Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Пітіченко Антоніни Миколаївни*
(ПІБ)

академічної групи *122-19-4*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка інформаційної системи підтримки ігроків з використанням технологій ReactJS та Java/Spring*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Ширін А.Л.*			
розділів:				
спеціальний	*доц. Ширін А.Л.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-4* (група) *Пітіченко Антоніни Миколаївни* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інформаційної системи підтримки ігроків з використанням технологій ReactJS та Java/Spring*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Ширін А.Л.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Пітіченко А.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 90 с., 34 рис., 13 табл., 3 дод., 21 джерело.

Об'єкт розробки: інформаційна система для підтримки гравців.

Мета кваліфікаційної роботи: розробка інформаційної системи підтримки гравців з використанням технологій ReactJS та Java/Spring.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні додатка, що надає можливість користувачам отримувати доступ до структурованих каталогів гайдів, актуальних новин, створені та влитті у спільноту гравців, а також в утворенні персоналізованого досвіду.

Актуальність інформаційної системи визначається популяризацію візуальних новел як жанру та мобільних ігор, а також у бажанні грати в такі ігри з підказками для економії часу/ігрової валюти/передчасного знання наслідків певних виборів тощо.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ВЕБДОДАТОК, FRONT-END, BACK-END, КОРИСТУВАЧ, ВІЗУАЛЬНА НОВЕЛА, ГАЙД, КОМПОНЕНТ.

ABSTRACT

Explanatory note: 90 pp., 34 fig., 13 table, 3 appendix, 21 sources.

Object of development: information system to support players.

The goal of the qualification work: the development of an information system for supporting players, which will focus on the use of ReactJS and Java/Spring technologies.

In the introduction, the analysis and current state of the problem is considered, the purpose of the qualification work and the field of its application are specified, the justification of the relevance of the topic is given, and the statement of the task is clarified.

In the first part, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second part, available solutions are analyzed, platforms for development are selected, program design and development are performed, program operation, algorithm and structure of its functioning are described, as well as program calling and loading, input and output data are determined, and the composition of technical means parameters is characterized.

In the economic part, the labor intensity of the developed information system is determined, the cost of work on creating the program is calculated, and the time for its creation is calculated.

The practical value is in creating an application that allows users to access structured catalogs of guides, up-to-date news, created and infused into the player community, and in creating a personalized experience.

The relevance of the information system is determined by the popularization of visual novels as a genre and mobile games, as well as in the desire to play such games with hints to save time/game currency/premature knowledge of the consequences of certain elections, etc.

Keywords: INFORMATION SYSTEM, WEB APPLICATION, FRONT-END, BACK-END, USER, VISUAL NOVEL, GUIDE, COMPONENT.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – Hypertext Markup Language;

CSS – Cascading Style Sheets;

БД – база даних;

DOM – Document Object Model;

JS – JavaScript;

JSX – JavaScript XML;

JSON – JavaScript Object Notation;

ГАЙД – посібник з проходження.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій в останні роки мав значний вплив на різні сфери нашого життя, у тому числі і на ігрову індустрію. Від появи візуальних новел, які поєднують в собі літературу та графічне мистецтво, цей жанр став дуже популярним серед шанувальників мобільних ігор. Водночас, гравці, які користуються цими новаторськими іграми, часто стикаються з певними труднощами. У зв'язку з цим, ця кваліфікаційна робота має на меті розробку інформаційної системи підтримки гравців, яка буде зосереджена на використанні технологій ReactJS та Java/Spring. Цей проект спрямований на створення зручного та ефективного інструменту, що зможе надати користувачам допомогу та підтримку під час проходження історій у візуальних новелах.

Актуальність теми полягає в тому, що на сьогоднішній день є значна популяризація цього жанру ігор [1], а також те, що не існує однозначної та повноцінної інформаційної системи підтримки гравців в галузі візуальних новел. Тому розробка такої системи є актуальною та перспективною.

В результаті виконання даної кваліфікаційної роботи буде створена інформаційна система підтримки гравців в галузі візуальних новел, яка забезпечить зручний доступ до необхідної інформації та покращить їхній досвід користування та проходження гри.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

На сьогоднішній день не для кого не секрет, що ігри є невід'ємною частиною нашого життя. Вони мають дуже довгу історію розвитку [2], від самих примітивних до найсучасніших. Вони розвивалися разом з нашим суспільством, разом з різноманітними технологічними досягненнями та змінами у культурі і, відповідно до цього, мають дуже великий вплив на людей. Це проявляється у багатьох аспектах, таких як: технологічний розвиток, як вже було сказано раніше, соціальна взаємодія, що допомагає розвивати комунікативні навички, безпосередній вплив на наш настрій, наші емоції, наші хобі і тому подібне.

Зокрема у світі існує дуже велика кількість різноманітних жанрів ігор: аркади, шутери, стратегії, рольові [3]. Але серед усіх можливих жанрів я хочу виділити лише один – візуальні новели. Візуальні новели – це ігри, які займають особливе місце, оскільки вони мають унікальні підходи до наративу та геймплею. Головною особливістю візуальних новел є наявність великої кількості текстового контенту, супроводжуваного ілюстраціями та анімацією, що означає те, що вони здатні комбінувати в собі відразу декілька елементів: літературу та інтерактивність відеоігор. Головна мета візуальних новел – розповісти захоплюючу історію із можливістю впливати на розвиток сюжету через вибори [4].

Одна із ігор такого жанру – Romance Club від компанії Your Story Interactive. Це мобільний додаток, який можна завантажити із Google Play та AppStore. У грі наявні на даний момент тридцять одна історія різного жанру: від банальної романтики та пригод до фентезі, фантастики, трилерів та жахів. Додаток має свою власну ігрову валюту та різні ігрові події, які допомагають йому заохочувати грати далі старих гравців та привертати увагу нових.

Romance Club є досить популярною грою серед молодих людей. Тільки Google Play має більше десяти мільйонів завантажень і не має ніякого сумніву, що і AppStore досить близький до цієї цифри. По грі роблять багато різноманітного контенту, наприклад, відео-проходження, або створюють групи по інтересам та спілкуються у цих групах. Навіть розробники мають свою офіційну продукцію: футболки, чашки, біжутерію та інше.

Кожна історія у додатку має свій сюжет та свої особливості. У грі є наявність декількох різних шляхів (так званих «статів»). Зазвичай, вони поділяються на декілька категорій:

- характер головного героя;
- вплив або авторитет;
- деякий різновид магічних здібностей або стиль боя;
- інколи можливість накопичення певної валюти та її витрати, якщо її виявиться у достатній кількості.

До того ж кожна історія має свою систему побудови відносин з іншими ігровими персонажами, а також певні «таємні» особливості, як то рівень стресу для головного героя, якщо історія має напружену сюжетну лінію, або щось подібне.

Усе це дозволяє гравцеві створювати свою власну історію та йти по ній своїм унікальним шляхом, а також самому обирати, які персонажі будуть супроводжувати його головного героя на цьому шляху. І дивлячись за це все, стає зрозуміло, що така велика кількість текстового контенту та складність сюжету можуть викликати певні проблеми або запитання у гравців. Для цього у світі візуальних новел існують гайди.

Гайди по візуальним новелам – це певні посібники, які надають користувачеві детальну інформацію про геймплей, персонажів, наратив та механіку гри. Вони можуть включати:

- стратегії проходження;
- відповіді на складні завдання або головоломки;

- описи різних варіантів розвитку сюжету відповідно до обраного шляху;
- детальні описи персонажів, їхні характеристики;
- підкази щодо побудови відносин з ігровими персонажами, як то дружніх, так і романтичних;
- секрети та приховані подробиці історії, які можуть бути важливими для розуміння сюжету;
- попередження про вибори, які мають великий вплив на сюжет та багато іншого.

Тобто гайди є цінними ресурсами для гравців, які хочуть отримати повне задоволення від гри, зрозуміти всі можливості сюжету та впливати на його розвиток. Вони допомагають гравцям зробити кращі вибори, часто зекономити ігрову валюту та отримати найбільше задоволення від проходження.

Найкращим ресурсом для перегляду гайдів – є інформаційна система з відповідною тематикою, оскільки вона:

- може надавати гравцям корисну та структуровану інформацію. Тобто гайди можуть бути розділені по історіям, жанрам цих історій та/або по більш вузьким категоріям, що спрощує пошук та робить перегляд контенту більш комфортним;
- може мати підтримку гравців. Тобто система могла б відповідати на самі часті запитання, а також надавати поради та рекомендації. Гравці можуть звертатись туди з питаннями щодо гри, проблемами, з якими вони зіткнулися, і, відповідно, отримувати поради та рішення для подальшого продовження гри;
- може мати додатковий контент. Не обов’язково повинні бути наявні виключно гайди, а також можуть бути присутні, наприклад, новини по відповідній тематиці, які розповідають но оновлення гри, про нововведення, про інтерв’ю з розробниками або інше;
- може допомогти у створенні спільноти гравців. Це означає, що це також може стати місцем для спілкування гравців, обміну думками, порадами

та досвідом. Гравці можуть обговорювати гру, ділитися своїми враженнями та знайомитись з іншими людьми зі схожими інтересами.

Тому темою моєї кваліфікаційної роботи є розробка інформаційної системи підтримки ігроків з використанням технологій ReactJS та Java/Spring саме для гри Romance Club.

1.2. Призначення розробки та галузь застосування

Як об’єкт розробки розглядається інформаційна система, яка буде орієнтована на підтримку гравців конкретної гри – Romance Club. За допомогою цієї системи користувач буде мати можливість перегляду детального проходження кожної окремої серії, тобто перегляду гайдів. Сервіс має містити поділ гайдів по усім історіям, які зараз наявні у додатку, функцію швидкого пошуку конкретного проходження, а також повинен бути наявний розділ з новинами, які пов’язані з розважальним додатком.

