**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ**
**Кафедра програмної інженерії**

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБФОРУМУ ПРИХИЛЬНИКІВ КОМП'ЮТЕРНИХ ІГОР»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
О.В. Свіркунов
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Лимаренко Ю.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри електроніки, інформаційних систем та програмного забезпечення ІННІ ЗНУ, доцент, к.т.н. Заяц В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

**ЗАТВЕРДЖУЮ**
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

# З А В Д А Н Н Я

## НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Свіркунову Олегу Віталійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебфоруму прихильників комп’ютерних ігор

керівник роботи Лимаренко Юлія Олексіївна, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка вебфоруму.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 26.12.2024 р.

## К А Л Е Н Д А Р Н И Й  П Л А Н

| № | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Розробка плану роботи. | 08.01.2025 | |
| | | | |
| 2. | Збір вихідних даних. | 13.01.2025 | |
| | | | |
| 3. | Обробка методичних та теоретичних | 28.01.2025 | |
| | джерел. | | |
| | | | |
| 4. | Розробка першого та другого розділу. | 03.04.2025 | |
| | | | |
| 5. | Розробка третього розділу. | 12.05.2025 | |
| | | | |
| 6. | Оформлення та нормоконтроль | 26.05.2025 | |
| | кваліфікаційної роботи бакалавра. | | |
| | | | |
| 7. | Захист кваліфікаційної роботи. | 19.06.2025 | |

Студент _____ О.В. Свіркунов
(підпис) (ініціали та прізвище)

Керівник роботи _____ Ю.О. Лимаренко
(підпис) (ініціали та прізвище)

## Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

# РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебфоруму прихильників комп’ютерних ігор»: 51 с., 28 рис., 9 джерел.

БЕКЕНД, ВЕБФОРУМ, ФРОНТЕНД, JAVASCRIPT, MYSQL, PHP.

Об’єкт дослідження – розробка тематичного вебфоруму для спілкування і зв’язку користувачів за тематикою комп’ютерних ігор.

Мета роботи: розробка зручного, функціонального та сучасного вебфоруму, що забезпечує комфортну та ефективну комунікацію між користувачами, які цікавляться тематикою комп’ютерних ігор.

Метод дослідження – методи програмної інженерії.

У кваліфікаційній роботі було наведено аналіз предметної області, огляд існуючих форумів та описано шлях створення проєкту за темою кваліфікаційної роботи. Результатом виконання кваліфікаційної роботи є вебфорум, розроблений з використанням сучасних методів веброзробки, мови програмування PHP, бібліотеки JavaScript, та системи керування базами даних MariaDB.

# SUMMARY

Bachelor’s qualifying paper “Development of a Web Forum for Computer Game Fans”: 51 pages, 28 figures, 9 references.

BACKEND, WEBFORUM, FRONTEND, JAVASCRIPT, MYSQL, PHP.

The object of the study is the development of a thematic web forum for communication and interaction among users interested in computer games.

The aim of the study is to develop a functional, and modern web forum that ensures comfortable and effective communication between users interested in computer games.

The method of research is software engineering methods.

The bachelor’s qualifying paper presents an analysis of the subject area, an overview of existing forums, and a description of the process of creating the project according to the topic of the qualifying paper.

The result of the work is a web forum developed using modern web development methods, the PHP programming language, the JavaScript library, and the MariaDB database management system.

**ЗМІСТ**

Завдання на кваліфікаційну роботу ..... 2
Реферат ..... 4
Summary ..... 5
Вступ ..... 8
1 Аналіз предметної області ..... 9
1.1 Характеристика предметної області ..... 9
1.2 Огляд існуючих вебфорумів ..... 10
1.2.1 Форум Reddit ..... 10
1.2.2 Steam Community Discussions ..... 11
1.2.3 GameSpot Forums ..... 12
1.2.4 Discord-сервери з елементами форуму ..... 12
1.3 Стек технологій ..... 13
1.3.1 Логіка предметної області ..... 13
1.3.2 База даних ..... 15
1.3.3 Користувацький інтерфейс ..... 15
1.4 Огляд середовища XAMPP ..... 16
1.4.1 Огляд XAMPP як інтегрованого середовища ..... 17
1.4.2 Використання XAMPP у контексті локальної розробки ..... 17
1.5 Висновки до розділу 1 ..... 18
2 Розробка вебфоруму ..... 19
2.1 Формування технічного завдання ..... 19
2.1.1 Список зацікавлених осіб та їх опитування ..... 19
2.1.2 Заповнення документу «Опис постановки задачі» ..... 20
2.2 Побудова діаграми варіантів використання ..... 22
2.3 Побудова діаграми послідовності ..... 26
2.4 Схема бази даних ..... 27
2.5 Висновки до розділу 2 ..... 29

3 Програмна реалізація ..... 30
3.1 Ініціалізація та налаштування проєкту ..... 30
3.2 Створення бази даних ..... 31
3.3 Програмна реалізація клієнтської частини ..... 34
3.4 Реалізація створення, редагування та видалення коментарів ..... 42
3.5 Висновки до розділу 3 ..... 48
Висновки ..... 50
Перелік посилань ..... 51

# ВСТУП

Розвиток сучасних інформаційних технологій та глобалізація інтернет-простору суттєво змінили способи взаємодії людей у різних сферах життя. Однією з найпопулярніших форм спілкування в Інтернеті є тематичні вебфоруми, які дозволяють обмінюватися думками, досвідом та інформацією між однодумцями. Серед таких платформ мене зацікавила тематика вебфоруму, присвячена комп'ютерним іграм.

Комп'ютерні ігри є важливим культурним та соціальним феноменом сьогодення. Вони об'єднують мільйони людей по всьому світу, формуючи численні ігрові спільноти, які активно обговорюють нові релізи, ігрові механіки, тактики, стратегії, новини індустрії та багато іншого. Саме тому виникає необхідність створення спеціалізованих платформ для організації ефективної комунікації та взаємодії серед прихильників комп'ютерних ігор.

У першому розділі кваліфікаційної роботи міститься характеристика предметної області, наведено приклади реалізації вебфорумів різної тематики, описаний їхній функціонал. Також описано стек технологій, за яким відбувається розробка вебфоруму.

У другому розділі описано технічне завдання, призначення розробки та технічні вимоги до проєкту. Наведено діаграми варіантів використання для різних груп користувачів, діаграми послідовності для розроблених функцій, ER-діаграму бази даних, описані етапи створення вебфоруму.

У третьому розділі зазначено реалізацію проєкту, включаючи клієнтську та серверну частини, розглянуто методи реалізації форуму, зокрема приділено увагу головній сторінці, спільній для сторінок шапки, та системі взаємодії з коментарями.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Характеристика предметної області

Вебфорум (Internet forum) представляє собою інтерактивну платформу в мережі Інтернет, призначену для організації тематичних обговорень, обміну інформацією та соціальної взаємодії між користувачами. Сучасні вебфоруми характеризуються широким спектром інструментів, що оптимізують комунікацію та систематизують контент, включаючи категорії, підкатегорії, теги, систему коментування, приватні повідомлення, а також додаткові функції для модерації та адміністрування.

Ключовою особливістю вебфорумів є структурована форма обговорень, що відрізняє їх від соціальних мереж чи чатів. На відміну від нескінченної стрічки повідомлень, вебфоруми організовують матеріали за розділами (наприклад, «Загальні обговорення», «Відеоігрові новини») або за тегами. Це дозволяє користувачам швидко знаходити актуальні теми та обговорення, поглиблено обговорювати конкретні питання та повертатися до наявної інформації навіть через тривалий час.

Управління обліковими записами та профілями є важливою складовою вебфорумів. Користувачі мають можливість реєструватися, створювати персональні сторінки з інформацією про себе, власними аватарами чи підписами. Цей функціонал сприяє формуванню сталої спільноти, де кожен учасник набуває власної ідентичності. Багато форумів впроваджують систему рейтингів (наприклад, кількість повідомлень, «лайки» чи «подяки»), що дозволяє відзначити внесок найактивніших чи найкорисніших учасників.

Взаємодія з інформацією на форумі зазвичай реалізується через публікацію повідомлень (постів) у вигляді тексту, зображень або посилань. Кожна тема (тред) містить ланцюжок таких повідомлень, упорядкованих за датою чи рейтингом. Вебфоруми можуть забезпечувати додаткові можливості,

такі як прикріплення файлів, вбудовані редактори з підтримкою BBCode чи Markdown, опитування, а також приватні та групові повідомлення.

Для багатьох інтернет-користувачів вебфорум є ключовим інструментом отримання та поширення знань, пошуку однодумців та обговорення найрізноманітніших питань. Завдяки довготривалій доступності матеріалів та зручному механізму пошуку, вебфорум дозволяє формувати «колективну пам'ять» спільноти. Новачки мають можливість знаходити відповіді на типові запитання без необхідності створювати повторні теми, а досвідчені учасники можуть вести детальні дискусії з великою кількістю посилань та прикріплених візуальних матеріалів.

Вебфоруми є універсальною платформою для створення тематичних спільнот з акцентом на тривале та впорядковане обговорення. У контексті ігрової тематики вебфорум дозволяє геймерам та зацікавленим особам не лише оперативно ділитися досвідом чи отримувати новини, а й підтримувати активну комунікацію, що сприяє розвитку ігрового середовища у цілому.

## 1.2 Огляд існуючих вебфорумів

У цьому підрозділі ми розглянемо найпоширеніші вебфоруми, що забезпечують взаємодію та обговорення між користувачами. Також буде проаналізовано ключові особливості, переваги, недоліки та можливості розширення їх функціоналу.

### 1.2.1 Форум Reddit

Reddit – це одна з найбільш популярних у світі платформ для обговорень, що містить тисячі підфорумів (сабредитів) на будь-яку тему, зокрема і комп’ютерні ігри. Для геймерів найбільш відомим є розділ r/gaming, але існують і спеціалізовані підрозділи, такі як r/pcgaming**,** r/truegaming**,** r/gamedev.

