнтській частині реалізовано за допомогою Next.js, що базується на принципах функціонального програмування. Це забезпечує передбачуваність інтерфейсу, спрощує тестування й підтримку. Поєднання таких рішень дозволяє створити надійний та ефективний WEB-ресурс, який придатний до масштабування та відповідає сучасним вимогам продуктивності [16].

У серверній частині WEB-ресурсу «Онлайн форум» потрібно реалізувати логіку управління користувачами, контентом, сповіщеннями та файлами. Ця частина складатиметься з ключових класів і сервісів, які забезпечать аутентифікацію, створення, редагування та видалення постів, коментарів, тем і секцій, а також нададуть змогу взаємодії користувачів через лайки і сповіщеня. Доцільно розглянути ключеві класи:

AuthController керуватиме авторизацією, реєстрацією, токенами та профілями користувачів, забезпечуючи безпечний доступ. Клас оброблятиме введені email і пароль, створюватиме нові облікові записи з підтвердженням пошти, завершуватиме сеанси, оновлюватиме токени, підтримуватиме авторизацію через сторонні сервіси, надсилатиме листи підтвердження дій, а також дозволятиме встановити новий пароль після перевірки.

AuthService забезпечуватиме логіку для авторизації та управління профілями. Оброблятиме авторизацію, реєстрацію, дані від сторонніх сервісів, оновлення токенів, надсилання листів для підтвердження пошти та формування об’єктів для токенів.

AuthController керуватиме авторизацією, реєстрацією, токенами та профілями користувачів, забезпечуючи безпечний доступ. Клас оброблятиме введені email і пароль, створюватиме нові облікові записи з підтвердженням пошти, завершуватиме сеанси, оновлюватиме токени, підтримуватиме авторизацію через сторонні сервіси, надсилатиме листи для підтвердження дій, а також дозволятиме встановлювати новий пароль після перевірки.

‘

AuthService забезпечуватиме логіку для авторизації та управління профілями. Оброблятиме авторизацію, реєстрацію, дані від сторонніх сервісів, оновлення токенів, надсилання листів для підтвердження пошти та формування об’єктів для токенів.

EmailConfirmationService відповідатиме за підтвердження пошти. Надсилатиме листи з токенами для верифікації, зміни пошти, підтверджуватиме існування пошти та оновлюватиме її в базі, а також генеруватиме токени певного типу.

PasswordRecoveryService оброблятиме відновлення пароля. Надсилатиме листи з токенами та інструкціями для скидання пароля і дозволятиме встановлювати новий пароль після перевірки.

TwoFactorAuthService займатиметься двофакторною авторизацією. Надсилатиме токени, перевірятиме введені коди та генеруватиме унікальні токени.

JwtTokenService керуватиме JWT-токенами, створюючи та перевіряючи їх валідність.

PasswordService шифруватиме паролі та порівнюватиме їх із зашифрованими версіями.

CookieService управлятиме cookies, додаючи та видаляючи токени, а також налаштовуючи їхні параметри, наприклад, термін дії та безпека.

UserController керуватиме профілями користувачів. Дозволятиме отримувати списки всіх користувачів, деталі профілю, оновлювати профіль, підтверджувати зміну пошти та видаляти облікові записи.

UserService управлятиме даними користувачів. Створюватиме нових користувачів, оновлюватиме їхні дані, паролі, пошту, отримуватиме інформацію за ідентифікатором або поштою, підтверджуватиме верифікацію пошти та видалятиме облікові записи.

SectionController відповідатиме за організацію секцій. Дозволятиме створювати, видаляти, отримувати список і оновлювати секції.

SectionService реалізовуватиме логіку управління секціями, включаючи їх створення, оновлення, видалення та отримання списку.

TopicController оброблятиме запити, пов’язані з темами. Дозволятиме отримувати, створювати, оновлювати та видаляти теми.

TopicService керуватиме логікою тем, забезпечуючи їх створення, оновлення, видалення та отримання за ідентифікатором.

PostController оброблятиме запити, пов’язані з публікаціями. Дозволятиме отримувати пости за темою, створювати, оновлювати та видаляти пости.

PostService реалізовуватиме логіку роботи з публікаціями. Дозволятиме виконувати дії з постами, збільшувати кількість переглядів, оновлювати кількість коментарів та останній коментар, а також готувати дані для сповіщень і коментарів.

TopicHierarchyService керуватиме ієрархією тем. Отримуватиме вкладені теми та їхніх нащадків, оновлюватиме статистичні лічильники та останні коментарі в гілці.

CommentController керуватиме коментарями. Дозволятиме отримувати коментарі до поста або користувача з пагінацією, виконувати дії з коментарями.

CommentService оброблятиме логіку коментарів. Забезпечуватиме їх створення, оновлення, видалення, отримання з пагінацією та підготовку даних для сповіщень.

LikeController керуватиме вподобаннями, дозволяючи перемикати їхній стан для коментарів.

LikeService реалізовуватиме логіку вподобань, дозволяючи ставити, видаляти та перемикати їх.

NotificationController керуватиме сповіщеннями. Дозволятиме отримувати сповіщення користувача (включно з непрочитаними), змінювати статус прочитання та видаляти сповіщення.

NotificationService оброблятиме логіку сповіщень. Створюватиме та видалятиме сповіщення про вподобання та нові коментарі, отримуватиме списки сповіщень, оновлюватиме статус прочитання та видалятиме сповіщення.

‘

UploadsController відповідатиме за завантаження та отримання файлів.

UploadsService оброблятиме та зберігатиме файли, отримуватиме їх із файлової системи та генеруватиме аватари з першої літери.

TokenService керуватиме токенами, створюючи, шукаючи та видаляючи їх за типом або поштою.

MailService надсилатиме листи для підтвердження реєстрації, зміни пошти, скидання пароля, двофакторної авторизації та інші листи з заданою темою і вмістом.

ProfileService керуватиме статистикою профілю, оновлюючи кількість коментарів для користувача, постів, тем або секцій.

AccountService управлятиме обліковими записами сторонніх провайдерів, створюючи їх, оновлюючи токени доступу та шукаючи за ідентифікатором і типом провайдера.

На рисунках 2.3 – 2.5 зображені UML-діаграми класів модулів серверної частини WEB-ресурсу «Онлайн форум».

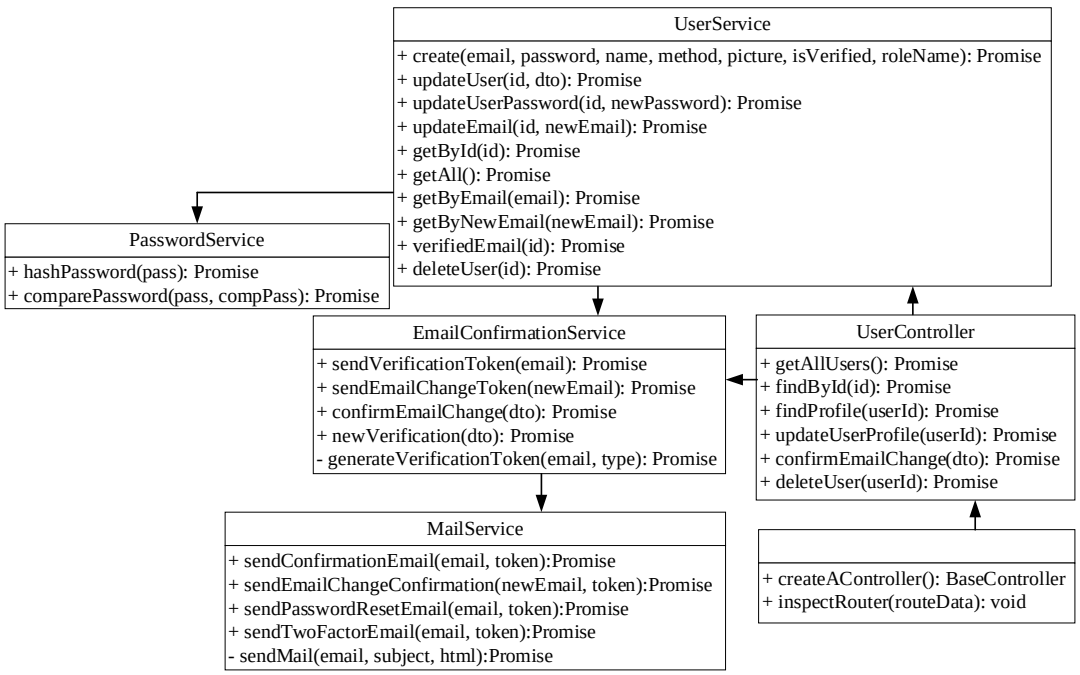

Рисунок 2.3 – UML-діаграма класів модуля роботи з профілем серверної частини

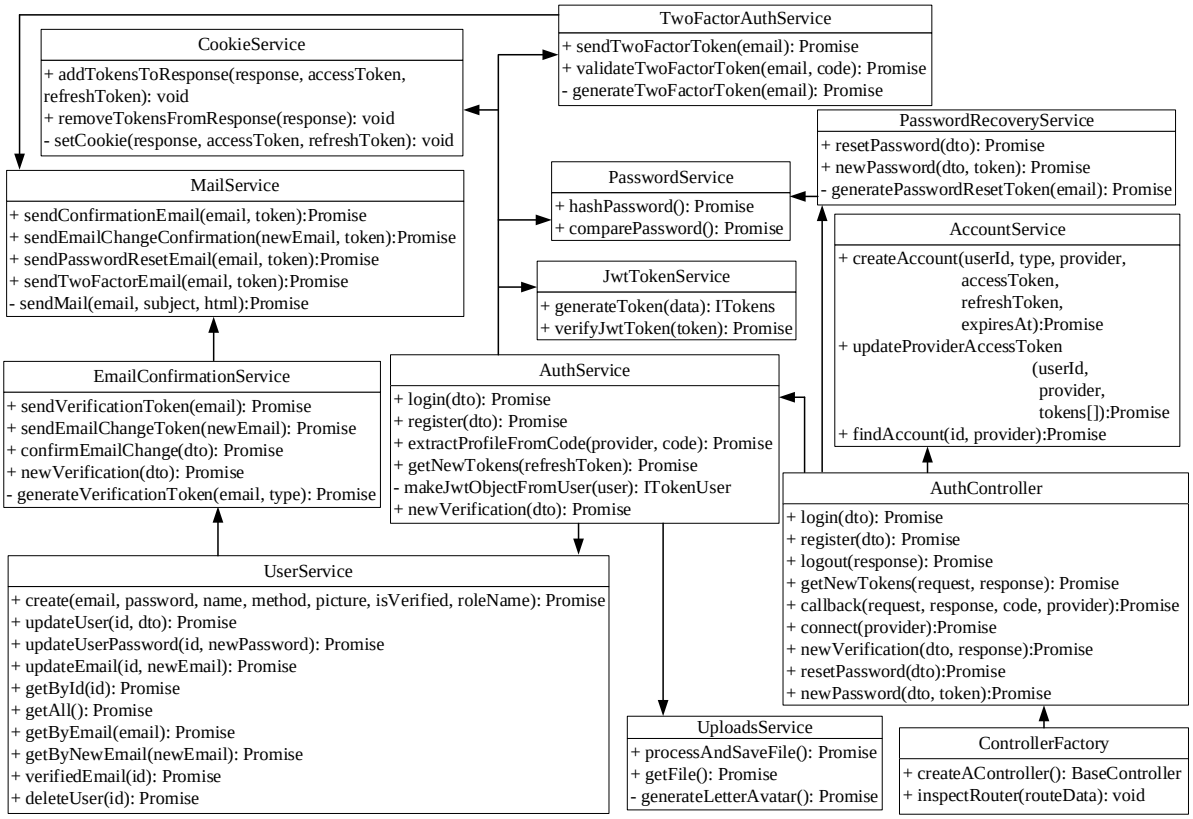

Рисунок 2.4 – UML-діаграма класів модуля роботи з аутентифікацією серверної частини

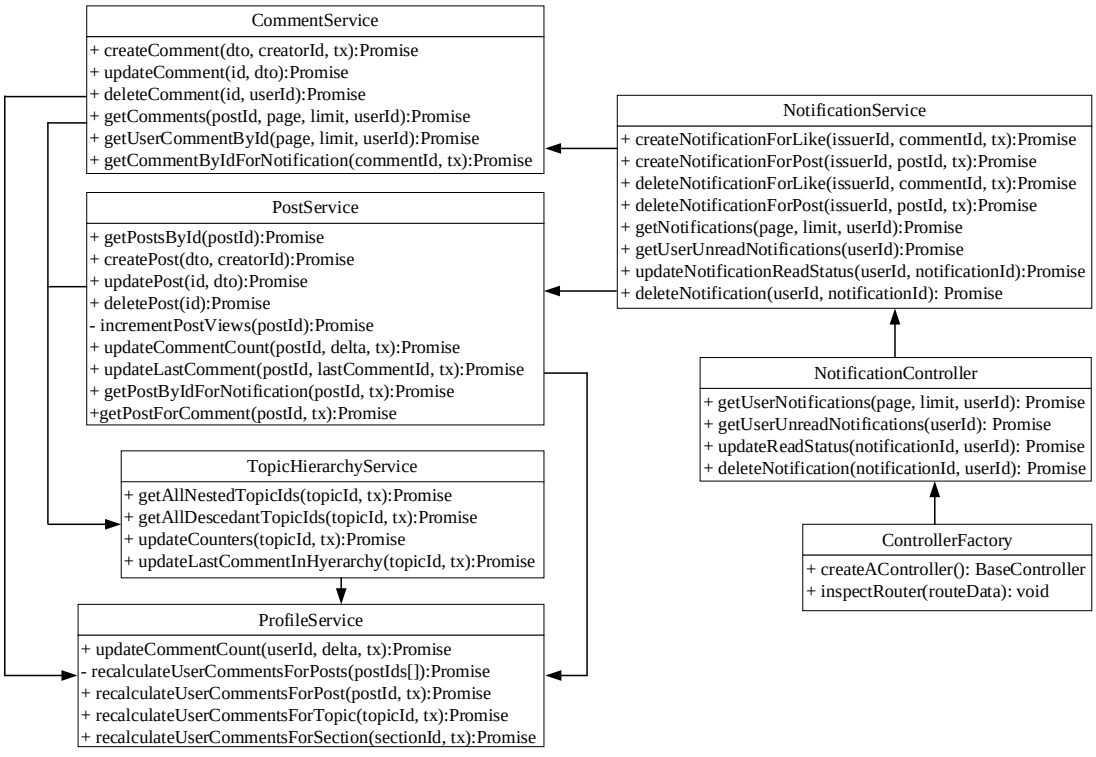

Рисунок 2.5 – UML-діаграма класів модуля роботи із сповіщеннями серверної частини

Отже, застосування уніфікованої мови моделювання дозволило чітко описати структуру основних компонентів серверної частини та їх взаємозв’язки. Було визначено ключові класи, що відповідатимуть за основний функціонал ресурсу. Створені UML-діаграми класів є важливим етапом проектування, який забезпечує визначеність та зручність реалізації функціональностей ресурсу.

2.3 Розробка ER-моделі бази даних

ER-підхід – це спосіб планування бази даних. Він допомагає зрозуміти, які дані будуть зберігатись і як вони пов’язані між собою. У цьому підході використовують спеціальні схеми, де об’єкти називають сутностями а зв’язки між ними – відношеннями. Такий підхід полегшує створення правильної та зручної бази даних [17].

Доцільно навести перелік сутностей, що описують предметну область «Онлайн форум»: аккаунт, коментар, лайк, сповіщення, пост, профіль, права, роль, секція, ключі доступу, тема, користувач.

За правилами побудови схем предметної області у вигляді ER-структур сутності зображують позначеними прямокутниками, асоціації – ромбами, а зв’язки між ними – ненаправленими ребрами, над якими може проставлятися ступінь зв’язку і необхідне пояснення [18].

Зв’язок ОДИН-ДО-ОДНОГО (1:1) використовується між такими сутностями:

1. Користувач – Акаунт, оскільки кожному користувачеві відповідає лише один акаунт. Тобто, для кожного користувача створюється один акаунт, і цей акаунт не може бути прив’язаний до кількох користувачів одночасно. У свою чергу, кожен акаунт ідентифікує конкретного користувача.

2. Користувач – Профіль, оскільки кожному користувачеві відповідає один профіль. Профіль містить інформацію про користувача, і жоден профіль не може бути прив’язаний до кількох користувачів одночавно.

Зв’язок ОДИН-ДО-БАГАТЬОХ (1:Б) або БАГАТО-ДО-ОДНОГО (Б:1) використовується між такими сутностями:

1. Користувач – Роль, оскільки один користувач може мати одну роль, але роль може належати кільком користувачам.

2. Користувач – Лайк, оскільки один користувач може поставити багато лайків, але кожен лайк належить одному користувачеві.

3. Користувач – Секція, оскільки користувач може створити багато секцій, але секція не обов’язково сворюється кожним користувачем.

4. Користувач – Тема, оскільки будь-який користувач може створити багато декілька тем, але кожна тема належить тільки одному користувачеві.

5. Користувач – Пост, оскільки користувач може публікувати багато потів, але кожен пост може бути створений лише одним користувачем.

6. Користуач – Коментар, оскільки користувач може залишати багато коментарів, але коментар належить тільки одному користувачеві.

7. Секція – Тема, оскільки секція може міст багато тем, але кожна тема може належати лише одній секції

8. Тема – Пост, оскільки одна тема може містити багато постів, але кожен пост належить тільки одній темі.

9. Пост – Коментар, оскільки один пост може мати багато коментарів, але кожен коментар належить лише одному посту.

10. Сповіщення – Користувач, оскільки один користувач може отримати багато сповіщень або створити багато сповіщень, але кожне сповіщення має лише одного отримувача та ініціатора.

11. Сповіщення – Пост, оскільки один пост може бути згаданий у багатьох сповіщеннях, але кожне сповіщення стосується лише одного поста.

12. Сповіщення – Коментар, оскільки один коментар може бути згаданий у багатьох сповіщеннях, але кожне сповіщення стосується одного коментаря.

Зв’язок БАГАТО-ДО-БАГАТЬОХ (Б:Б) використовується лише між такими сутностями:

‘

1. Роль – Право, оскільки роль може мати багато прав, а одне право може належати кільком ролям.

Сутність «Токени» взагалі не має зв’язку з будь-якими сутностями, оскільки вона є службовою і використовується виключно для підтвердежння дій.

При описі наявних сутностей для створення ER-моделі з’ясовано, що наявні усі типи зв’язків. Перед поданням ER-моделі доцільно навести таблицю характеристик зв’язків предметної області «Онлайн форум» (табл. 2.1).

Таблиця 2.1 – Характеристики відношень між сутностями предметної області «Онлайн форум»

№	Ім’я першої сутності	Тип зв’язку	Ім’я другої сутності	Ім’я зв’язку	Клас належності
1	Користувач	1:1	Акаунт	Належить	Обов’язковий
2	Користувач	1:1	Профіль	Належить	Обов’язковий
3	Користувач	Б:1	Роль	Належить	Обов’язковий
4	Користувач	1:Б	Лайк	Належить	Можливий
5	Користувач	1:Б	Секція	Належить	Можливий
6	Користувач	1:Б	Тема	Належить	Можливий
7	Користувач	1:Б	Пост	Належить	Можливий
8	Користувач	1:Б	Коментар	Належить	Можливий
9	Секція	1:Б	Тема	Належить	Можливий
10	Тема	1:Б	Пост	Належить	Можливий
11	Пост	1:Б	Коментар	Належить	Можливий
12	Сповіщення	Б:1	Користувач	Належить	Можливий
13	Сповіщення	Б:1	Пост	Належить	Можливий
14	Сповіщення	1:Б	Коментар	Належить	Можливий
15	Роль	Б:Б	Право	Має	Можливий

Представлення ER-моделі для бази даних «Онлайн форум» зображено на рисунку 2.6.

‘

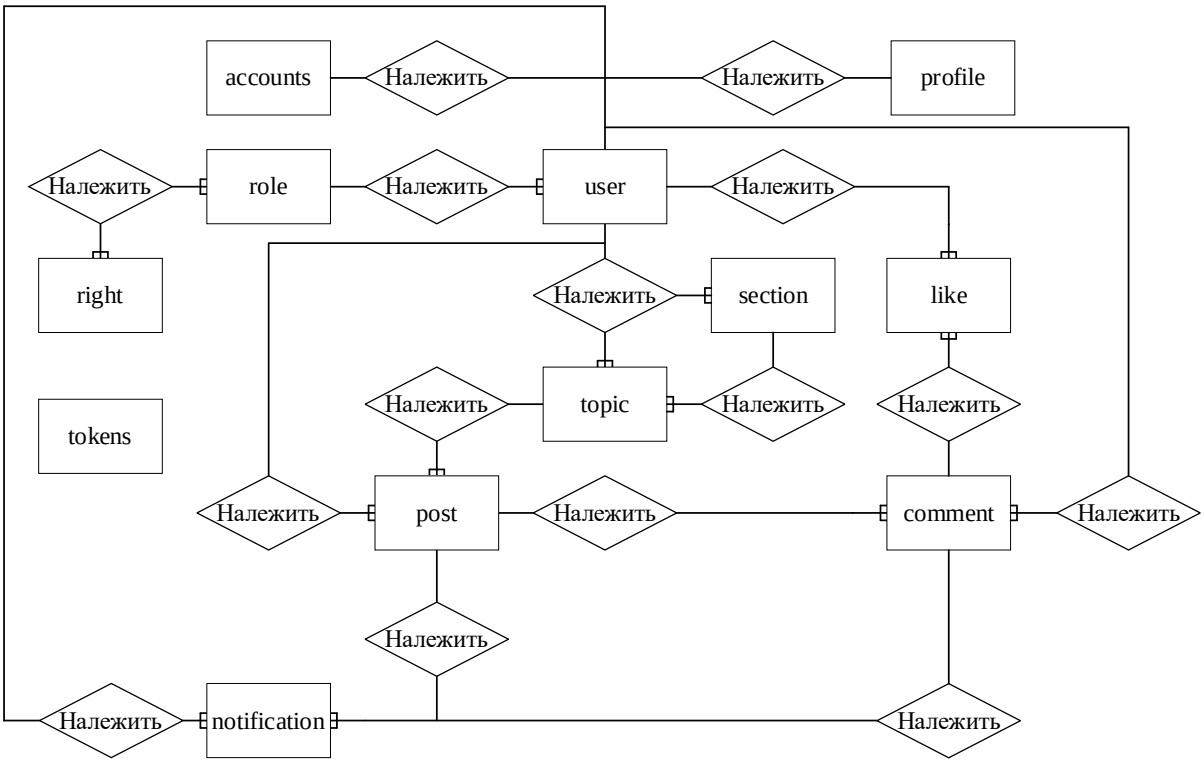

Рисунок 2.6 – ER-модель предметної області «Онлайн форум»

Наступним кроком потрібно побудувати ER-діаграму предметної області. Будь-якому екземпляру сутності «Користувач» відповідає лише один екземпляр сутності «Роль», тому між ними має бути встановлений обв’язковий тип зв’язку. Екземпляру сутності «Роль» може відповідати декілька екземплярів сутності «Право», а одне «Право» може бути пов’язане з кількома екземплярами сутності «Роль», тому їх зв’язок має бути необов’язковим. Представлення такого фрагменту ER-діаграми для бази даних зображено на рисунку 2.7.

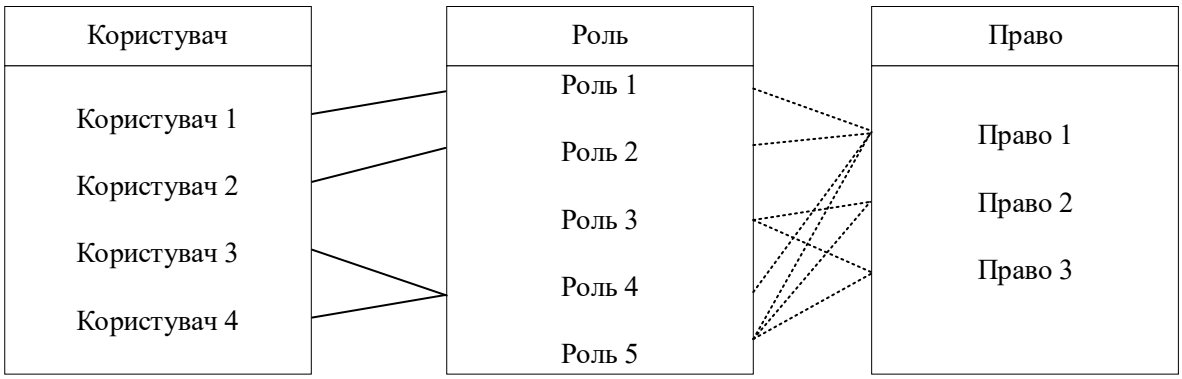

Рисунок 2.7 – Фрагмент ER-діаграми, що демонструє зв’язки між екземплярами сутностей предметної області «Онлайн форум»

2.4 Розробка схеми алгоритму роботи WEB-ресурсу «Онлайн форум»

Розробка алгоритму є ключовим етапом створення програмної системи, оскільки визначає послідовність дій для досягнення мети. Він описує обробку вхідних даних, прийняття рішень, переходи між станами та формування результатів. Для візуалізації роботи модуля WEB-ресурсу «Онлайн форум» використано блок-схему, що дозволяє чітко показати основні етапи, логіку та розгалуження, полегшуючи реалізацію програмного коду [19].

Схему алгоритму функціонування WEB-ресурсу «Онлайн форум» зображено на рисунку 2.8.

І випадок

Алгоритм реєстрації користувача складається з таких кроків (рис. 2.8):

Крок 1. Якщо користувач натиснув кнопку «Register», то перейти до кроку 2, інакше – до кроку 14.

Крок 2. Користувач вводить своє ім’я у відповідне поле форми реєстрації.

Крок 3. Користувач вводить свою електронну пошту у відповідне поле форми реєстрації.

Крок 4. Користувач вводить пароль у відповідне поле форми реєстрації.

Крок 5. Користувач натискає кнопку «Register».

Крок 6. Якщо ім’я або електронна пошта вже існують у базі даних, то перейти до кроку 7, інакше – до кроку 8.

Крок 7. Сервер повертає повідомлення про помилку, і користувач повертається до кроку 2.

Крок 8. Сервер створює новий запис користувача в базі даних.

Крок 9. Сервер надсилає лист із токеном підтвердження на електронну пошту користувача.

Крок 10. Якщо користувач перейшов за посиланням у листі, то перейти до кроку 11, інакше перейти до кроку 1.

Крок 11. Якщо токен валідний, то перейти до кроку 12, інакше перейти до кроку 13.

Крок 12. Сервер оновлює статус користувача і перейти до кроку 14.

Крок 13. Сервер повертає повідомлення про помилку реєстрації і перейти до кроку 9.