Розроблена інформаційна система буде призначена для гравців гри Romance Club, для збільшення її популярності серед людей, а також особливо для збільшення популярності саме україномовного контенту по обраній грі.

1.3. Підстава для розробки

Підставами для розробки «виконання кваліфікаційної роботи» є:

- освітня програма «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» на дипломування денної форми навчання №350-с від 16.05.2023;
- завдання на кваліфікаційну роботу на тему: «Розробка інформаційної системи підтримки ігроків з використанням технологій ReactJS та Java/Spring».

1.4. Постановка завдання

Завданням цієї кваліфікаційної роботи є розробка інформаційної системи для підтримки гравців з використанням технологій ReactJS та Java/Spring. Система повинна надавати користувачам зручний доступ до детальних проходжень, персонажів, новин та інших можливих аспектів.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт має дотримуватися наступних функціональних вимог:

- формування вебсторінок для відображення відповідного контенту з серверної частини;
- управління цим контентом, тобто його редагування, видалення та додавання нового контенту (для адміністратора);
- управління акаунтами та коментарями, тобто можливість штрафувати та видаляти їх (для адміністратора);
- авторизація та реєстрація;
- залишення коментарів;
- оцінка коментарів;
- оцінка кожного контенту по рейтингу;
- можливість залишати скарги на контент та/або коментарі;
- певна система сповіщень як для користувача, який має акаунт, так і для адміністратора.

1.5.2. Вимоги до інформаційної безпеки

Певні вимоги інформаційної безпеки для коректної роботи інформаційної системи:

- стабільна робота системи;

- забезпечення конфіденційності даних користувача;
- валідація та надійне збереження даних;
- заборона доступу звичайному користувачу до функцій адміністратора;
- функції перевірки введених даних.

1.5.3. Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги для коректної роботи інформаційної системи на персональному комп'ютері користувача:

- процесор зі швидкістю тактування 3 ГГц тачотирма ядрами;
- 8 GB оперативної пам'яті;
- 256 GB вільного місця на жорсткому диску;
- стабільний доступ до мережі Internet;
- операційна система Windows/MacOs/ Linux;
- веббраузер Opera/Google Chrome/Microsoft Edge та інші.

1.5.4. Вимоги до інформаційної та програмної сумісності

У розробці інформаційної системи будуть використанні наступні засоби:

- Figma – це популярний інструмент для створення дизайну, макетів та інтерфейсів сайтів;
- ReactJS – це JavaScript-бібліотека, яка дозволяє створювати динамічні компоненти та забезпечити взаємодію з користувачем;
- Java/Spring – фреймворк мови програмування Java, який надає різноманітні інструменти та бібліотеки для спрощення процесу розробки додатків;
- MariaBD – база даних, яка пропонує широкий набір функцій для зберігання та обробки даних, включаючи підтримку структурованих та невеликих даних, текстових полів, зображень і т.д.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Результатом даної кваліфікаційної роботи є розроблена інформаційна системи підтримки гравців, призначення якої буде полягати у наданні користувачам зручного та цінного джерела інформації, порад та рекомендацій щодо геймплею, персонажів, сюжету та інших аспектів візуальних новел.

Основні функціональні призначення системи є наступними:

- навігація та структурування: система надає користувачам зручну навігацію та зрозумілу структуру пошуку. Користувачі можуть легко знаходити гайди та/або новини до конкретних історій;
- детальні описи історій: система надає докладні описи до кожної історії, включаючи інформацію про кількість епізодів, жанри, автора, сюжет, персонажів, геймплей та інші важливі аспекти. Користувачі можуть отримати повну картину про історію перед її вибором для проходження;
- рейтинг та коментарі: система дозволяє користувачам залишати рейтинги та коментарі до гайдів. Це допомагає іншим користувачам, а також власникам сайту оцінити якість та популярність кожного гайду, а також створити власну спільноту гравців;
- користувацькі облікові записи: система надає можливість користувачам створювати облікові записи, що дозволяють зберігати улюблені гайди, отримувати сповіщення по оновлення, бачити свої коментарі тощо. Це створює персоналізований досвід для кожного користувача;
- новини: система надає актуальні новини про новини, які пов’язані з Romance Club, включаючи оголошення про нові релізи, зміни та оновлення. Користувачі можуть слідкувати за останніми подіями і бути завжди в курсі.

Усі ці функції та можливості спільно створюють повноцінну систему для підтримки гравців. А також надають корисну інформацію, зручність,

безперервне спілкування та задоволення користувачам, одночасно сприяючи зростанню впливу мобільної гри Romace Club на ринку візуальних новел, а також популяризацію україномовного контенту по ній.

2.2. Опис застосованих математичних методів

Даний проект не розрахований на використання будь-яких складних математичних методів та функцій, які би потребували особливої уваги та опису у даній записці. Були використанні лише самі прості арифметичні методи для розрахування рейтингів коментарів та контенту на сторінках.

2.3. Опис використаних технологій та мов програмування

Front-end частина інформаційної системи була реалізована за допомогою JavaScript, React, React-Redux, React-Router, Tailwind CSS.

React є однією з найпопулярніших бібліотек JavaScript, яка призначена для розробки інтерфейсів користувача. React дозволяє розробникам будувати вебдодатки, які відповідають за відображення даних на сторінці в реальному часі. Вебсторінка, побудована з використанням React, складається з набору незалежних компонентів, кожен з яких відповідає за свою частину інтерфейсу. Компонент може містити HTML-код, CSS-стилі та JavaScript-логіку [5].

Ключовий принцип React – це використання віртуального DOM. Коли стан компонента змінюється, React створює віртуальне представлення сторінки в пам'яті, порівнює його з реальним DOM та визначає необхідні зміни, які потрібно внести. Після цього React здійснює зміни в реальному DOM лише тих елементів, які, відповідно, були змінені. Це дозволяє покращити продуктивність додатків і зменшити навантаження на браузер [6].

React також підтримує використання JSX – спеціального розширення JavaScript, що дозволяє писати HTML-подібний код прямо в JavaScript-файлах [7].

Redux є бібліотекою управлінням станом для JavaScript-додатків, особливо для тих, які використовують фреймворк React. Redux працює на основі концепцій одностороннього потоку даних. Це означає, що дані в додатку зберігаються в єдиному store. Компоненти React можуть отримувати доступ до даних зі store і оновлювати їх через «actions» – об'єкти, які описують зміни, що відбуваються у додатку. При зміні даних Redux використовує «reducers» [8]. Приклади використання Redux наведено на рисунках 2.1 - 2.2.

```
import { configureStore } from "@reduxjs/toolkit";
import { userReducers } from "../pages/auth/store";

export default configureStore({
    reducer: {
        user: userReducers
    }
})
```

Рис. 2.1. Простий приклад використання Redux

React Router – це бібліотека, яка надає засоби для маршрутизації в додатках React. Вона дозволяє легко визначати маршрути та управляти навігацією між сторінками додатку. Основна мета Router – забезпечити відображення відповідного компонента для кожного маршруту, який може бути присутній в додатку. Це дає змогу створювати багатосторінкові додатки зі змінним вмістом, де відповідний компонент відображається залежно від URL-адреси [9]. Приклад використання Router наведений на рисунку 2.2.

```
export default createBrowserRouter(createRoutesFromElements(
    <Route element={<Provider store={store} children={<Outlet />} />}>
        <Route path="/" element={<Layout />}>
            <Route index element={<Home />} />
            <Route path="stories" element={<Stories />} />
            <Route path="news" element={<News />} />
            <Route path="account-admin" element={<AdminAccount />} />
            <Route path="account-user" element={<UserAccount />} />
        </Route>

        <Route path="/login" element={<Login />} />
        <Route path="/signup" element={<SignUp />} />
    </Route>
))
```

Рис. 2.2. Простий приклад використання Redux та Router

Основні функціональності Router є наступними [10]:

- вкладені маршрути: дозволяє створювати складні структури маршрутизації, де окремі компоненти можуть бути відображені в залежності від глибини URL-шляху;
- динамічні маршрути: дозволяє визначити шаблон для маршруту, використовуючи параметри в унікальних частинах URL-адреси;
- переадресація та 404 сторінка: надає компоненти для переадресації і обробки сторінок, які не знайдено;
- програмна навігація: надає зручні методи для програмної навігації між сторінками.

Tailwind CSS – молодий, але дуже популярний CSS-фреймворк, який пропонує інноваційний підхід до стилізації вебдодатків. Він відрізняється від традиційних CSS-фреймворків тим, що зосереджується на наборі низькорівневих класів, які можна використовувати для швидкої розробки та кастомізації інтерфейсу. Використання Tailwind CSS з React додає гнучкості і продуктивності, що спрощує розробку стильного та сучасного інтерфейсу для вебдодатків [11]. Приклад використання Tailwind CSS у React наведений на рисунку 2.3.

```
const App = () => {
    return (
        <div className="bg-blue-500 p-4">
            <h1 className="text-white text-2xl">Привіт, Tailwind CSS і React!</h1>
            <p className="text-white mt-2">Це приклад використання Tailwind CSS разом з React.</p>
            <button className="bg-white text-blue-500 px-4 py-2 mt-4 hover:bg-blue-500 hover:text-white">
                Натисни мене!
            </button>
        </div>
    );
};
```

Рис. 2.3. Простий приклад використання Tailwind CSS у React

Основні переваги використання Tailwind CSS у React-проектах [12]:

- скорочений час розробки: надає широкий набір класів, які використовуються для стилізації елементів і компонентів. Завдяки цьому,

розробнику не потрібно витрачати час на написання власних CSS-стилів або пошук відповідних класів;

- гнучкість та кастомізація: дозволяє легко кастомізувати вигляд елементів, змінюючи значення класів або налаштовуючи конфігурацію;
- респонсивний дизайн: дозволяє легко застосувати класи, які працюють тільки на певних розмірах екрану, що дозволяє розробляти адаптивні інтерфейси без необхідності в додатковому CSS-коді.