Основні можливості та особливості полягають у механізмі голосування за контент (upvote/downvote), що дозволяє користувачам піднімати популярні пости і «топити» менш цікаві, можливістю публікації як текстових, так і мультимедійних постів (зображень, відео, посилань), акценті на швидкоплинному контенті: за лічені години новий пост може набрати тисячі голосів і стати «гарячою темою», однак з часом він зникає з перших сторінок.

Перевагами можна назвати інтегровану система пошуку за сабредитами та ключовими словами та зручну для користувачів мобільну версію та офіційні додатки. До недоліків слід віднести складність структурованого зберігання даних (теми швидко змінюються та губляться в потоці нових постів) та відсутність класичної деревовидної структури форуму, що ускладнює довготривалі обговорення.

### 1.2.2 Steam Community Discussions

Steam Community – офіційний форум платформи Steam, що створює окремі розділи для кожної гри, доступної в сервісі Valve. Тут гравці можуть спілкуватися щодо конкретного тайтлу, ставити технічні питання, обговорювати нові оновлення або шукати компанію для мультиплеєра.

Його основні можливості полягають у автоматичній прив’язці до ігрової бібліотеки (користувачі бачать розділи тих ігор, якими володіють у Steam), офіційний темах розробників (студії самостійно публікують новини, патчноути й отримують зворотний зв’язок) та пошуку і сортуванні за популярністю, датою або тегами.

Серед переваг слід виділити повну інтеграцію з ігровим профілем, можливість перегляду ігрової статистики користувача, зручну система маркування питань як «вирішено», що особливо корисно для технічної підтримки, та високий рівень безпеки та захисту від спаму завдяки Steam-акаунтам.

Але є і недоліки: обговорення дуже роздроблені за конкретними іграми, міститься дублювання одних і тих самих обговорень через недостатньо розвинену індексацію, та спрямованість виключно на ігри зі Steam.

### 1.2.3 GameSpot Forums

GameSpot – великий ігровий портал, що поєднує новини, рецензії, відео та власні форуми. Тематично охоплює консольні та комп'ютерні ігри, технології, залізо, кіберспорт і багато іншого.

Серед основних можливостей міститься категоризація за платформами (PC, PlayStation, Xbox, Nintendo) та жанрами ігор (RPG, FPS, Adventure), містяться розділи для обговорення оглядів, вивчення гайдів, участі в опитуваннях, користувачі можуть коментувати статті основного сайту й продовжувати обговорення на форумі.

Серед переваг мається велика база користувачів, що дає змогу отримати швидку відповідь та експертну допомогу, розгалужена модерація, яка підтримує порядок та заохочує аналітичні обговорення, та наявність офіційних тем від редакції GameSpot, які часто висвітлюють головні новини індустрії. Недоліками є надмірна кількість розділів, через що деякі тематичні гілки стають малопомітними та рекламний контент на сторінках, який відволікає від обговорень.

### 1.2.4 Discord-сервери з елементами форуму

Discord не є класичним вебфорумом, але він заслуговує уваги, так як у ньому є гілки для обговорень з можливістю розміщення файлів та зображень, та тематика як самого Discord, так і його наповнення, переважно належить відеоігровій тематиці.

Основні можливості серверу Discord полягають у миттєвій комунікації в реальному часі, як у текстовому, так і у відео/аудіо форматі. Discord дає змогу створювати канали за темами й підкатегоріями, а також “Threads” – короткочасні чи довготривалі гілки, присвячені конкретному питанню в межах каналу. Ця функція нагадує форумні треди, проте реалізована у форматі чат-інтерфейсу. Має підтримку обміну файлами та посиланнями (зображення, відео, документи), інтеграцію ботів для відстежування статистики, автоматичної модерації, вітанню новачків та інших завдань.

Адміністратори сервера можуть створювати розгалужену ієрархію прав: від модераторів, які стежать за порядком, до спеціальних ролей із доступом до закритих каналів. Це дозволяє тримати організований простір навіть на дуже великому сервері.

Завдяки офіційним додаткам для Android та iOS користувачі отримують оперативні сповіщення про нові повідомлення чи треди, що сприяє активнішому обговоренню та швидшій взаємодії.

## 1.3 Стек технологій

У цьому розділі ми розглянемо основні технології, що використовуватимуться для розробки вебфоруму. Зокрема, буде показано, як ми обрали інструменти для фронтенду, бекенду та бази даних, а також обґрунтовано це рішення з точки зору продуктивності, зручності розробки та сумісності з потребами форуму.

### 1.3.1 Логіка предметної області

При створенні вебфоруму основна частина логіки предметної області зосереджена на таких ключових аспектах:

– керуванні обліковими записами (реєстрація, авторизація);

– організації структури обговорень (тем, розділів, підрозділів);
– системі ролей та прав (адміністратори, модератори, звичайні користувачі);
– підтримці інструментів для модерації контенту (видалення або редагування небажаних постів).

Для реалізації логіки предметної області було обрано мову PHP, яка є однією з найпоширеніших мов для розробки вебзастосунків та забезпечує широкий набір готових бібліотек, пакетів і активну спільноту розробників.

PHP демонструє високу ефективність взаємодії з MySQL/MariaDB через розширення PDO та MySQL, що ідеально підходить для зберігання та обробки даних форуму, таких як користувачі, пости та теми. Крім того, відносна простота освоєння цієї мови значно полегшує командну розробку й прискорює створення базової функціональності.

Основні завдання логіки предметної області полягають в:

– управлінні користувачами (створення облікових записів, вхід, вихід, відновлення паролю, перевірка електронної пошти та інші операції, пов’язані з профілем);
– модерації та адмініструванні (реалізація ролей і прав, захист від спаму, надання можливості банити порушників, закривати чи закріплювати теми, редагувати пости);
– структурі форуму (створення ієрархії розділів наприклад, «Загальні обговорення», «Оголошення», «Технічна допомога», управління тегами для контенту);
– системі сповіщень (інформування користувачів про відповіді на їхні пости, згадки, особисті повідомлення), безпеці (перевірка автентичності запитів, захист від CSRF, SQL-ін’єкцій, XSS).

PHP підходить для реалізації вищевказаних завдань предметної області завдяки вбудованій підтримці сесій, форм та обробки запитів, що дозволяє легко керувати користувачами, автентифікацією й авторизацією. Його гнучка взаємодія з базами даних через PDO або MySQL забезпечує збереження структури форуму, повідомлень, тегів і дозволяє реалізувати контроль ролей.

Серед переваг PHP слід виділити велике різноманіття фреймворків: Laravel, Symfony, CodeIgniter дає змогу обрати потрібний ступінь абстракції та швидкості розробки. PHP має просту інтеграцію з фронтендом завдяки вбудованим шаблонизаторам Blade, Twig, або іншим механізмам генерації HTML, швидку розробку базових CRUD-операцій для постів, коментарів і профілів [1].

### 1.3.2 База даних

Форум потребує надійного сховища для даних про користувачів, пости, коментарі та іншу інформацію. Для цієї мети було обрано MariaDB, що є форком MySQL і відзначається високою сумісністю з нею. У контексті форумних сценаріїв реляційна база даних (таблична модель) є оптимальним рішенням: теми і пости мають чітку структуру й логічно пов’язані між собою. MariaDB забезпечує високу швидкість CRUD-запитів за умови правильної побудови індексів [2].

MariaDB має значну схожість з MySQL за рахунок подібності синтаксису і внутрішніх механізмів, що суттєво спрощує процеси перенесення чи інтеграції з готовими скриптами і прикладами для MySQL. Це дозволяє використовувати широку базу знань та готові рішення, що існують для MySQL, що значно прискорює розробку та підвищує надійність системи [3].

### 1.3.3 Користувацький інтерфейс

Форум має бути зручним та інтуїтивним для широкого кола користувачів, також не має завдання містити у собі рухомі візуальні елементи, векторні зображення та подібні їм складні елементи. Вирішено застосовувати стандартні вебтехнології HTML5, CSS3 і JavaScript з можливим доповненням бібліотек, таких як jQuery, для спрощення DOM-маніпуляцій.

jQuery – це популярна JavaScript-бібліотека, створена для спрощення роботи з HTML-документами, обробки подій, анімацій та AJAX-запитів. Вона дозволяє розробникам писати менше коду для досягнення тих самих результатів, які у звичайному JavaScript потребували б значно більше рядків. Однією з головних переваг jQuery є кросбраузерна сумісність – вона усуває багато проблем, пов'язаних із відмінностями реалізації JavaScript у різних браузерах. Серед основних задач, які jQuery спрощує: маніпуляція DOM-деревом, валідація форм, приховування й показ елементів, обробка натискань миші або клавіш.

Завдяки зручному та лаконічному синтаксису, jQuery активно використовувалась у веброзробці протягом тривалого часу. Для невеликих або середніх проєктів вона дозволяє швидко реалізувати інтерфейсну логіку без необхідності в складних фреймворках. До того ж jQuery має велику спільноту та численні плагіни, які полегшують реалізацію типових функцій – від слайдерів до модальних вікон [4].

У якості переваги, усі сучасні браузери стабільно відображають HTML/CSS/JS, тому не виникає залежностей від певних додатків чи обмежень платформи, достатньо «чистої» верстки і невеликого JavaScript-коду, щоб створювати сторінки для форуму. Також слід виділити підтримку AJAX для часткового оновлення сторінок, тобто, підвантаження нових коментарів без перезавантаження. Також існує можливість застосування Fetch API чи jQuery.ajax [5, 6].

### 1.4 Огляд середовища XAMPP

Розгортання локального середовища розробки, а також початкове тестове розгортання вебзастосунків можуть бути суттєво спрощені, якщо застосовувати інтегровані програмні пакети. Одним із найпопулярніших рішень є XAMPP – кросплатформний дистрибутив, що об'єднує вебсервер

Apache, мову PHP, систему керування базами даних MariaDB та кілька додаткових інструментів. XAMPP має доволі просте встановлення та оновлення за необхідності, а інтегровані в нього елементи регулярно оновлюються. Його використання дозволяє швидко налаштувати робоче середовище та зосередитися безпосередньо на логіці вебзастосунку.