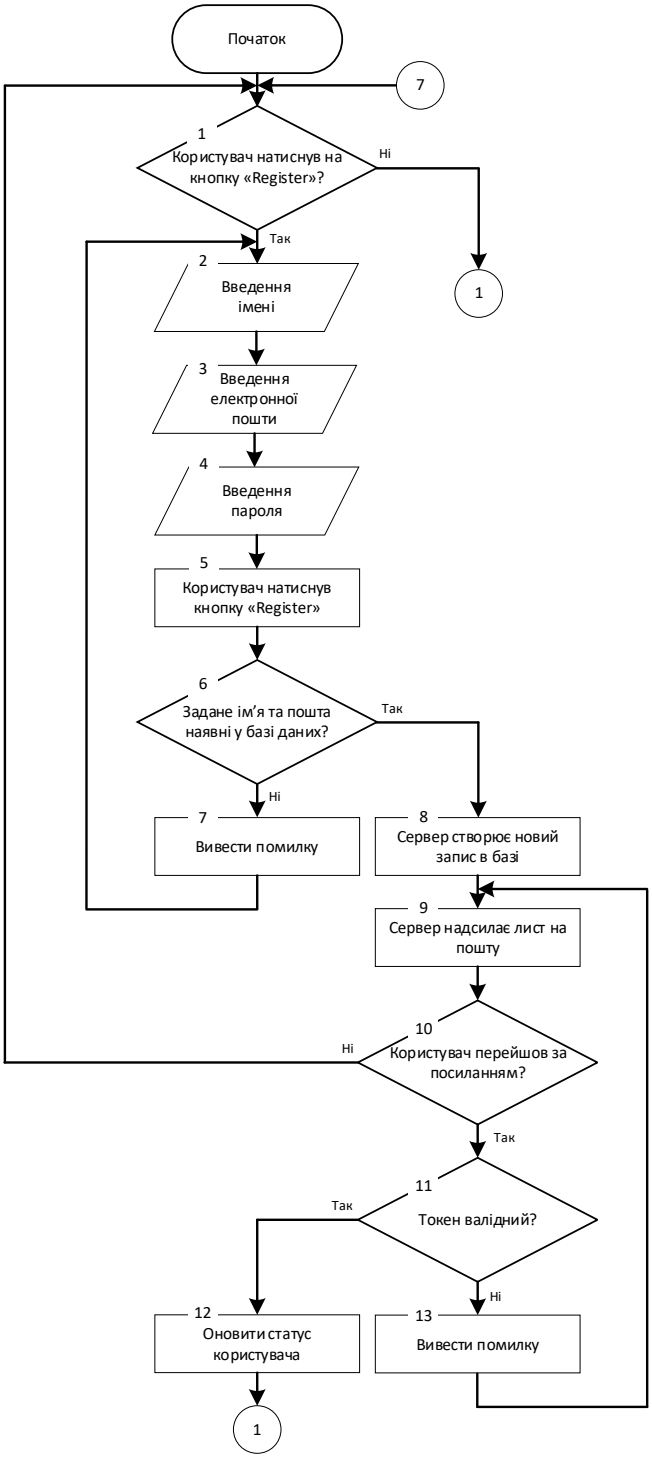

Рисунок 2.8 – Схема алгоритму роботи програмного модуля

‘

II випадок

Алгоритм аутентифікації користувача складається з таких кроків (рис. 2.8, аркуш 2):

Крок 14. Якщо користувач натиснув кнопку «Login», то перейти до кроку 15, інакше переходить до кроку 26.

Крок 15. Користувач вводить свою електронну пошту у відповідне поле форми аутентифікації.

Крок 16. Користувач вводить свій пароль у відповідне поле форми аутентифікації.

Крок 17. Якщо у користувача увімкнена двофакторна аутентифікація, то перейти до кроку 18, інакше перейти до кроку 22.

Крок 18. Сервер надсилає код на електронну пошту.

Крок 19. Користувач вводить отриманий код у відповідне поле.

Крок 20. Якщо код валідний, то перейти до кроку 22, інакше перейти до кроку 21.

Крок 21. Сервер повертає повідомлення про помилку, і перейти до кроку 18.

Крок 22. Якщо електронна пошта та пароль відповідають запису в базі даних, то перейти до кроку 23, інакше перейти до кроку 24.

Крок 23. Сервер повертає повідомлення про помилку авторизації і перейти до кроку 15.

Крок 24. Сервер генерує пару токенів доступу.

Крок 25. Сервер додає токени до http only cookies.

Крок 57. Користувач натискає на кнопку «Delete».

Крок 58. Користувачу показує новий список коментарів.

Крок 59. Якщо користувач продовжує роботу з форумом, то перейти до кроку 1.

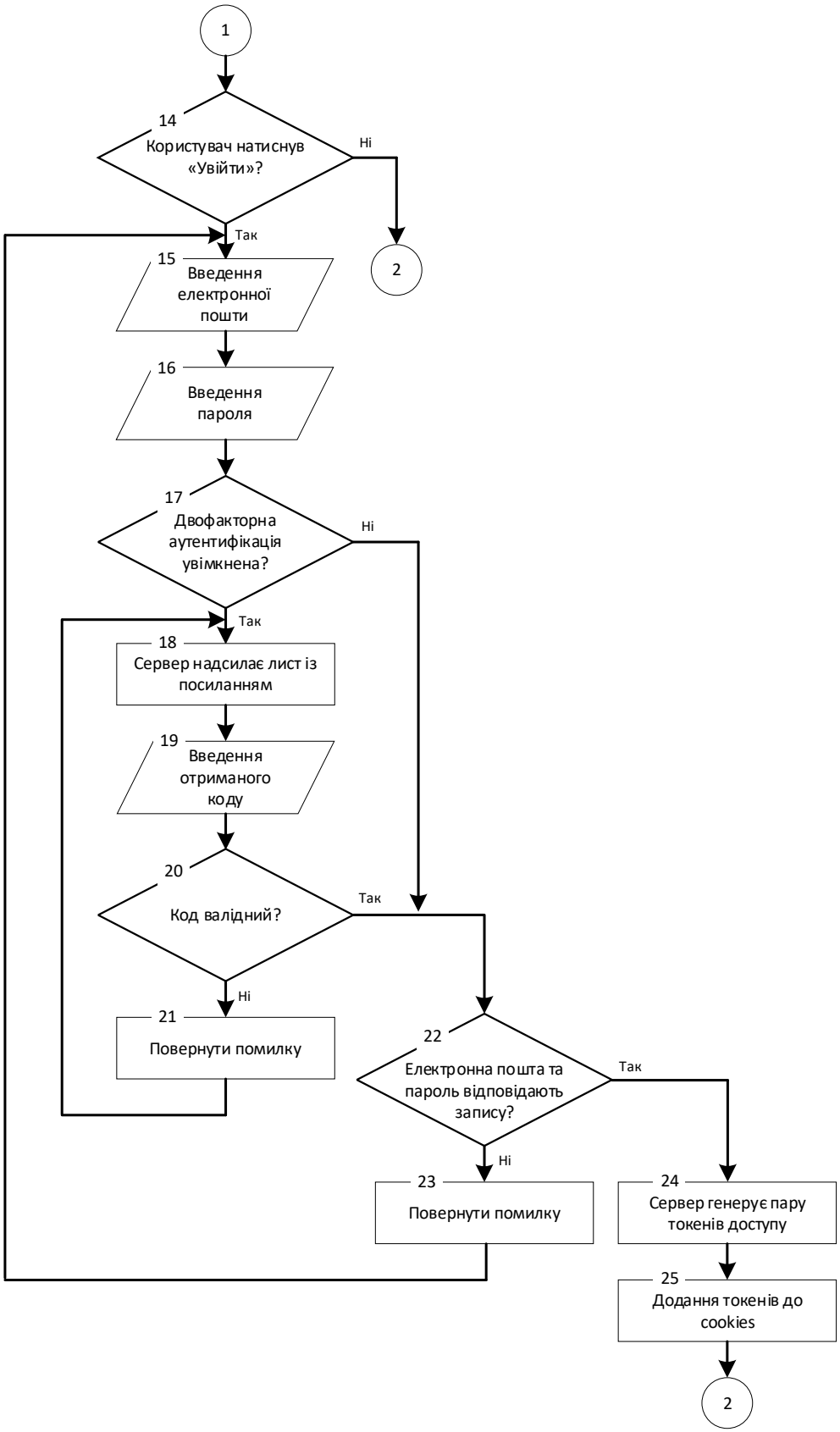

Рисунок 2.8, аркуш 2

III випадок

Алгоритм створення та управління темою складається з таких кроків (рис. 2.8, аркуш 3):

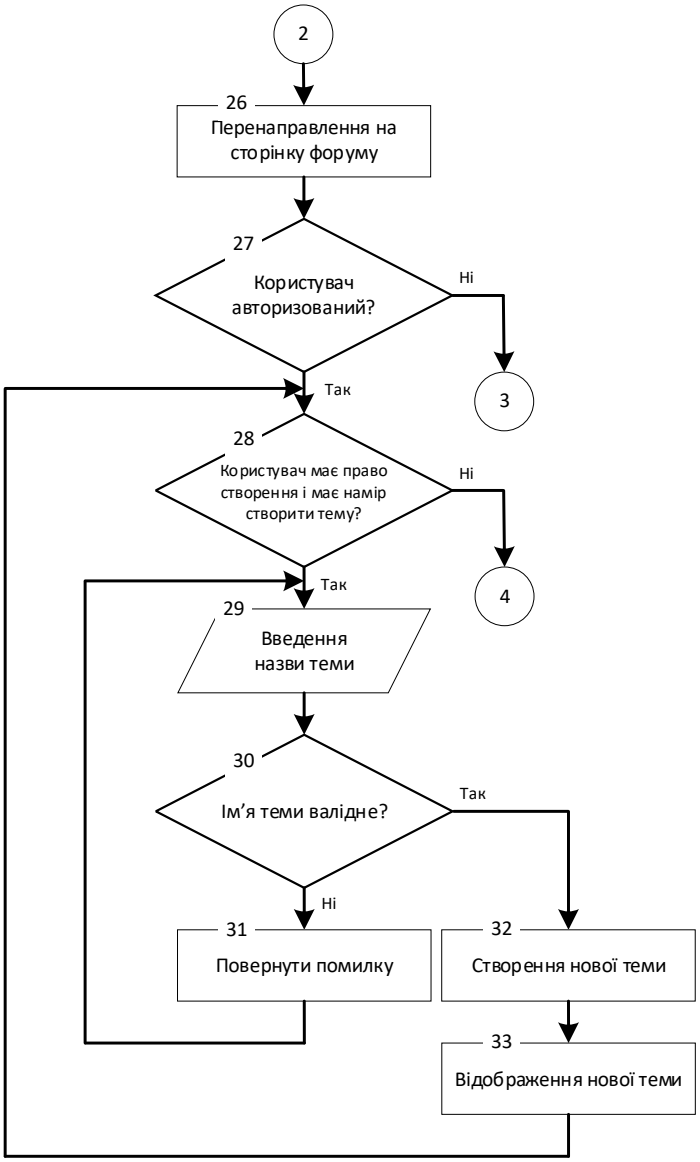

Рисунок 2.8, аркуш 3

Крок 26. Сервер перенаправляє користувача на сторінку форуму.

Крок 27. Якщо користувач авторизований, то перейти до кроку 28, інакше перейти до кроку 43.

Крок 28. Якщо користувач має право створення і бажає створити тему, то перейти до кроку 29, інакше перейти до кроку 34.

Крок 29. Користувач вводить назву теми.

Крок 30. Якщо ім’я теми валідне, то перейти до кроку 32, інакше – до кроку 31.

Крок 31. Сервер повертає повідомлення про помилку створення і перейти до кроку 29.

Крок 32. Сервер створює нову тему.

Крок 33. Користувачу відображує нову тему.

Крок 34. Якщо користувач має право редагування і бажає оновити тему, то перейти до кроку 35, інакше перейти до кроку 40 (рис. 2.8, аркуш 4).

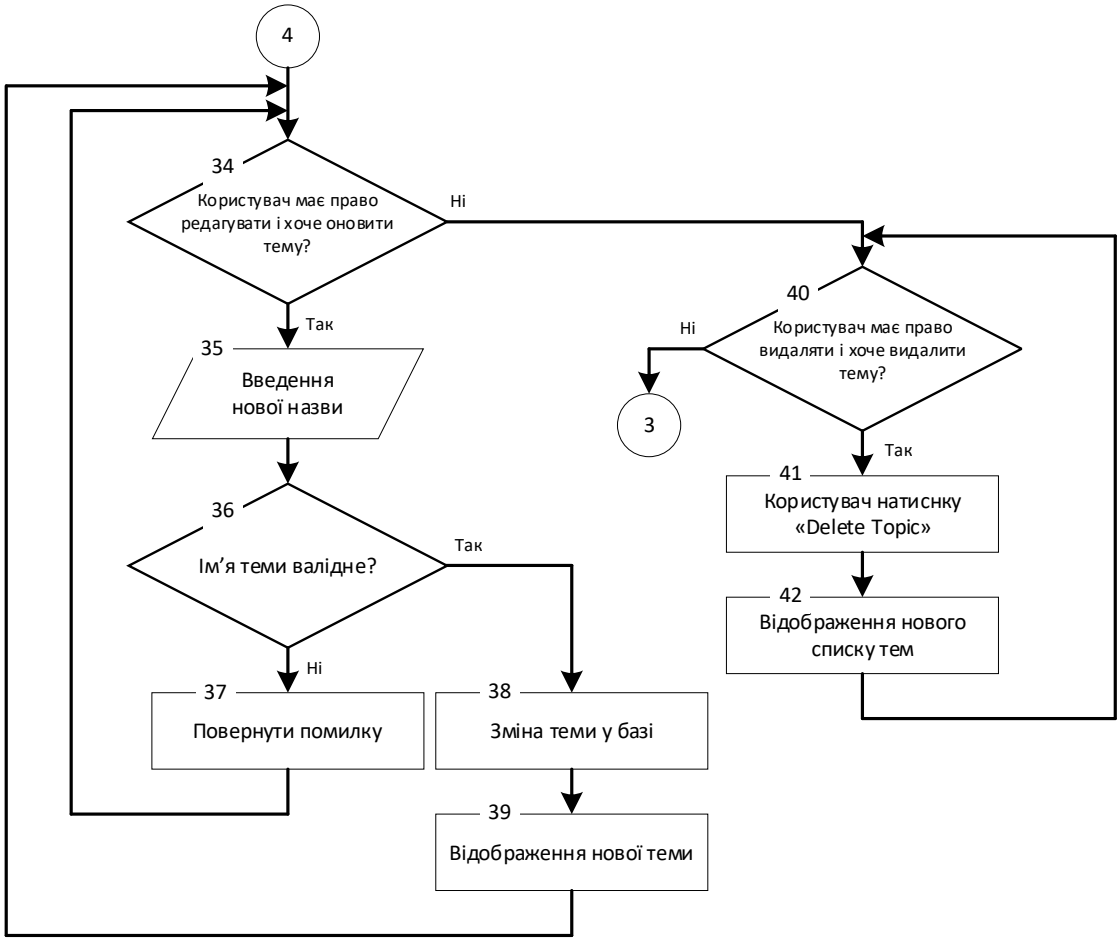

Рисунок 2.8, аркуш 4

Крок 35. Користувач вводить нову назву теми.

Крок 36. Якщо ім’я теми валідне, то перейти до кроку 38, інакше – до кроку 37.

‘

Крок 37. Сервер повертає повідомлення про помилку оновлення і перейти до кроку 34.

Крок 38. Сервер оновлює запис у базі даних.

Крок 39. Користувачу відображує оновлену тему.

Крок 40. Якщо користувач має право видаляти і бажає видалити тему, то перейти до кроку 41, інакше перейти до кроку 43.

Крок 41. Користувач натискає на кнопку «Delete Topic».

Крок 42. Користувачу показує новий список тем.

IV випадок

Алгоритм створення та управління коментарем складається з таких кроків (рис. 2.8, аркуш 5):

Крок 43. Якщо користувач авторизований, то перейти до кроку 44 інакше перейти до кроку 59.

Крок 44. Якщо користувач бажає створити коментар, то перейти до кроку 45, інакше перейти до кроку 50.

Крок 45. Користувач вводить новий коментар.

Крок 46. Якщо введені дані коректні, то перейти до кроку 48, інакше – до року 47.

Крок 47. Сервер повертає повідомлення про помилку створення і перейти до кроку 45.

Крок 48. Сервер створює новий запис у базі даних.

Крок 49. Користувачу відображається новий коментар.

Крок 50. Якщо користувач має право оновлення і бажає оновити коментар, то перейти до кроку 51, інакше перейти до кроку 56 (рис. 2.8, аркуш 6).

Крок 51. Користувач оновлює вміст коментаря.

Крок 52. Якщо введені дані коректні, то перейти до кроку 53, інакше – до кроку 54.

Крок 53. Сервер повертає повідомлення про помилку оновлення і перейти до кроку 50.

Крок 54. Сервер оновлює запис у базі даних.

Крок 55. Користувачу відображається оновлений коментар.

Крок 56. Якщо користувач має право видаляти і бажає видалити коментар, то перейти до кроку 57, інакше перейти до кроку 59.

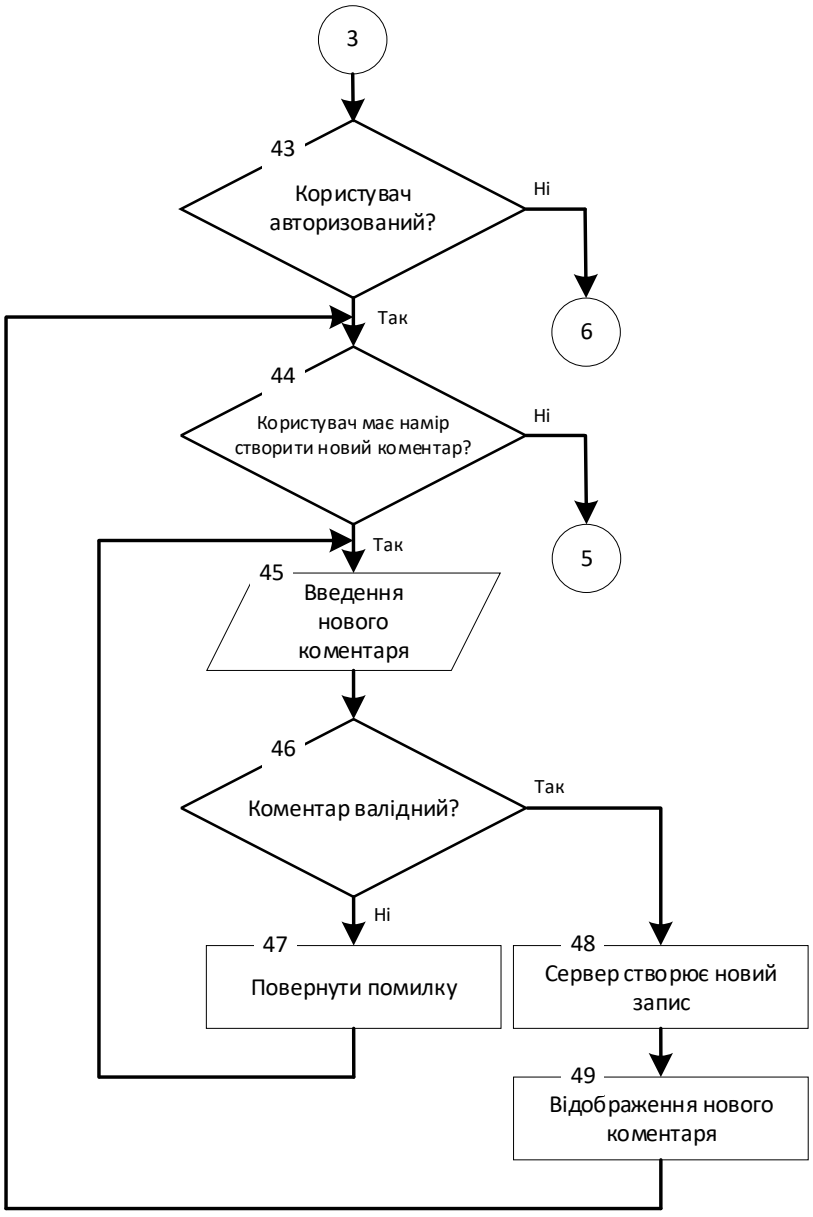

Рисунок 2.8, аркуш 5

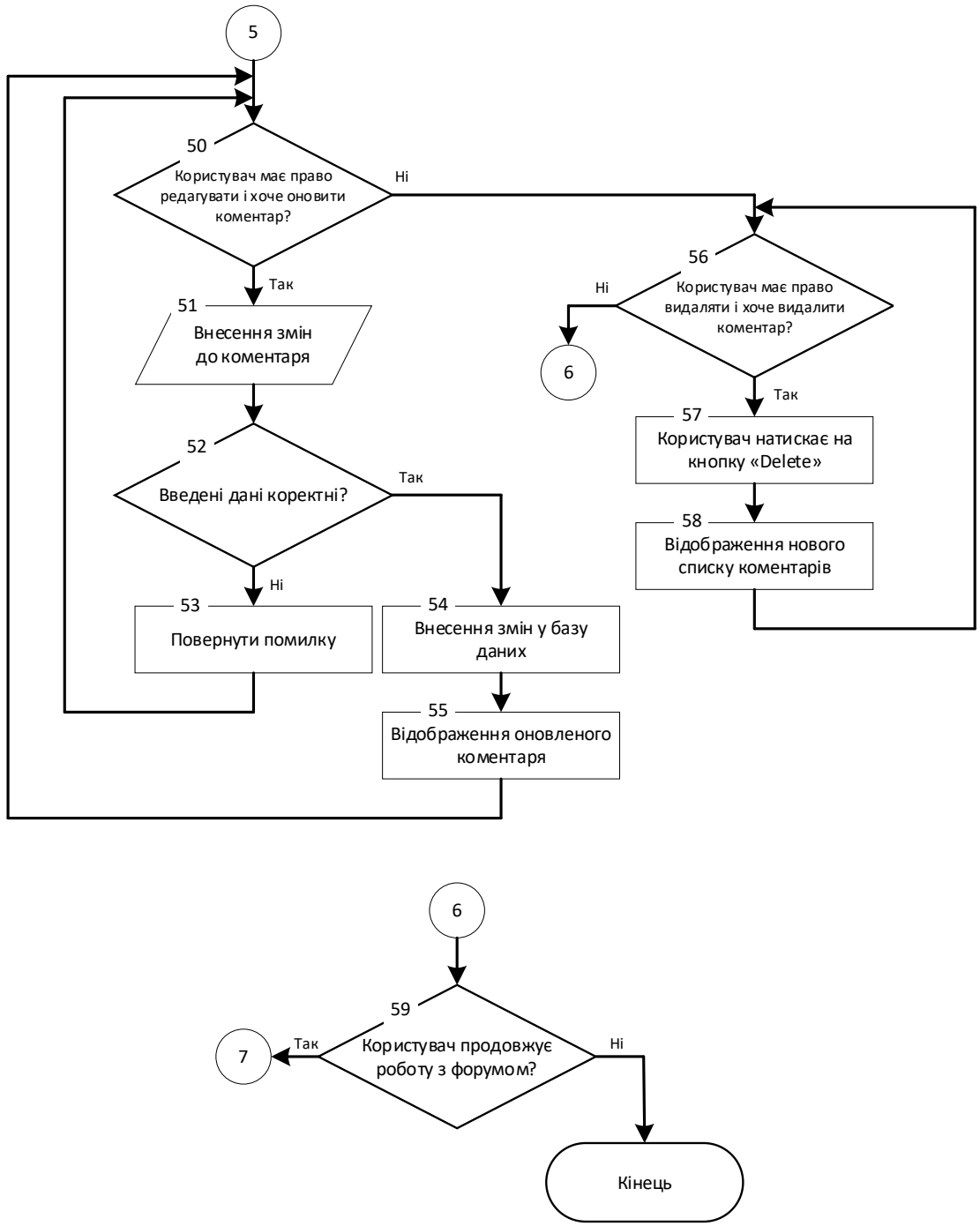

Рисунок 2.8, аркуш 6

Отже, у результаті розробки схеми алгоритму функціонування WEB-ресурсу «Онлайн форум» було детально описано логіку базових процесів, які забезпечують працездатність ресурсу. Побудовані схеми алгоритмів відображають ключові етапи взаємодії користувача з форумом, перевірку введених даних, обробку запитів на сервері та обробку ситуацій.

2.5 Висновок

У цьому розділі було розроблено структуру програмного модуля WEB-ресурсу «Онлайн форум», що забезпечує чітку організацію розробки шляхом визначення ключових компонентів системи та їхнього взаємозв’язку. Було встановлено, що доцільним є використання клієнт-серверної архітектури.

Клієнтська частина включає модулі для аутентифікації, управління профілями, постами, темами та секціями.

Серверна частина обробляє логіку, наприклад, взаємодію з базою даних та керування сповіщеннями. Такий поділ сприяє модульності, полегшує масштабування та допомагає підтримувати ресурс.

Для детального опису серверної частини створено UML-діаграми основних класів, які візуалізують структуру модулів, їхні методи та взаємозв’язки. Такі діаграми є інструкціями, які полегшують розуміння системи для розробників та сприяють ефективній реалізації.

Розроблено ER-модель бази даних, яка включає усі сутності та визначає їх тип зв’язку, що полегшує проектування бази даних і розуміння взаємозв’язків між сутностями.

Крім того, розроблено схеми алгоритмів для ключових процесів, таких як реєстрація, аутентифікація, створення та управління темами і коментарями. Ці схеми алгоритмів чітко описують послідовність дій, що зменшує ризик помилок, оптимізує продуктивність та сприяє зрозумілому плануванню розробки.

Запропонована структура та підходи дозволять створити надійну основу для розробки WEB-ресурсу «Онлайн форум», який відповідає сучасним стандартам продуктивності, безпеки та зручності. Така структура дозволить легко додавати нові функції у майбутньому.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-РЕСУРСУ «ОНЛАЙН ФОРУМ»

3.1 Обґрунтування вибору мови та бібліотек програмування

Програмна реалізація WEB-ресурсу «Онлайн форум» передбачає створення клієнтської та серверної частин. У цьому розділі буде проведено аналіз і порівняння різних мов програмування, а також обґрунтовано вибір найбільш ефективних технологій окремо для клієнтського та серверного застосунків. При виборі мов розробки особливу увагу буде приділено таким критеріям, як швидкодія, масштабованість, безпека та зручність подальшої підтримки системи.

Для клієнтського частини ресурсу порівняємо мови програмування TypeScript та JavaScript.

JavaScript є основною мовою для WEB-розробки, яка підтримується будь-якими сучасними браузерами та широко використовується для створення інтерактивних WEB-сторінок чи додатків. Проте зі зростанням складності WEB-додатків з’явилася потреба в інструментах, які полегшують розробку та підтримку великих кодових баз. TypeScript, розроблений Microsoft, є вдосконаленням JavaScript, що додає статичне типування, інтерфейси та модулі, роблячи його особливо придатним для масштабних проєктів [20, 21].

У таблиці 3.1 наведено основні відмінності між мовами програмування TypeScript та JavaScript.

Таблиця 3.1 – Порівняння мов програмування TypeScript та JavaScript

Критерій порівняння	TypeScript	JavaScript
Статична типізація	+	-
Підтримка об’єктно-орієнтованого програмування	+	+
Популярність	+	+
Динамічне виявлення помилок	+	-
Ручне управління пам’яттю	-	-

‘

Продовження таблиці 3.1

Критерій порівняння	TypeScript	JavaScript
Інтеграція з сучасними IDE	+	±
Спрощений синтаксис	±	+
Масштабованість	+	±
Складність використання	±	+
Безпека коду	+	-
Документація коду	+	±
Підтримка сучасних бібліотек	+	+

Відповідно до результатів таблиці 3.1, TypeScript забезпечує статичну типізацію, виявляючи помилки на етапі компіляції, в той час як JavaScript, будучи динамічно типізованою мовою, підвищує ризик помилок під час виконання. Обидві мови підтримують об'єктно-орієнтоване програмування з класами, спадкуванням та інкапсуляцією. JavaScript має простіший синтаксис і ширшу базу користувачів, що полегшує освоєння новачками. TypeScript, попри потребу вивчення типів, покращує структуру коду, рефакторинг, автодоповнення та навігацію, зменшуючи помилки завдяки перевірці типів. Обидві мови мають автоматичний збір сміття. TypeScript підтримує самодокументування через типи та інтерфейси, тоді як JavaScript потребує додаткових інструментів [20, 21].

Отже, TypeScript є оптимальним вибором для клієнтського застосунку завдяки типізації, яка знижує кількість помилок у розробці та експлуатації. Підтримка сучасних фреймворків і високі можливості рефакторингу сприяють продуктивності розробників. TypeScript полегшує командну роботу завдяки чіткій структуризації коду.

Хоча JavaScript простіший у вивченні, але для створення складного та надійного WEB-застосунку вибір TypeScript є обґрунтованим. Його використання підвищує якість, безпеку та ефективність розробки клієнтської частини онлайн форуму.

Для розробки клієнтської частини «Онлайн форуму» розглянемо React та Next.js.

React – це бібліотека для побудови інтерфейсів користувача, розроблена Facebook, яка дозволяє створювати компоненти та керувати станом ресурсу.

Next.js – це фреймворк, побудований на основі React, який пропонує розширені функції для розробки повноцінних WEB-додатків, включаючи серверний рендеринг, маршрутизацію та оптимізацію продуктивності.