Back-end частина інформаційної системи була реалізована за допомогою Java, Spring, MariaBD.

Java є однією з найпопулярніших і широко використовуваних мов програмування у світі. Вона відома своєю надійністю, масштабованістю та широким спектром застосувань [13].

Основні переваги Java є наступними:

- простота вивчення та використання: має простий і зрозумілий синтаксис, який полегшує розробку програм і зменшує ймовірність помилок, має чіткі синтаксичні правила і стислу семантику;
- об'єктно-орієнтований підхід: базується на об'єктно-орієнтованому підході, де основний акцент робиться на обробці даних (об'єктів) та взаємодії між ними;
- надійність: має вбудовану перевірку помилок під час компіляції, що дозволяє виявити помилки ще до виконання програми;
- незалежність від платформи: є платформою-незалежною мовою програмування. Це означає, що програми, написані на Java, можуть працювати на будь-якій підтримуваній платформі без необхідності змінювати код;
- динамічність та адаптивність: має багатий набір бібліотек та фреймворків, які полегшують розробку програм;
- мережеві можливості: має вбудовану підтримку мережевих операцій для створення мережевих додатків, таких як вебсервери, клієнт-серверні програми, протоколи зв'язку та інші.

Один із найпопулярніших фреймворків Java для розробки вебдодатків – це Spring Framework. Spring є високорівневим фреймворком, який надає розробникам потужні інструменти для створення ефективних та масштабованих додатків [14].

Основні переваги Java/Spring є наступними:

- швидка і легка розробка: надає простий та ефективний спосіб розробки додатків на Java;
- автоналаштування компонентів: автоматично налаштовує компоненти додатка, що дозволяє писати менше конфігураційного коду;
- широкий вибір плагінів: має велику кількість плагінів та доповнень, що полегшують роботу з базами даних, кешуванням, обробкою вебзапитів та багато іншого;
- легкий доступ до баз даних: надає простий спосіб взаємодії з різними базами даних, такими як MySQL, Oracle, MongoDB, Redis та інші;
- інтеграція з екосистемою Spring: інтегрується з іншими фреймворками та інструментами в екосистемі Spring, такими як Spring Boot, Spring Data, Spring Security та багато інших.

MariaDB є відкритою реляційною базою даних. Вона розроблена з метою забезпечити стабільну, швидку та надійну систему управління базами даних з великим набором функцій [15].

Основні переваги MariaDB є наступними [16]:

- висока продуктивність: пропонує оптимізований та швидкий движок бази даних, який забезпечує ефективну обробку запитів та високу швидкість роботи;
- розширена підтримка функцій: надає багатий набір функцій, що робить її потужним інструментом для розробки баз даних;
- висока надійність: пропонує механізми забезпечення високої доступності і надійності даних;
- велика спільнота та підтримка: має активну спільноту розробників, що надає підтримку, оновлення та вирішення проблем.

2.4. Опис структури системи та алгоритмів її функціонування

Структура представленої інформаційної системи (тобто схема руху або опис користувачів між сторінками або компонентами системи) у даній кваліфікаційній роботі є досить стандартною у порівнянні з будь-яким вебдодатком. Це зроблено для того, щоб користувачі відчували комфорт при використанні продукту та бачили перед собою інтуїтивно-зрозумілий інтерфейс.

Загалом структура є майже завжди однаковою, наявні лише незначні зміни при використанні системи авторизованого користувача та неавторизованого. Більш детально розглянути саму структуру та вище згадані зміни можна на рисунках 2.4 та 2.5, які представлені нижче.

На рисунку 2.4 видно, що структура системи складається з наступних частин:

- головної сторінки сайту;
- сторінки каталогу усіх історій, по яким наявні гайди, а також: сторінки з більш детальною інформацією про конкретну історію; сторінки з детальними проходженнями історії по сезонам та серіям; сторінки з детальним проходженням по відносинах з іншими персонажами;
- сторінка з переліком усіх новин, які пов’язані з діяльністю додатку або гри, з можливістю переходу на новину для більш детального ознайомлення з текстом;
- сторінки для авторизації та реєстрації користувачів, якщо вони бажають мати особистий кабінет.

Рис. 2.4. Структура системи неавторизованого користувача

Інформаційна система має шапку (так званий header), яка розміщена на кожній сторінці (окрім авторизації та реєстрації), що зображені на рисунку. Завдяки цьому користувач, з представленої вище структури, може завжди з будь-якого місця повернутися на три основні сторінки: головну, історії та новини, що спрощує переміщення по системі. Також шапка має перехід на сторінки, які призначені для створення або вхід у особистий кабінет, з яких можна повернутися тільки на головну.

Як вже було сказано, такий вигляд перед собою буде мати користувач, який не має акаунту на сайті або просто не авторизований. Натомість рисунок 2.5 показує структуру, яку буде мати перед собою вже зареєстрований користувач.

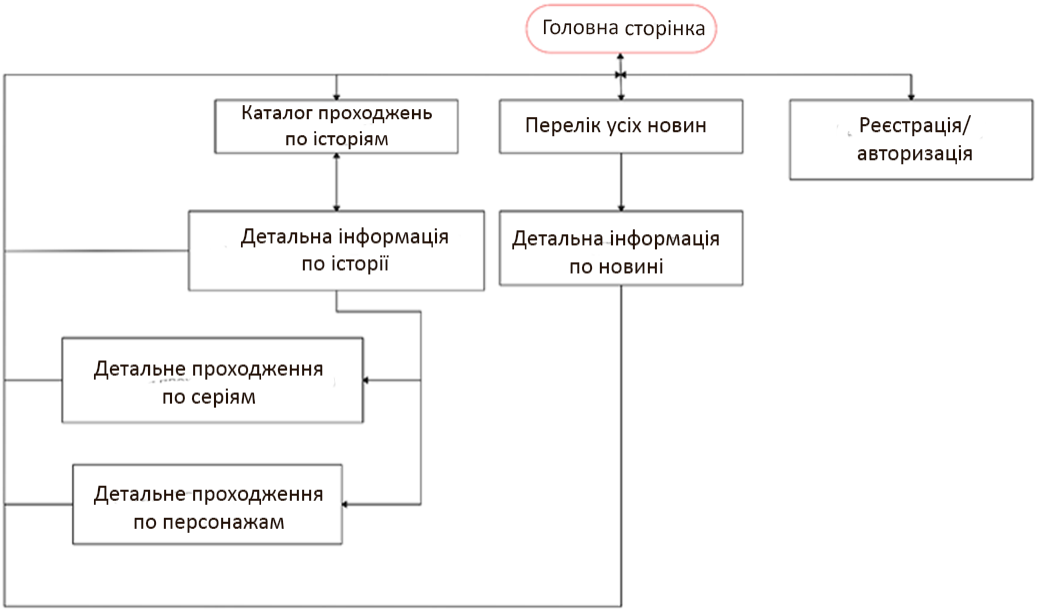

Рис. 2.5. Структура системи авторизованого користувача

Вигляд системи майже не змінився, за винятком того, що тепер користувач буде мати постійний доступ з кожної сторінки до свого особистого акаунту, для швидкого переходу до нього відповідно.

Нижче можна побачити рисунки 2.6 та 2.7 з Use Case діаграмами, які відображають функціонування системи та взаємодію користувача з нею відповідно до ролі.

Рисунок 2.6 демонструє взаємодію користувача з додатком, яке дозволяє йому мати наступні можливості:

- авторизацію або реєстрацію, якщо він бажає;
- детальний перегляд новин та гайдів та деякі додаткові можливості, а також: збереження окремих гайдів; оцінка кожного гайду/новини по рейтингу; залишення коментарів під гайдами/новинами та/або скарг;
- доступ до власного акаунту, якщо наявна реєстрація, а також деякі додаткові можливості: перегляд особистої інформації, можливість її зміни та запит на видалення акаунту; перегляд папки «збережене» для швидкого доступу до потрібних гайдів та видалення звідти непотрібних вже компонентів; перегляд власних коментарів, які коли-небудь були опубліковані на сайті;

перегляд сповіщень та їх розділ на прочитані та непрочитані, а також відмічення відповідних сповіщень як «почитані».