### 1.4.1 Огляд XAMPP як інтегрованого середовища

XAMPP позиціонується як комплексне рішення «під ключ» для веброзробки, де користувачеві не потрібно окремо налаштовувати й синхронізувати компоненти системи. Це особливо зручно в ситуаціях, коли пріоритетом є швидкий старт розробки без витрат часу на складні інсталяції та конфігурації.

Взаємодія між PHP, вебсервером і базою даних забезпечується автоматично, оскільки всі компоненти пакету одразу оптимізовані для спільної роботи.

У ключових ідеях XAMPP міститься швидке налаштування (після встановлення дистрибутиву розробник отримує вебсервер Apache, PHP та MariaDB у готовому вигляді з можливістю запускати й зупиняти ці сервіси за допомогою зручної контрольної панелі) та кросплатформність (XAMPP доступний для Windows, Linux і macOS, що надає універсальність команді розробників з різними операційними системами).

### 1.4.2 Використання XAMPP у контексті локальної розробки

Враховуючи потреби вебфоруму, такі як оптимальна швидкодія, простота налаштування та можливість легкого розповсюдження середовища серед учасників команди, XAMPP забезпечує весь необхідний базовий

функціонал. Розробники можуть відразу після встановлення пакету запускати Apache та MariaDB, створювати PHP код, а потім тестувати його роботу через локальний браузер.

Хоча мікросервісний або контейнеризований підхід набуває популярності у масштабних проєктах, для середніх і невеликих рішень, таких як вебфорум із помірними вимогами, XAMPP залишається оптимальним вибором. При необхідності розширення чи збільшення навантаження, код проєкту можна як перенести у Docker контейнери, так і залишити на XAMPP, оскільки навіть при зростанні проєкту функціональність цього середовища залишається стабільною і надійною.

Важливо зазначити, що проєкт, розроблений у середовищі XAMPP, у подальшому може бути легко перенесений на хостинг із встановленим LAMP (Linux, Apache, MySQL, PHP) або в контейнерне середовище без необхідності у змінах коду.

## 1.5 Висновки до розділу 1

У цьому розділі було проаналізовано ключові особливості вебфорумів як платформ для тематичних обговорень та обміну інформацією, розглянуто приклади вже існуючих рішень та обґрунтовано вибір технологій для реалізації форуму. Крім того, для швидкого й зручного розгортання локального середовища розробки обрано XAMPP, що дозволяє сконцентруватися на безпосередній реалізації функціональності вебфоруму.

# 2 РОЗРОБКА ВЕБФОРУМУ

## 2.1 Формування технічного завдання

Перед початком проєктування вебфоруму необхідно сформувати технічне завдання, а саме: надати питання зацікавленим сторонам, з’ясувати їх побажання, цілі проєкту, межі виконання, функціональні та нефункціональні вимоги.

### 2.1.1 Список зацікавлених осіб та їх опитування

Перед початком роботи необхідно відповісти на наступні питання:

– «Хто є користувачем системи?» – користувачами вебфоруму є його відвідувачі та адміністрація;
– «Хто є замовником (покупцем) системи?» – замовником є адміністрація вебфоруму;
– «На кого ще вплинуть результати роботи системи?» – на всіх користувачів системи;
– «Хто буде оцінювати і приймати систему, коли вона буде представлена і розгорнута?» – замовник;
– «Чи існують інші внутрішні чи зовнішні користувачі системи, чиї потреби необхідно врахувати?» – потреби модераційного відділу;
– «Хто буде займатися супроводом нової системи?» – або той, хто систему створив, або людина, яка буде працювати з адміністрацією у подальшому.

Далі, необхідно провести опитування з зацікавленими особами стосовно розроблюваного проєкту.

Далі наведено приклад анкетування зацікавленої особи:

– ім’я: Іван;
– найменування організації: не належу до організацій;
– найменування структурного підрозділу: адміністрація;
– посада: майбутній адміністратор;
– «Які Ваші посадові обов’язки?» – робота з модераційним відділом, самостійна модерація форуму за необхідністю, організація роботи та вирішення подальших проблем;
– «Що є продуктом Вашої діяльності?» – стабільна робота вебфоруму.

### 2.1.2 Заповнення документу «Опис постановки задачі»

*Проблема (проблеми), що викликала необхідність у розробці*: необхідність у зборі людей та оголошення різноманітної інформації, спілкування.

*Визначення цілей продукту та потреб користувачів*: комфортний для використання вебфорум, де можна спілкуватися та оголошувати різні теми для обговорення.

*Зацікавлені персони та користувачі*: замовник, розробник.

*Межі системи* (середовище, платформа та оточення, в якому буде працювати): вебсторінка, середовище браузеру, бази даних SQL.

*Обмеження системи* (задачі, які не будуть вирішуватись): поки що невідомо.

*Цільовий сегмент ринку*: розробка вебсторінки.

*Нефункціональні вимоги*:

– короткий час відклику;
– пропускна здатність;
– зручність у використанні

– аутентифікація та авторизація;
– захист даних;
– можливість розширення функціональності без повної переробки коду;
– підтримка різних браузерів для відображення.

*Функціональні вимоги*:

– здатність користувача зареєструватися або увійти у систему;
– здатність користувача створювати теми для обговорення;
– здатність редагувати створені обговорення;
– здатність коментувати обговорення та вносити зміни;
– здатність знайти обговорення по назві та здатність знайти користувача;
– можливість перегляду профілю користувача та його обговорень;
– можливість роботи з картинками та різним форматом тексту у обговоренні чи коментарі;
– можливість адміністрації, точніше модерації, у редагуванні наповнення вебфоруму.

*Профілі користувачів*: адміністратор, модератор, користувач.

*Сценарії використання*: адміністратор має повний доступ до системи, а саме: оголошення користувача до групи модерації, створення, редагування та видалення обговорень, своїх чи чужих, створення, редагування та видалення коментарів, пошук за обговоренням, користувачем, та тегами, можливість редагування елементів профілю, редагування нікнейму, пошти та аватару. Модератор назначається адміністратором та має можливість редагування та видалення обговорень чи коментарів інших людей, створення обговорень та коментарів, пошук за обговоренням та користувачем. Користувач може створювати обговорення чи коментарі, але не має можливості редагування чужого обговорення чи коментаря, може виконувати пошук за обговоренням та користувачем.

## 2.2 Побудова діаграми варіантів використання

Діаграма варіантів використання є одним із базових інструментів моделювання програмного забезпечення, який широко застосовується для візуального представлення функціональності системи з точки зору користувача. Основна мета такої діаграми полягає у фіксації вимог до системи на ранніх етапах проєктування, а також у наочному відображенні того, як різні типи користувачів взаємодіють із системою через певні сценарії використання. Діаграма варіантів використання дозволяє зосередитися не на внутрішній реалізації функцій, а на діях, які повинна підтримувати система з точки зору логіки предметної області та клієнта.

Елементами діаграми є актори, варіанти використання та зв’язки між ними. Актори представляють ролі, які користувачі або зовнішні системи виконують при взаємодії з елементами проєкту. Це можуть бути реальні люди, зовнішні пристрої або автоматизовані інформаційні системи. Варіанти використання описують функціональні можливості системи – тобто що саме користувачі можуть зробити, наприклад, «зареєструватися», «створити публікацію», «залишити коментар». Зв’язки показують взаємодію між акторами та варіантами використання, а також показують відносини між самими сценаріями через механізми «include» (обов’язкове включення одного сценарію в інший) та «extend» (опціональний функціонал).

Розглянемо основні переваги використання.

*Зрозумілість*. Однією з головних переваг діаграм варіантів використання є їхня висока зрозумілість для широкої аудиторії. Такі діаграми використовують прості візуальні елементи – акторів, варіанти використання, стрілки зв’язку – що робить їх доступними як для розробників програмного забезпечення, так і для замовників, менеджерів проєктів та беспосередніх користувачів. Вони дозволяють отримати загальне уявлення про функціональність системи без потреби заглиблюватися у технічні деталі коду та внутрішню архітектуру проєкту.

*Виявлення вимог на ранніх етапах*. Діаграми варіантів використання суттєво допомагають у процесі аналізу та виявлення вимог до системи ще на перших етапах розробки. Завдяки візуалізації взаємодії між користувачами та системою можна одразу визначити основні функціональні можливості, які необхідно реалізувати, і зрозуміти логіку роботи проєкту.

Також, раннє виявлення вимог дозволяє краще планувати тестування, так як на основі варіантів використання легко формулювати майбутні тест-кейси. Як результат, підвищується загальна якість програмного продукту і зменшується кількість помилок на пізніх етапах розробки.

*Покращення комунікації між учасниками проєкту*. Ще однією важливою перевагою діаграм варіантів використання є те, що вони виступають універсальним засобом комунікації між усіма учасниками розробки: замовниками, розробниками, дизайнерами, тестувальниками та менеджерами.

Це особливо корисно під час проведення зустрічей чи презентацій, коли необхідно узгоджувати деталі проєкту, визначити обов’язкові та бажані функції, визначитися з пріорітетами реалізації. Наявність чіткої та узгодженої візуальної моделі дозволяє уникати непорозумінь та, як наслідок, економити час.

*Підвищення якості архітектури програмного забезпечення*. Якісне створення діаграм варіантів використання значною мірою впливає на якість архітектури майбутнього програмного забезпечення. Вони допомагають чітко зрозуміти, які саме функціональні компоненти будуть потрібні, як користувачі взаємодіятимуть із системою, де можуть виникати складні сценарії, а де можливе спрощення логіки.

Завдяки діаграмам можна одразу передбачити точки розширення системи, що дає змогу в майбутньому додавати нові функції без радикальної перебудови вже існуючого коду, що зменшує часові витрати на модернізацію проєкту та збільшує життєвий цикл програмного забезпечення.

Нижче представлені діаграми варіантів використання для незареєстрованого користувача (рис. 2.1), зареєстрованого користувача (рис. 2.2) та адміністратора (рис. 2.3).