У таблиці 3.2 наведено основні відмінності між бібліотекою React та фреймворком Next.js 15.

Таблиця 3.2 – Порівняння мов програмування TypeScript та JavaScript

Критерій порівняння	React	Next.js
Серверний рендеринг (SSR)	-	+
Статична генерація (SSG)	-	+
Вбудована маршрутизація	-	+
Гнучкість налаштування	+	±
Легка інтеграція	+	+
Оптимізація продуктивності	-	+
Розмір початкового коду	Низький	Вищий
Підтримка розгортання на сервері	-	+

Відповідно до результатів таблиці 3.2, Next.js забезпечує розширену функціональність для сучасних WEB-додатків, включаючи серверний рендеринг, статичну генерацію сторінок та автоматичну оптимізацію ресурсів. Він має вбудовану маршрутизацію на основі файлової системи, що значно спрощує розробку масштабованих додатків без потреби у додаткових бібліотеках, наприклад, таких як React Router.

React надає вищий рівень гнучкості та дозволяє розробникам самостійно обирати архітектуру проєкту, однак для реалізації повноцінного WEB-додатка на

React часто доводиться додатково інтегрувати сторонні бібліотеки для маршрутизації, SSR та оптимізації, що збільшує складність налаштувань.

Next.js, хоч і вимагає дещо глибшого розуміння концепцій SSR, SSG та підходів до рендерингу сторінок, але значно полегшує розробку, оптимізацію та розгортання WEB-додатків.

Таким чином, для розробки клієнтської частини «Онлайн форуму» було обрано Next.js, оскільки він забезпечує кращу швидкість завантаження сторінок, розширені можливості оптимізації, підтримку серверного рендерингу, а також гарну інтеграцію з TypeScript. Це рішення сприятиме підвищенню продуктивності, SEO-оптимізації та покращенню досвіду користувачів.

Для розробки серверної частини «Онлайн форуму» розглянемо TypeScript, Python та Go. Основними критеріями вибору стали: продуктивність, масштабованість, легкість розробки та підтримки, безпека та екосистема.

У таблиці 3.3 наведено основні відмінності між цими мовами програмування.

Таблиця 3.3 – Порівняння мов програмування TypeScript, Python та Go

Критерій порівняння	TypeScript	Python	Go
Швидкодія	+	-	+
Масштабованість	+	±	+
Паралелізм	±	-	+
Легкість розробки	+	+	±
Безпека	+	±	+
Екосистема	+	+	±
Різноманіття бібліотек	+	+	±
Інтеграція з базами даних	+	+	±

Згідно з результатами таблиці 3.3 видно, що кожна з мов має свої сильні сторони.

Python вирізняється простотою синтаксису та великою кількістю бібліотек, що робить його популярним вибором для швидкої розробки прототипів та

наукових застосунків. Проте його продуктивність є нижчою порівняно з іншими мовами, що може стати проблемою для високонавантажених систем.

Go, розроблений компанією Google, забезпечує високу продуктивність та чудову підтримку паралельних обчислень через горутини. Однак розробка на Go вимагає більшого часу для опанування концепцій мови та має менш розвинену екосистему для WEB-розробки порівняно з TypeScript [22].

TypeScript у поєднанні з Node.js надає високий рівень продуктивності завдяки асинхронній обробці подій, що ідеально підходить для обробки великої кількості одночасних запитів. Статична типізація TypeScript підвищує безпеку коду, зменшує кількість помилок та спрощує подальший рефакторинг і підтримку проєкту.

Враховуючи наведені переваги, для серверної частини «Онлайн форуму» було обрано TypeScript, оскільки він забезпечує баланс між високою продуктивністю, масштабованістю, безпекою та швидкістю розробки. Крім того, використання однієї мови як для клієнтської, так і для серверної частини значно спрощує розробку, командну роботу та підтримку.

Для розробки серверної частини «Онлайн форуму» розглянемо NestJS та Express.

Express є мінімалістичним WEB-фреймворком для Node.js, який забезпечує базову функціональність для створення серверних застосунків.

NestJS, у свою чергу, побудований на основі Express і використовує архітектурні принципи, запозичені з корпоративних рішень, таких як Angular, зокрема інверсію керування, модульність та декоратори.

У таблиці 3.4 наведено основні відмінності між фреймворками NestJS та Express [23, 24].

Таблиця 3.4 – Порівняння фреймворків NestJS та Express

Критерій порівняння	NestJS	Express
Рівень абстракції	Високий	Низький
Структура проєкту	Диктується фреймворком	Вільна

‘

Продовження таблиці 3.4

Критерій порівняння	NestJS	Express
Підтримка TypeScript	+	±
Легкість використання	±	+
Гнучкість налаштувань	±	+
Швидкість розробки масштабних додатків	+	±
Вбудована підтримка валідації, авторизації, обробки помилок	+	-

Відповідно до результатів таблиці 3.4, Express є простим і гнучким рішенням для побудови серверних застосунків. Його головною перевагою є легкість та мінімалізм, що дозволяє самостійно обирати структуру проєкту та бібліотеки для розширення функціональності. Однак, із зростанням складності застосунку в Express може з'являтися проблема погано структурованого коду, що ускладнює підтримку великих систем.

NestJS, навпаки, пропонує чітку структуру проєкту за допомогою модулів, сервісів та контролерів. Він реалізує підтримку об'єктно-орієнтованого програмування і патернів розробки, таких як інверсія керування та залежностей, що суттєво полегшує розробку великих, масштабованих і підтримуваних застосунків. Крім того, NestJS має нативну підтримку TypeScript та широкий спектр офіційних модулів, наприклад, для роботи з мікросервісами, WebSockets, GraphQL та іншими технологіями.

Таким чином, для серверної частини «Онлайн форуму» було обрано NestJS, оскільки він забезпечує модульну структуру, високу масштабованість та зручність супроводу великих проєктів. Його використання сприяє швидшій розробці та кращій підтримуваності програмного продукту в довгостроковій перспективі.

3.2 Обґрунтування вибору мови програмування для управління організованою базою даних

Для забезпечення ефективного управління організованою базою даних «Онлайн форуму» важливим є вибір мови запитів, яка дозволяє зручно виконувати операції створення, читання, оновлення та видалення даних. Основними варіантами для розгляду є Structured Query Language – традиційна мова для роботи з реляційними базами даних та Object Query Language – мова запитів, орієнтована на об'єктно-орієнтовані бази даних.

У таблиці 3.5 наведено порівняння основних характеристик SQL та OQL [25, 26].

Таблиця 3.5 – Порівняння мов запитів SQL та OQL

Критерій порівняння	SQL	OQL
Поширеність	+	-
Природність для ООП	-	+
Наявність готових інструментів і бібліотек	+	-
Складність використання	±	±
Підтримка транзакцій	+	-
Оптимізація запитів	+	±
Зручність інтеграції з ORM-інструментами	+	+
Необхідність нормалізації даних	+	-
Відношення між даними	Зовнішні ключі	Посилання між об’єктами

Відповідно до результатів таблиці 3.5, SQL залишається основним стандартом для взаємодії з реляційними базами даних завдяки своїй широкій підтримці, розвиненій екосистемі інструментів і гнучким можливостям оптимізації запитів. Однак традиційний SQL менш природно інтегрується із об'єктно-орієнтованими мовами програмування.

OQL, з одного боку, краще відповідає парадигмі об'єктно-орієнтованого програмування, дозволяючи працювати безпосередньо з об'єктами замість таблиць, але з іншого боку, його поширення є обмеженим, а підтримка серед популярних систем керування базами даних є мінімальною.

Враховуючи переваги обох підходів, у проєкті було прийнято рішення використовувати Prisma ORM. Prisma абстрагує взаємодію з базою даних, дозволяючи працювати у стилі об'єктно-орієнтованого програмування за допомогою зручного API, що генерується автоматично на основі схеми бази даних. При цьому Prisma генерує SQL-запити у фоновому режимі, забезпечуючи високу продуктивність та підтримку транзакцій. Завдяки статичній типізації й інтеграції з TypeScript, Prisma підвищує безпеку та передбачуваність.

Таким чином, використання Prisma ORM на основі SQL забезпечує найкращий баланс між сумісністю, продуктивністю та зручністю розробки для серверної частини «Онлайн форуму».

3.3 Обґрунтування вибору середовища розробки

Для розробки клієнтської та серверної частин WEB-ресурсу «Онлайн форуму» розлянемо кілька популярних середовищ: Visual Studio Code, WebStorm та Sublime Text. Основними критеріями оцінки стали: підтримка TypeScript, інтеграція з системами керування версіями, підтримка розширень, швидкодія, зручність налагодження, а також наявність інструментів для роботи з базами даних та REST API [27–29].

В таблиці 3.6 подано порівняльна характеристика середовищ розробки. Відповідно до результатів наведених у таблиці 3.6, Visual Studio Code демонструє високу гнучкість, легкість розширення і гарну підтримку всіх необхідних технологій для проєкту.

WebStorm є потужним рішенням, але потребує платної ліцензії та складніший у налаштуванні для новачків.

Таблиця 3.6 – Порівняння середовищ розробки

Критерій прівняння	Visual Studio Code	WebShtorm	Sublime Text
Підтримка TypeScript	+	+	±
Інтеграція з Git	+	+	±
Наявність інструментів дебагінгу	+	+	-
Кількість розширень	+	±	±
Підтримка Node.js, NestJS, React, Next.js	+	+	-
Вартість	Безкоштовне	Платне	Частково
Зручність налаштування	+	±	+

Sublime Text має високу швидкодію, однак потребує багато сторонніх налаштувань для повноцінної роботи з сучасними WEB-технологіями та обмежену підтримку інструментів для складних проєктів.

Таким чином, вибір Visual Studio Code для середовища розробки, який забезпечує найкращий баланс між функціональністю, продуктивністю та зручністю для розробки WEB-ресурсу «Онлайн форум», є досить гарним рішенням.

3.4 Розробка клієнтської та серверної частин

Розробка WEB-ресурсу «Онлайн форум» базується на клієнт-серверній архітектурі, де клієнтська частина взаємодіє з серверною завдяки HTTP. Для створення такого ресурсу застосовано сучасні технології, які забезпечують високу продуктивність, розширюваність та зручність розробки.

Серверна логіка реалізована за допомогою фреймворку NestJS, який є досить потужним інструментом для створення серверних застосунків на основі Node.js та TypeScript. NestJS забезпечує модульну структуру, підтримує принцип інверсії керування та розділення відповідальностей, а також дозволяє легко

впроваджувати захист, авторизацію, валідацію даних та взаємодію з базами даних.

Для зберігання даних використовується реляційна база PostgreSQL, це забезпечує надійне збереження структурованої інформації. Доступ до бази реалізований через PrismaORM, що є інструментом об’єктно-реляційного відображення, це спрощує роботу з даними. Prisma використовує декларативний синтаксис для опису моделей, їхніх полів, типів і зв’язків у файлі схеми на основі якого генерується типізований клієнт для ефективних запитів. Це полегшує операції, підвищує безпеку коду завдяки статичній типізації, зменшує ризик помилок і підтримує зручне керування міграціями структури бази даних [30].

У рамках проєкту реалізовані такі ключові моделі:

Модель User відповідає за користувачів, які мають облікові записи, аватари, ролі та зв’язки з іншими сутностями. Post містить інформацію про дописи, зокрема заголовок, дату створення, кількість переглядів і коментарів. Comment реалізує коментарі до дописів з підтримкою оновлення, вподобань та збереження історії редагування. Section і Topic формують логічну ієрархічну структуру для організації контенту. Like реалізує систему вподобань коментарів користувачами. Модель Notification використовується для роботи з повідомленнями. Role і Right відповідають за ролі та права доступу, що дозволяє ефективно розмежовувати функціональні можливості користувачів. Token забезпечує обробку тимчасових токенів для підтвердження дій, а модель Account використовується для інтеграції з OAuth-провайдерами.

Серверна частина дотримується принципу розділення відповідальностей: кожна функціональність реалізована у окремому модулі, що включає контролери, сервіси та модулі.

Першим є модуль аутентифікації AuthModule, який відповідає за всі процеси аутентифікації, реєстрації, керування токенами, OAuth та відновлення паролю. Вхідною точкою в цей модуль є контролер AuthController, який обробляє відповідні HTTP-запити. Контролер визначає набір маршрутів для роботи з логікою, однак сам по собі контролер не містить логіки, а лише отримує

відповідні дані та викликає відповідні методи сервісів і повертає результат клієнту.

Для кращого розуміння окремих функіональних можливостей доцільно перед наведенням програмної реалізації розглянути діаграму активності методу login (рис. 3.1).

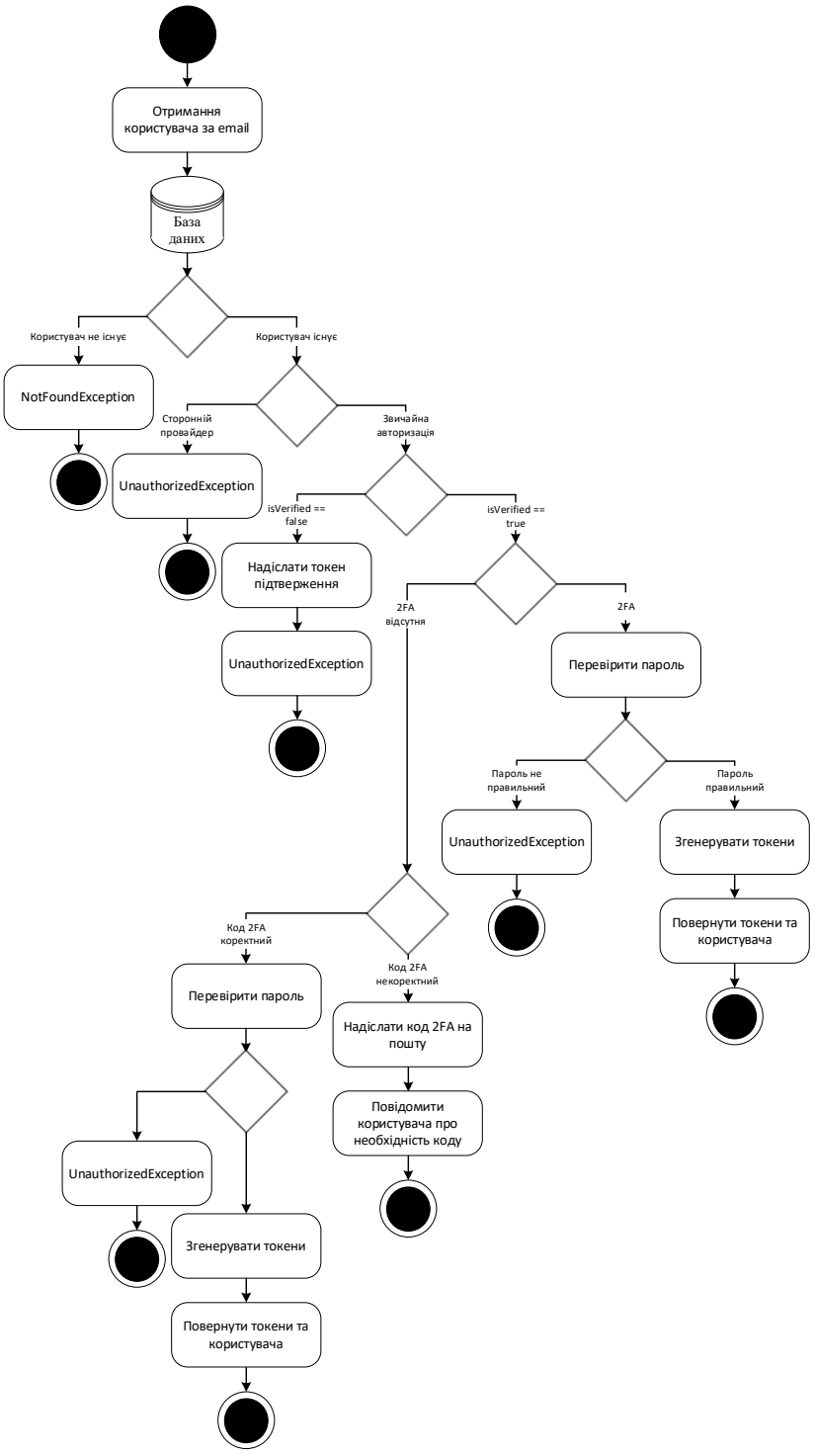

Рисунок 3.1 – Діаграма діяльності для методу login сервісу AuthService

‘

Така діаграма демонструє основні етапи аутентифікації користувача, включно з перевіркою наявності облікового запису, підтвердженням електронної пошти, обробкою двофакторної аутентифікації та генерацією токенів доступу. Це дозволяє навести загальну логіку без необхідності заглиблюватись у програмний код.

Нижче наведено фрагмент реалізації AuthService:

```
@Injectable()
export class AuthService {
  async login(dto: LoginDto) {
    const user = await this.userService.getByEmail(
      dto.email,
    );
    if (!user || !user.password)
      throw new NotFoundException(
        'User not found. Ensure the user exists and the password is properly set, then try again.',
      );
    if (user && user.method != 'CREDENTIALS') {
      throw new UnauthorizedException(
        'This email is registered via an external provider. Please log in using the respective login option.',
      );
    }
    if (!user.isVerified) {
      await this.emailConfirmationService.sendVerificationToken(
        user.email,
      );
      throw new UnauthorizedException(
        'Your email is not confirmed. Please check your mail and confirm the address.',
      );
    }
    if (user.isTwoFactorEnabled) {
      if (!dto.code) {
        await this.twoFactorAuthService.sendTwoFactorToken(
          user.email,
```

```
      );
      return {
        message:
          'Check your email. Two-factor authentication code required',
      };
    }
    await this.twoFactorAuthService.validateTwoFactorToken(
      user.email,
      dto.code,
    );
  }
```

Для забезпечення авторизації використовується декоратор Authorization, який застосовує RightsGuard для перевірки наявності в користувача відповідних прав доступу, що вказуються як його аргументи.

Реалізація декоратора має такий зміст:

```
@Injectable()
export class RightsGuard implements CanActivate {
  constructor(private readonly reflector: Reflector) {}
  async canActivate(context: ExecutionContext) {
    const requiredRights = this.reflector.getAllAndOverride<
      string[]
    >(RIGHTS_KEY, [
      context.getHandler(),
      context.getClass(),
    ]);
    if (!requiredRights) return true;
    const request = context.switchToHttp().getRequest();
    const realUserRights: TAuthUser = request.user;
    if (
      realUserRights &&
      !realUserRights.role.rights.some((right) =>
        requiredRights.includes(right),
      )
    ) {
```

‘

```
        throw new ForbiddenException(
          'You do not have the required permissions. Please contact your administrator if you believe this is an error.',
        );
      }
      return true;
    }
  }
```

Модуль CommentModule відповідає за роботу з коментарями користувачів. Він тісно пов’язаний з постами, профілями користувачів, ієрархією тем та сповіщень. Використовує сервіси інших модулів: PostService, ProfileService, TopicHierarchyService та NotificationService. CommentService надає наступний набір методів для роботи з коментарями:

- отримання списку коментарів для поста з підтримкою пагінації;
- отримання коментарів, створених певним користувачем;
- створення нового коментаря з оновленням пов’язаних лічильників і сповіщеннями;
- оновлення вмісту існуючого коментаря;
- видалення коментаря з автоматичним оновленням стану поста та структури теми;

Додатково, при отриманні коментарів враховується, чи лайкнув їх поточний користувач і додається інформація про загальну кількість лайків.

Фрагмент реалізації цього модуля має такий зміст:

```
@Injectable()
export class CommentService {
  async createComment(
    dto: CreateCommentDto,
    creatorId: string,
    tx?: Prisma.TransactionClient,
  ) {
    const executeCommentCreation = async (
      prismaTx: Prisma.TransactionClient,
```

‘

```
  ) => {
    const post = await this.postService.getPostForComment(
      dto.postId,
      prismaTx,
    );
    const comment = await prismaTx.comment.create({
      data: {
        message: dto.message,
        postId: dto.postId,
        creatorId,
      },
      select: commentRowsToSelect
    });
    await this.profileService.updateCommentCount(
      comment.creatorId,
      1,
      prismaTx,
    );
}
```

Наступним є модуль CookieModule, який відповідає за управління токенами аутентифікації на рівні HTTP cookies. Цей модуль не має власного контролера і не взаємодіє з клієнтом безпосередньо. Його основне завдання – встановлення, оновлення та видалення токенів у відповідях сервера.

MailModule є внутрішнім модулем, який відповідає за відправку службових електронних листів у системі. Він інкапсулює логіку побудови та надсилання листів для підтвердження email-адреси, зміни email, скидання пароля та двофакторної автентифікації.

Модуль LikeModule відповідає за функціональність лайків коментарів у системі. Він містить логіку додавання, видалення лайків, забезпечуючи взаємодію користувачів з контентом. Використовується іншими частинами системи, наприклад, у модулях коментарів чи сповіщень.

‘

Наступним є модуль роботи з нотифікаціями NotificationModule, який відповідає за створення, оновлення, отримання та видалення сповіщень у системі. Надає такі функції:

- отримання сповіщень;
- оновлення статусу прочитання конкретного сповіщення;
- видалення сповіщень із перевіркою прав доступу;
- створення/видалення сповіщень для лайків коментарів;
- створення/видалення сповіщень для коментарів до постів.

Фрагмент програмної реалізації має такий зміст:

```
@Injectable()
export class NotificationService {
  async getUserUnreadNotifications(userId: string) {
    return this.prisma.notification.count({
      where: { recipientId: userId, read: false, },
    });
  }
  async updateNotificationReadStatus(
    userId: string,
    notificationId: string,
  ) {
    const notification =
      await this.prisma.notification.findUnique({
        where: { id: notificationId },
      });
}
```

PasswordModule відповідає за хешування паролів та їх перевірку. Забезпечує безпечну обробку паролів користувачів та побудований на основі бібліотеки bcrypt.

PostModule керує постами, забезпечуючи їх створення, оновлення, видалення та отримання інформації, а також взаємодію з коментарями та ієрархією тем. Використовує PrismaService для роботи з базою даних і співпрацює з CommentService, ProfileService та TopicHierarchyService.

‘

PostService обробляє запити: отримання даних поста за ідентифікатором, створення поста з назвою, прив’язкою до теми, початковим коментарем і оновленням лічильників теми, оновлення назви поста, видалення поста з оновленням лічильників і коментарів, отримання даних для коментарів, оновлення останнього коментаря, зміну лічильника коментарів і отримання ID автора для сповіщень. Операції, що потребують цілісності, виконуються в транзакціях.

ProfileModule відповідає за керування профілями користувачів, зокрема за підрахунок і оновлення кількості коментарів. Модуль взаємодіє з PrismaService для роботи з базою даних та TopicHierarchyService для обробки ієрархії тем. Вхідною точкою є ProfileService, який обробляє операції.

SectionModule відповідає за керування секціями в системі. Модуль забезпечує створення, оновлення, видалення та отримання інформації про секції, а також взаємодіє з ProfileService для перерахунку коментарів користувачів. Вхідною точкою є SectionService, який використовує PrismaService для роботи з базою даних.

TokenModule відповідає за керування токенами в системі, зокрема їх створення, пошук та видалення. Модуль використовує PrismaService для взаємодії з базою даних. Вхідною точкою є TokenService, який обробляє операції.

TopicModule керує темами, забезпечуючи їх створення, оновлення, видалення та отримання інформації. Взаємодіє з PrismaService для роботи з базою даних, ProfileService для перерахунку коментарів і TopicHierarchyService для оновлення ієрархії тем. TopicService обробляє запити: отримання деталей теми, створення теми з назвою та прив’язкою до секції чи батьківської теми, оновлення назви теми після перевірки, видалення теми з перерахунком коментарів, лічильників і оновленням останнього коментаря в транзакції.

TopicHierarchyModule управляє ієрархією тем, отримуючи пов’язані теми та оновлюючи їхню метаінформацію. TopicHierarchyService виконує операції: отримання ідентифікаторів усіх батьківських і дочірніх тем, включаючи саму тему, оновлення ідентифікатора останнього коментаря для теми та її батьків на

основі найновішого коментаря, а також перерахунок кількості постів і коментарів для теми та її батьків.

UserModule відповідає за зберігання, отримання та оновлення даних користувачів. UserController взаємодіє з UserService, який обробляє пошук користувачів за поштою чи ідентифікатором, створення нових користувачів, оновлення профілю, видалення облікових записів, підтвердження пошти та перевірку прав доступу.

Next.js 14 − це сучасний фреймворк для розробки WEB-застосунків, який використовується для створення WEB-ресурсу «Онлайн форум». Next.js забезпечує потужний набір інструментів для створення швидких, SEO-оптимізованих і масштабованих WEB-застосунків із підтримкою серверного рендерингу, статичної генерації та інкрементальної статичної регенерації. Основні особливості Next.js 14, застосовані в проєкті, включають:

- використання ISR для періодичного оновлення даних, таких як списки секцій, тем та постів, що забезпечує швидке завантаження та актуальність контенту;
- сучасна система маршрутизації на основі файлової структури, яка забезпечує гнучке управління маршрутами форуму, включаючи сторінки автентифікації, профілів і тем;
- захист від атак XSS за допомогою бібліотеки DOMPurify для санітарної обробки HTML-контенту та CSRF шляхом зберігання токенів у HTTP-only cookies;
- чітка структура проєкту з використанням серверних і клієнтських компонентів для ефективної розробки та підтримки.

Проєкт використовує App Router для організації маршрутів через файлову систему в папці app/. Вхідною точкою є файл page.tsx, який відображає головну сторінку форуму з секціями. Структура маршрутів включає багато сторінок.

Сторінки аутентифікації скдадаються з: /auth/login, /auth/register, /auth/new-password, /auth/reset-password, /auth/new-verification для управління входом, реєстрацією та відновленням паролів;

‘

Сторінки профілю складаються з: /profile/[id], /profile/email-verification, /profile/notifications для перегляду та управління профілем користувача;

Сторінки контенту складаються з: /post/[id], /topic/[id] для відображення окремих постів і тем;

Сторінки користувачів складаються з: /users для перегляду списку користувачів.

Маршрути організовано так, щоб забезпечити інтуїтивну навігацію та швидкий доступ до контенту форуму.

Для управління доступом до маршрутів використовується middleware, яке перевіряє наявність refreshToken у cookies. Middleware перенаправляє авторизованих користувачів зі сторінок автентифікації на головну сторінку форуму, блокує доступ неавторизованих користувачів до адмін-панелі та оновлює токени при необхідності. Фрагмент реалізації middleware має такий зміст:

```
export async function middleware(request: NextRequest) {
  const { url, cookies } = request;
  const refreshToken = cookies.get('refreshToken')?.value;
  const isAuthPage = url.includes('/auth');
  if (isAuthPage && refreshToken) {
    return NextResponse.redirect(
      new URL(DASHBOARD_PAGES.FORUM, url),
    );
  }
  if (refreshToken) {
    const response = NextResponse.next();
    const tokenConfirmation = await fetch(
      `${process.env.SERVER_URL}/auth/login/get-new-tokens`,
      {
        method: 'POST',
        credentials: 'include',
        headers: {
          'Content-Type': 'application/json',
```

‘

```
          Cookie: request.cookies.toString(),
        },
      },
    );
```

Для оптимізації швидкості завантаження сторінок форуму, таких як список секцій, використовується ISR. Це дозволяє періодично оновлювати статично згенеровані сторінки без необхідності повного перезбирання. Приклад реалізації сторінки секцій має такий зміст:

```
export const revalidate = 3600;
async function fetchSections() {
  let sections = await sectionService.getSections();
  sections = sections.sort((a, b) => {
    return new Date(b.createdAt).getTime() - new Date(a.createdAt).getTime();
  });
  return sections;
}
export const metadata: Metadata = {
  title: 'BDR Forum | Sections',
};
export default async function Page() {
  const sections = await fetchSections();
  return (
      {sections.map((section) => (
        <SectionList key={section.id} section={section} />
      ))}
  );
}
```

Для захисту від XSS-атак використовується бібліотека DOMPurify, яка санітізує HTML-контент, введений користувачами.