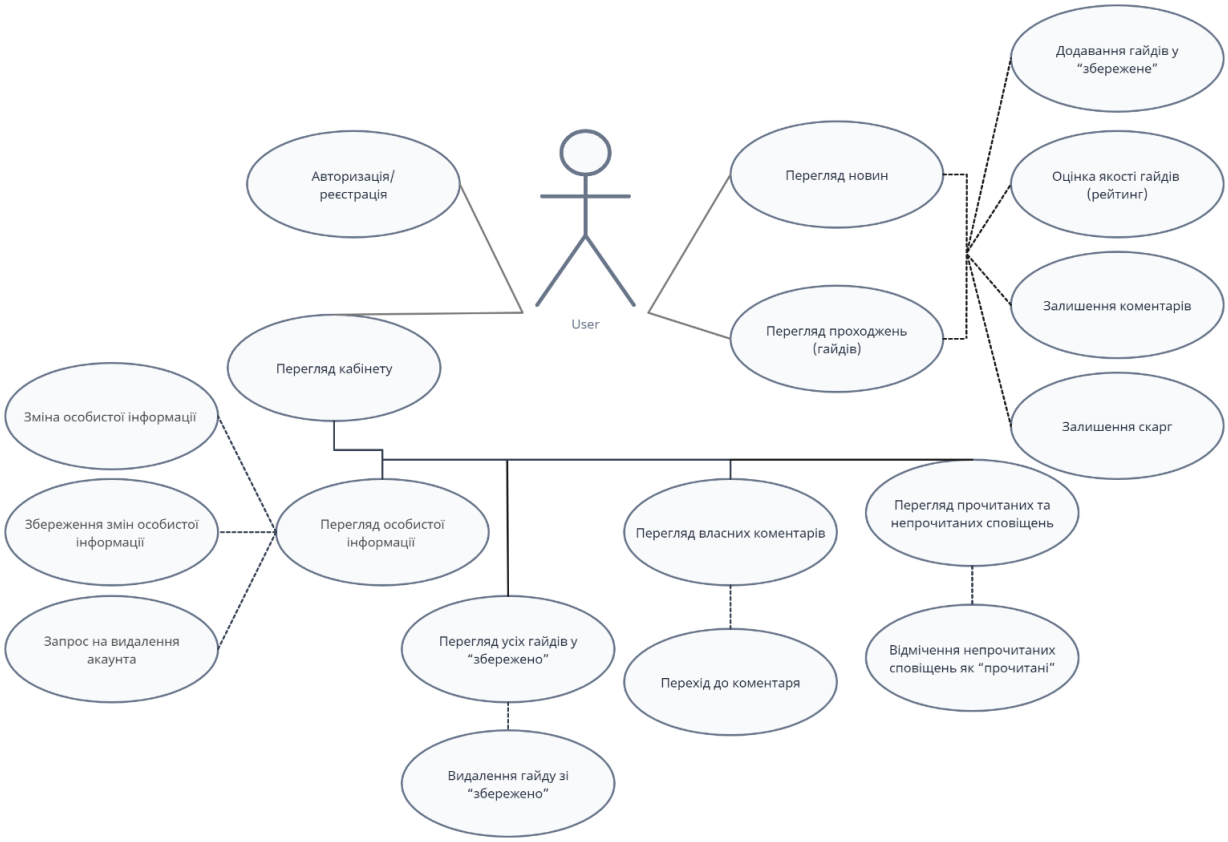

Рис. 2.6. Use Case діаграма звичайного користувача

Рисунок 2.7 демонструє натомість взаємодію та можливості вже адміністратора. Відповідно, деякі дії користувача такій ролі вже недоступні, але також у деякий випадках адміністратор має більш розширений спектр дій:

- перегляд новин/гайдів присутній, як і залишення оцінки або коментаря, але відсутня можливість зберігати контент та надсилати скарги;
- кабінет має більш розширений функціонал, аніж користувач, наприклад, з'являються наступні дії: перегляд усіх коментарів, які були опубліковані на сайті, а також акаунтів усіх користувачів з можливістю видалити ці компоненти або оштрафувати при порушенні певних правил; також додається можливість управління контентом сайту: його редагування, видалення та/або додавання нового.

Увесь інший функціонал перекликається з користувальницьким.

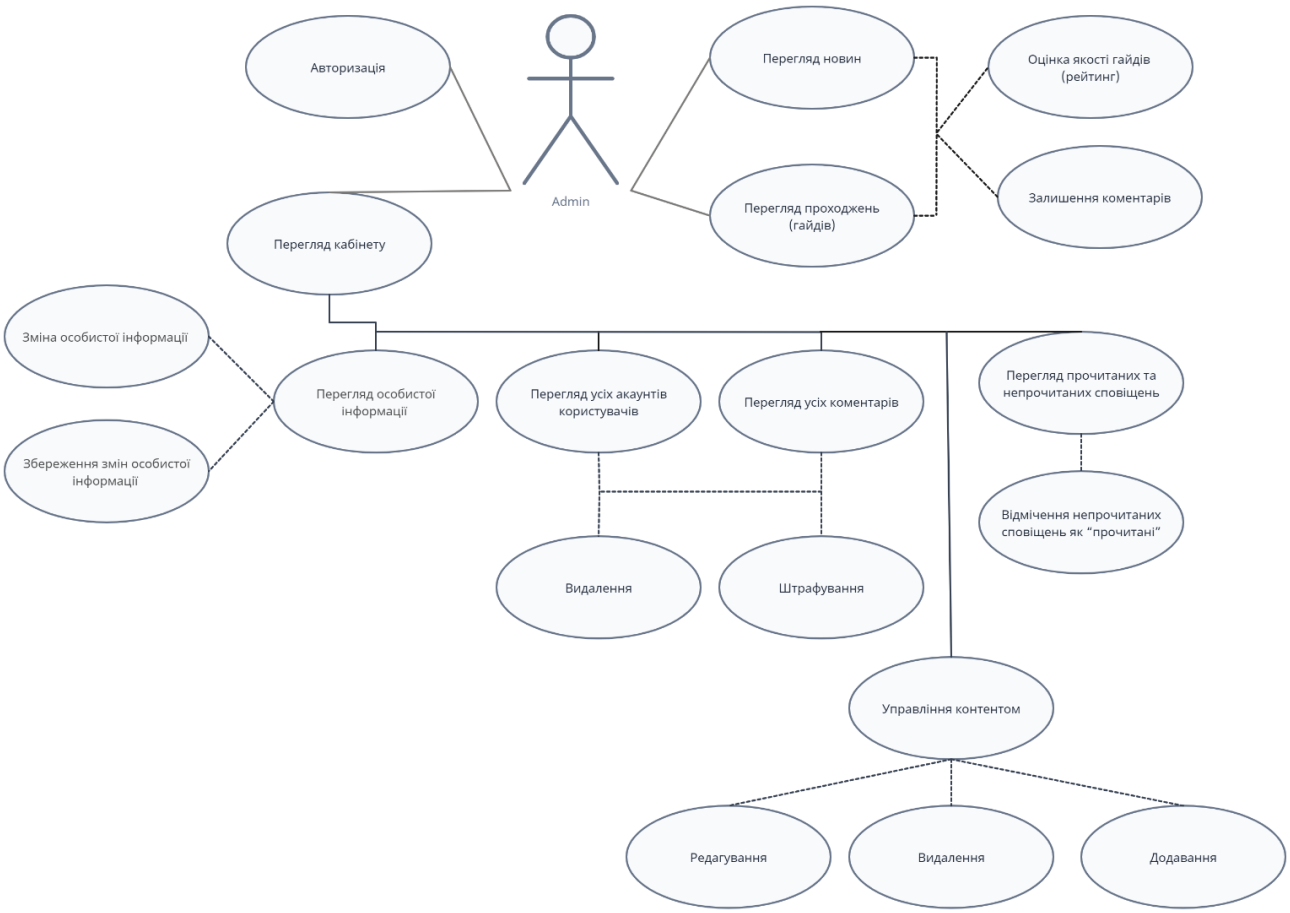

Рис. 2.7. Use Case діаграма адміністратора

Також є дуже важливим представлення логічної структури БД (бази даних). Структура бази даних є описом організації даних у БД без урахування фізичної реалізації. Діаграма логічної структури БД є графічним уявленням цієї структури. Відповідно діаграму структури БД можна побачити на рисунку 2.8.

Вона допомагає візуалізувати схему бази даних та зрозуміти, які сутності існують, як вони пов'язані та які атрибути вони мають. Це важливий інструмент при проектуванні та аналізі баз даних. Більш детально ця структура буде розібрана у наступному пункті.

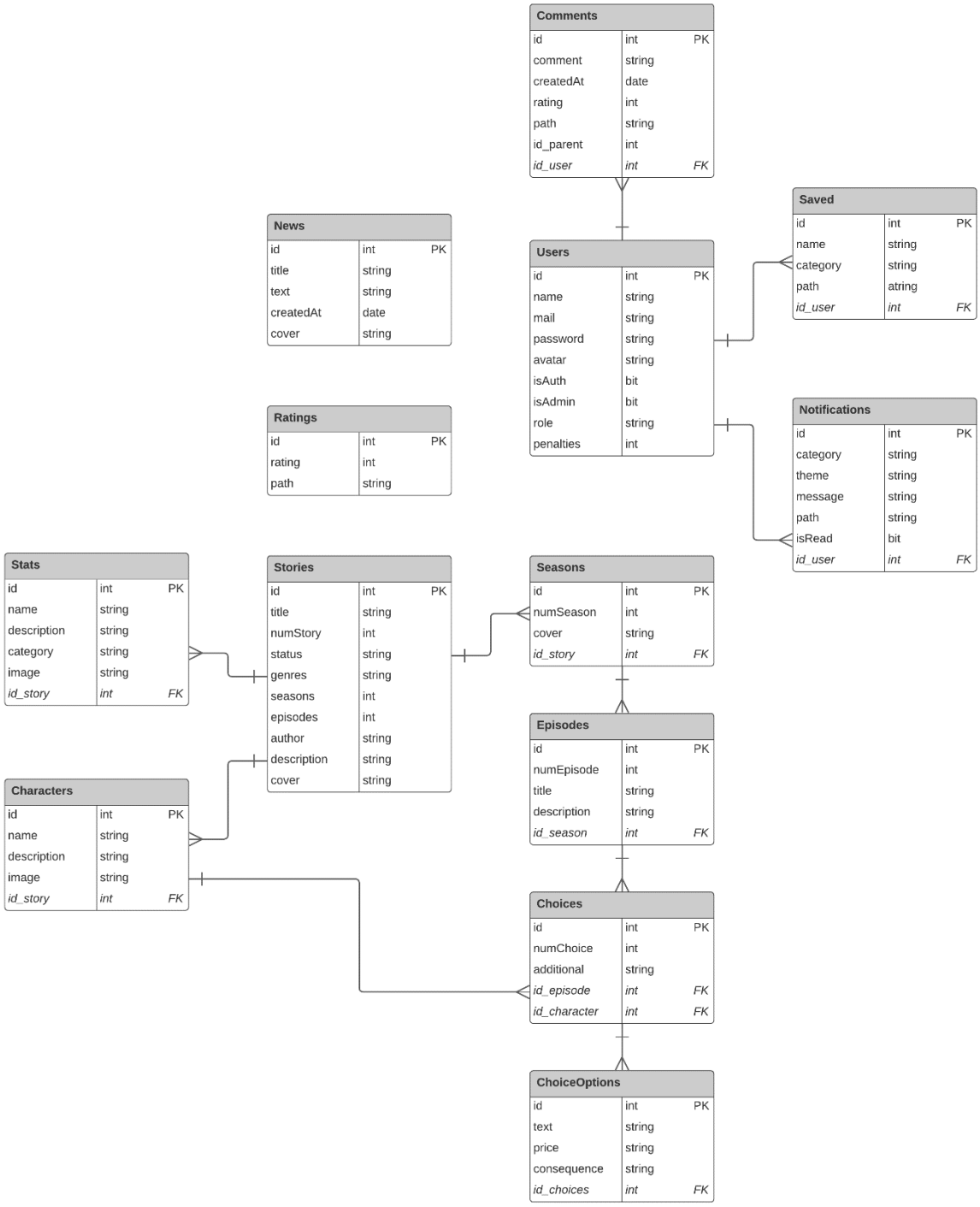

Рис. 2.8. Логічна структура БД

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними програми є дані, які користувач надсилає на сервер для обробки, збереження та отримання конкретної інформації.