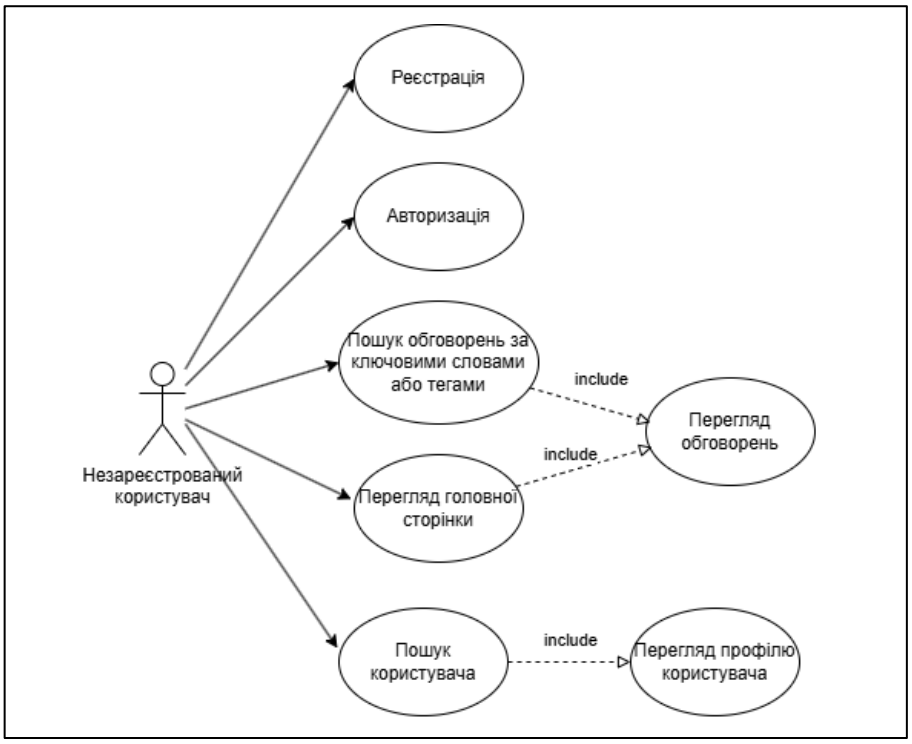



Рисунок 2.1 – Діаграма варіантів використання для незареєстрованого користувача

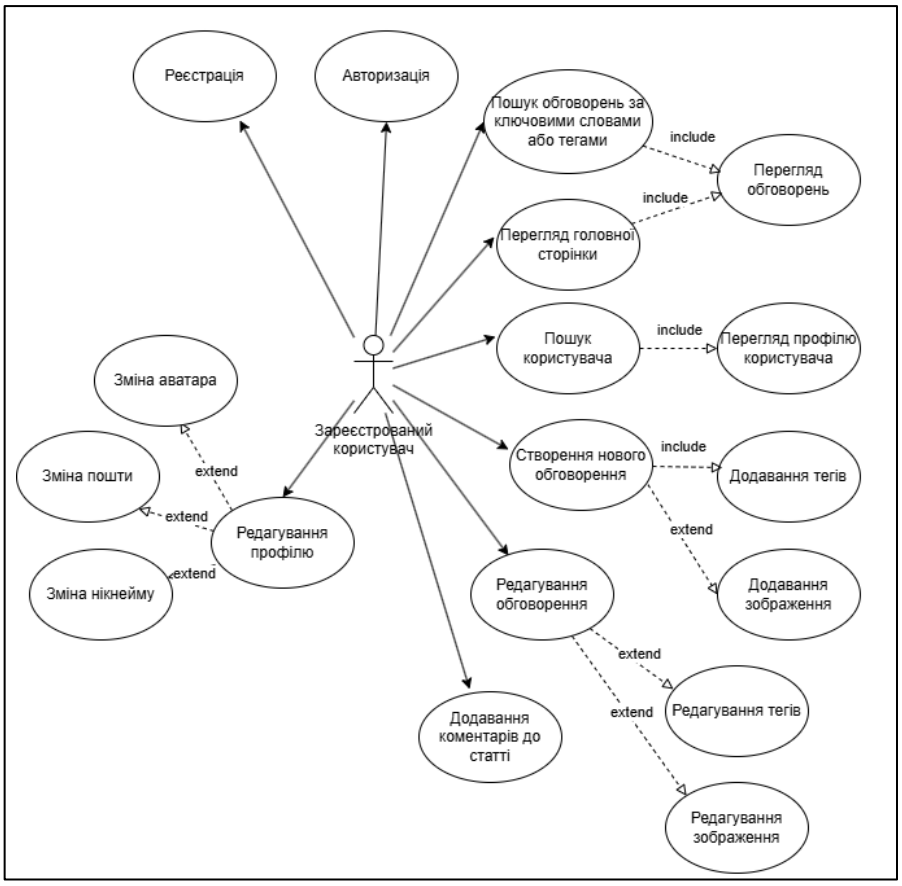



Рисунок 2.2 – Діаграма варіантів використаннядля зареєстрованого користувача

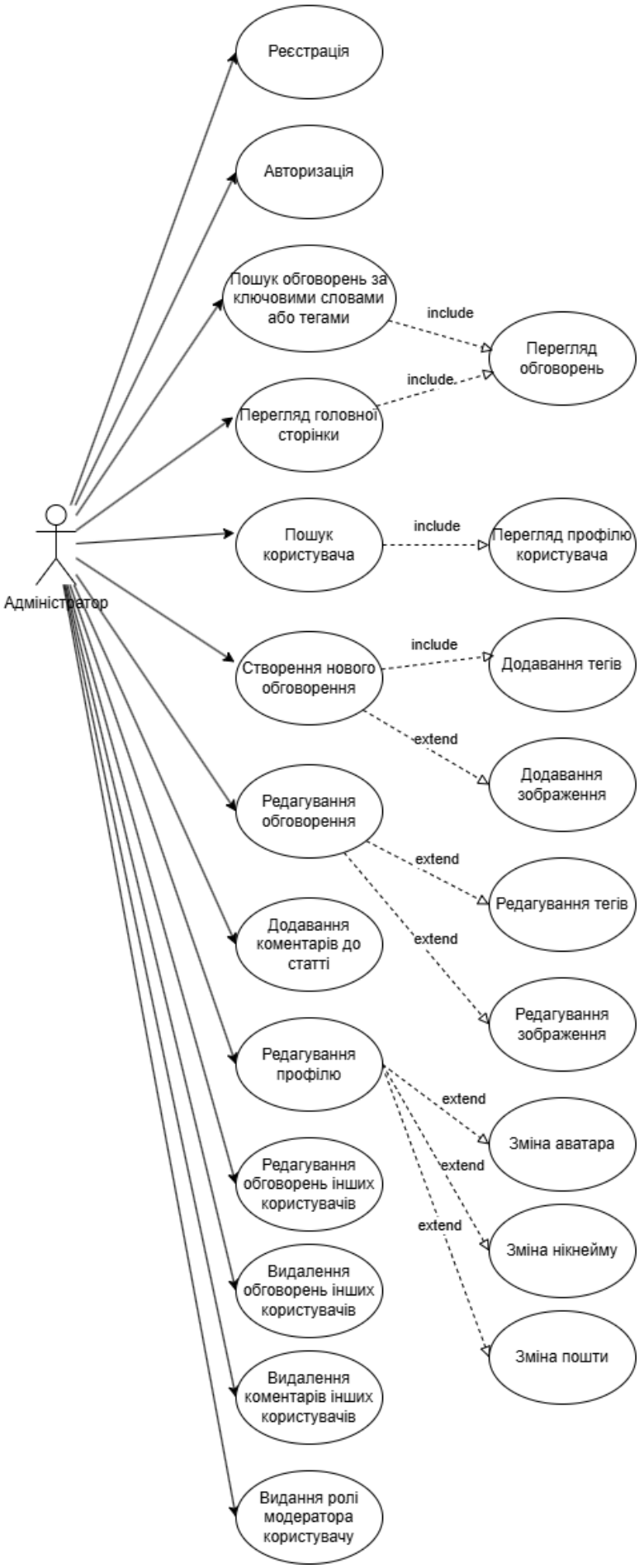



Рисунок 2.3 – Діаграма варіантів використання для адміністратора

## 2.3 Побудова діаграми послідовності

Діаграма послідовності – це один із типів UML-діаграм, який моделює взаємодію об’єктів у часовій послідовності. Вона показує, які об’єкти беруть участь у сценарії, в якій черговості передаються повідомлення, і як змінюються стани системи з часом.

Основною метою діаграми послідовності є відображення логіки виконання функціоналу: які компоненти системи взаємодіють між собою під час виконання певної задачі, і які повідомлення між ними передаються.

Основні переваги діаграм послідовності:

– візуалізація логіки взаємодії: діаграма дозволяє швидко зрозуміти, які об’єкти задіяні в процесі, та в якій черговості виконуються їхні дії;
– раннє виявлення помилок: якщо щось нелогічне у послідовності або відсутній важливий етап, це можна побачити ще на стадії проєктування;
– полегшення реалізації: діаграма допомагає програмістам зрозуміти, як правильно організувати методи і виклики між модулями;
– документування системи: використовується як частина технічної документації для пояснення роботи ключових сценаріїв системи.

Нижче наведені діаграми послідовності для реєстрації користувача (рис. 2.4), пошуку обговорень (рис. 2.5) та створення обговорення (рис. 2.6).