Для захисту від CSRF токени зберігаються в HTTP-only cookies, що унеможливлює їх доступ через JavaScript, а також забезпечує безпечну передачу через захищені запити.

‘

3.5 Тестування роботи програми та аналіз результатів

Тестування програми та аналіз результатів – ключовий етап розробки програмного забезпечення, який забезпечує його якість, надійність і відповідність вимогам. Тестування дозволяє виявити помилки, перевірити функціональність і оцінити продуктивність системи. Аналіз результатів допомагає інтерпретувати дані тестів, визначити проблемні місця та розробити рекомендації для вдосконалення програми, забезпечуючи її стабільність і зручність для користувачів. Розроблений WEB-ресурс був ретельно протестований, із проведенням понад 100 тестових запусків. Після запуску ресурсу користувач опиниться на головній сторінці. На рисунку 3.2 зображена головна сторінка ресурсу.

Рисунок 3.2 – Загальний вигляд інтерфейсу головної сторінки ресурсу

У неавторизованого користувача доступна функція лише перегляду вмісту, тому перейдемо до сторінок аутентифікації. Сторінка авторизації наведена на рисунку 3.3, а реєстрації – рисунку 3.5.

Відповідно до рисунку 3.3, користувач має змогу авторизуватись завдяки електронній пошті та паролю або Google OAuth.

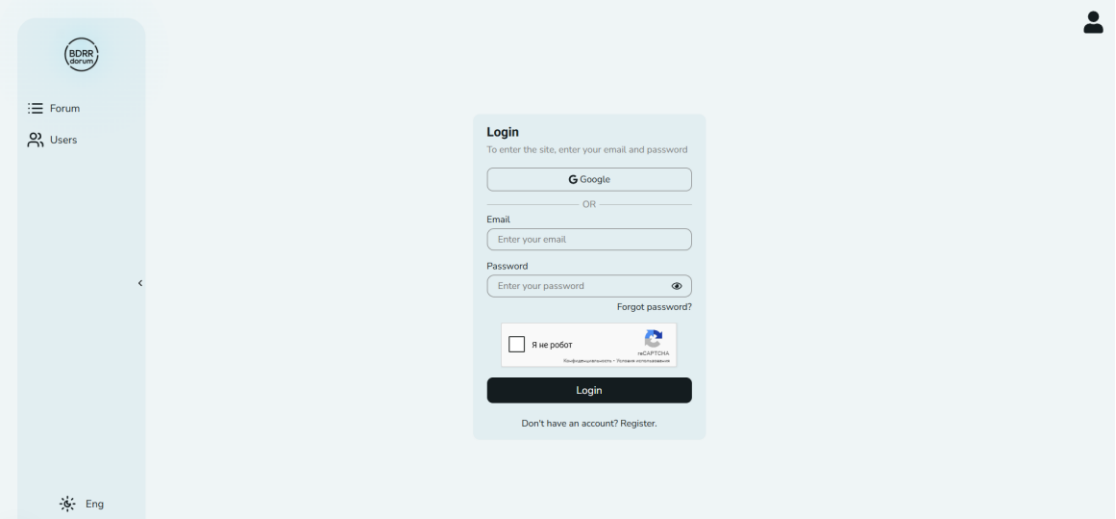

Рисунок 3.3 – Інтерфейс сторінки авторизації користувачів на WEB-ресурсі

Якщо для користувача увімкнено двофакторну автентифікацію, на його електронну адресу буде надіслано лист із кодом підтвердження. Після цього користувача буде перенаправлено на сторінку введення коду двофакторної автентифікації. Користувачу необхідно буде ввести отриманий код у потрібному місці, і у разі, якщо код правильний, то його направить на головну сторінку ресурсу.

Якщо користувач забув пароль, то він має змогу відновити його. Для відновлення йому потрібно натиснути на кнопку «Frogot password?». Сторінка для відновлення паролю наведена на рисунку 3.4.

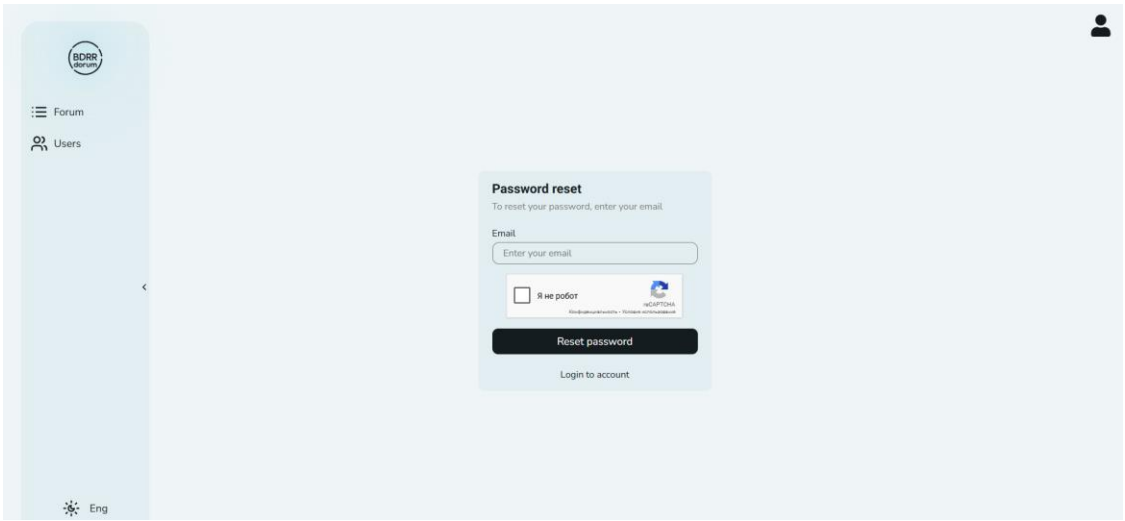

Рисунок 3.4 – Інтерфейс сторінки відновлення паролю

‘

Після того, як користувач введе електронну пошту та натисне «Reset oassword» йому на електронну пошту надійде лист з посиланням. Якщо користувач перейде за посиланням, що надійшло у листі відновлення паролю, то він буде направлений на сторінку вставновленя нового паролю.

Після натискання на кнопку «Continue», якщо пароль успішно зміниться, то користувача направить на головну сторінку форуму.

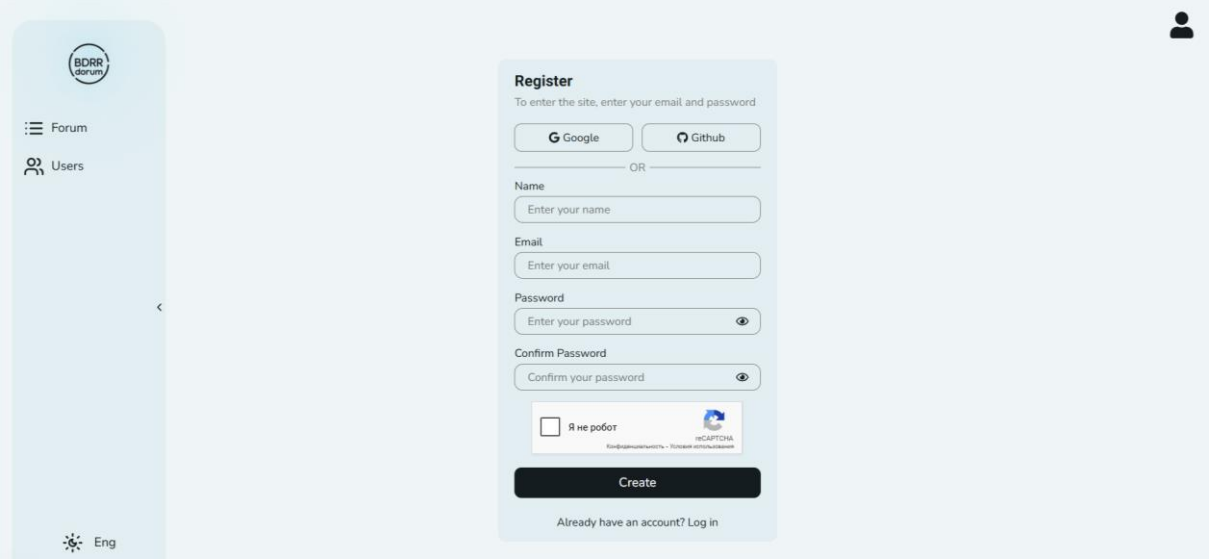

Рисунок 3.5 – Інтерфейс сторінки реєстрації користувачів на WEB-ресурсі

Після відкриття посилання, що надсилається під час реєстрації, користувач підтверджує пошту і має змогу продовжити роботу з форумом в якості користувача з можливістю маніпулювання власними коментарями та постами.

Подальше тестування проводитемо від аккаунту адміністратора, щоб протестувати увесь функціонал.

З головної сторінки користувач може перейти до перегляду тем, що розміщені у секціях. Сторінка однієї з тем наведена на рисунку 3.6.

У темі, що наведена на рисунку 3.6 наявна підтема і відсутні пости, тому перейдемо у дочірню тему, яка містить більш конкретизовану інформацію.

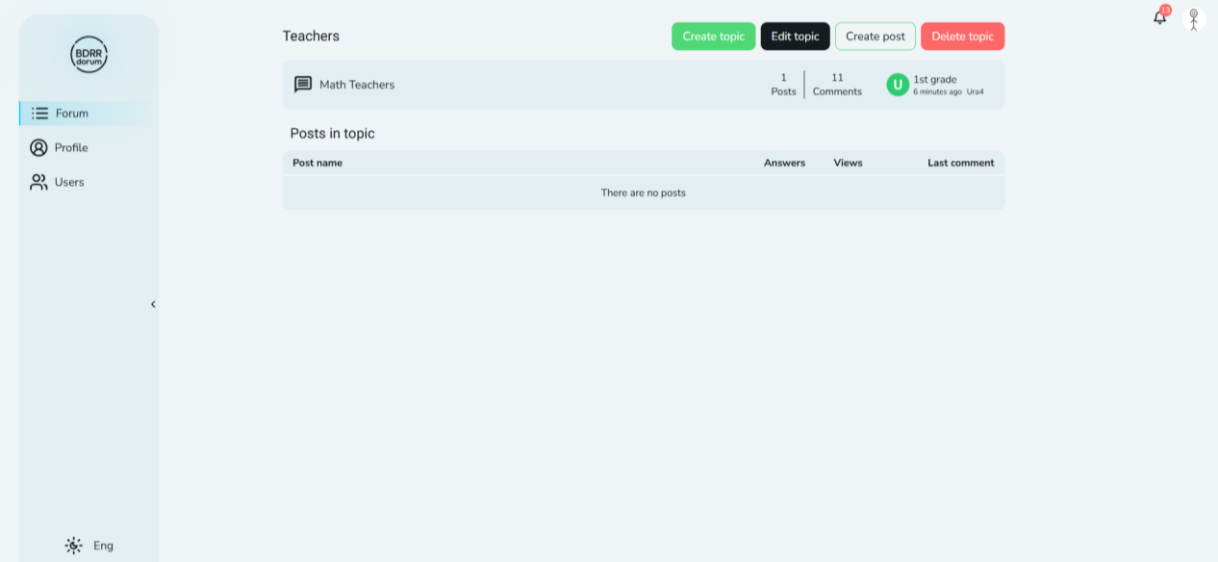

Рисунок 3.6 – Інтерфейс сторінки теми «Teachers» у секції «121 – Software Engineering»

На рисунку 3.7 наведена підтема «Math Teachers». Відповідно до рисунку 3.7, у цій темі наявний пост, тому перейдемо до нього. Пост наведений на рисунку 3.8.

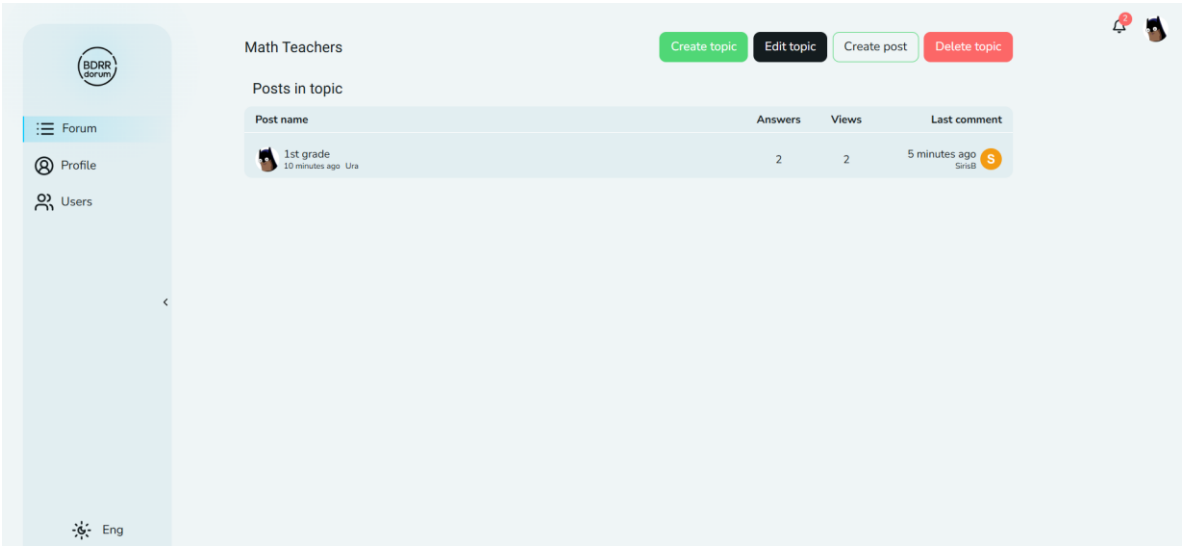

Рисунок 3.7 – Інтерфейс сторінки теми «Math Teachers»

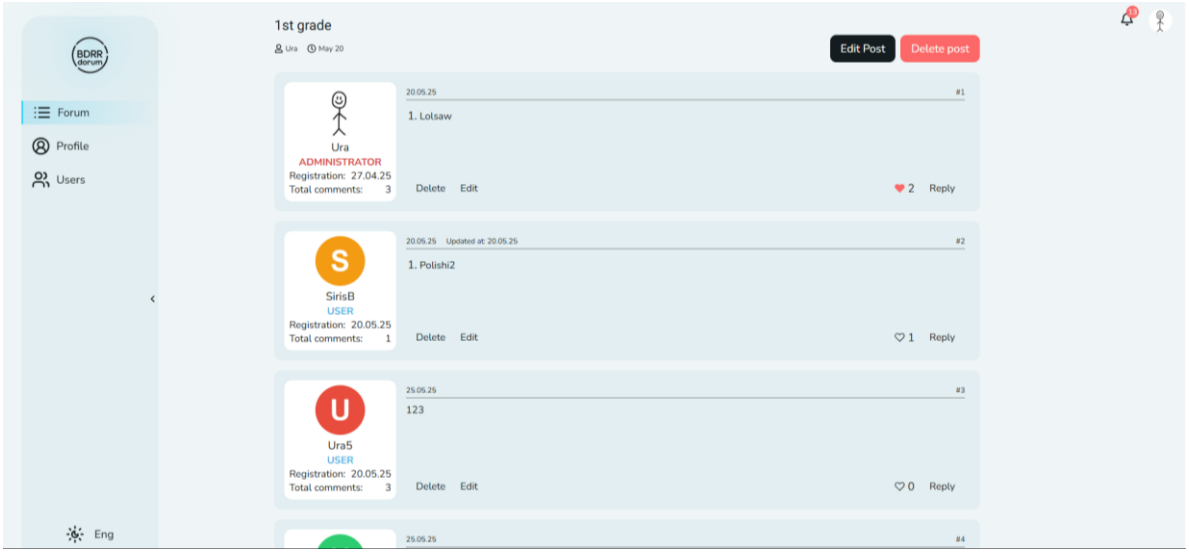

Рисунок 3.8 – Інтерфейс сторінки поста, який розміщений у підтемі

Користувач з роллю має більший набір функцій, наприклад, він може маніпулювати не лише власним контентом, а й чужим. На рисунку 3.9 наведено інтерфейс користувача з розширеним доступом, а на рисунку 3.10 – із звичайним.

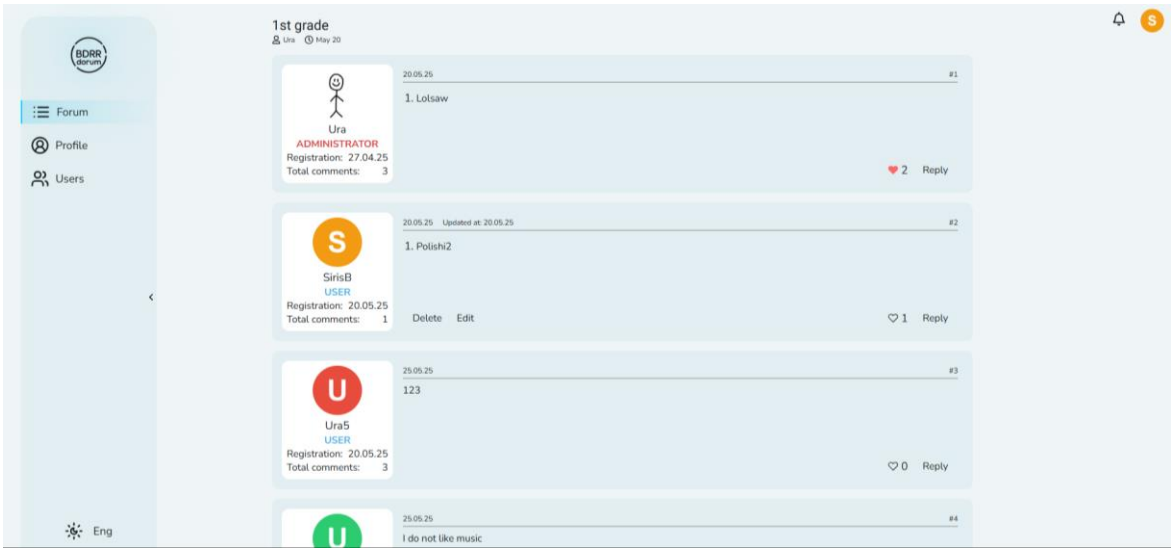

Рисунок 3.9 – Інтерфейс сторінки поста користувача з звичайними правами доступу

Якщо користувач має право на створення секцій, то він може створити нову секцію, ввівши назву і натиснувши на кнопку «Create». Процес створення секції наведено на рисунку 3.10, а нова секція – на рисунку 3.11.

‘

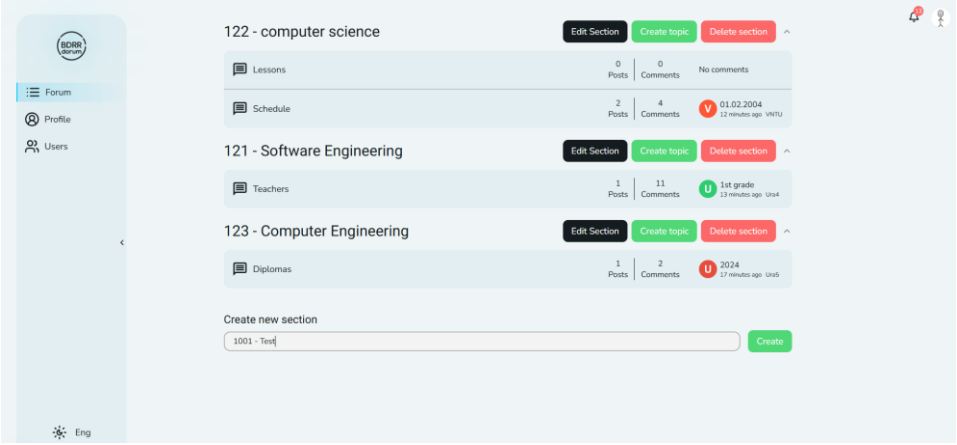

Рисунок 3.10 – Форма створення нової секції

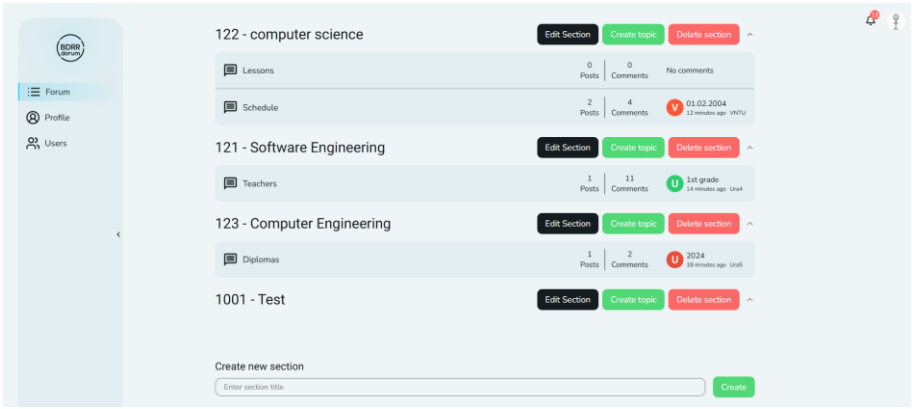

Рисунок 3.11 – Інтерфейс ресурсу після створення нової секції

Якщо користувач має право на створення тем, то він може створити нову тему у будь-якій секції чи підтему у будь-якій темі, натиснувши на кнопку «Create Topic». Інтерфейс для створення нової теми наведено на рисунку 3.12.

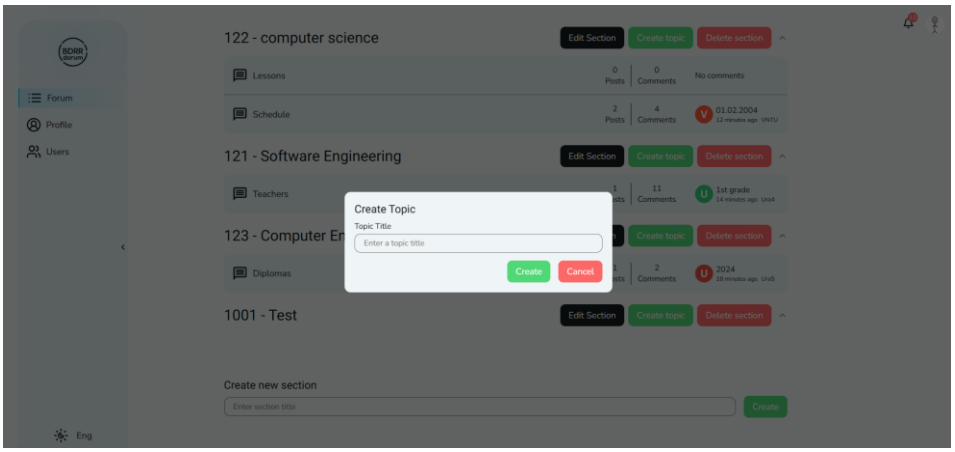

Рисунок 3.12 – Модальне вікно створення нової теми

‘

Якщо користувач бажає створити власний пост, то йому необхідно натиснути на кнопку «Create Post». Інтерфейс для створення нового поста наведено на рисунку 3.13.

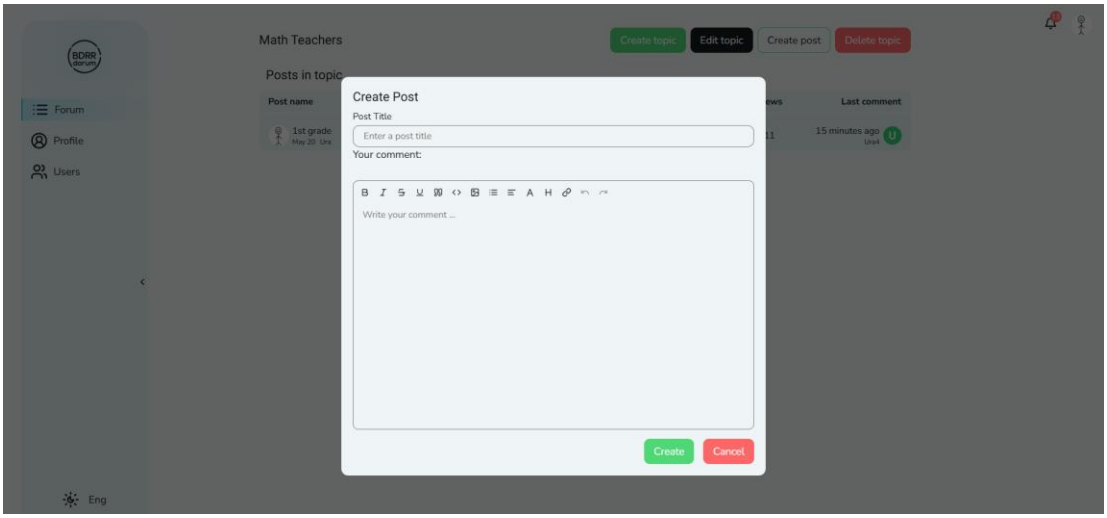

Рисунок 3.13 – Модальне вікно створення нового поста

Для редагування коментаря потрібно перейти до потрібного коментаря та натиснути на кнопку «Edit». Процес редагування коментаря наведений на рисунку 3.14.

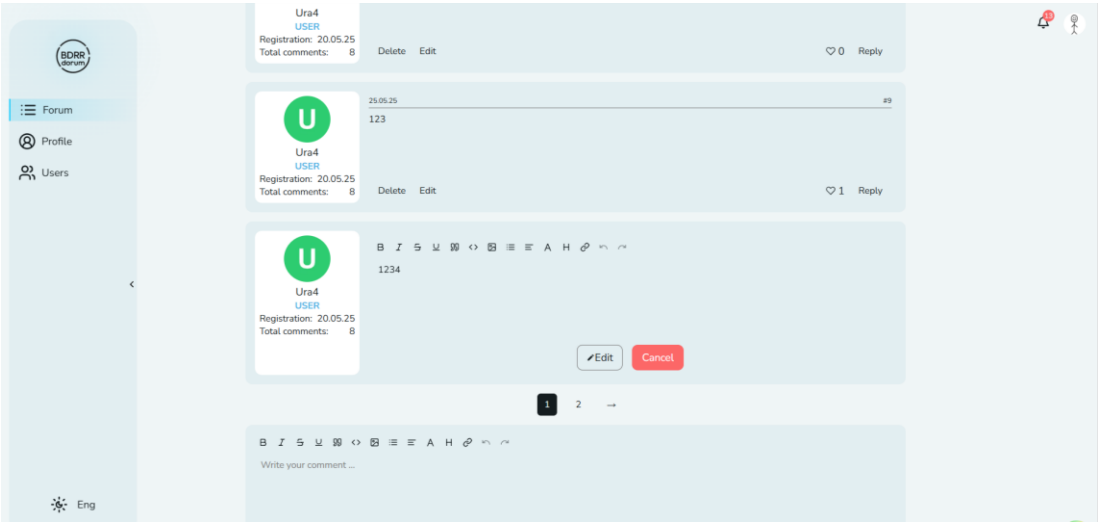

Рисунок 3.14 – Процес редагування коментаря

‘

Користувач має можливість переглядати свій профіль або профіль іншого користувача. У разі перегляду власного профілю доступні функції редагування та видалення. Приклад інтерфейсу профілю наведено на рисунку 3.15.