Основна точка входу в програму реалізована за допомогою React-компонентів, які отримують дані від користувача через введення у форму або за допомогою натискання на певні кнопки. Далі ці дані передаються до back-end частини додатку. На back-end стороні, обробляються вхідні дані, передаються та зберігаються до бази даних.

Для отримання вихідних даних back-end частина програми виконує запит до бази даних для отримання інформації. Результат запиту є JSON файл з конкретними даними, який надсилається до клієнтської частини для відображення. Ці дані приймаються у вигляді властивостей (props) і використовуються для побудови інтерфейсу користувача.

На даний момент у базі даних існує декілька схем сутностей (таблиць з даними) з прописаними у них атрибутами, типами даних та зв’язками.

У таблиці 2.1 представлено дані полів БД «Users» з основною інформацією про зареєстрованого у системі користувача: ім’я, пошта, пароль, аватар, значення чи авторизований та чи адміністратор, роль та кількість штрафів. Це є вхідні дані, які вносяться під час реєстрації користувача.

У таблиці 2.2 представлено дані полів БД «Stories» з основною інформацією про каталог історій системі: назва, порядковий номер, статус (завершена чи ні), жанри, кількість сезонів та серій, автора, короткий опис та обкладинку. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

Таблиця 2.1

Поля БД «Users»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
name	String	required
mail	String	required, unique
password	String	required
avatar	String	-
isAuth	Bit	0/1
iaAdmin	Bit	0/1
role	String	user/admin
penalties	Int	0/1/2/3/4/5

Таблиця 2.2

Поля БД «Stories»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
title	String	required, unique
numStory	Int	required, unique
status	String	completed/incomplete
genres	String	-
seasons	Int	-
episodes	Int	-
author	String	-
description	String	-
cover	Spring	-

У таблиці 2.3 представлено дані полів БД «Characters» з даними по персонажам, з якими можна взаємодіяти у історії. Основна інформацію у

системі наступна: імя персонажа, короткий опис, зображення персонажа. Також присутній зв'язок M:1 з атрибутом Stories, оскільки декілька персонажів можуть належати лише одній історії, а одна історія може мати декілька персонажів. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

У таблиці 2.4 представлено дані полів БД «Seasons» з даними по сезонам конкретної історії, яка містить наступну основну інформацію у системі: номер сезону та обкладинку сезону. Також присутній зв'язок M:1 з атрибутом Stories, оскільки декілька сезонів можуть належати одній історії, а кожна історія може мати декілька сезонів. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

Таблиця 2.3

Поля БД «Characters»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_story	Int	required
name	String	required
description	String	-
image	String	-

Таблиця 2.4

Поля БД «Seasons»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
Id_story	String	required
numSeason	Int	required
cover	String	-

У таблиці 2.5 представлено дані полів БД «Episodes» з даними по серіям історій, яка містить наступну основну інформацію у системі: номер епізоду, його назва та короткий опис. Також присутній зв'язок M:1 з атрибутом Seasons, оскільки декілька епізодів можуть належати лише одному сезону, а один сезон може мати декілька епізодів. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

У таблиці 2.6 представлено дані полів БД «Choices»з даними по типу вибору, яка містить наступну основну інформацію у системі: номер вибору, а також деяку додаткову інформацію, якщо вона потрібна. Також присутній зв'язок M:1 з атрибутом Episodes, оскільки декілька виборів можуть належати лише одній конкретній серії, а одна серія може мати декілька виборів. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

У таблиці 2.7 представлено дані полів БД «ChoiceOptions» з даними по виборам, які можна здійснити у історії. Ця таблиця містить наступну основну інформацію у системі: опис вибору, можливу ціну (якщо наявна) та наслідки вибору. Також присутній зв'язок M:1 з атрибутом Choices, оскільки декілька варіантів виборів можуть належати лише одному конкретному вибору, а однин вибір може мати декілька різних параметрів. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

Таблиця 2.5

Поля БД «Episodes»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_season	Int	required
numEpisode	Int	required
title	String	-
description	String	-

Таблиця 2.6

Поля БД «Choices»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_episode	Int	required
id_character	Int	-
numChoice	Int	required
additional	String	-

Таблиця 2.7

Поля БД «ChoiceOptions»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_choices	Int	required
text	String	required
price	String	-
consequence	String	-

У таблиці 2.8 представлено дані полів БД «News» з даними по детальній інформації по новині, яка містить наступну основну інформацію у системі: заголовок новини, основний текст новини, дату публікації та обкладинку новини. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

У таблиці 2.9 представлено дані полів БД «Comments» з даними по коментарях, які знаходяться на сайті. Ця таблиця містить наступну основну інформацію у системі: текст коментаря, яким користувачем він був залишений, коли був створений, його рейтинг, шлях (тобто на якій сторінці він знаходиться), а також чи є він відповіддю на інший коментар. Також присутній зв'язок M:1 з атрибутом Users, оскільки декілька коментарів можуть належати лише одному конкретному користувачу, а однин користувач може мати

декілька різних коментарів. Це є вхідними даними, які залишають користувачі, а також вихідними для відображення на сторінці.

Таблиця 2.8

Поля БД «News»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
title	String	required
text	String	required
createdAt	Date	required
cover	String	-

Таблиця 2.9

Поля БД «Comments»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_user	Int	required
comment	String	required
createdAt	Date	required
rating	Int	-
path	String	required
id_parent	Int	-

У таблиці 2.10 представлено дані полів БД «Ratings» з даними по рейтингу кожної статті (гайду або новині), яка знаходиться на сайті. Ця таблиця містить наступну основну інформацію у системі: значення самої оцінки (тобто рейтингу), а також шлях (тобто на якій сторінці вона знаходиться). Це є вхідними даними, які залишають користувачі, а також вихідними для відображення на сторінці.

У таблиці 2.11 представлено дані полів БД «Stats» з даними по різним параметрам (статам), які дозволяють проходити історії по різним шляхам та варіаціям. Ця таблиця містить наступну основну інформацію у системі: назву стату, його опис, категорію, а також зображення стату. Також присутній зв'язок M:1 з атрибутом Stories, оскільки декілька статів можуть належати лише одній конкретній історії, а одна сторія може мати декілька різних статів. Це є вхідними даними, які вносить адміністратор, а також вихідними для відображення на сторінці.

У таблиці 2.12 представлено дані полів БД «Saved» з даними по гайдам та/або новинам, які знаходяться у користувачів в акаунті у «збережено» для більш швидкого доступу. Ця таблиця містить наступну основну інформацію у системі: назву гайду або заголовок новини, категорію та шлях до сторінки. Також присутній зв'язок M:1 з атрибутом Users, оскільки декілька елементів із «збережено» можуть належати лише одному конкретному користувачу, а однин користувач може мати декілька різних елементів у «збережено». Це є вихідними даними для відображення на сторінці.

Таблиця 2.10

Поля БД «Ratings»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
rating	Int	required
path	String	required

Таблиця 2.11

Поля БД «Stats»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_story	Int	required
name	String	required
description	String	-
category	String	character/ability/authority
image	String	-

Таблиця 2.12

Поля БД «Saved»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_user	Int	required
name	String	required
category	String	story/guide/news
path	String	required

У таблиці 2.13 представлено дані полів БД «Notifications» з даними по сповіщенням, які можуть бачити у себе користувачі. Ця таблиця містить наступну основну інформацію у системі: категорію сповіщення, тему та саме повідомлення сповіщення, шлях (якщо ще сповіщення про оновлення або відповідь на коментар, а також позначення чи сповіщення є прочитаним. Також присутній зв'язок M:1 з атрибутом Users, оскільки декілька сповіщень можуть належати лише одному конкретному користувачу, а однин користувач може мати декілька різних сповіщень. Це є вихідними даними для відображення на сторінці.

Таблиця 2.13

Поля БД «Notifications»

Назва	Тип даних	Обмеження
id	Int	primary, required, unique
id_user	Int	required
category	String	required
theme	String	comment/penalty/request/ update/complaint
message	String	required
path	String	-
isRead	Bit	0/1

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби.

Для розробки інформаційної системи була використана персональна електронно-обчислювальна машина (ЕОМ) з наступними характеристиками:

- операційна система: Windows 10;
- тип системи: 64-розрядна операційна система, процесор на базі архітектури x64;
- процесор: Intel® Core i5-9300H CPU @ 2.40GHz;
- ОЗП: 8 ГБ;
- відеокарта: Intel® UHD Graphics 630.

А також додаткові засоби вводу-виведення інформації: комп’ютерна миша, клавіатура, робочий монітор.

2.6.2. Використані програмні засоби.