Рисунок 2.4 – Діаграма послідовності за функцією реєстрації

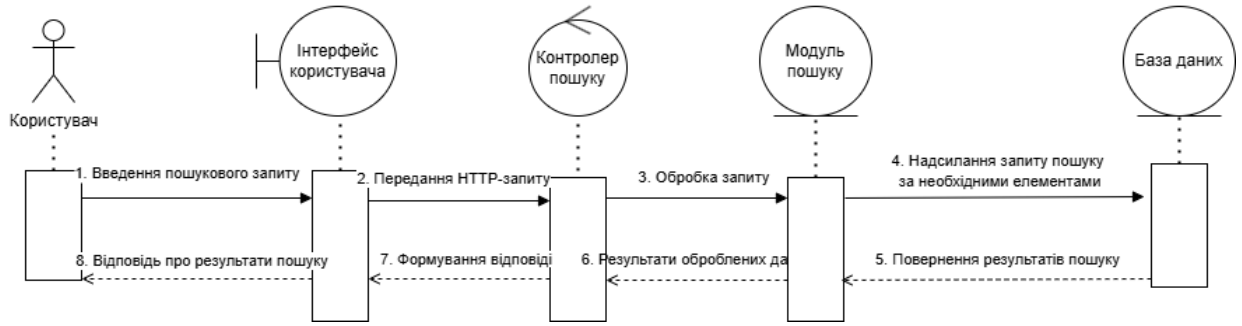



Рисунок 2.5 – Діаграма послідовності за функцією пошуку за назвою, тегами

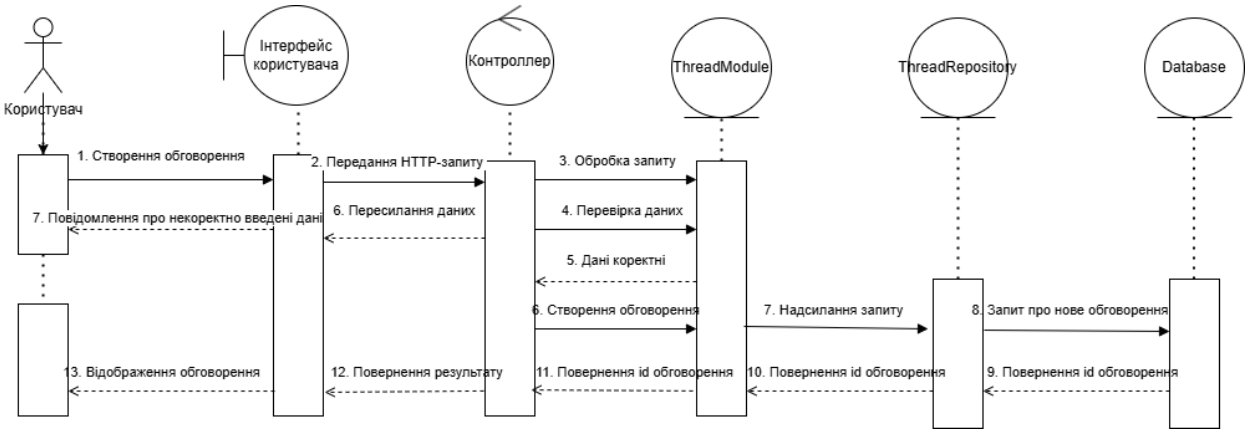



Рисунок 2.6 – Діаграма послідовності за функцією створення обговорення

### 2.4 Схема бази даних

У процесі розробки вебфоруму за тематикою комп’ютерних ігор була спроєктована реляційна база даних.

Розглянемо основні таблиці схеми бази даних.

*Таблиця users*. Таблиця зберігає інформацію про зареєстрованих користувачів форуму. Поле id має тип даних int, первинний ключ користувача. Поле username має тип даних varchar, ім’я користувача. Поле email має тип даних varchar, електронна пошта. Поле password має тип даних varchar, хеш пароля. Поле nickname має тип даних varchar, найменування користувача для інших користувачів форуму. Поле avatar_path має тип даних varchar, шлях до аватара користувача.

*Таблиця articles*. Таблиця містить дані про створені обговорення. Поле id має тип даних int, первинний ключ обговорення. Поле title має тип даних

varchar, заголовок обговорення. Поле content має тип даних text, основний вміст статті. Поле author_id має тип даних int, ключ автора обговорення. Поле author має тип даних varchar, автор статті. Поле created_at має тип даних timestamp, дата і час створення обговорення. Поле image_path має тип даних varchar, шлях до прикріпленого зображення.

*Таблиця comments.* Таблиця містить коментарі до обговорень. Поле id має тип даних int, первинний ключ коментаря. Поле content має тип даних text, текст коментаря. Поле article_id має тип даних int, ідентифікатор статті, до якої залишено коментар. Поле user_id має тип даних int, ключ автора коментаря. Поле created_at має тип даних timestamp, дата і час створення коментаря. Поле edited_at має тип даних datetime, дата і час редагування коментаря.

*Таблиця tags.* Таблиця містить коментарі до обговорень. Поле id має тип даних int, первинний ключ тегу. Поле name має тип даних varchar, назва тегу.

*Таблиця article_tags.* Таблиця містить коментарі до обговорень. Поле article_id має тип даних int, ідентифікатор статті. Поле tag_id має тип даних int, ідентифікатор тегу.

Нижче наведено ER-діаграму бази даних (рис. 2.7).

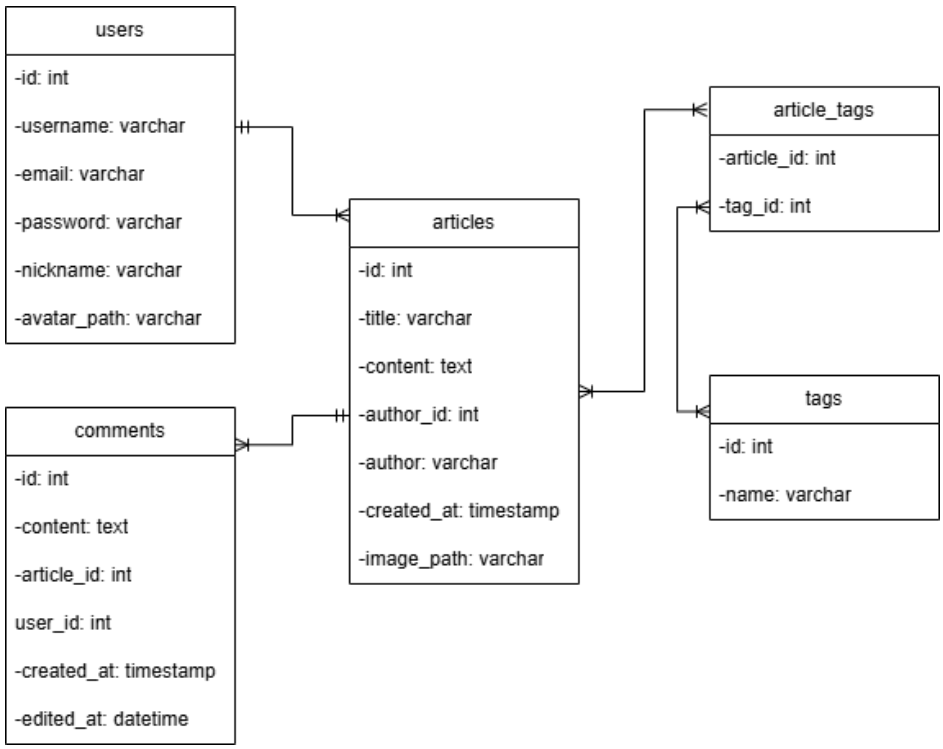



Рисунок 2.7 – ER-діаграма бази даних

## 2.5 Висновки до розділу 2

У другому розділі кваліфікаційної роботи було сформульовано технічне завдання, визначено функціональні та нефункціональні вимоги, розроблено діаграми варіантів використання для різних типів користувачів системи, побудовано діаграми послідовності для ключових функцій додатку.

Також виконано фізичне проєктування бази даних, розроблено її реляційну структуру та побудовано ER-діаграму, яка відображає зв’язки між основними сутностями: користувачами, обговореннями, коментарями та тегами.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Ініціалізація та налаштування проєкту

На початковому етапі розробки вебфоруму було виконано базову ініціалізацію проєкту:

а) на локальній машині встановлено програмне забезпечення XAMPP, що включає Apache, PHP, MySQL;

б) створено базову структуру проєкту, яка включає такі директорії:
   – public/ – для основних сторінок (index.php, discussion.php, create_article.php);
   – includes/ – для конфігураційних та допоміжних файлів (config.php, header.php);
   – uploads/ – для збереження завантажених аватарів та зображень статей;

в) підключено базу даних через файл includes/config.php, у якому налаштовано підключення до MySQL за допомогою PDO;

г) реалізовано основну автентифікацію користувачів (реєстрація, вхід у систему) через файли register.php та login.php;

д) для забезпечення захисту доступу до особистих сторінок використовується перевірка активності сесії (session_start + перевірка $_SESSION) [7].

База даних проєкту має такі основні таблиці:

– users – збереження даних користувачів;
– articles – збереження статей користувачів;
– comments – збереження коментарів до статей;
– tags та article_tags – реалізація тегування статей.

Для взаємодії з базою даних усі запити виконуються через підготовлені вирази (PDO::prepare).

Реалізовано базові функції CRUD:

– створення, редагування, видалення статей та коментарів;

– зміна особистих даних користувача, таких як нікнейм, email, аватар.

Інтерфейс користувача створено за допомогою HTML5 та CSS3 без використання сторонніх фреймворків.

Для завантаження аватарів і обрізки зображень використовується бібліотека Cropper.js [8].

### 3.2 Створення бази даних

Для створення бази даних вебфоруму було використано систему керування базами даних MariaDB, яка є форком популярної системи MySQL. MariaDB була обрана через її стабільність, відкритість, високу продуктивність і сумісність із великою кількістю середовищ розробки.

База даних була спроєктована із дотриманням принципів нормалізації даних, що дозволяє мінімізувати дублювання інформації та забезпечити логічну структуру зберігання об’єктів. Для взаємодії серверної частини додатка з базою даних було використано розширення PDO (PHP Data Objects) у мові PHP. Використання PDO дозволяє працювати з базою даних безпосередньо через підготовлені запити (prepared statements), що суттєво підвищує безпеку застосунку і допомагає уникнути атак типу SQL-ін’єкцій [9].

При проєктуванні бази даних застосовувалися зовнішні ключі **(**FOREIGN KEY**)** для забезпечення зв’язності між таблицями та автоматичного забезпечення цілісності даних при видаленні або зміні записів. Наприклад, при видаленні обговорення автоматично видаляються пов’язані коментарі та записи зв’язку з тегами.

Таким чином, архітектура бази даних була побудована таким чином, щоб забезпечити:

– масштабованість;

– підтримку відношень між об’єктами системи;

– оптимізацію запитів через використання індексів та структурованих зв’язків.