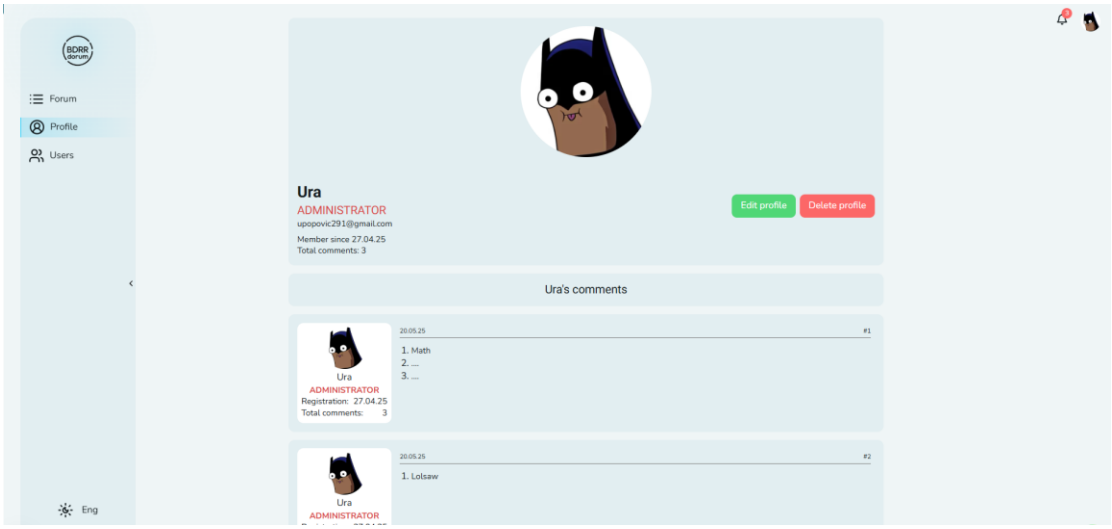

Рисунок 3.15 – Інтерфейс сторінки профіля користувача

Для редагування профіля потрібно натиснути на кнопку «Edit Profile». На рисунку 3.16 наведено вікно редагування профіля.

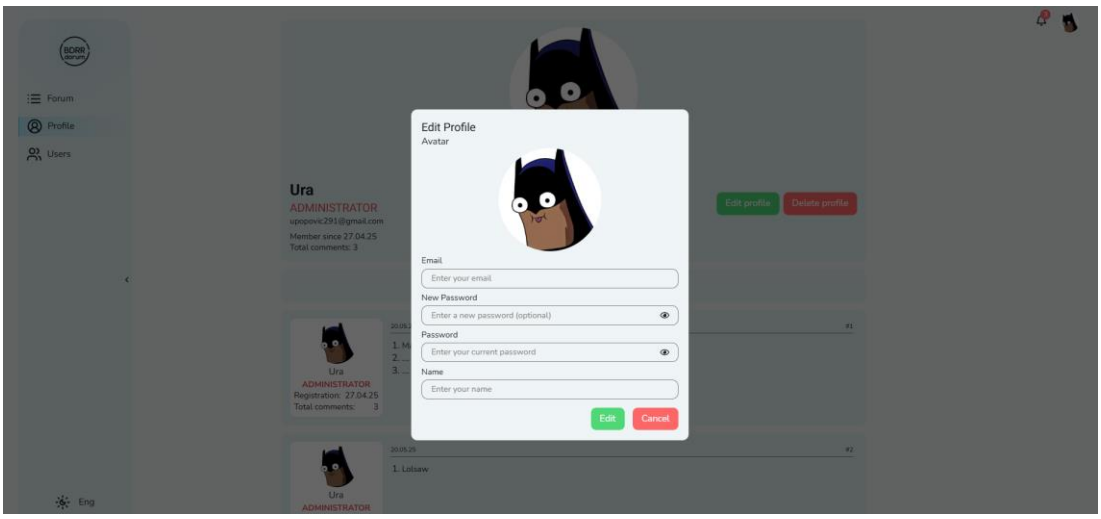

Рисунок 3.16 – Модальне вікно для редагування профілю

Щоб змінити аватар, користувач має натиснути на поточне зображення профілю. Після цього відкриється вікно вибору файлу, де можна обрати нове

‘

зображення з пристрою. Далі користувачеві буде надано можливість обрізати зображення відповідно до власних уподобань. Інтерфейс обрізання зображення показано на рисунку 3.17.

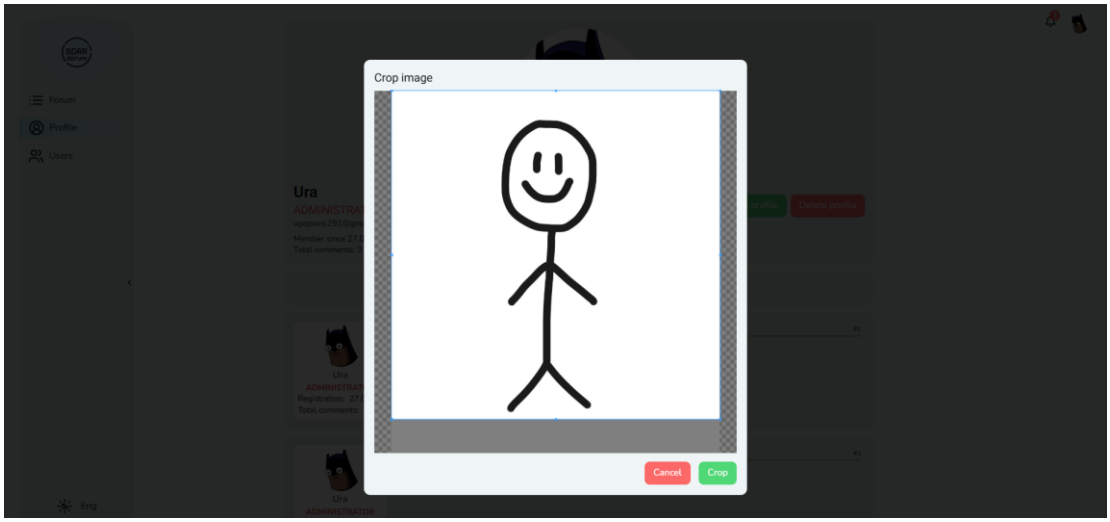

Рисунок 3.17 – Модальне вікно для оновлення аватарки користувача

Отже, розроблений форумний WEB-ресурс «Онлайн форум», реалізований з використанням технологій Next.js та NestJS, успішно пройшов тестування і відповідає всім заданим функціональним та нефункціональним вимогам.

Тепер доцільно провести порівняння з аналогами. В таблиці 3.7 подано результати тестування ресурсу та аналогів.

В результаті тестування було розглянуто такі аспекти як: наявність серверного рендерингу; можливості SEO-оптимізації; підтримка масштабованості; гнучкість API; використання сучасних технологій; базування на рушії «XenForo»; підтримка української локалізації.

Відповідно до результатів тестування, що наведені у таблиці 3.7, було проведено порівняльне тестування розробленого ресурсу з іншими форумами.

Розроблений ресурс переважає в більшості ключових аспектів, важливих для сучасних ресурсів. Наприклад, система підтримує повноцінний серверний рендеринг через Next.js, що позитивно впливає на швидкість завантаження та

SEO. На відміну від аналогів, які обмежено використовують SSR або працюють на клієнтському рендерингу чи застосовують інші методи оптимізації.

Таблиця 3.7 – Результати тестування WEB-ресурсу та аналогів

Аспект	Розроблений WEB-ресурс	UKRAINE GTA	Політичний форум	Red Forum
Наявність серверного рендерингу	+	±	±	±
Можливості SEO-оптимізації	+	±	±	±
Підтримка масштабованості	+	+	±	+
Гнучкість API	+	-	±	-
Використання сучасних технологій	+	±	-	±
Базування на рушії «XenForo»	-	+	-	+
Підтримка української локалізації	-	±	±	±

Важливою особливістю ресурсу є його гнучкість API, реалізована на основі NestJS. Це дозволяє інтегрувати сторонні сервіси, реалізовувати власну логіку без обмежень. Ресурси, побудовані на руші XenForo, значно обмежені у цьому аспекті, наприклад, доступ до API відсутній, або вимагає окремих платних плагінів, що ускладнює розширення функціоналу.

З точки зору використання сучасних технологій, розроблений ресурс має переваги, оскільки реалізований на новітньому стеку, що забезпечує кращу продуктивність, зручність підтримки, подальше розширення та наявність великої кількості ком’юніті. Інші форуми частково використовують застарілі технології або обмежені можливостями рушія.

Єдиним недоліком розробки є відсутність української локалізації. Варто вказати, що у XenForo локалізація також здійснюється лише через плагіни або ручний переклад шаблонів, що може ускладнювати оновлення та модифікацію системи. Розроблений ресурс надає розширений функціонал у 3 напрямах: швидке завантаження та SEO-оптимізацію завдяки серверному рендерингу на Next.js, гнучке API на NestJS для інтеграції та розширення, а також сучасний технологічний стек, який забезпечує високу продуктивність і масштабованість.

Таким чином, аналіз показав, що розроблений WEB-ресурс «Онлайн форум» перевершує аналоги за більшістю ключових критеріїв.

3.6 Висновок

У цьому розділі було проведено порівняльний аналіз технологій для розробки клієнтської та серверної частин WEB-ресурсу «Онлайн форум», а також обґрунтовано вибір інструментів і підходів для реалізації проєкту.

Для клієнтської частини обрано TypeScript завдяки статичній типізації, яка зменшує кількість помилок на етапі компіляції, полегшує рефакторинг і підтримує сучасні фреймворки. Next.js обрано через підтримку серверного рендерингу, вбудованої маршрутизації та оптимізації продуктивності, що сприяє швидкому завантаженню сторінок і SEO-оптимізації.

Для серверної частини було обрано TypeScript у поєднанні з NestJS завдяки високій продуктивності, модульній структурі, підтримці об’єктно-орієнтованого програмування та зручності масштабування. NestJS забезпечує чітку організацію коду, вбудовану підтримку валідації, авторизації та обробки помилок, що полегшує розробку і підтримку великих проєктів.

Для управління базою даних було обрано SQL у поєднанні з Prisma ORM завдяки широкій підтримці реляційних баз даних, високій продуктивності та зручному API, який абстрагує SQL-запити, забезпечуючи безпеку і передбачуваність роботи з даними.

Середовищем розробки обрано Visual Studio Code через його безкоштовність, підтримку TypeScript, інтеграцію з Git, велику кількість розширень і зручність налагодження, що забезпечує ефективну розробку клієнтської та серверної частин.

Проведено тестування WEB-ресурсу, яке підтвердило відповідність функціональним і нефункціональним вимогам. Порівняння з аналогами показало переваги розробленого ресурсу. Таким чином, розроблений WEB-ресурс перевершує аналоги за більшістю ключових критеріїв.

Отже, розроблений WEB-ресурс «Онлайн форум» має 3 ключові переваги порівняно з аналогами: швидке завантаження та SEO-оптимізація завдяки серверному рендерингу на Next.js, гнучке API на NestJS для легкої інтеграції та розширення, а також сучасний стек технологій, що забезпечує високу продуктивність і масштабованість.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було розроблено WEB-ресурс «Онлайн форум», який забезпечує зручну та швидку платформу для взаємодії користувачів, обміну знаннями тощо. Усі поставлені завдання виконано в повному обсязі: обґрунтовано доцільність розробки WEB-ресурсу «Онлайн форум, спроектовано WEB-ресурс «Онлайн форум, програмно реалізовано WEB-ресурс «Онлайн форум», виконано тестування програми та проведено аналіз отриманих результатів; розроблено інструкцію користувача.

Для обґрунтування доцільності розробки WEB-ресурсу було проаналізовано існуючі аналоги: «UKRAINE GTA», «Політичний форум» та «Red Forum». Виявлено їх основні недоліки, зокрема вузьку спеціалізацію, погану адаптацію під мобільні пристрої, повільне завантаження сторінок та проблеми з SEO. На основі аналізу обґрунтовано доцільність створення універсального WEB-ресурсу, який поєднує сучасні підходи до рендерингу.

Під час проектування системи створено клієнт-серверну архітектуру, яка забезпечує розмежування обов’язків між клієнтом і сервером. Для візуалізації структури створено UML-діаграми класів, які чітко описують компоненти системи та їхні взаємозв’язки, що полегшує розуміння проєкту. Розроблена ER-модель бази даних чітко визначає сутності та їхні зв’язки, що спрощує проектування та взаємодію з даними. Також розроблено схеми алгоритмів ключових процесів, таких як реєстрація, аутентифікація, створення тем і коментарів. Запропонована структура забезпечує модульність, масштабованість і можливість легкого додавання нових функцій у майбутньому.

Було проведено порівняльний аналіз технологій і обґрунтовано вибір інструментів. TypeScript обрано через статичну типізацію, яка зменшує кількість помилок і полегшує рефакторинг. Next.js забезпечує швидке завантаження сторінок і SEO-оптимізацію завдяки серверному рендерингу, тоді як NestJS пропонує модульну структуру та зручність масштабування. SQL у поєднанні з Prisma ORM забезпечує безпечну та ефективну роботу з реляційними базами

даних. Середовищем розробки обрано Visual Studio Code за його універсальність і підтримку необхідних інструментів. Проведено повне тестування системи, у результаті чого підтверджено її відповідність вимогам. Для зручності розроблено інструкцію, яка містить покрокові інструкції, що допоможуть взаємодіяти з ресурсом.

Мета роботи, яка полягала в розширенні функціональних можливостей WEB-ресурс «Онлайн форум» досягнута за рахунок введення таких 3 додаткових компонентів функціоналу, а саме: швидке завантаження та SEO-оптимізація завдяки серверному рендерингу на Next.js, гнучке API на NestJS для легкої інтеграції та розширення, а також сучасний стек технологій, що забезпечує високу продуктивність і масштабованість.

Робота виконана у повному обсязі відповідно до поставлених цілей, вимог та методичних рекомендацій щодо підготовки бакалаврських кваліфікаційних робіт. Усі етапи були реалізовані належним чином, що підтверджує якість, завершеність і практичну цінність розробленого рішення [31].

Отже, розроблений WEB-ресурс «Онлайн форум» відповідає заданим вимогам, забезпечує ефективну комунікацію та взаємодію користувачів, а також повністю реалізує поставлену мету – розширює функціональні можливості платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Попович Ю. Ю., Крилик Л. В. Перспективи використання серверного рендерингу у контексті розробки WEB-ресурсу «Онлайн форум». *LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації.* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23328/19293 (дата звернення: 12.05.2025).

2. Свідоцтво про реєстрацію авторського права на твір № № 135891. Комп'ютерна програма «WEB-ресурс «Онлайн-форум»» / Крилик Л. В., Попович Ю. Ю., дата реєстрації 07.05.2025 р.

3. Порівнюємо способи генерації сторінок: CSR, SSR, SSG, ISR. Гайд на основі стеку React. URL: https://dou.ua/forums/topic/41585/ (дата звернення: 12.05.2025).

4. When to Use SSR or SSG? | Understanding Rendering Options. URL: https://www.azion.com/en/learning/performance/when-to-use-ssr-or-ssg/ (дата звернення: 12.05.2025).

5. Server Side Rendering: Benefits, Use Cases, and Best Practices. URL: https://cloudinary.com/guides/automatic-image-cropping/server-side-rendering-benefits-use-cases-and-best-practices#:~:text=Websites%20with%20large%20amounts%20of,a%20more%20responsive%20user%20interface (дата звернення: 12.05.2025).

6. What Is A Forum? A Beginner's Guide (Definition + Examples). URL: https://www.mightynetworks.com/resources/online-forum (дата звернення: 13.05.2025).

7. Disadvantages of Forum Web Hosting. URL: https://websitehosting.com/guide/disadvantages-of-forum-web-hosting/ (дата звернення: 14.05.2025).

8. 8 advantages and disadvantages of forums. URL: https://entrepreneurshipera.com/advantages-and-disadvantages-of-forums/ (дата звернення: 14.05.2025).

‘

9. UkraineGta. URL: https://forum.ukraine-gta.com.ua/ (дата звернення: 15.05.2025).

10. Політичний ФОРУМ. URL: https://forum.pravda.com.ua/ (дата звернення: 15.05.2025).

11. Red Forum. URL: https://red-forum.com/ (дата звернення: 15.05.2025).

12. revalidatePath. URL: https://nextjs.org/docs/app/api-reference/functions/revalidatePath (дата звернення: 15.05.2025).

13. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 15.05.2025).

14. Об’єктно-Орієнтоване Програмування (ООП). Пояснюємо чотири основні принципи. URL: https://career.softserveinc.com/uk-ua/stories/what-is-object-oriented-programming-oop-explaining-four-major-principles (дата звернення: 15.05.2025).

15. Введення в ООП. URL: https://w3schoolsua.github.io/hyperskill/oop-intro.html#gsc.tab=0 (дата звернення: 15.05.2025).

16. Функціональне програмування та приклади його використання. URL: https://foxminded.ua/funktsionalne-prohramuvannia/ (дата звернення: 15.05.2025).

17. Методичні вказівки до виконання курсової роботи з дисципліни «Організація баз даних та знань» для студентів денної та заочної форм навчання за спеціальністю «Комп’ютерні науки» / Уклад.: Т. О. Савчук, О. В. Ольшанська. Вінниця : ВНТУ, 2021. 38 с.

18. Поняття ER-моделі. Поняття сутності (entity). Атрибути. Види атрибутів. URL: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/#google_vignette (дата звернення: 17.05.2025).

19. Алгоритм: що це таке і для чого він потрібен. URL: https://mathema.me/blog/algoritm-shho-cze-take/ (дата звернення: 15.05.2025).

20. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 15.05.2025).

21. TypeScript for the New Programmer. URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html (дата звернення: 15.05.2025).

22. Effective Go. URL: https://go.dev/doc/effective_go (дата звернення: 15.05.2025).

23. Express 5.1.0. Fast, unopinionated, minimalist web framework for Node.js. URL: https://expressjs.com/ (дата звернення: 15.05.2025).

24. Hello, nest! URL: https://nestjs.com/ (дата звернення: 16.05.2025).

25. Введення в SQL. URL: https://w3schoolsua.github.io/sql/sql_intro.html#gsc.tab=0 (дата звернення: 17.05.2025).

26. Object Query Language. URL: https://www.ibm.com/docs/en/networkmanager/4.2.0?topic=reference-object-query-language (дата звернення: 17.05.2025).

27. Why did we build Visual Studio Code? URL: https://code.visualstudio.com/docs/editor/whyvscode#:~:text=With%20support%20for%20hundreds%20of,navigate%20your%20code%20with%20ease (дата звернення: 17.05.2025).

28. WebStorm Features. URL: https://www.jetbrains.com/webstorm/features/ (дата звернення: 17.05.2025).

29. What is Sublime Text? What are its benefits? URL: https://sourcebae.com/blog/what-is-sublime-text-what-are-its-benefits/ (дата звернення: 17.05.2025).

30. What is Prisma ORM? URL: https://www.prisma.io/docs/orm/overview/introduction/what-is-prisma (дата звернення: 17.05.2025).

31. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. Вінниця : ВНТУ, 2023. (PDF, 54 с).

‘

ДОДАТКИ

ДОДАТОК А (обов'язковий)
Протокол перевірки кваліфікаційної роботи

Назва роботи: Розробка WEB-ресурсу «Онлайн форум»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3,40 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А. А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О. К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В. С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Крилик Л. В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Попович Ю. Ю.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (обов’язковий)

Лістинг програми

```
@Injectable()
export class AuthService {
  async login(dto: LoginDto) {
    const user = await this.userService.getByEmail(
      dto.email,
    );
    if (!user || !user.password)
      throw new NotFoundException(
        'User not found. Ensure the user exists and the password is properly set, then try again.',
      );
    if (user && user.method != 'CREDENTIALS') {
      throw new UnauthorizedException(
        'This email is registered via an external provider. Please log in using the respective login
option.',
      );
    }
    if (!user.isVerified) {
      await this.emailConfirmationService.sendVerificationToken(
        user.email,
      );
      throw new UnauthorizedException(
        'Your email is not confirmed. Please check your mail and confirm the address.',
      );
    }
    if (user.isTwoFactorEnabled) {
      if (!dto.code) {
        await this.twoFactorAuthService.sendTwoFactorToken(
          user.email,
        );
        return {
          message:
            'Check your email. Two-factor authentication code required',
        };
      }
      await this.twoFactorAuthService.validateTwoFactorToken(
        user.email,
        dto.code,
      );
    }
    const tokens = this.jwtTokenService.generateTokens(
      this.makeJwtObjectFromUser(user),
    );
    const { password, ...response } = user;
    return { response, ...tokens };
  }
  async extractProfileFromCode(
    provider: string,
```

```
    code: string,
  ) {
    const providerInstance =
      this.providerService.findByService(provider);
    const profile =
      await providerInstance.findUserByCode(code);
    if (!profile) {
      throw new NotFoundException(
        'Profile not found or invalid code. Please verify the authorization code and ensure that it
corresponds to a valid user profile.',
      );
    }
    const account = await this.accountService.findAccount(
      profile.id,
      provider,
    );
    let user = null;
    const providerKey =
      profile.provider.toUpperCase() as keyof typeof AuthMethod;
    if (account) {
      user = await this.userService.getById(account.userId);
      if (user) {
        const tokens = this.jwtTokenService.generateTokens(
          this.makeJwtObjectFromUser(user),
        );
        return {
          user,
          ...tokens,
        };
      }
    }
    if (!oldUser) {
      user = await this.userService.create(
        profile.email,
        '',
        profile.name,
        authMethod,
        profile.picture,
        true,
        'USER',
      );
    } else {
      user = oldUser;
    }
    const tokens = this.jwtTokenService.generateTokens(
      this.makeJwtObjectFromUser(user),
    );
    return {
      user,
      ...tokens,
    };
  }
'
```

```
  async register(dto: RegisterDto) {
    const oldUser = await this.userService.getByEmail(
      dto.email,
    );
    if (oldUser)
      throw new ConflictException(
        'User already exist. Please ensure that the user has not been registered previously, and use a
different identifier if needed.',
      );
    const pictureUrl =
      await this.uploadsService.generateLetterAvatar(
        dto.name[0],
      );
    await this.emailConfirmationService.sendVerificationToken(
      user.email,
    );
    return true;
  }
}
@Injectable()
export class AccountService {
  constructor(private prisma: PrismaService) {}

  async findAccount(id: string, provider: string) {
    return await this.prisma.account.findFirst({
      where: {
        id,
        provider,
      },
    });
  }
  async updateProviderAccessToken(
    userId: string,
    provider: string,
    tokens: {
      accessToken: string;
      refreshToken?: string;
      expiresAt: number;
    },
  ) {
    await this.prisma.account.updateMany({
      where: {
        userId: userId,
        provider: provider,
      },
      data: {
        accessToken: tokens.accessToken,
        refreshToken: tokens.refreshToken ?? undefined,
        expiresAt: new Date(tokens.expiresAt),
      },
    });
  }
```

‘

```
}
@Injectable()
@Injectable()
export class CommentService {
  async getUserCommentsById(
    page: number,
    limit: number,
    userId: string,
  ) {
    const skip = (page - 1) * limit;
    const comments = await this.prisma.comment.findMany({
      where: { creatorId: userId },
      skip,
      take: limit,
      orderBy: {
        createdAt: 'asc',
      },
      select: commentToSelect
    });
    const totalComments = await this.prisma.comment.count({
      where: { creatorId: userId },
    });
    const likes = await this.prisma.like.findMany({
      where: {
        userId,
        commentId: {
          in: comments.map((comment) => comment.id),
        },
      },
    });
    return {
      data: comments.map((comment) => {
        const { _count, ...commentOtherFields } = comment;
        return {
          ...commentOtherFields,
          likedByMe: likes.find(
            (like) => like.commentId === comment.id,
          ),
          likeCount: _count.likes,
        };
      }),
      total: totalComments,
      page,
      limit,
      totalPages: Math.ceil(totalComments / limit),
    };
  }
  async getCommentByIdForNotification(
    commentId: string,
    tx?: Prisma.TransactionClient,
  ) {
    const prisma = tx || this.prisma;
```

‘

```
  const comment = await prisma.comment.findUnique({
    where: { id: commentId },
    select: { creatorId: true, postId: true },
  });
  if (!comment) {
    throw new NotFoundException(
      'Comment not found. Please try again.',
    );
  }
  return comment;
}
async createComment(
  dto: CreateCommentDto,
  creatorId: string,
  tx?: Prisma.TransactionClient,
) {
  const executeCommentCreation = async (
    prismaTx: Prisma.TransactionClient,
  ) => {
    const post = await this.postService.getPostForComment(
      dto.postId,
      prismaTx,
    );
    const comment = await prismaTx.comment.create({
      data: {
        message: dto.message,
        postId: dto.postId,
        creatorId,
      },
      include: commentToInclude
    });
    await this.profileService.updateCommentCount(
      comment.creatorId,
      1,
      prismaTx,
    );
    await this.postService.updateCommentCount(
      dto.postId,
      1,
      prismaTx,
    );
    await this.postService.updateLastComment(
      post.id,
      comment.id,
      prismaTx,
    );
    await this.topicHierarchyService.updateLastCommentInHierarchy(
      post.topicId,
      prismaTx,
    );
    await this.topicHierarchyService.updateCounters(
      post.topicId,
```

‘

```
      prismaTx,
    );
    if (post.totalComments >= 1) {
      await this.notificationService.createNotificationForPost(
        creatorId,
        dto.postId,
        prismaTx,
      );
    }
    return {
      ...comment,
      likeCount: 0,
      likedByMe: false,
    };
  };
  if (tx) {
    return await executeCommentCreation(tx);
  }
  return await this.prisma.$transaction(
    async (prismaTx) => {
      return await executeCommentCreation(prismaTx);
    },
  );
}
async updateComment(id: string, dto: UpdateCommentDto) {
  const existingComment =
    await this.prisma.comment.findFirst({
      where: {
        id,
      },
    });
  if (!existingComment) {
    throw new NotFoundException(
      `Comment not found. Please try again.`,
    );
  }
  return await this.prisma.comment.update({
    where: { id },
    data: {
      message: dto.message,
    },
    include: commentToInclude
  });
}
async deleteComment(id: string, userId: string) {
  return await this.prisma.$transaction(async (tx) => {
    const existingComment = await tx.comment.findFirst({
      where: { id },
    });
    const post = await this.postService.getPostForComment(
      existingComment.postId,
      tx,
```

```
      );
      if (!existingComment) {
        throw new NotFoundException(
          'Comment not found. Please try again.',
        );
      }
      const comment = await tx.comment.delete({
        where: { id },
      });
      await this.postService.updateCommentCount(
        comment.postId,
        -1,
        tx,
      );
      if (post.lastCommentId == comment.id) {
        const latestComment = await tx.comment.findFirst({
          where: { postId: post.id },
          orderBy: { createdAt: 'desc' },
          select: { id: true },
        });
        await this.postService.updateLastComment(
          post.id,
          latestComment?.id || null,
          tx,
        );
        await this.topicHierarchyService.updateLastCommentInHierarchy(
          post.topicId,
          tx,
        );
      }
      await this.topicHierarchyService.updateCounters(
        post.topicId,
        tx,
      );
      await this.profileService.updateCommentCount(
        comment.creatorId,
        -1,
        tx,
      );
      await this.notificationService.deleteNotificationForPost(
        userId,
        post.id,
        tx,
      );
      return comment;
    });
  }
}
```

‘

ДОДАТОК В (обов'язковий)

ГРАФІЧНА ЧАСТИНА

«РОЗРОБКА WEB-РЕСУРСУ «ОНЛАЙН ФОРУМ»»

Виконав: студент 4 курсу, групи 2КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Попович Ю. Ю.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)
«03» 06 2025 р.