Для розробки інформаційної системи були використанні наступні програмні засоби:

- Figma Desktop: це програма для дизайну та прототипування, яка дозволяє створювати високоякісні вебдизайни, макети інтерфейсів користувача та спільно працювати з командою над проектами. Вона надає широкі можливості для створення та редагування графічних елементів, включаючи векторні зображення, інтерактивні компоненти та анімацію. Figma також має можливість спільно працювати над проектами в реальному часі, дозволяючи команді спільно редагувати та коментувати дизайни [17];

- Microsoft Paint: це базовий растровий графічний редактор, який протягом багатьох років є частиною операційної системи Microsoft Windows. Він надає прості інструменти для створення та редагування зображень, таких як пензлі, форми, текст і кольори. MS Paint в основному використовується для основних завдань обробки зображень, таких як кадрування, зміна розміру та малювання простих форм;

- Visual Studio Code: це безкоштовне і легке у використанні середовище розробки, розроблене компанією Microsoft. Воно надає потужні можливості для написання, редагування та налагодження коду в різних мовах програмування. VS Code має багато корисних функцій, таких як автодоповнення коду, керування версіями, інтегровані термінали та підтримка розширень, що дозволяють розширити його можливості та налаштувати під потреби розробника [18];

- HeidiSQL: це безкоштовний клієнт для управління базами даних MySQL, MariaDB, Microsoft SQL Server та PostgreSQL. Він надає інтуїтивно зрозумілий інтерфейс для взаємодії з базами даних, дозволяючи виконувати різні операції, такі як створення таблиць, запити SQL, імпорт та експорт даних, настроювання прав доступу та багато іншого. HeidiSQL також має підтримку роботи з віддаленими серверами баз даних і надає зручний спосіб взаємодії з ними [19].

2.6.3. Виклик та завантаження програми.

Для ініціювання та розгортання необхідно виконати наступні кроки:

- забезпечити наявність підходящого по технічним характеристикам серверного обладнання та доменного імені, шляхом оренди або придбання;
- встановити на серверне обладнання необхідне для роботи серверне програмне забезпечення;
- налаштувати серверне програмне забезпечення відповідно до наявних потреб;
- використовуючи FTP клієнт, підключитися до сервера та перенести всі скомпільовані файли програми;
- підключити доменне ім'я та встановити необхідні сертифікати на сервері;
- увімкнути сервер, щоб розпочати його роботу.

2.6.4. Опис інтерфейсу користувача.

При відкритті додатку користувач спершу потрапляє на головну сторінку, зображення якої можна побачити на рисунку 2.9, та бачить основну інформацію, яка може знадобитися для швидкого пересування по останнім оновленим елементам або для ознайомлення для тих користувачів, які ще не відома гра та її геймплей. Інтерфейс є інтуїтивно зрозумілим та має комфортну та ніжну кольорову палітру для очей.

Рис. 2.9. Головна сторінка

На кожній сторінці, окрім сторінок з формами для реєстрації та авторизації, знаходиться шапка (header) з декількома основними елементами: текстовим логотипом сайту, головною сторінкою, історіями, новинами та акаунтом.

«Історії» та «Новини» представляють собою каталог усіх історій та новин відповідно, які на даний момент наявні у базі даних. Якщо перейти на сторінку з історією або новиною, то можна побачити більш детальну інформацію про

відповідний елемент. Також історії мають додаткові сторінки з гайдами, які розбитті по сезонам, та гайдами з персонажами, з якими доступна взаємодія у грі.

Зображення відповідних сторінок, які описані вище, можна побачити на рисунках 2.10 - 2.17.

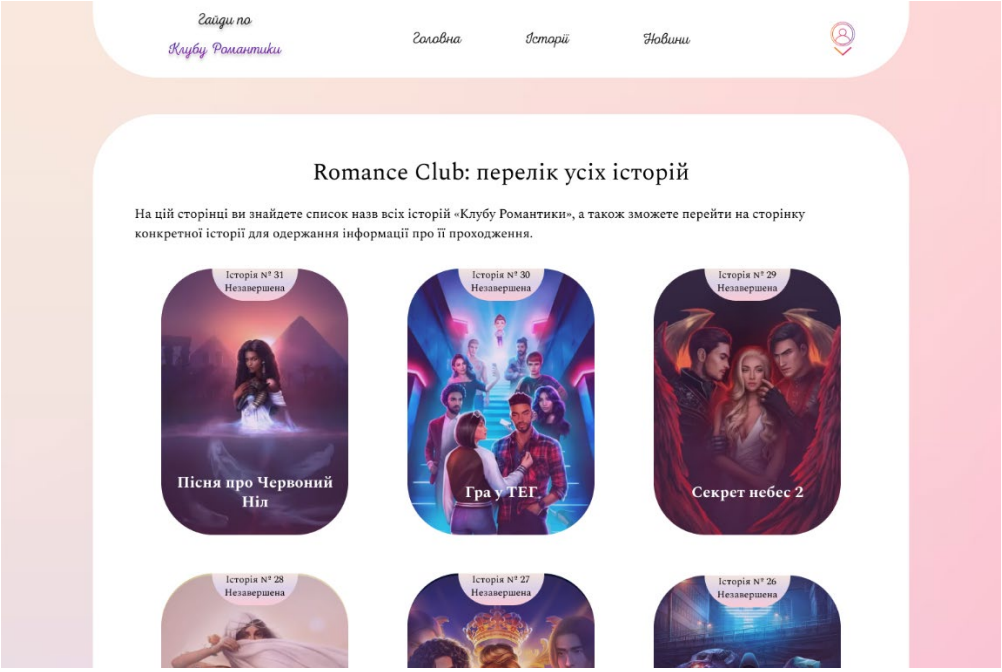

Рис. 2.10. Сторінка з каталогом історій

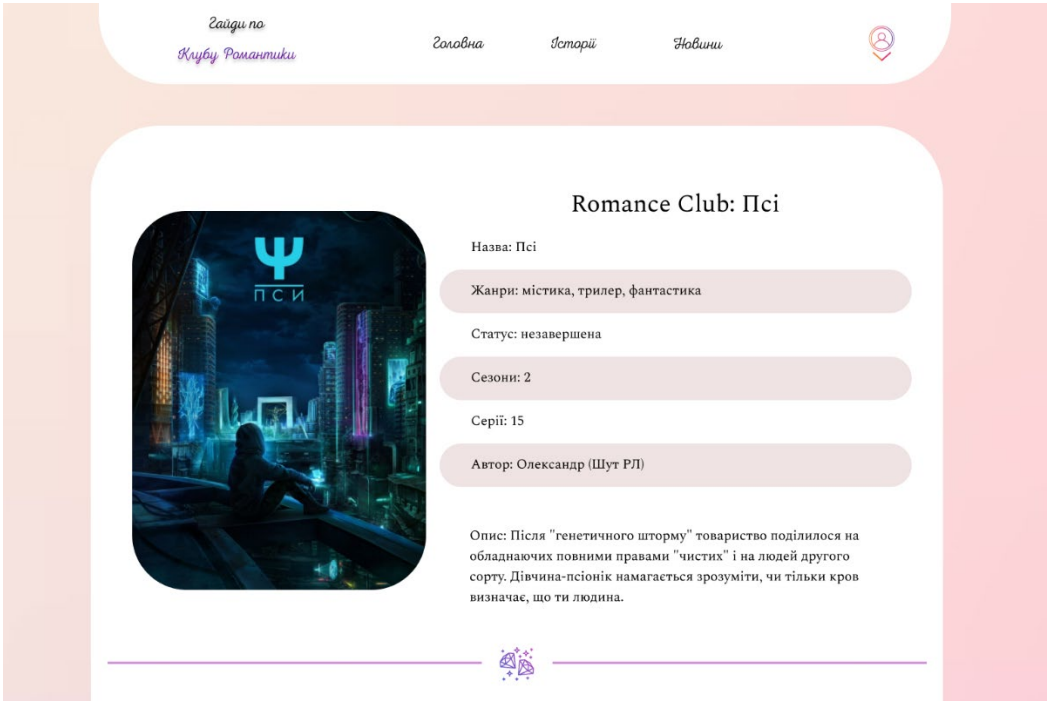

Рис. 2.11. Сторінка з детальною інформацією про історію

Особливості ігрового процесу:

Інформація про стати:

Імпульс: стат, який відповідає за характер; головна героїня більш емоційна та іронічна, відає перевагу духовним цінностям.

Контроль: стат, який відповідає за характер; головна героїня більш спокійна та стримана, відає перевагу матеріальним цінностям.

Наявність балансу: відсутній

Форма: стат, який відповідає за здібності; дозволяє головній героїні спотворювати та деформувати об'єкти.

Вектор: стат, який відповідає за здібності; дозволяє головній героїні рухати об'єкти по прямій лінії від себе або до себе.

Наявність балансу: відсутній

Комбінація статів: будь-яка.

Інформація про фаворитів:

- Кей (перша поява (1x01)
- Даніель (перша поява 1x05)
- Йонас (перша поява 1x01)
- Іво (перша поява 1x01)

Рис. 2.12. Сторінка з більш детальною інформацією про історію

Покрокові гайди по сезонам:

Гайди по фаворитам:

Рис. 2.13. Сторінка з інформацією про кількість проходжень та персонажів

Зміст

- Серія 1
- Серія 2
- Серія 3
- Серія 4

Серія 1 - Доказів достатньо

Лу розповідає, як все сталося.

Вибір №1: *(можна поставити всі запитання)*

- Можна мені кави? - **Нічого.**
- Як його вбили? - **Закриваємо діалог.**
- Навіщо мені когось вбивати? - **Нічого.**

Вибір №2:

- Навіщо? - +**1 контроль.**
- У в'язницю не відправите? - +**1 імпульс.**

Вибір №3: *(вибір одягу)*

- Фіолетовий худі - **Нічого.**
- В'язаний худі (**34 алмази**) - **Нічого.**
- Комбінезон із замком (**56 алмазів**) - **Нічого.**

Вибір №4:

Рис. 2.14. Сторінка з варіантами відповіді на діалоги та наслідками

Серії: 15

Автор: Олександр (Шут РЛ)

Опис: Після "генетичного шторму" товариство поділилося на обладнаючих повними правами "чистих" і на людей другого сорту. Дівчина-псіонік намагається зрозуміти, чи тільки кров визначає, що ти людина.