На рисунках 3.1 – 3.5 наведено програмну реалізацію створення основних таблиць бази даних.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    avatar_path VARCHAR(255),
    role ENUM('user', 'moderator', 'admin') DEFAULT 'user'
);
```

Рисунок 3.1 – Таблиця users

Таблиця призначена для зберігання інформації про зареєстровних користувачів форуму. Її структурні елементи:

– id – унікальний ідентифікатор користувача;

– username, email – дані для ідентифікації та авторизації користувача;

– password_hash – захищений хеш пароля;

– avatar_path – шлях до аватара користувача;

– role – роль користувача в системі (звичайний користувач, модератор або адміністратор).

```
CREATE TABLE articles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    author VARCHAR(255) NOT NULL,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    image_path VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Рисунок 3.2 – Таблиця articles

Таблиця містить інформацію про створені теми обговорень. Її структурні елементи:

– id – унікальний ідентифікатор статті;
– author – ім’я користувача, який створив обговорення;
– title – заголовок теми;
– content – текст обговорення;
– image_path – шлях до прикріпленого зображення;
– created_at – дата та час створення.

```
CREATE TABLE comments (
   id INT AUTO_INCREMENT PRIMARY KEY,
   article_id INT NOT NULL,
   author VARCHAR(255) NOT NULL,
   content TEXT NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (article_id) REFERENCES articles(id) ON DELETE CASCADE
);
```

Рисунок 3.3 – Таблиця comments

Таблиця містить коментарі користувачів до статей. Її структурні елементи:

– id – унікальний ідентифікатор коментаря;
– article_id – прив’язка до відповідної статті;
– author – автор коментаря;
– content – зміст коментаря;
– created_at – дата і час створення коментаря.

```
CREATE TABLE tags (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(255) UNIQUE NOT NULL
);
```

Рисунок 3.4 – Таблиця tags

Таблиця призначена для зберігання унікальних тегів обговорень. Її структурні елементи:

– id – унікальний ідентифікатор тега;
– name – назва тега.

```
CREATE TABLE article_tags (
    id INT AUTO_INCREMENT PRIMARY KEY,
    article_id INT NOT NULL,
    tag_id INT NOT NULL,
    FOREIGN KEY (article_id) REFERENCES articles(id) ON DELETE CASCADE,
    FOREIGN KEY (tag_id) REFERENCES tags(id) ON DELETE CASCADE
);
```

Рисунок 3.5 – Таблиця article_tags

Таблиця зв’язку між статтями та тегами. Її структурні елементи:

– id – унікальний ідентифікатор запису;

– article_id – посилання на статтю;

– tag_id – посилання на тег.

### 3.3 Програмна реалізація клієнтської частини

На рисунках 3.6 – 3.15 відображено створену реалізацію головної сторінки та шапки профілю форуму.

Файл index.php відповідає за відображення головної сторінки форуму, де виводяться всі наявні обговорення. Спочатку ми підключаємо такі елементи, як конфігурацію бази даних та загальне меню сайту (рис. 3.6).

```
<?php
session_start();
require '../includes/config.php';
include '../includes/header.php';
```

Рисунок 3.6 – Підключення необхідних залежностей

session_start() – ініціалізує сесію для користувача. Це дозволяє використовувати такі глобальні змінні, як $_SESSION, для збереження авторизаційних даних.

require '../includes/config.php' – підключення конфігураційного файлу для доступу до бази даних через PDO. Він містить об’єкт $pdo, який буде використовуватись у SQL-запитах.

include '../includes/header.php' – підключає загальну шапку сайту та її елементи, такі як меню навігації, модальні вікна логіну/реєстрації.

SQL запит з рисунку 3.7 витягує заголовки, тексти, зображення, дати публікації обговорень зі зв’язаних таблиць.

```
try {
   $sql = "SELECT
             articles.id,
             articles.title,
             articles.content,
             articles.image_path,
             articles.created_at,
             users.username,
             GROUP_CONCAT(tags.name SEPARATOR ', ') AS tags
          FROM articles
          JOIN users ON articles.author_id = users.id
          LEFT JOIN article_tags ON articles.id = article_tags.article_id
          LEFT JOIN tags ON article_tags.tag_id = tags.id
          GROUP BY articles.id
          ORDER BY articles.created_at DESC";
   $stmt = $pdo->query($sql);
   $articles = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
   $articles = [];
   echo "Помилка завантаження статей: " . $e->getMessage();
}
?>
```

Рисунок 3.7 – Отримання обговорень з бази даних

JOIN users – приєднує інформацію про автора статті.

LEFT JOIN article_tags + tags – дозволяє підключити всі теги, які прив’язані до обговорення.

GROUP_CONCAT(tags.name SEPARATOR ', ') AS tags – об’єднує всі теги для одного обговорення у строку через кому.

GROUP BY articles.id – згруповує записи для унікальності кожної статті.

ORDER BY created_at DESC – відсортовано за датою, новіші статті вище.

fetchAll(PDO::FETCH_ASSOC) повертає всі записи як асоціативний масив.

На рисунку 3.8 за допомогою представленого коду перевіряється, чи масив $articles не порожній. Якщо жодної статті немає – показується заклик створити перше обговорення.

```
<main>
  <?php if (empty($articles)): ?>
    <section class="news-preview">
      <p>Поки що немає обговорень. Створіть перше!</p>
    </section>
  <?php endif; ?>
</main>
```

Рисунок 3.8 – Перевірка та вивід статей

Цей блок динамічний і дозволяє адаптувати інтерфейс до стану бази даних.

Простий футер, який додає завершальний блок для кожної сторінки (рис. 3.9). Можна використовувати для контактів, авторських прав, коротких посилань на основні сторінки сайту.

```
<footer>
  <p>© 2024 Web Forum for Computer Game Fans </p>
</footer>
```

Рисунок 3.9 – Підвал сторінки

Скрипти для управління відкриттям вікон логіну та реєстрації через jQuery представлено на рисунку 3.10.

$(document).ready(function() { ... }) , $("#login-button").off("click").on("click", function(e) { ... }) , $("#auth-panel").fadeToggle(), $("#register-form").hide();, $("#login-form").show(), $(".auth-panel").on("click", function(e) { ... }) використовують jQuery для запуску коду, виборці елементів, видаленні і додання обробників, анімації,динамічного управління модальними вікнами логіну/реєстрації, та виборці по класу.

fadeToggle() дозволяє плавно показувати або ховати блок логіну.

hide() / .show() – відбувається перемикання між формами входу й реєстрації на клієнті, без перезавантаження сторінки.

off("click").on("click") – гарантує, що попередні обробники кліку не будуть дублюватися.

```
<script>
   $(document).ready(function () {
      $("#login-button").off("click").on("click", function (e) {
         e.preventDefault();
         $("#auth-panel").fadeToggle();
         $("#register-form").hide();
         $("#login-form").show();
      });

      $("#show-register").off("click").on("click", function (e) {
         e.preventDefault();
         $("#login-form").hide();
         $("#register-form").show();
      });

      $("#show-login").off("click").on("click", function (e) {
         e.preventDefault();
         $("#register-form").hide();
         $("#login-form").show();
      });

      $(".auth-panel").off("click").on("click", function (e) {
         if (e.target.id === "auth-panel") {
            $("#auth-panel").fadeOut();
         }
      });
   });
</script>
</body>
</html>
```

Рисунок 3.10 – Скрипти для управління відкриттям вікон логіну та реєстрації через jQuery

Файл header.php відповідає за відображення шапки сайту, меню навігації, панель авторизації і модальне вікно пошуку.

```
<?php
if (session_status() === PHP_SESSION_NONE) {
   session_start();
}
require '../includes/config.php';
?>
```

Рисунок 3.11 – Ініціалізації сесії та підключення бази даних

session_status() === PHP_SESSION_NONE – перевірка, чи запущена сесія. Якщо не запущена – запускаємо.

session_start() – ініціалізує сесію для відстеження стану користувача.

require '../includes/config.php'; – підключає налаштування підключення до бази даних через PDO.

```
<header class="main-header">
  <h1>Форум</h1>
  <nav>
    <a href="/forum/public/index.php">Головна</a>
    <a href="#" id="search-button">Пошук</a>
    <?php if (isset($_SESSION['username'])): ?>
      <a href="/forum/includes/create_article.php">Створити обговорення</a>
      <a href="/forum/public/account.php">Акаунт</a>
      <a href="#" onclick="document.getElementById('logout-form').submit();" class="logout-button">Вихід</a>
      <form id="logout-form" action="/forum/includes/logout.php" method="POST" style="display: none;"></form>
    <?php else: ?>
      <a href="#" id="login-button">Логін</a>
    <?php endif; ?>
  </nav>
</header>
```

Рисунок 3.12 – Шапка сайту і навігація

<header class="main-header"> – контейнер верхньої частини сайту, містить назву та меню навігації.

<nav> – навігаційне меню, де розміщуються посилання на основні розділи сайту.

<a href="/forum/public/index.php">Головна</a> – посилання на головну сторінку форуму.

<a href="#" id="search-button">Пошук</a> – кнопка для відкриття модального вікна пошуку.

<?php if (isset($_SESSION['username'])): ?> – умовна перевірка: якщо користувач залогінений та у сесії збережено логін, то відображаються додаткові посилання.

<a href="/forum/includes/create_article.php">Створити обговорення</a> – посилання на сторінку створення нового обговорення, для авторизованого користувача.

<a href="/forum/public/account.php">Акаунт</a> – посилання на сторінку особистого кабінету користувача.

<a href="#" onclick="document.getElementById('logout-form').submit();" class="logout-button">Вихід</a> – посилання на вихід, яке при кліку викликає .submit() для прихованої форми виходу, використовується JavaScript для безпечного виходу через POST.

<form id="logout-form" action="/forum/includes/logout.php" method="POST" style="display: none;"></form> – прихована форма для виконання POST-запиту на вихід із системи, щоб уникнути проблем безпеки, не через GET.

<?php else: ?> – у випадку, коли користувач не авторизований.