Вінниця ВНТУ – 2025 рік

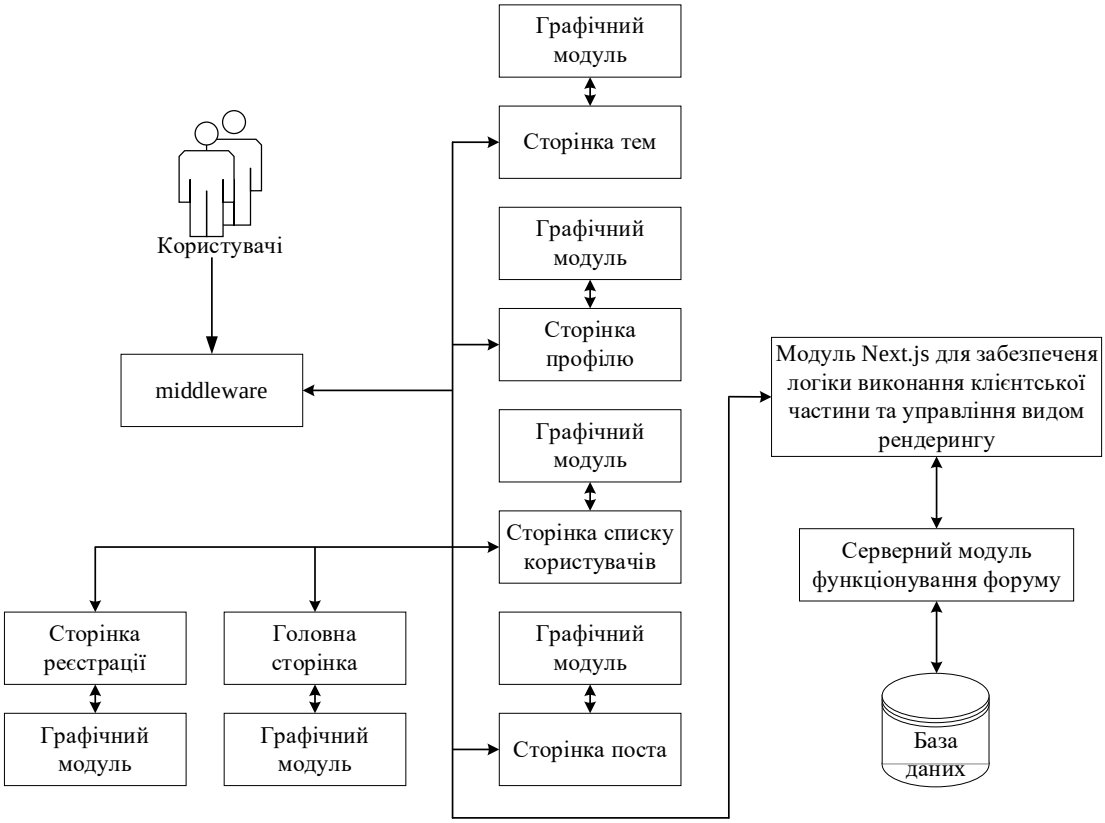

Рисунок В.1 − Загальна структурна схема функціонування системи

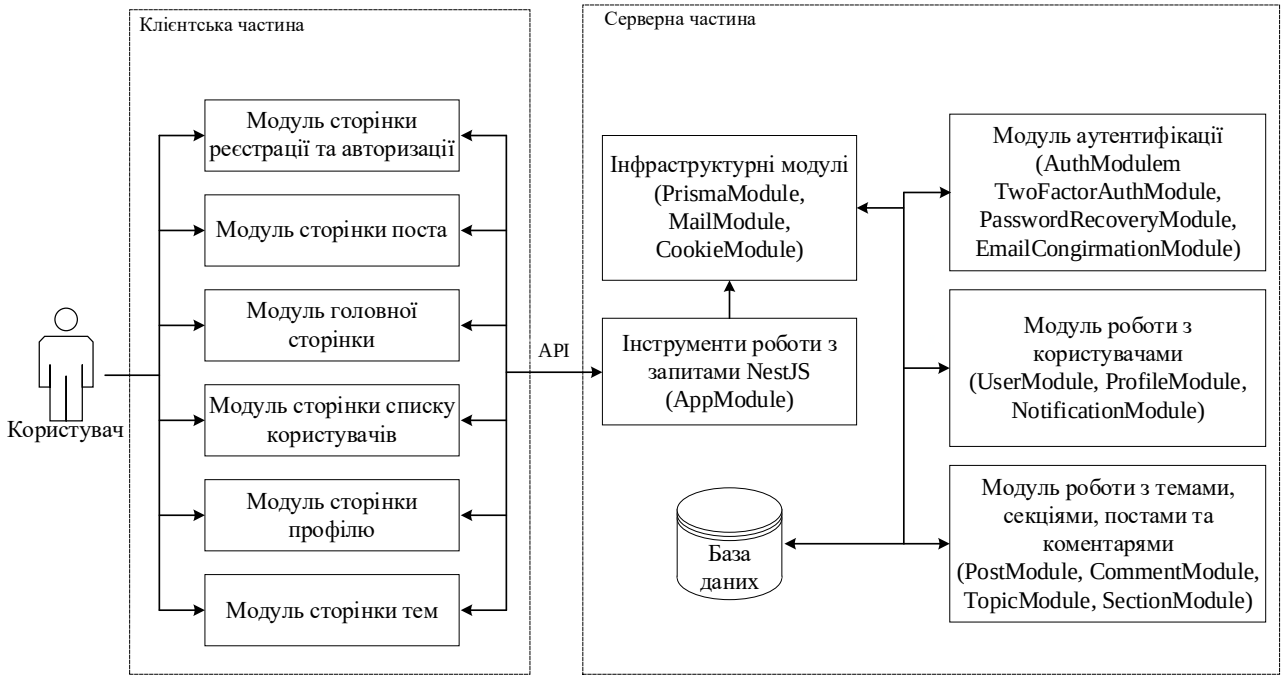

Рисунок В.2 –Загальна структурна схема компонентів програмного модуля «Онлайн форум»

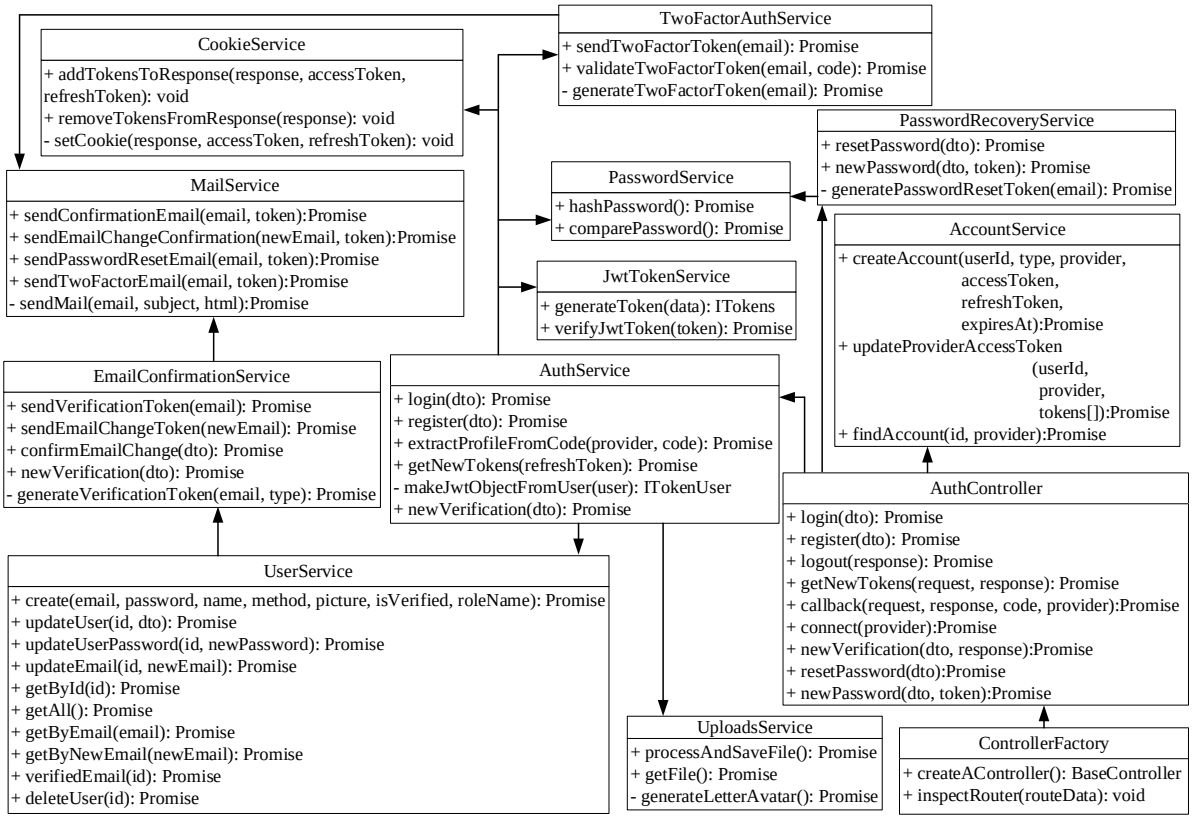

Рисунок В.3 – UML-діаграма класів модуля роботи з аутентифікацією серверної частини

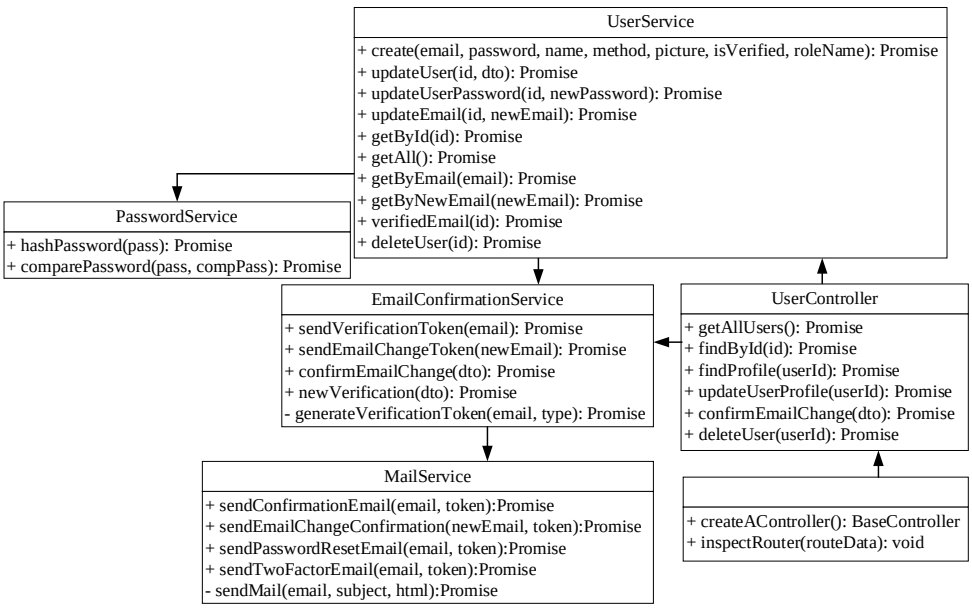

Рисунок В.4 – UML-діаграма класів модуля роботи з профілем серверної частини

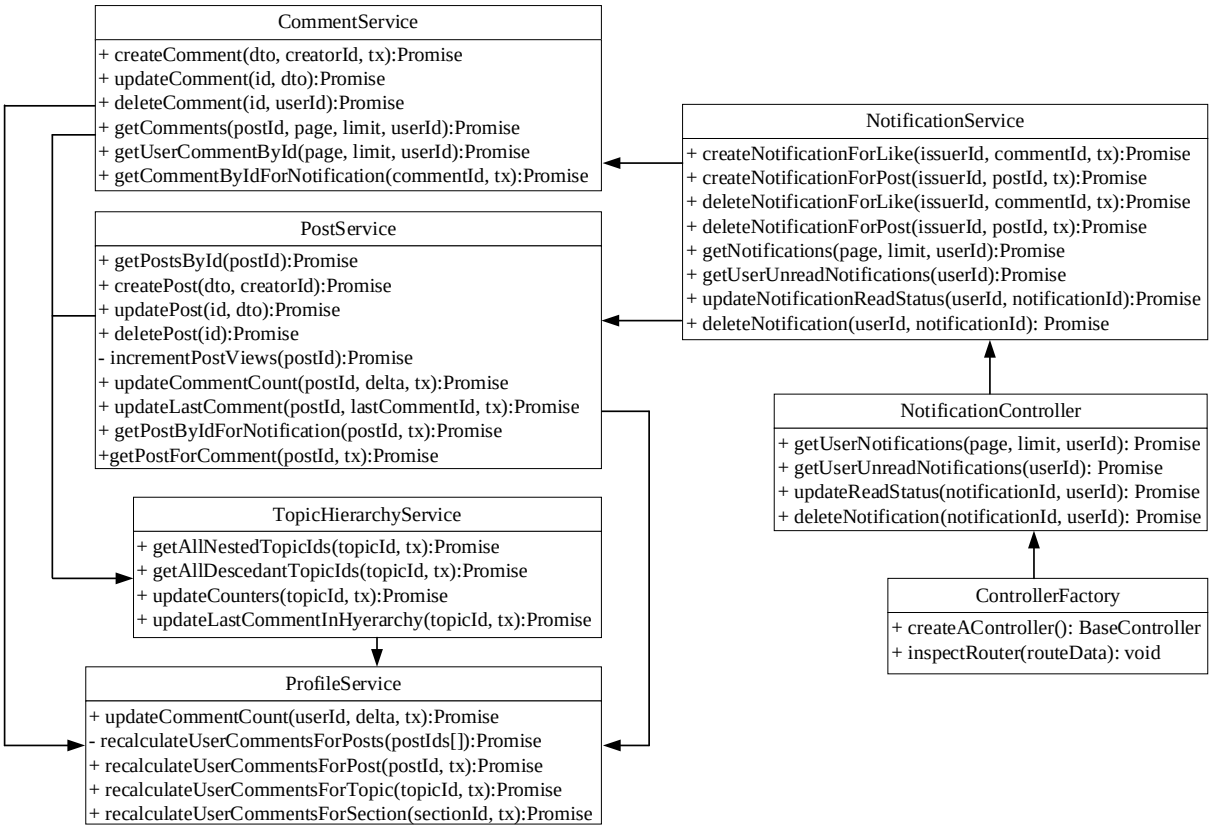

Рисунок В.5 – UML-діаграма класів модуля роботи із сповіщеннями серверної частини

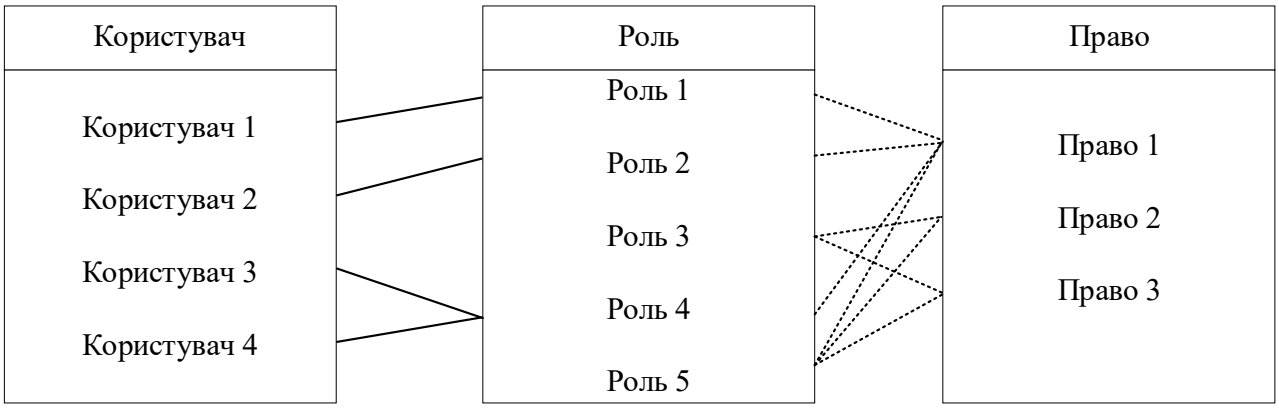

Рисунок В.6 – Фрагмент ER-діаграми, що демонструє зв’язки між екземплярами сутностей предметної області «Онлайн форум»

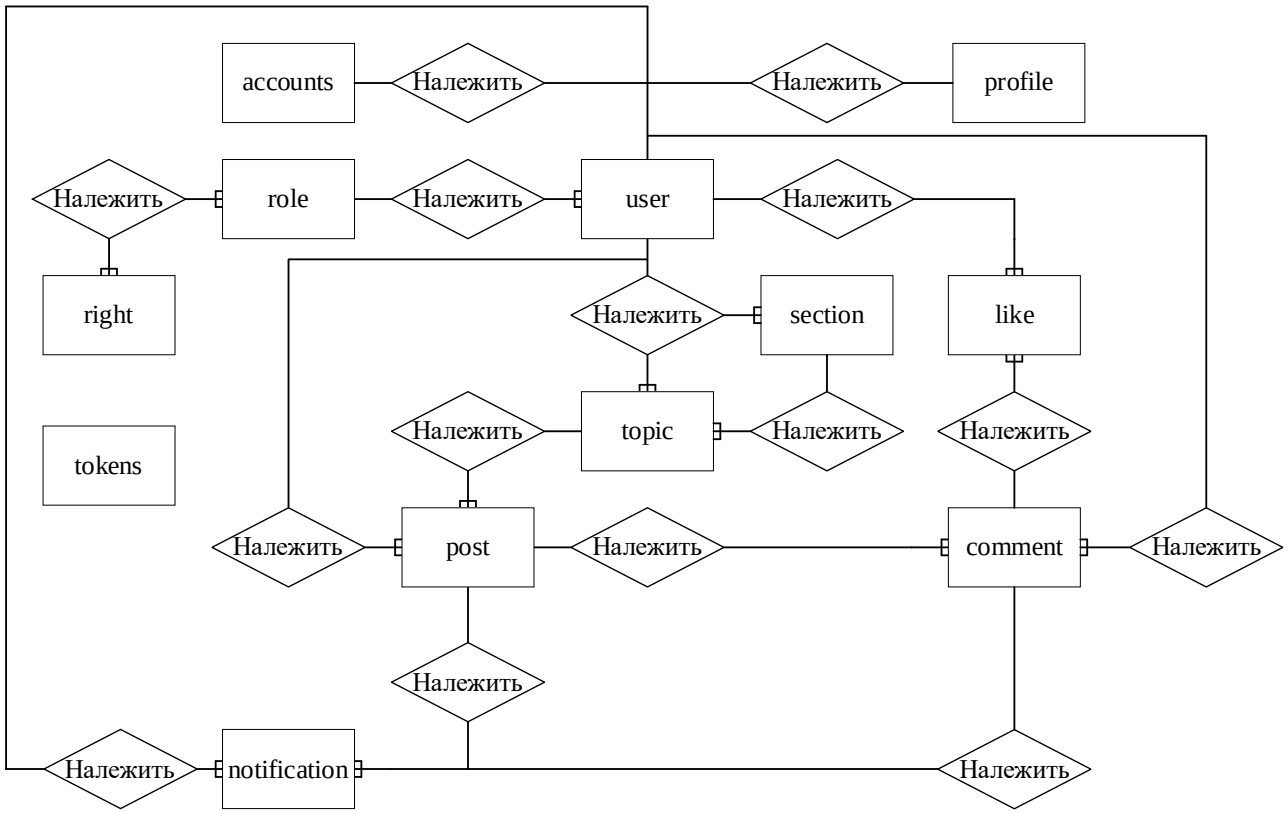

Рисунок В.7 –ER-модель предметної області «Онлайн форум»

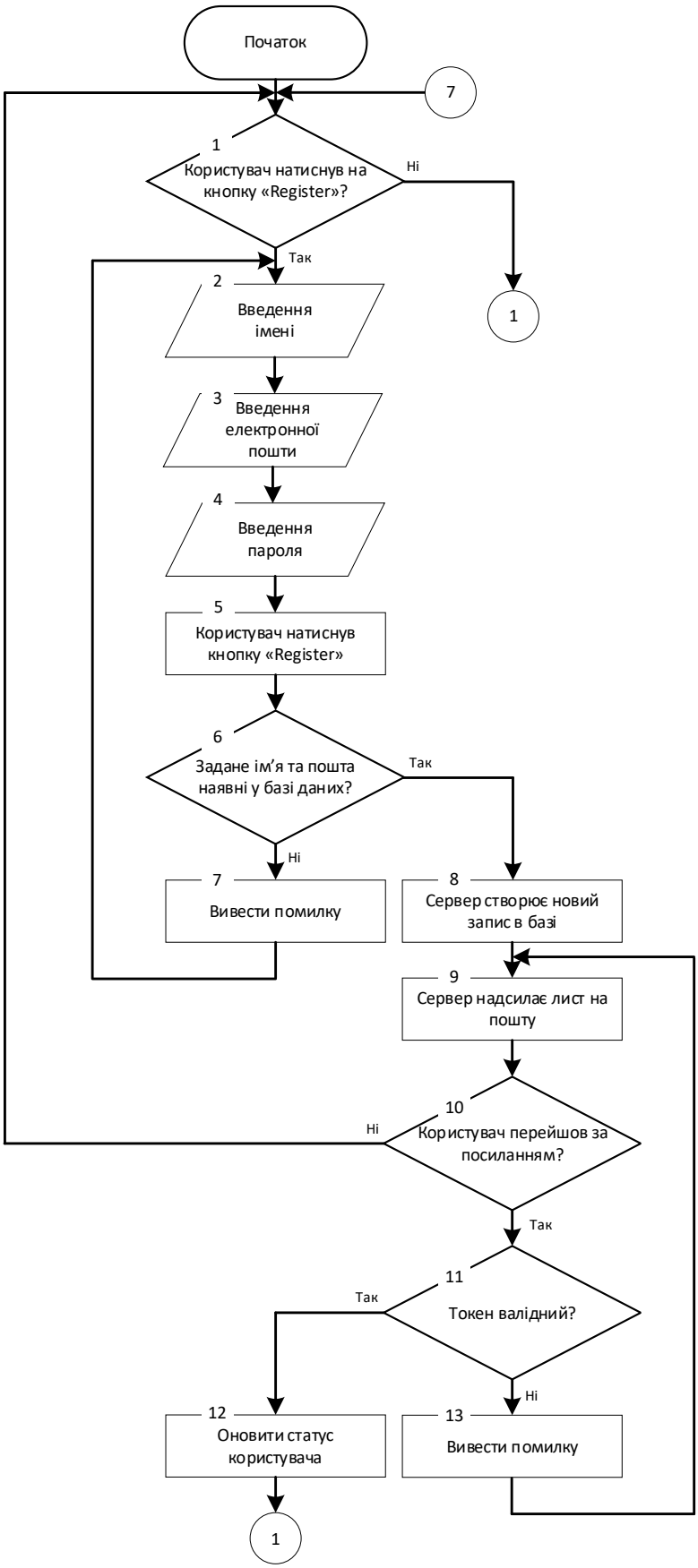

Рисунок В.8 – Схема алгоритму роботи програмного модуля WEB-ресурсу «Онлайн форум»