Кей Стоун - персонаж історії "Псі"

Кей Стоун - емпатик (зчитує емоційний стан людини за допомогою дотиків), співробітник Корпуса Діяльності, напарник та друг/фаворит для главної героїні історії. У цій статті представлені вибори для прокачки саме **романтичних стосунків**.

Зміст

Сезон 1:

- Серія 1
- Серія 2
- Серія 3

Сезон 1 - серія 1

Вибір №1:

- Ходімо (**27 алмазів**) - Побачимо сцену спілкування з Кеєм в будинку Лу. У 3 серії поліпшимо відносини з Кеєм.

Рис. 2.15. Сторінка з гайдом по персонажу

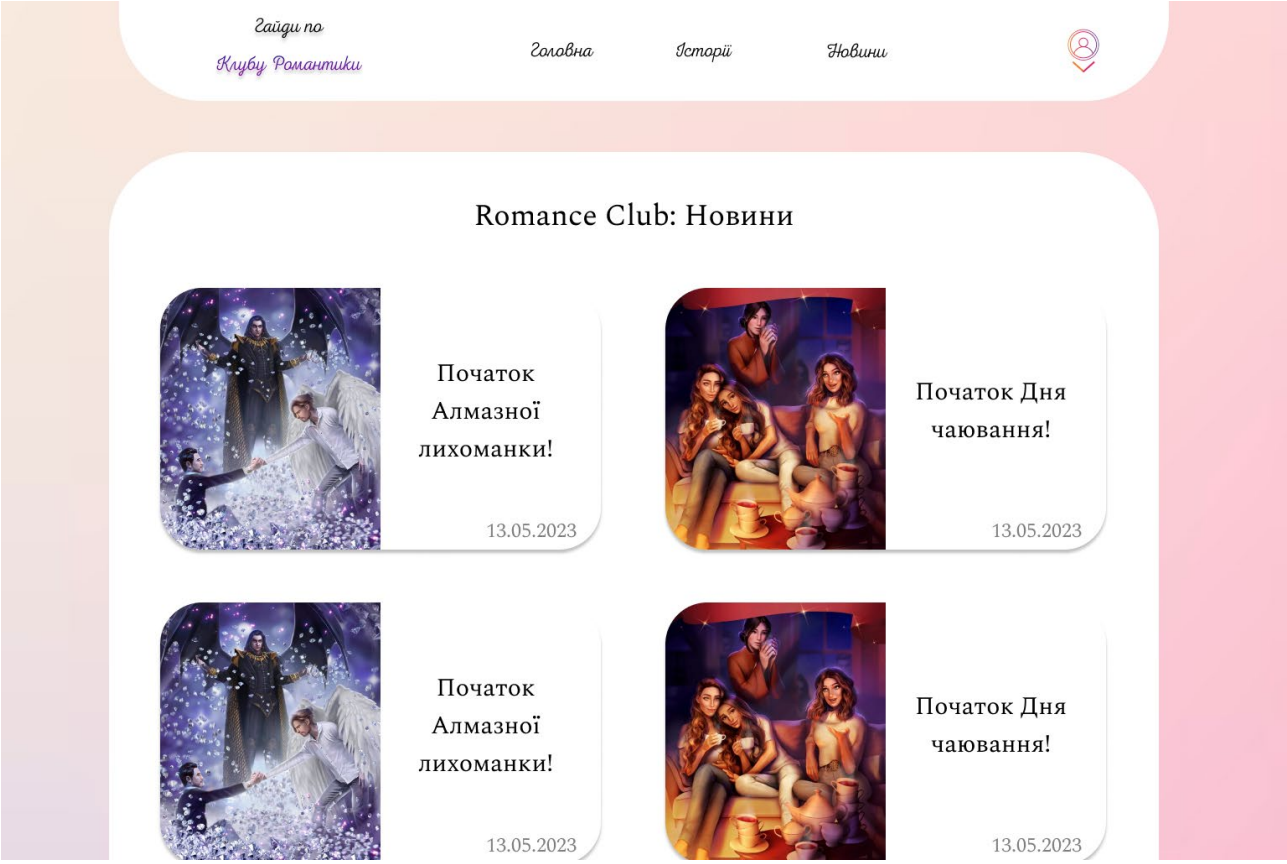

Рис. 2.16. Сторінка з каталогом новин

Кожна сторінка з гайдом та новиною має деякі додаткові функції такі як:

- оцінка контенту: користувачі можуть оцінити надану інформацію, вказавши наскільки якісною вони її вважають, а також побачити загальний рейтинг контенту, який розраховується по всім наданим оцінкам. Оцінка доступна лише зареєстрованим користувачам, незареєстровані можуть лише бачити загальний рейтинг;

- додавання у збережено: додавання статті у збережено, для швидкого доступу до неї зі свого акаунту. Ця функція також доступна лише авторизованим користувачам;

- кнопка скарги: відкриття форми та посилання скарги адміністратору, якщо користувач знайшов помилку або має інші претензії.

Зображення наведених вище функцій можна побачити на рисунках 2.17 - 2.18.

Рис. 2.17. Вигляд рейтингу та додаткових скарги і додавання у збережено

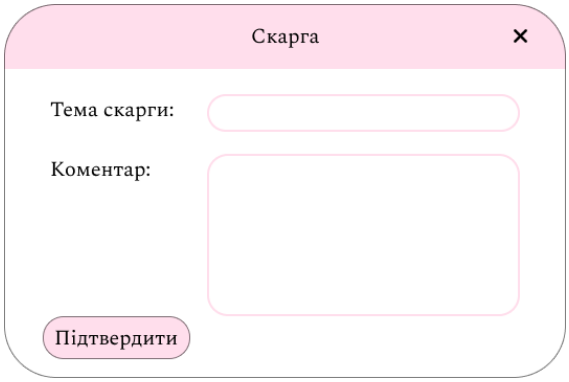

Рис. 2.18. Форма для скарги на контент або для штрафування

Також сторінки з новинами та гайдами мають у кінці розділ з коментарями. Користувачі можуть залишити свою думку та вступити у дискую з іншими відвідувачами сайту. Окрям того, що на коментарі можна відповідати, їх також можна оцінювати та/або скаржитися, якщо вважається, що коментар порушує правила спільноти.

Зображення наведених вище функцій можна побачити на рисунках 2.19 - 2.20.

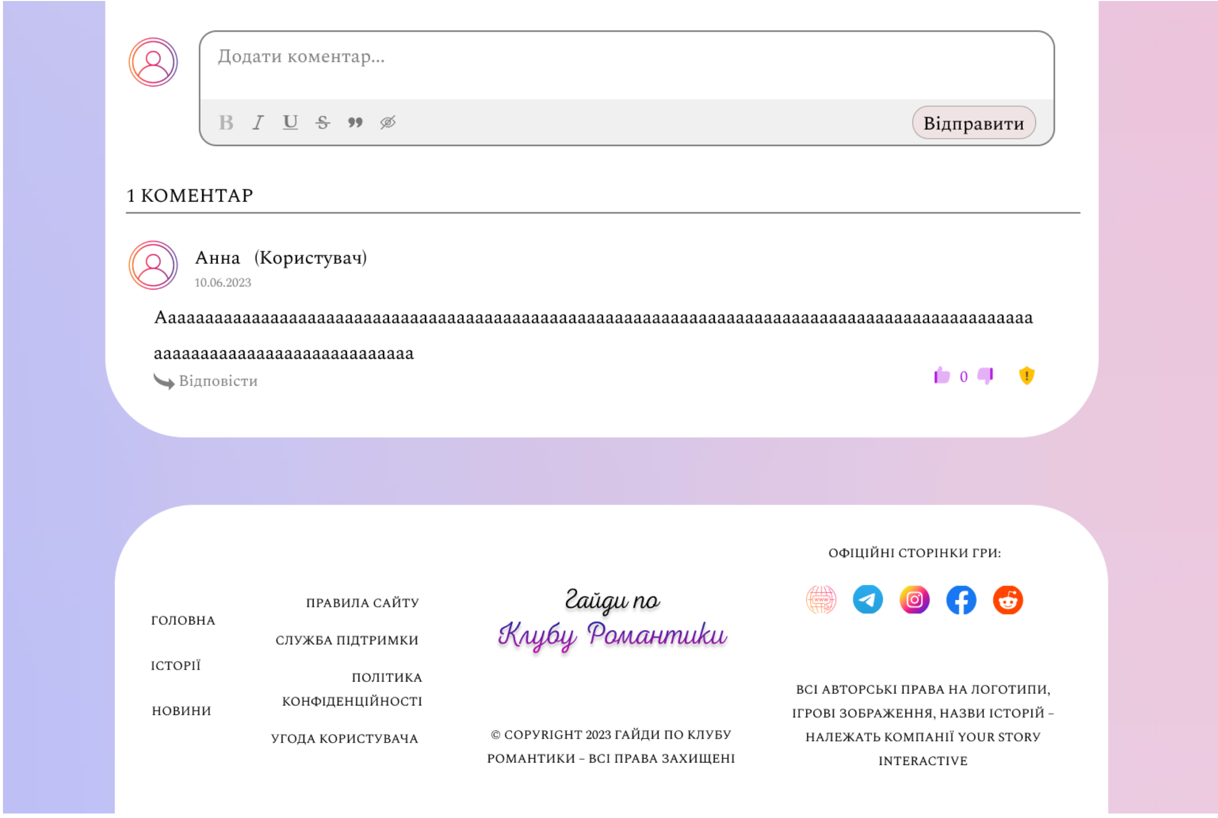

Рис. 2.19. Вигляд коментарів та підвалу

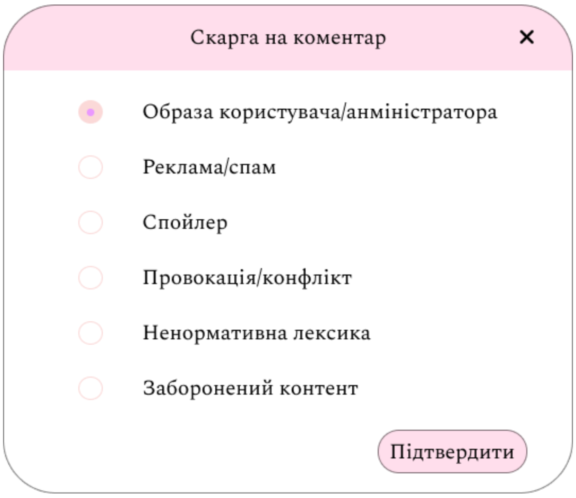

Рис. 2.20. Форма для скарги на коментар

Також на кожній сторінці, окрім сторінок з формами для реєстрації та авторизації, знаходиться підвал (footer) з декількома основними елементами: повтором елементами з шапки, деякі відомості про правила сайту, посилання на офіційні сторінки гри тощо. Зображення підвалу можна побачити на рисунку 2.19.