<a href="#" id="login-button">Логін</a> – кнопка для відкриття форми логіну, відображається лише незалогіненим користувачам, обробляється через jQuery по id="login-button".

```
<div class="auth-panel" id="auth-panel" style="display:none;">
  <div class="auth-container">
    <!-- Login Form -->
    <div id="login-form" class="auth-form">
      <h2>Login</h2>
      <form action="/forum/includes/login.php" method="POST">
        <input type="text" name="username" placeholder="Ім’я користувача" required>
        <input type="password" name="password" placeholder="Пароль" required>
        <button type="submit">Увійти</button>
        <p>Немає акаунту? <a href="#" id="show-register">Зареєструйтесь</a></p>
      </form>
    </div>
```

Рисунок 3.13 – Панель для логіну та реєстрації

 – контейнер модального вікна авторизації, спочатку прихований, з’являється при натисканні кнопки "Логин". Має id, за яким його обробляє JavaScript.

 – внутрішній контейнер, що містить логін – форму і форму реєстрації.

 – форма для авторизації користувача. Має id="login-form", який використовується для її відображення або приховування через JavaScript.

<h2>Login</h2> – заголовок блоку логіну.

<form action="/forum/includes/login.php" method="POST"> – HTML-форма, яка надсилає логін і пароль методом POST на серверний обробник /includes/login.php.

<input type="text" name="username" placeholder="Ім’я користувача" required> – обов’язкове поле для введення логіна.

<input type="password" name="password" placeholder="Пароль" required> – поле для введення пароля, прихованого типу, також обов’язкове.

<button type="submit">Увійти</button> – кнопка надсилання форми, яка ініціює запит авторизації.

Немає акаунту? <a href="#" id="show-register">Зареєструйтесь</a> – текст з посиланням на реєстрацію. Посилання має id="show-register", яке використовується скриптом для переключення на форму реєстрації.

```
<div id="register-form" class="auth-form" style="display:none;">
        <h2>Register</h2>
        <form action="/forum/includes/register.php" method="POST">
           <input type="text" name="username" placeholder="Ім’я користувача" required>
           <input type="email" name="email" placeholder="Электронна пошта">
           <input type="password" name="password" placeholder="Пароль" required>
           <button type="submit">Зареєструватися</button>
           <p>Уже є акаунт? <a href="#" id="show-login">Увійти</a></p>
        </form>
     </div>
  </div>
</div>
```

Рисунок 3.14 – Форма для реєстрації

 – контейнер форми реєстрації. Спочатку прихований через display: none,

відображається при натисканні "Зареєструйтесь" у формі логіну. Має id, який використовується в JavaScript для перемикання між формами.

<h2>Register</h2> – заголовок блоку форми реєстрації.

<form action="/forum/includes/register.php" method="POST"> – HTML-форма для реєстрації. Дані надсилаються методом POST на серверний обробник /includes/register.php.

<input type="text" name="username" placeholder="Ім’я користувача" required> – обов’язкове поле для введення логіна нового користувача.

<input type="email" name="email" placeholder="Электронна пошта"> – поле для введення електронної пошти. Необов’язкове, тому тут відсутній атрибут required.

<input type="password" name="password" placeholder="Пароль" required> – поле для введення пароля. Тип password приховує символи, обов’язкове для заповнення.

<button type="submit">Зареєструватися</button> – кнопка надсилання форми, ініціює запит на реєстрацію користувача.

Уже є акаунт? <a href="#" id="show-login">Увійти</a> – текст із посиланням на повернення до форми логіну. Має id="show-login" – його обробляє скрипт для перемикання назад на форму авторизації.

```
<div id="search-modal" class="modal" style="display:none;">
  <div class="modal-content">
    <form action="/forum/public/search.php" method="GET" style="width: 100%; display: flex; justify-content: center; align-items: center;">
      <input type="text" name="query" class="search-input" placeholder="Введіть ключові слова" required>
      <button type="submit" class="search-button">Шукати</button>
    </form>
  </div>
</div>
```

Рисунок 3.15 – Модальне вікно пошуку

 – контейнер модального вікна пошуку.

 – внутрішній блок вікна. Містить саму форму пошуку. Центрується на екрані.

<form action="/forum/public/search.php" method="GET" – HTML-форма для пошуку, яка відправляє запит методом GET на сторінку /public/search.php. GET використовується, щоб параметр запиту query з’являвся в URL.

<input type="text" name="query" class="search-input" placeholder="Введіть ключові слова" required> – поле введення пошукового запиту.

<button type="submit" class="search-button">Шукати</button> – кнопка для запуску пошуку.

### 3.4 Реалізація створення, редагування та видалення коментарів

На рисунках 3.16 – 3.21 зазначено реалізацію для створення, редагування та видалення коментарів.

```
if ($action === 'add_comment') {
    $comment_content = trim($_POST['content'] ?? '');
    $user_id = $_SESSION['user_id'] ?? null;
    if ($user_id && !empty($comment_content)) {
        try {
            $sql = "INSERT INTO comments (content, article_id, user_id, created_at)
                    VALUES (?, ?, ?, NOW())";
            $stmt = $pdo->prepare($sql);
            $stmt->execute([$comment_content, $id, $user_id]);
            header("Location: discussion.php?id=$id");
            exit();
        } catch (PDOException $e) {
            echo "<p style='color:red;'>Помилка додання коментаря: "
                . htmlspecialchars($e->getMessage()) . "</p>";
        }
    }
}
```

Рисунок 3.16 – PHP частина створення коментаря

if ($action === 'add_comment') – перевіряє, чи була відправлена форма додавання коментаря. Змінна $action надходить із прихованого поля <input name="action" value="add_comment">.

$comment_content = trim($_POST['content'] ?? '') – отримує вміст коментаря з POST-запиту, прибираючи зайві пробіли. Якщо поле відсутнє, підставляється порожній рядок.

$user_id = $_SESSION['user_id'] ?? null – витягує ID поточного авторизованого користувача з сесії. Якщо користувач не авторизований, буде null.

if ($user_id && !empty($comment_content)) – перевірка: користувач повинен бути авторизований, а текст коментаря – не порожній.

$sql = "INSERT INTO comments (content, article_id, user_id, created_at), VALUES (?, ?, ?, NOW())" – SQL-запит для вставки нового коментаря. Поля: текст, ID статті, ID користувача, поточна дата/час.

$stmt = $pdo->prepare($sql), $stmt->execute([$comment_content, $id, $user_id]) – підготовка та виконання запиту з передачею параметрів для захисту від SQL-ін’єкцій.

header("Location: discussion.php?id=$id") – після успішного додавання коментаря – перенаправлення назад на сторінку обговорення.

catch (PDOException $e) { ... } – обробляє винятки, які можуть виникнути під час виконання SQL-запиту. PDOException – це спеціальний тип помилок, пов’язаних із роботою бази даних через PDO.

echo " Помилка додання коментаря: ", . htmlspecialchars($e->getMessage()) . " " – виводить повідомлення про помилку в червоному кольорі на сторінці. Функція htmlspecialchars() захищає від XSS-атак, перетворюючи спецсимволи у безпечні HTML-коди. $e->getMessage() повертає текст опису помилки, що сталася.

```
elseif ($action === 'edit_comment') {
    $comment_id = $_POST['comment_id'] ?? null;
    $updated_content = trim($_POST['content'] ?? '');
    if ($comment_id && !empty($updated_content)) {
        try {
            $sql = "UPDATE comments
                    SET content = ?, edited_at = NOW()
```

```
                WHERE id = ? AND article_id = ?";
            $stmt = $pdo->prepare($sql);
            $stmt->execute([$updated_content, $comment_id, $id]);
            header("Location: discussion.php?id=$id");
            exit();
        } catch (PDOException $e) {
            echo "<p style='color:red;'>Помилка збереження коментаря: "
                . htmlspecialchars($e->getMessage()) . "</p>";}
    }
}
```

Рисунок 3.17 – PHP частина редагування коментаря

elseif ($action === 'edit_comment') – перевіряє, чи користувач виконує дію редагування коментаря. Вона спрацьовує, коли в POST-запиті надійшло action=edit_comment.

$comment_id = $_POST['comment_id'] ?? null – отримує id коментаря, який редагується, або null, якщо параметр відсутній.

$updated_content = trim($_POST['content'] ?? '') – отримує оновлений текст коментаря з форми. trim() прибирає пробіли на початку та в кінці рядка.

if ($comment_id && !empty($updated_content)) { – перевірка: якщо id коментаря заданий і новий текст не порожній, переходимо до оновлення в БД.

$sql = "UPDATE comments , SET content = ?, edited_at = NOW(), WHERE id = ? AND article_id = ?"; – SQL-запит для оновлення вмісту коментаря та встановлення часу редагування edited_at = NOW(). Додатково перевіряється, що цей коментар належить поточній статті article_id = ?.

$stmt = $pdo->prepare($sql), $stmt->execute([$updated_content, $comment_id, $id]) – підготовка та виконання запиту з параметрами: новий текст, id коментаря і id статті.

header("Location: discussion.php?id=$id") – після успішного редагування виконується переадресація назад на сторінку обговорення, щоб уникнути повторного надсилання форми.

catch (PDOException $e) { echo " Помилка збереження коментаря: " . htmlspecialchars($e->getMessage()) . " "; } – обробка помилки під час оновлення, повідомлення виводиться червоним кольором з деталями помилки.

```
<form method="POST" action="discussion.php?id=<?= htmlspecialchars($id) ?>"
    class="add-comment-form">
  <textarea name="content" required class="new-comment-textarea"></textarea>
  <button type="submit" class="forum-button blue">Відправити</button>
  <input type="hidden" name="action" value="add_comment">
</form>
```

Рисунок 3.18 – Форма додавання коментаря

<form method="POST" action="discussion.php?id=<?= htmlspecialchars($id) ?>" class="add-comment-form"> – форма додавання нового коментаря. Відправляє дані методом POST на файл discussion.php з параметром id, який ідентифікує поточну статтю.

<textarea name="content" required class="new-comment-textarea"></textarea> – текстове поле для вводу коментаря. Атрибут required забороняє відправку порожнього поля. Клас new-comment-textarea відповідає за стилізацію та автозбільшення поля на основі вмісту.

<button type="submit" class="forum-button blue">Відправити</button> – кнопка відправки форми. Класи відповідають за зовнішній вигляд кнопки в стилі форуму.