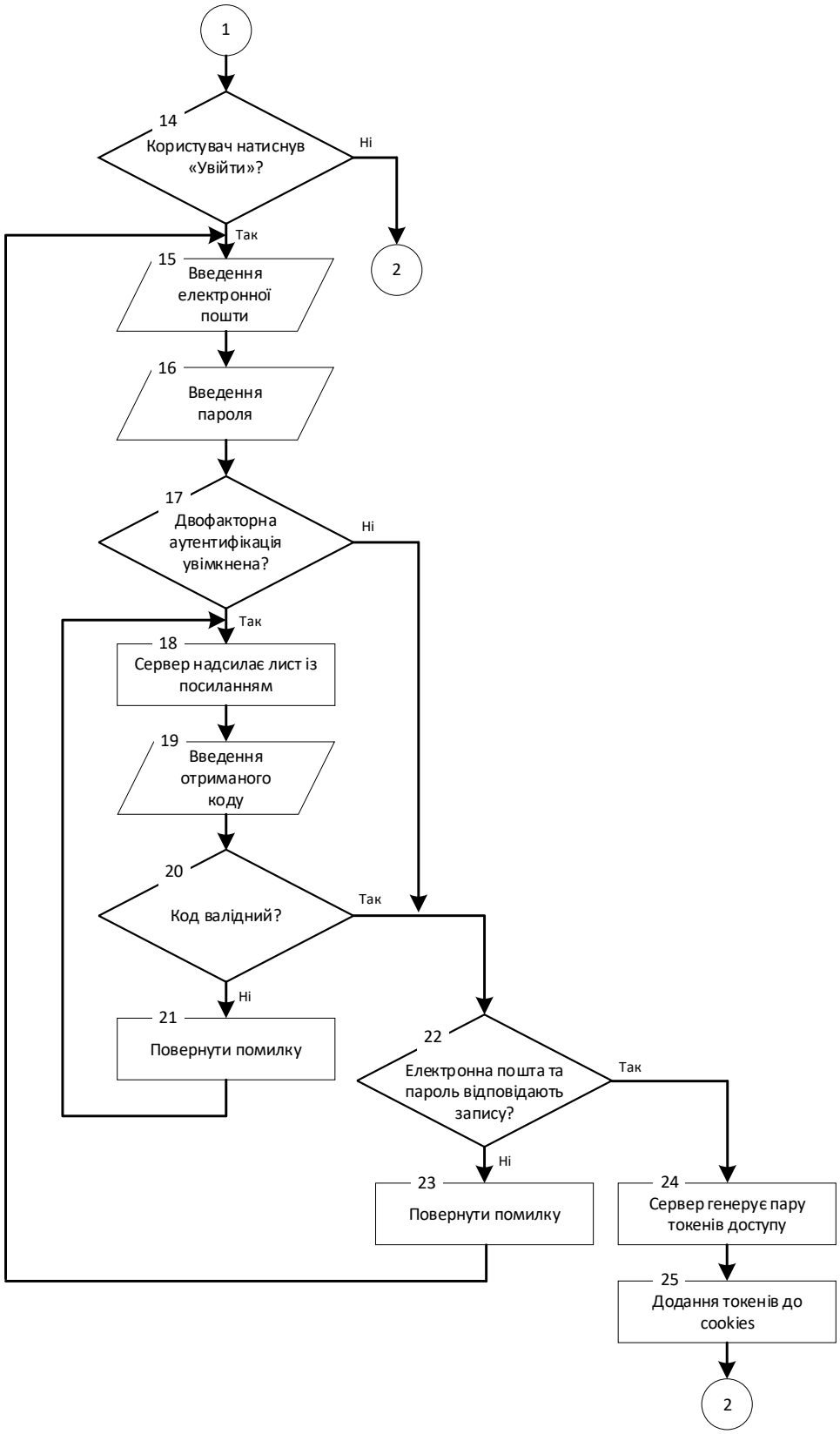

Рисунок В.8, аркуш 2

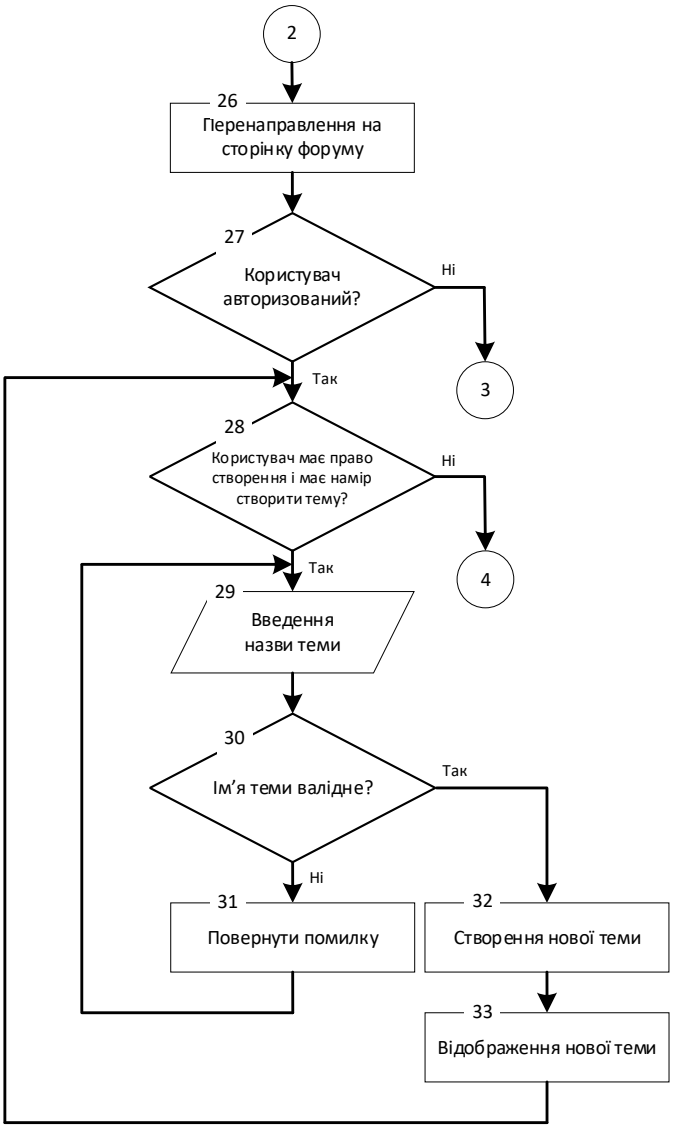

Рисунок В.8, аркуш 3

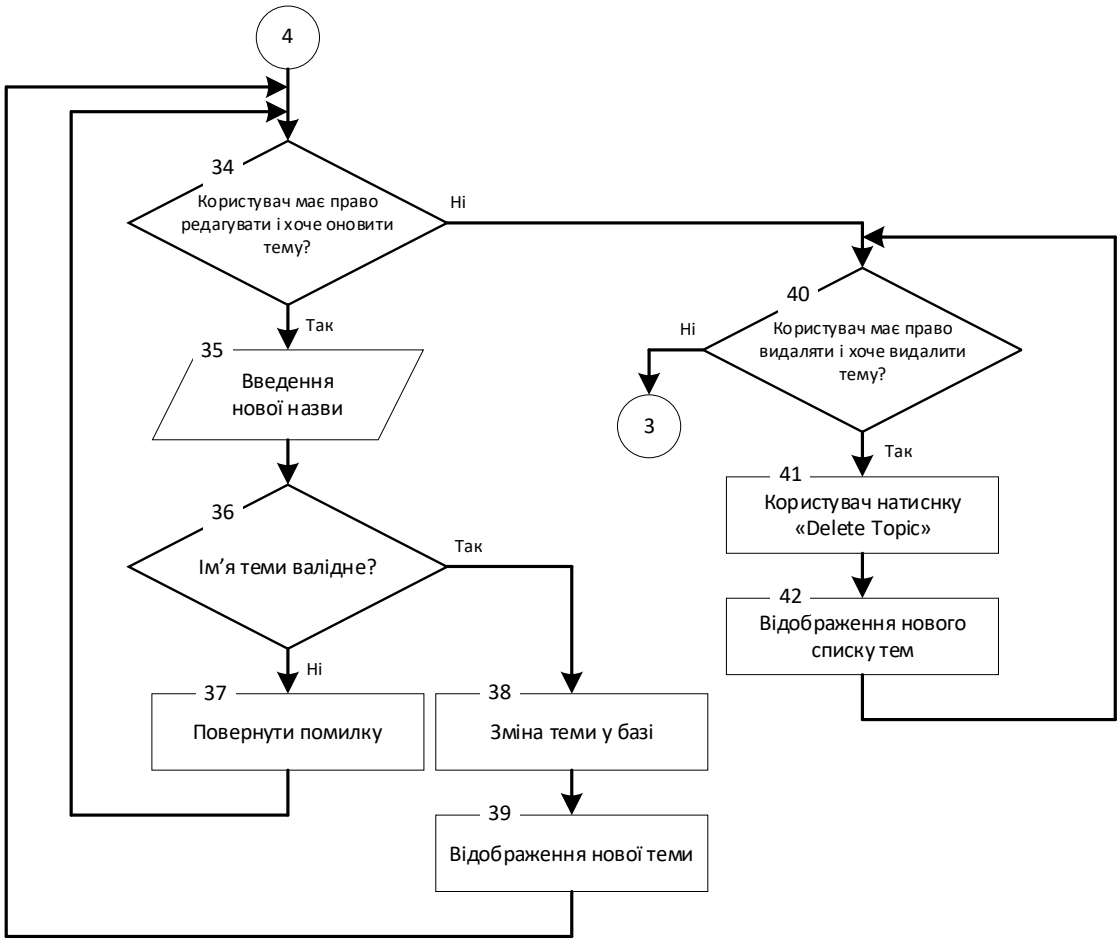

Рисунок В.8, аркуш 4

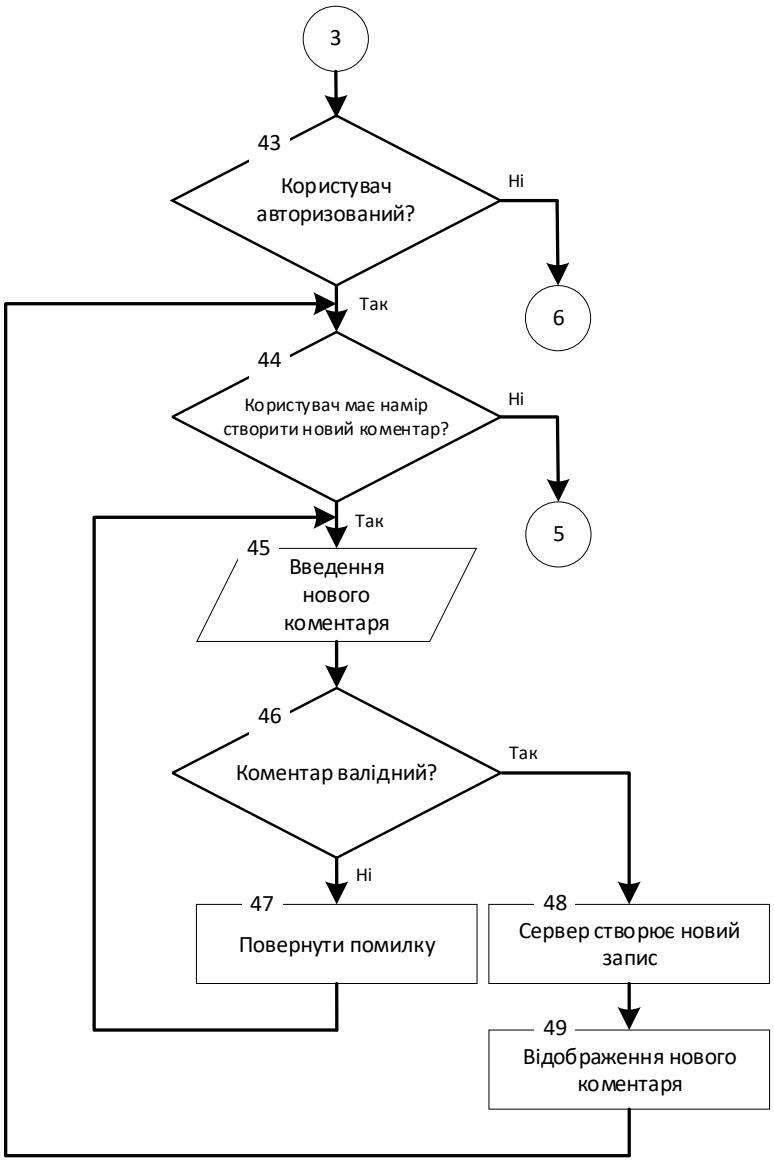

Рисунок В.8, аркуш 5

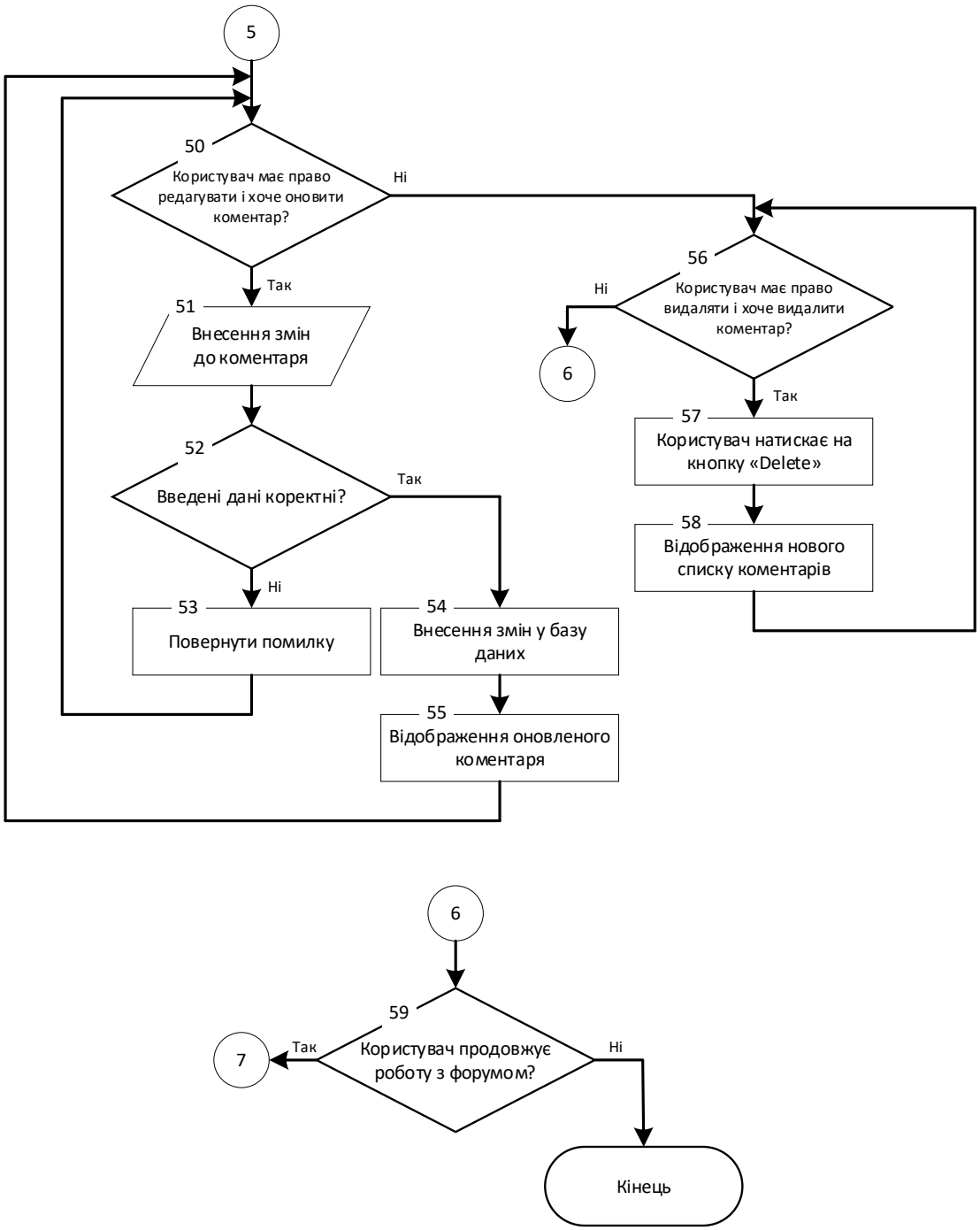

Рисунок В.8, аркуш 6

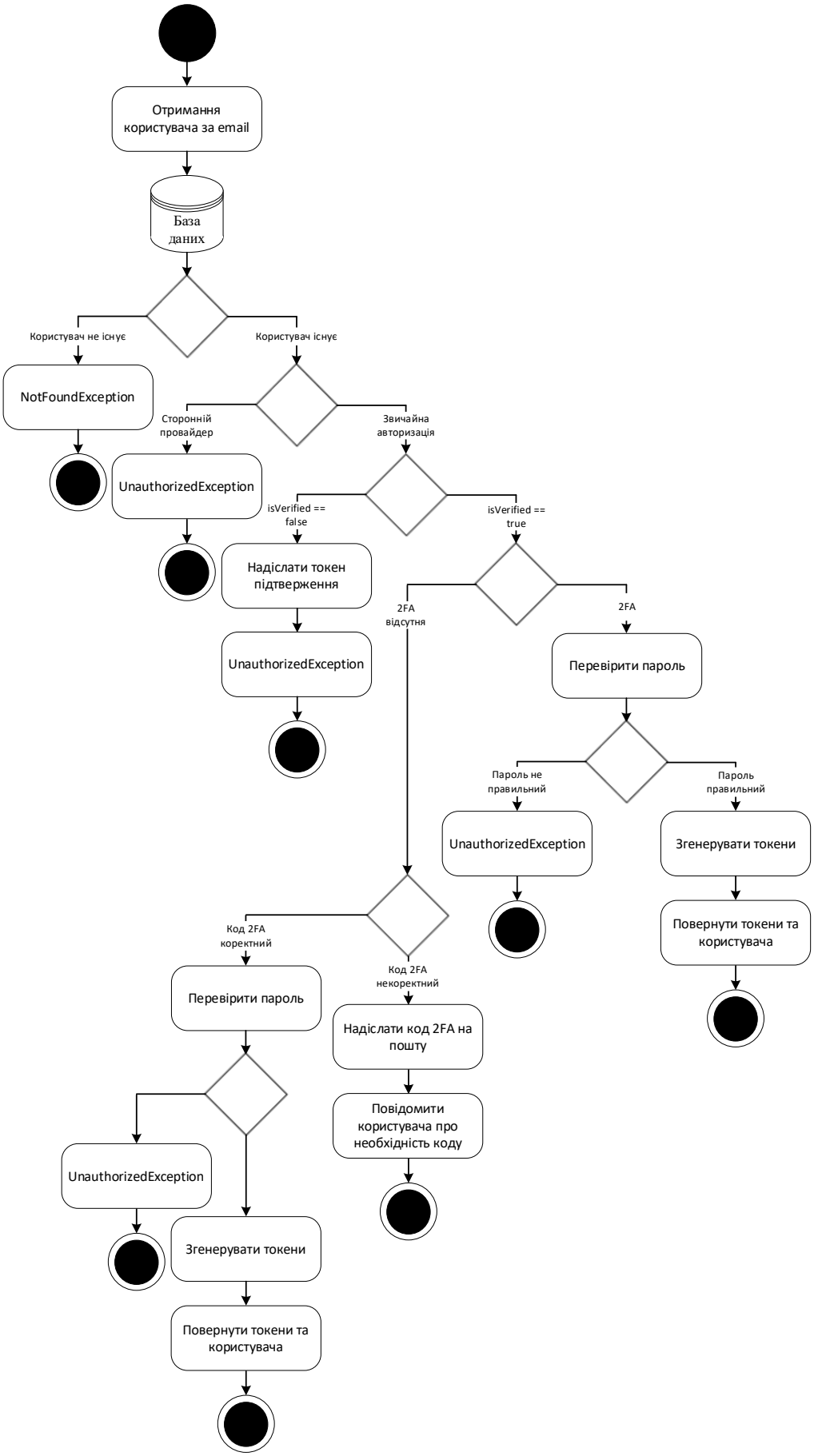

Рисунок В.9 – Діаграма діяльності для методу login сервісу AuthService

Рисунок В.10 – Загальний вигляд інтерфейсу головної сторінки

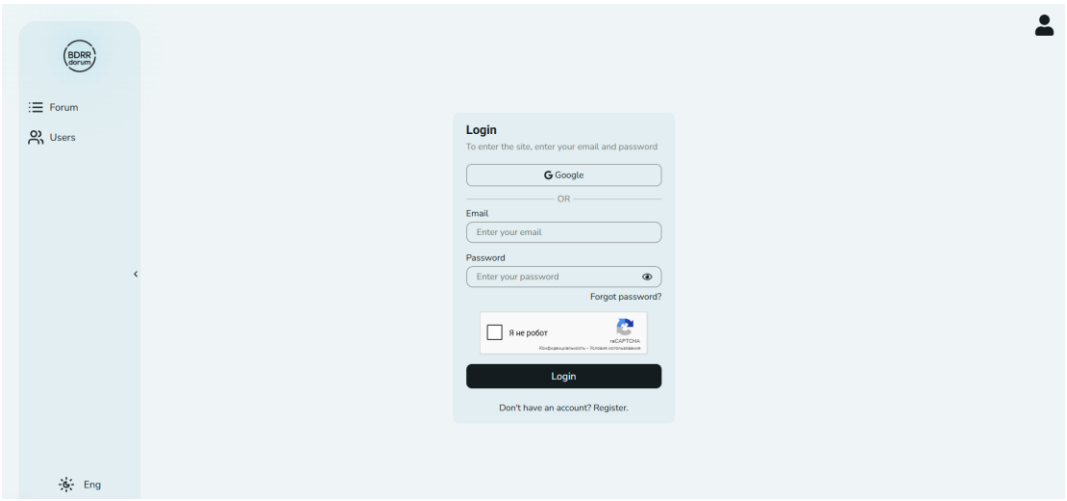

Рисунок В.11 – Інтерфейс сторінки авторизації користувачів на WEB-ресурсі

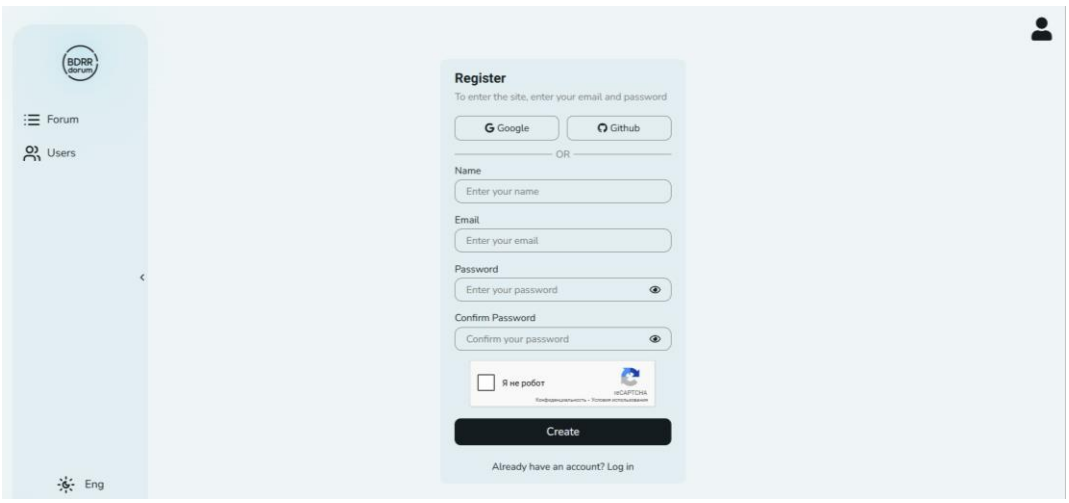

Рисунок В.12 – Інтерфейс сторінки реєстрації користувачів на WEB-ресурсі

‘

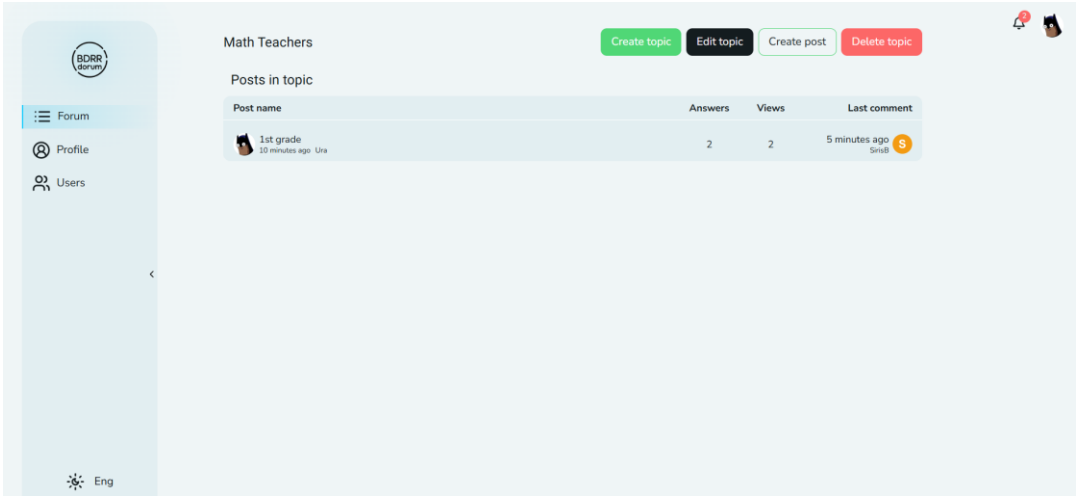

Рисунок В.13 – Інтерфейс сторінки теми «Math Teachers»