<input type="hidden" name="action" value="add_comment"> – приховане поле, яке передає інформацію про те, що ця форма призначена для створення нового коментаря. Це значення пізніше перевіряється у PHP-коді if ($action === 'add_comment').

```
<form class="edit-comment-form" method="POST" action="discussion.php?id=<?= htmlspecialchars($id) ?>">
  <textarea name="content" rows="1" required><?= htmlspecialchars($comment['content']) ?></textarea>
  <button type="submit" class="forum-button blue save-button" style="display:none;">Зберегти</button>
  <button type="button" class="forum-button blue cancel-button" style="display:none;">Скасування</button>
  <a href="delete_comment.php?id=<?= $comment['id'] ?>" class="forum-button red delete-button-new" style="display: none;">Видалити</a>
  <input type="hidden" name="comment_id" value="<?= $comment['id'] ?>">
  <input type="hidden" name="action" value="edit_comment">
</form>
```

Рисунок 3.19 – Форма редагування коментаря

<form class="edit-comment-form" method="POST" action="discussion.php?id=<?= htmlspecialchars($id) ?>"> – форма редагування

коментаря. Дані надсилаються методом POST на файл discussion.php з параметром id, що відповідає ID статті. Клас edit-comment-form дозволяє JavaScript управляти цією формою для показу або приховування вмісту.

<textarea name="content" rows="1" required><?= htmlspecialchars($comment['content']) ?></textarea>- поле вводу з поточним текстом коментаря. Значення вставляється автоматично з БД. required гарантує, що користувач не відправить порожній коментар.

<button type="submit" class="forum-button blue save-button" style="display:none;">Зберегти</button>- кнопка для підтвердження змін. Спочатку прихована через display:none, показується при натисканні кнопки "Редагувати» через JavaScript.

<button type="button" class="forum-button blue cancel-button" style="display:none;">Скасування</button>- кнопка для скасування редагування. Також спочатку прихована. При натисканні скасовує зміну і повертає попередній вигляд.

<a href="delete_comment.php?id=<?= $comment['id'] ?>" class="forum-button red delete-button-new" style="display: none;">Видалити</a>- посилання для видалення коментаря. Це новий варіант кнопки, який з’являється лише під час редагування, щоб дати змогу видалити прямо під час правки. Працює через перехід на delete_comment.php.

<input type="hidden" name="comment_id" value="<?= $comment['id'] ?>"> – приховане поле, що містить ID коментаря. Потрібне для ідентифікації того, який саме коментар редагується на сервері.

<input type="hidden" name="action" value="edit_comment">- ще одне приховане поле, яке вказує серверу, що ця форма призначена для редагування, а не додавання коментаря.

```
<a href="delete_comment.php?id=<?= $comment['id'] ?>"
   class="forum-button red delete-button-old"
   onclick="localStorage.setItem('scrollPos', window.pageYOffset); return confirm(Ви впевнені, що хочете видалити коментар?');">
   Видалити
</a>
```

Рисунок 3.20 – Форма видалення коментаря

href="delete_comment.php?id=<?= $comment['id'] ?>" – перехід на окрему серверну сторінку delete_comment.php, яка виконує видалення коментаря з бази даних. В параметр id передається ідентифікатор конкретного коментаря.

class="forum-button red delete-button-old" – стилізація кнопки через CSS клас forum-button і додаткове маркування delete-button-old для ідентифікації у скриптах.

onclick="localStorage.setItem('scrollPos', window.pageYOffset); return confirm(...);":

- localStorage.setItem(...) – зберігає поточну позицію прокрутки сторінки, щоб після видалення і перезавантаження залишитись на тому ж місці;
- return confirm(...) – показує користувачу підтвердження на видалення. Якщо користувач натисне «Скасувати», дія не виконується. Якщо «Ок» – браузер переходить за посиланням і запускає видалення.

```
<?php
session_start();
require '../includes/config.php';
$id = $_GET['id'] ?? null;
if (!$id) {
   die("Помилка: ID коментаря не вказано.");}
try {
   $sql = "SELECT * FROM comments WHERE id = ?";
   $stmt = $pdo->prepare($sql);
   $stmt->execute([$id]);
   $comment = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$comment) {
      http_response_code(404);
      die("Помилка: коментар не знайдено.");}
   if ($_SESSION['username'] !== 'root' && $_SESSION['user_id'] != $comment['user_id']) {
      http_response_code(403);
      die("У вас немає прав на видалення цього коментаря.");}
   $sql = "DELETE FROM comments WHERE id = ?";
   $stmt = $pdo->prepare($sql);
   $stmt->execute([$id]);
   header("Location: discussion.php?id=" . $comment['article_id']);
   exit();
} catch (PDOException $e) {
   die("Помилка видалення коментаря: " . $e->getMessage());}
?>
```

Рисунок 3.21 – Безпосередня реалізація видалення коментаря

$id = $_GET['id'] ?? null – отримує параметр id коментаря з URL, якщо його немає – змінна буде null.

if (!$id) {die("Помилка: ID коментаря не вказано.");} – перевіряє наявність id. Якщо не вказано, скрипт завершується з повідомленням про помилку.

try {$sql = "SELECT * FROM comments WHERE id = ?"; $stmt = $pdo->prepare($sql); $stmt->execute([$id]); $comment = $stmt->fetch(PDO::FETCH_ASSOC) – формує та виконує SQL-запит, щоб знайти коментар у таблиці comments за переданим id.

if (!$comment) {http_response_code(404); die("Помилка: коментар не знайдено.");} – перевіряє, чи знайдено коментар у базі. Якщо ні – встановлюється HTTP-статус 404 і виводиться повідомлення про помилку.

if ($_SESSION['username'] !== 'root' && $_SESSION['user_id'] != $comment['user_id']) – перевіряє, чи має користувач право видаляти коментар. Це можливо лише для адміністратора (username === 'root') або автора коментаря (user_id == $comment['user_id']). Якщо обидві умови не виконуються, доступ забороняється.

http_response_code(403); – встановлює HTTP-статус 403 Forbidden, що означає заборону доступу до дії.

$stmt = $pdo->prepare($sql); – готує SQL-запит до виконання через механізм prepared statements – захист від SQL-ін’єкцій.

} catch (PDOException $e) {- обробка помилок PDO, що можуть виникнути під час видалення, наприклад, якщо база недоступна або запит некоректний.

### 3.5 Висновки до розділу 3

У цьому розділі було детально розглянуто програмну реалізацію вебфоруму за тематикою комп’ютерних ігор. Було виконано:

- ініціалізацію проєкту в середовищі XAMPP, організовано структуру директорій, підключено базу даних за допомогою PDO;
- реалізовано серверну логіку для роботи з обговореннями та коментарями: додавання, редагування, видалення, зберігання в базі даних;
- забезпечено авторизацію та реєстрацію користувачів, а також доступ до персонального кабінету на основі сесій;
- реалізовано клієнтську частину з використанням HTML, CSS, JavaScript та jQuery – для динамічного відображення форм авторизації, редагування та видалення коментарів, а також роботи з модальними вікнами;
- забезпечено безпеку обробки даних: усі запити до бази даних реалізовані через підготовлені вирази, що знижує ризики SQL-ін’єкцій; контролюється доступ до функцій редагування та видалення за роллю користувача.

# ВИСНОВКИ

В даній кваліфікаційній роботі розроблено функціональний вебфорум за тематикою комп’ютерних ігор. Розглянуті ключові етапи створення проєкту: аналіз предметної області, розробка вебфоруму та програмна реалізація проєкту.

На першому етапі було проведено аналіз предметної області – онлайн форумів, їхньої структури, особливості реалізації, та визначено їхні переваги й недоліки. Це дозволило обґрунтувати вибір технологій та методів реалізації форуму.

У другому розділі розглянуто архітектуру вебзастосунку, створено структуру бази даних, таблиці та зв’язки між ними. Також були спроєктовані базові сценарії використання – створення обговорень, реєстрація, авторизація, написання та редагування коментарів.

У третьому розділі безпосередньо реалізовано вебфорум за допомогою стеку PHP, MySQL, HTML, CSS та JavaScript. Розроблено зручний інтерфейс користувача, реалізовано автентифікацію та авторизацію, систему обговорень, коментування, пошуку, а також редагування й видалення контенту згідно з правами доступу. Реалізовано динамічну обробку подій на клієнтській стороні, зокрема редагування коментарів без перезавантаження сторінки. Під час роботи були дотримані принципи модульності, структурованості коду, застосовано підготовлені SQL-запити для захисту від SQL-ін’єкцій.

У результаті виконання поставлених завдань створено функціональний, стабільний та зручний вебфорум, що дозволяє користувачам обмінюватися думками, створювати нові обговорення, коментувати й взаємодіяти в межах ігрової спільноти.

# ПЕРЕЛІК ПОСИЛАНЬ

1. PHP Manual – офіційна документація мови PHP. URL: https://www.php.net/manual/en/index.php (дата звернення: 04.03.2025).

2. MariaDB Server Documentation – документація по СУБД MariaDB. URL: https://mariadb.com/kb/en/library/ (дата звернення: 05.03.2025).

3. MySQL Documentation – документація по СУБД MySQL. URL: https://mariadb.com/kb/en/documentation/ (дата звернення: 05.03.2025).

4. jQuery API Documentation. URL: https://api.jquery.com/ (дата звернення: 10.03.2025).

5. HTML Living Standard – офіційний стандарт HTML. URL: https://html.spec.whatwg.org (дата звернення: 14.03.2025).

6. JavaScript Guide – офіційна документація JavaScript. URL: https://developer.mozilla.org/en/docs/Web/JavaScript (дата звернення: 14.03.2025).

7. PHP Sessions Explained – для реалізації сесій. URL: https://www.php.net/manual/en/book.session.php (дата звернення: 17.03.2025).

8. Cropper.js Guide – офіційний гайд по бібліотеці для обробки зображень. URL: https://fengyuanchen.github.io/cropperjs/guide.html (дата звернення: 21.03.2025).

9. SQL Injection Prevention Cheat Sheet – запобігання SQL-ін’єкціям. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 25.03.2025).