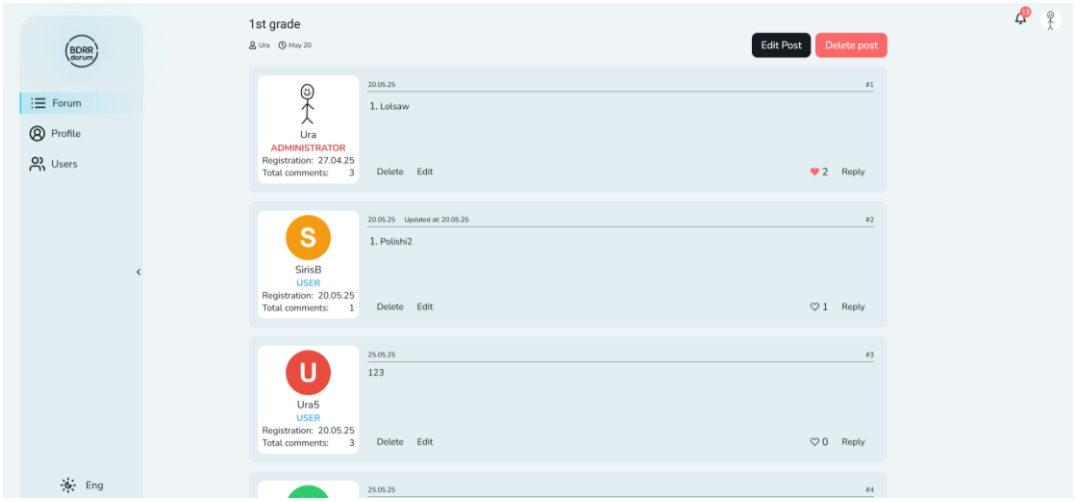

Рисунок В.14 – Інтерфейс сторінки поста, який розміщений у підтемі

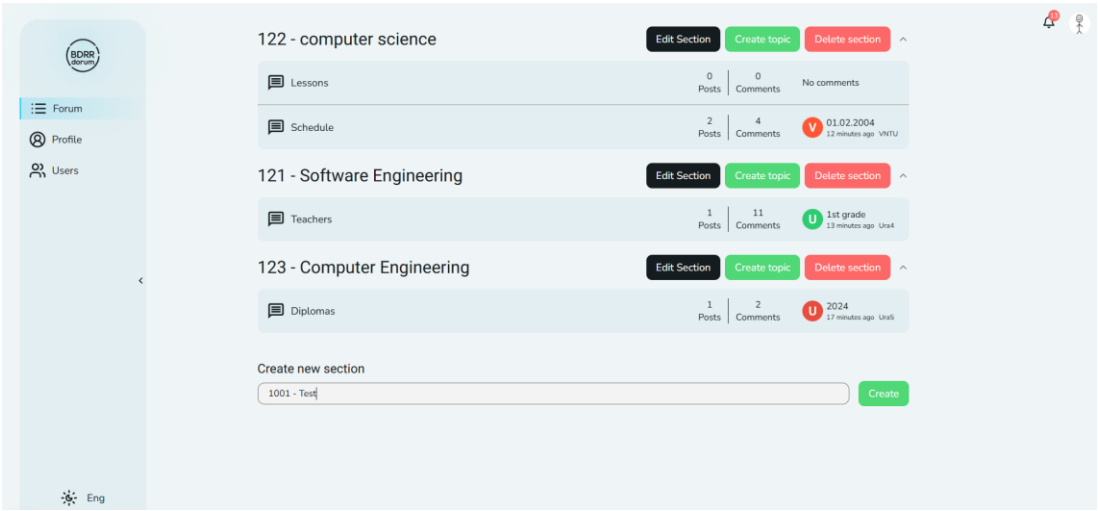

Рисунок В.15 – Форма створення нової секції

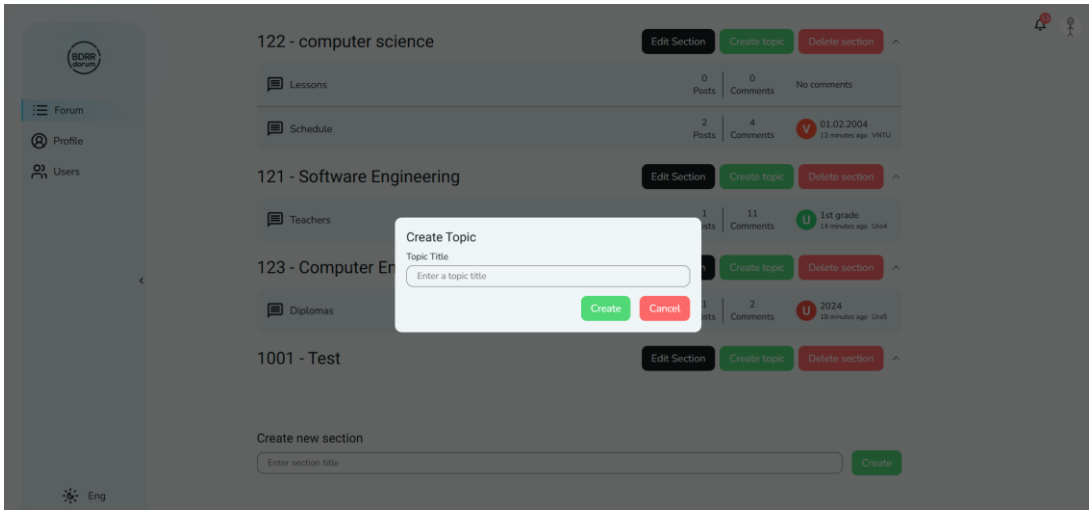

Рисунок В.16 – Модальне вікно створення нової теми

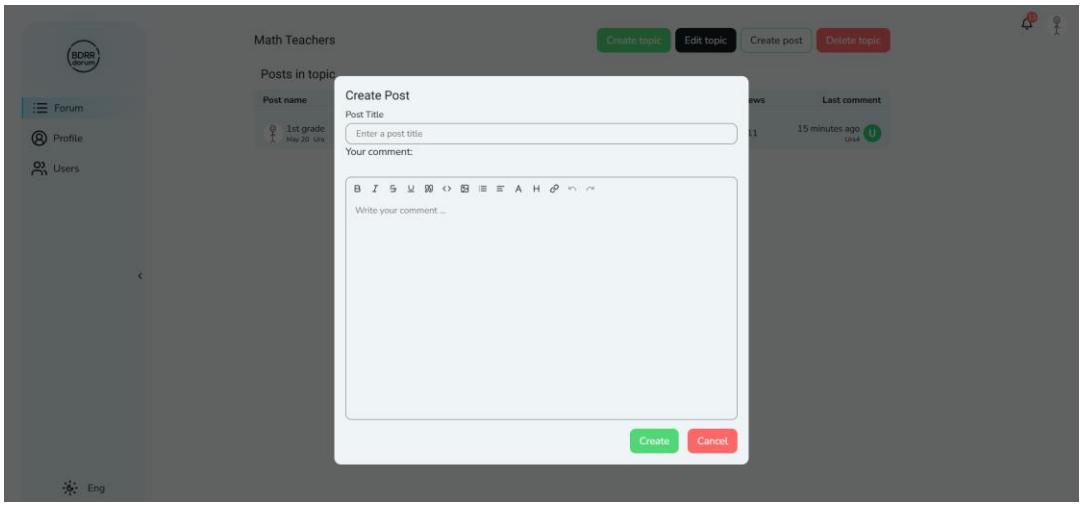

Рисунок В.17 – Модальне вікно створення нового поста

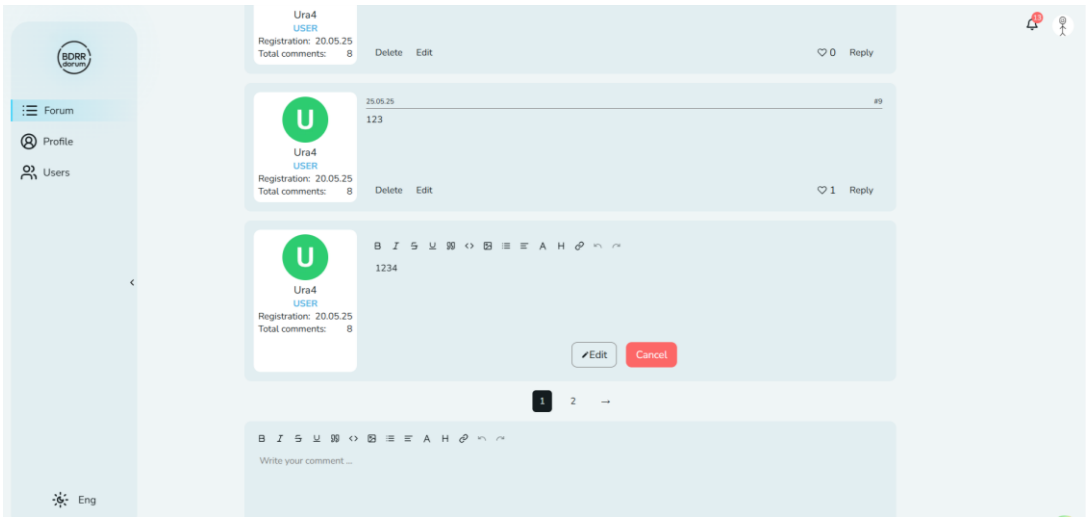

Рисунок В.18 – Процес редагування коментаря

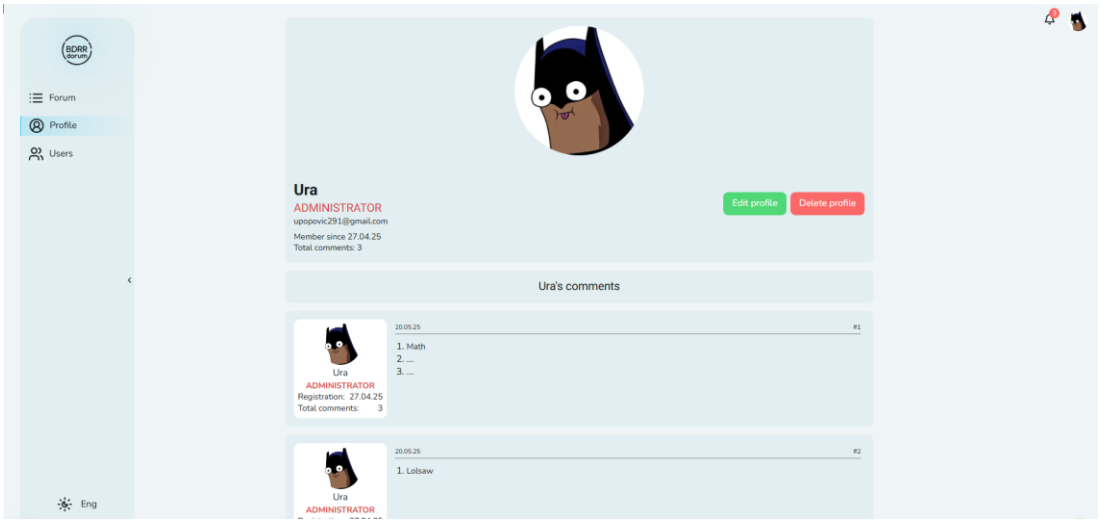

Рисунок В.19– Інтерфейс сторінки профілю користувача

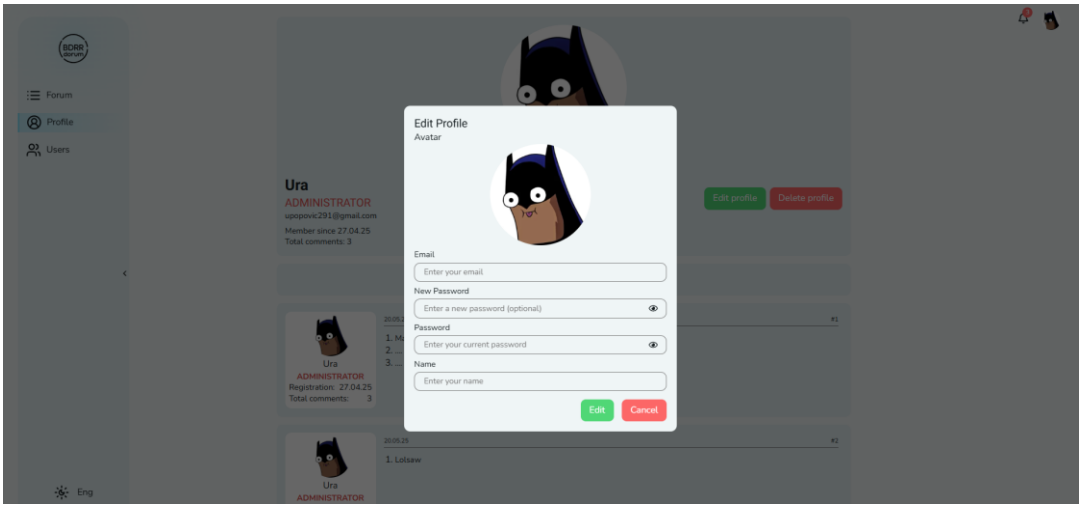

Рисунок В.20 – Модальне вікно для редагування профілю

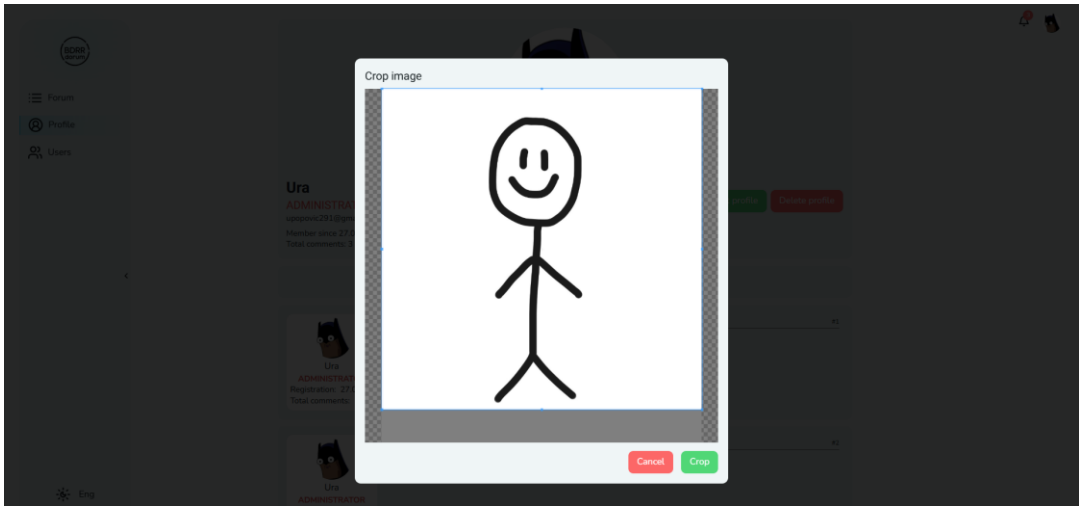

Рисунок В.21 – Модальне вікно для оновлення аватарки користувача

ДОДАТОК Г (довідниковий)

Інструкція користувача

Після запуску ресурсу з’являється головна сторінка, яка зображена на рисунку Г.1.

Рисунок Г.1 – Загальний вигляд інтерфейсу головної сторінки WEB-ресурсу «Онлайн форум»

Для відкриття функціональних можливостей користувачу необхідно увійти в аккаунт. Для цього потрібно перейти на сторінку авторизації (рис. Г.2) чи реєстрації (рис. Г.3).

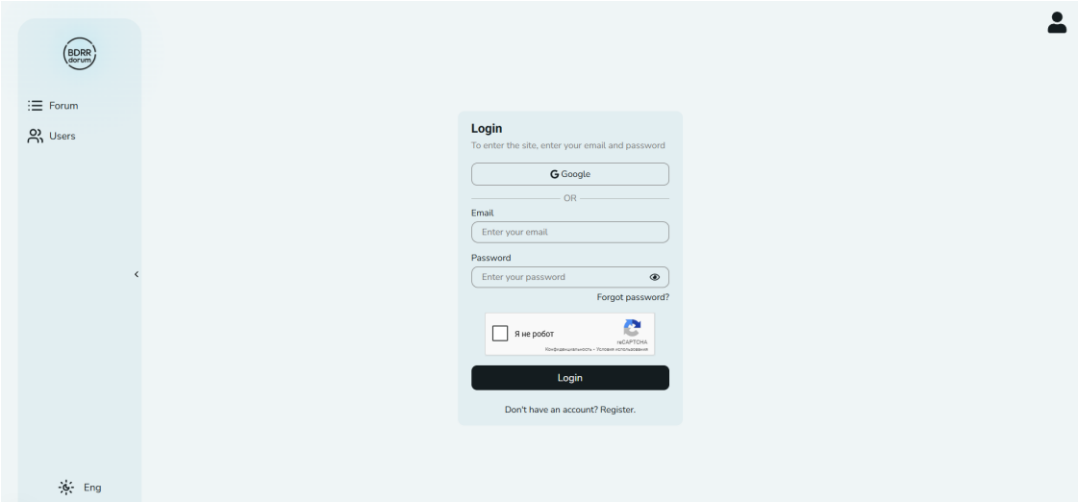

Рисунок Г.2 – Інтерфейс сторінки авторизації користувачів на WEB-ресурсі

‘

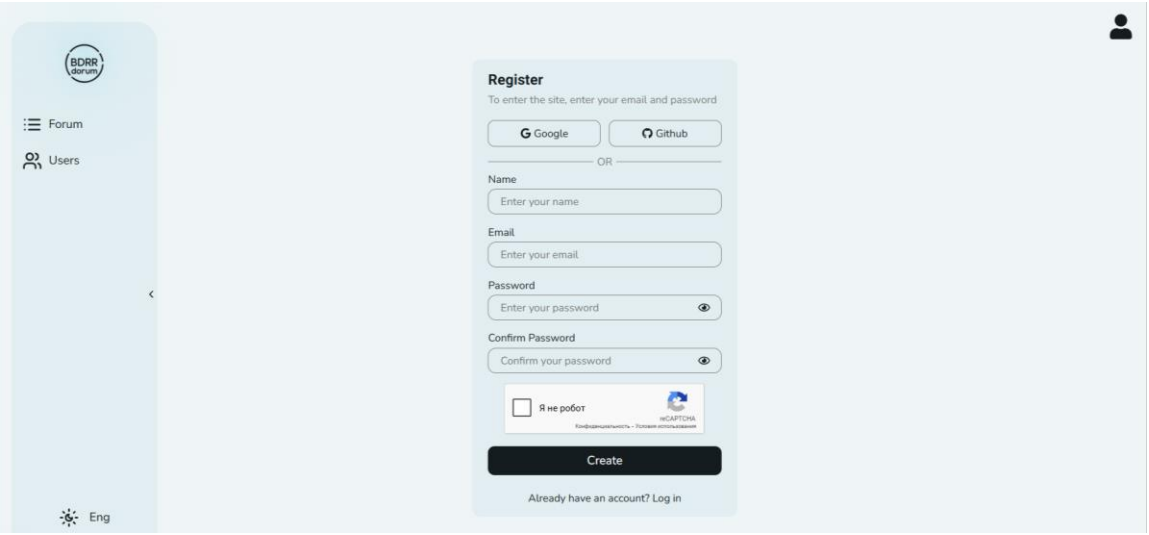

Рисунок Г.3 – Інтерфейс сторінки реєстрації користувачів

Якщо у користувача увімкнена двофакторна авторизація, то на його пошту надійде лист з кодом підтвердження. Цей код потрібно буде ввести у відповідне поле. Якщо користувачеві потрібно відновити пароль, то доцільно перейти на спеціальну сторінку, де потрібно ввести свою електронну адресу. На цю адресу буде надіслано код підтвердження. Після введення коду у відповідному полі з'явиться можливість встановити новий пароль. У разі успішного встановлення, користувача буде перенаправлено на головну сторінку ресурсу.

Користувач може перейти до будь-якої теми, натиснувши на неї. Після чого його направить на сторінку тем. Сторінка тем наведена на рисунку Г.4.

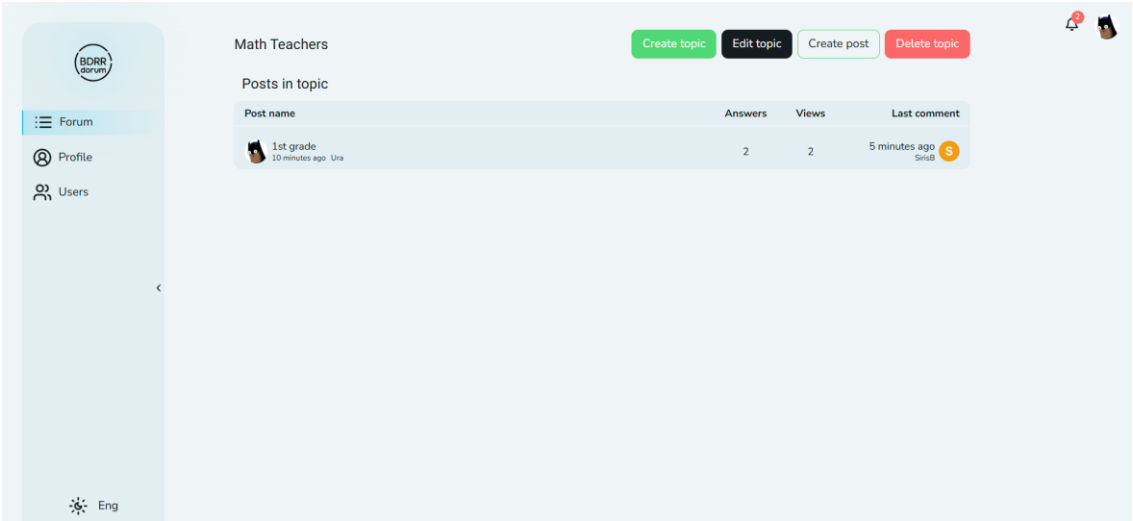

Рисунок Г.4 – Інтерфейс сторінки теми «Math Teachers»

‘

Користувач може перейти до будь-якого поста, натиснувши на нього. Після чого його направить на сторінку поста. Сторінка поста наведена на (рис. Г.5).

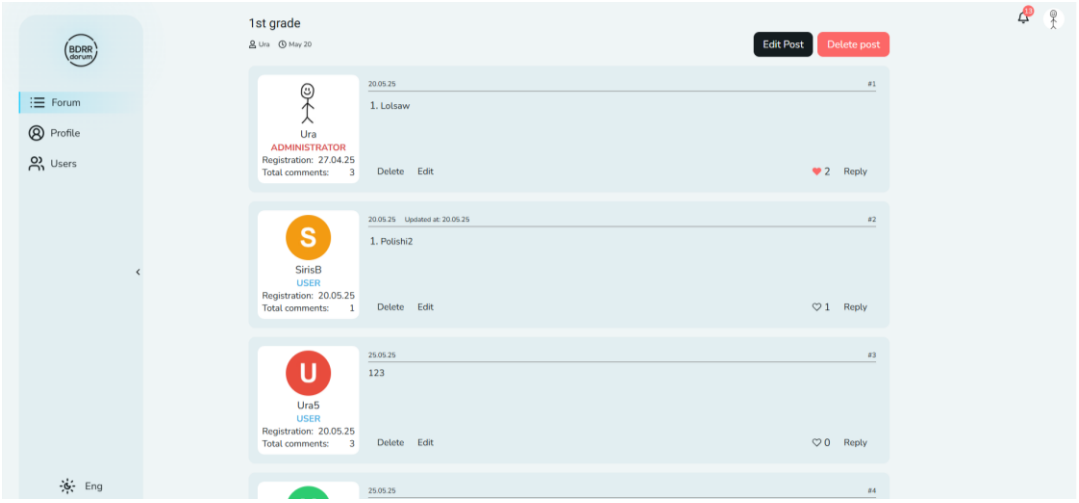

Рисунок Г.5 – Інтерфейс сторінки поста

Якщо користувач має роль з розширеним доступом, то він матиме змогу створювати секції. Форма для створення секції наведена на рисунку Г.6. У результаті успішного створення секції інтерфейс застосунку оновиться і буде містити нову секцію.

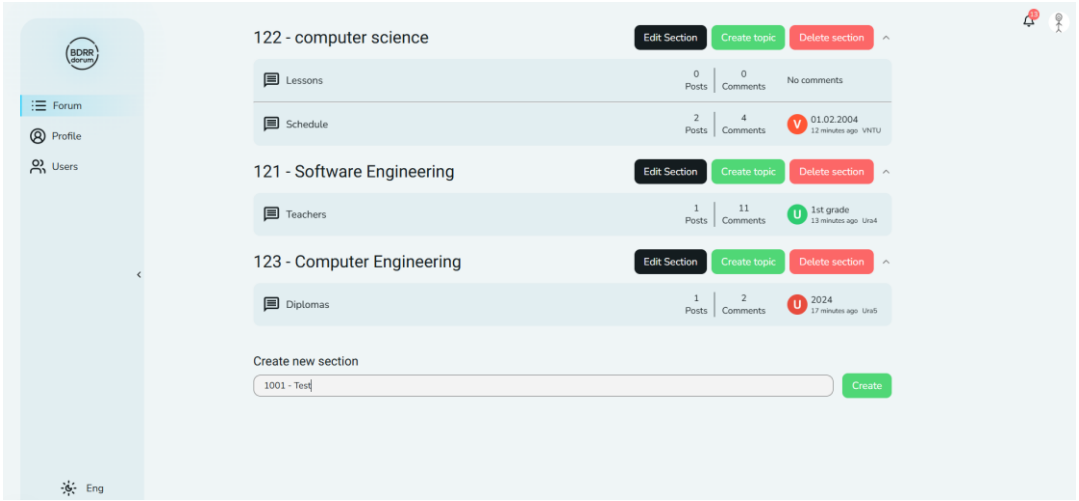

Рисунок Г.6 – Форма створення нової секції

‘

Щоб створити нову тему, потрібно натиснути у потрібному місці на кнопку «Create topic» та заповнити форму, яка буде надана. Форма створення нової теми наведена на рисунку Г.7.

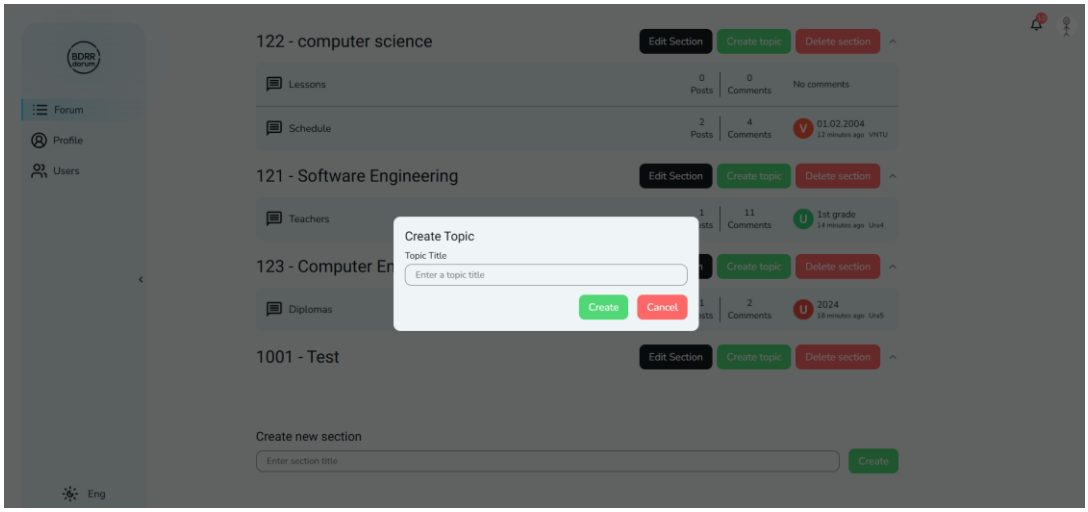

Рисунок Г.7 – Модальне вікно створення нової теми

Щоб створити новий пост, потрібно натиснути у потрібному місці на кнопку «Create post» та заповнити форму, що буде надано. Форма створення нового поста наведена на рисунку Г.8.

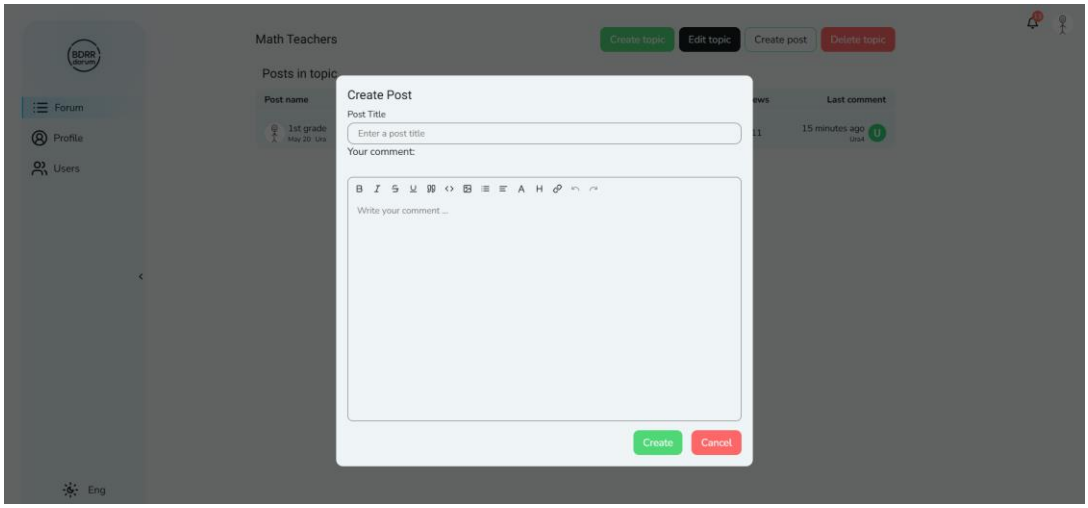

Рисунок Г.8 – Модальне вікно створення нового поста

‘

Щоб внести зміни до існуючого коментаря, потрібно натиснути на кнопку «Edit». У результаті буде оновлено карточку коментаря, щоб мати змогу внести зміни до коментаря. Форму оновлення коментаря наведено на рисунку Г.9.

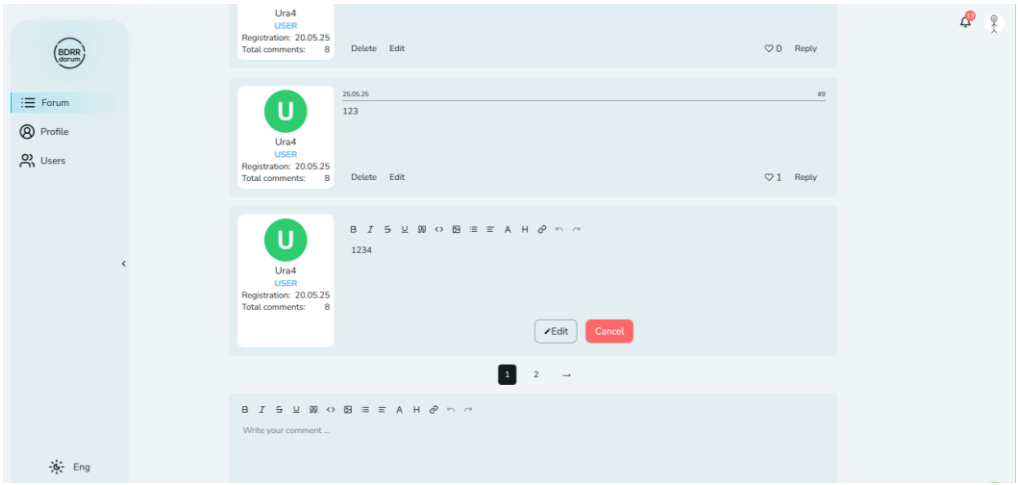

Рисунок Г.9 – Форма редагування коментаря

Щоб оновити профіль користувача, потрібно натиснути на кнопку «Edit Profile», а щоб овновити – «Delete Profile».

Для оновлення інформації у профілі користувача потрібно внести зміни у відповідні поля, виконати усі вказівки, якщо вони будуть наведені та натиснути на кнопку «Edit». Модальне вікно редагування профіля наведена на рисунку Г.10.

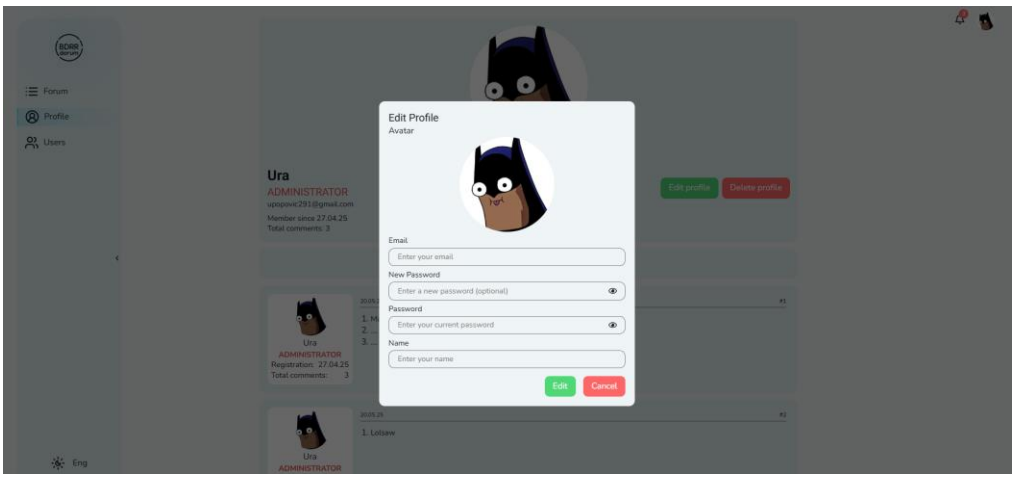

Рисунок Г.10 – Модальне вікно для редагування профілю

‘

Щоб оновити зображення користувача, потрібно натиснути на відповідний аватар, який розміщений у модальному вікні редагування (рис. Г.10), після чого потрібно обрати на персональному пристрої потрібне зображення. У результаті буде надано змогу обрізати зображення. Модальне вікно для оновлення аватару наведено на рисунку Г.11.

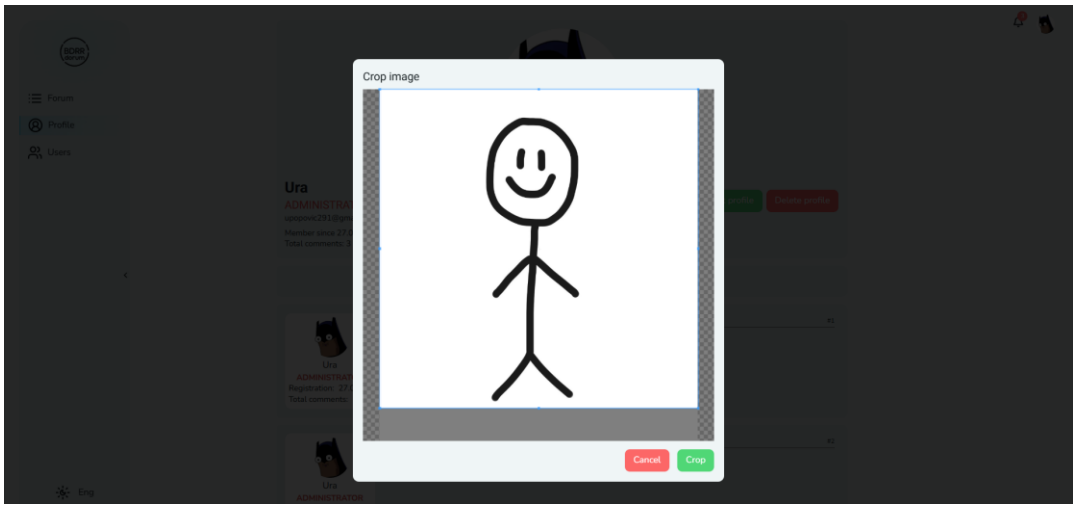

Рисунок Г.11 – Модальне вікно для оновлення аватарки користувача

ДОДАТОК Д

Свідоцтво про реєстрацію авторського права на твір

УКРАЇНА

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 135891

Комп'ютерна програма «WEB-ресурс «Онлайн-форум»

(вид, назва твору)

Автор (співавтори) Крилик Людмила Вікторівна, Попович Юрій Юрійович

(прізвище, ім'я, по батькові (за наявності), псевдонім (за наявності))

Авторські майнові права належать спільно Крилик Людмила Вікторівна, вул. Політехнічна, 3, кв. 314, м. Вінниця, 21021; Попович Юрій Юрійович, вул. Космонавтів, 54/4, м. Козятин, Вінницька обл., 22102

(прізвище, ім'я, по батькові (за наявності) фізичної особи / найменування юридичної особи, адреса)

Дата реєстрації 7 травня 2025 р.

Директор Державної організації «Український національний офіс інтелектуальної власності та інновацій» Олена ОРЛЮК

УКРНОІВІ