Як було сказано раніше, шапка сайту містить посилання на акаунт. Це посилання може набувати двох різних виглядів:

- для незареєстрованих та неавторизованих користувачів буде з’являтися випадаючий список з пропозицією авторизації або реєстрації;
- якщо користувач авторизований, то він побачить випадаючий список з пропозиціями перейти до особистого кабінету або вийти з акаунту.

При вибору варіантів з авторизацією або реєстрацією користувач буде переправлений на відповідні форми. Для реєстрації користувач мусить вказати бажане ім’я, яке буде відображатися на сайті, пошту та пароль. Авторизація здійснюється за існуючими поштою та паролем у базі даних системі. В іншому ж випадку користувач зможе перейти до сторінки з особистим кабінетом.

Зображення наведених вище функцій можна побачити на рисунках 2.21 - 2.24.

Рис. 2.21. Випадаючий список неавторизованого користувача

Рис. 2.22. Форма авторизації

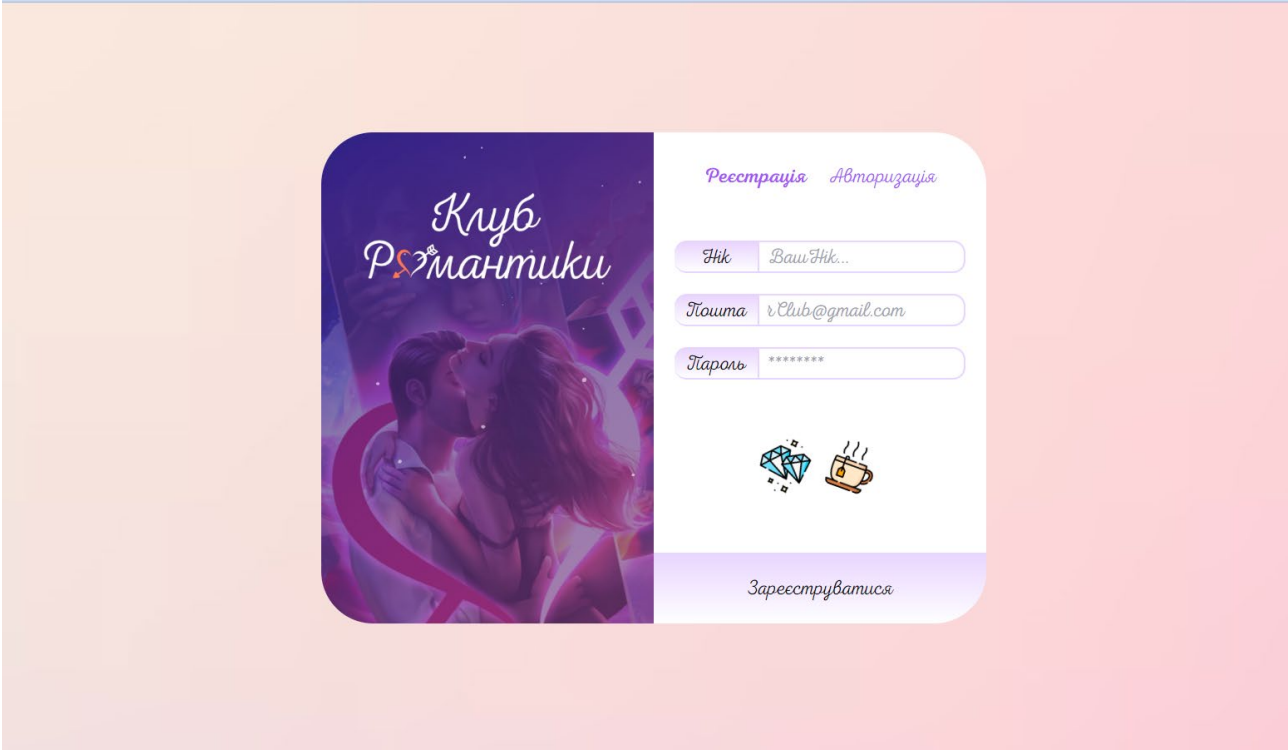

Рис. 2.23. Форма реєстрації

Рис. 2.24. Випадаючий список авторизованого користувача

Вигляд особистих кабінетів відрізняється в залежності від ролі користувача у системі. Звичайний відвідувач має наступний функціонал:

- особисті дані: дані, які були введенні під час реєстрація. По бажанню, їх можна змінити. Крім цього є можливість завантажити зображення аватару та побачити невелику статистику про кількість власних коментарів на сайті, гайдів у збережено, а також кількість штрафних балів. Також можна відправити запит на видалення акаунту адміністратору;
- моє збережене: тут відображається список статей, які були додані користувачем. Це дозволяє швидко перейти до них та скорочує час пошуку. Також можна здійснити пошук конкретної статті;
- мої коментарі: тут відображається список власних коментарів користувача з можливістю швидкого переходу до сторінки, на якій вони знаходяться. Також можна здійснити пошук конкретного коментаря;
- мої сповіщення: тут знаходиться список сповіщень. Користувач може отримувати наступні сповіщення: сповіщення про штрафи від адміністратора, сповіщення про оновлення статей, які знаходяться у «збережено», а також сповіщення про відповіді на коментарі. Сповіщення поділяються на прочитані та непрочитані, відповідно Також можна здійснити пошук конкретного сповіщення.

Зображення наведених вище функцій можна побачити на рисунках 2.25 - 2.28.

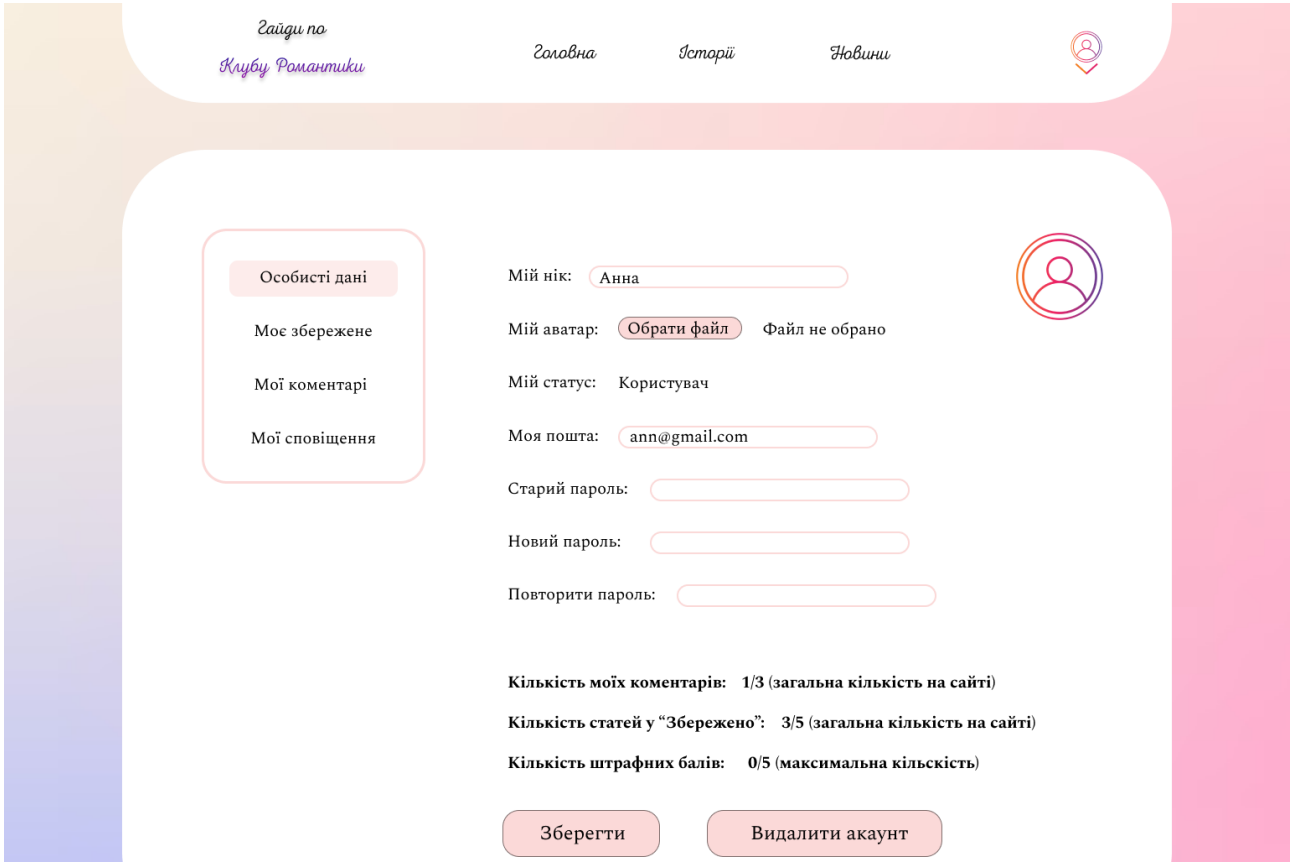

Рис. 2.25. Особисті дані користувача

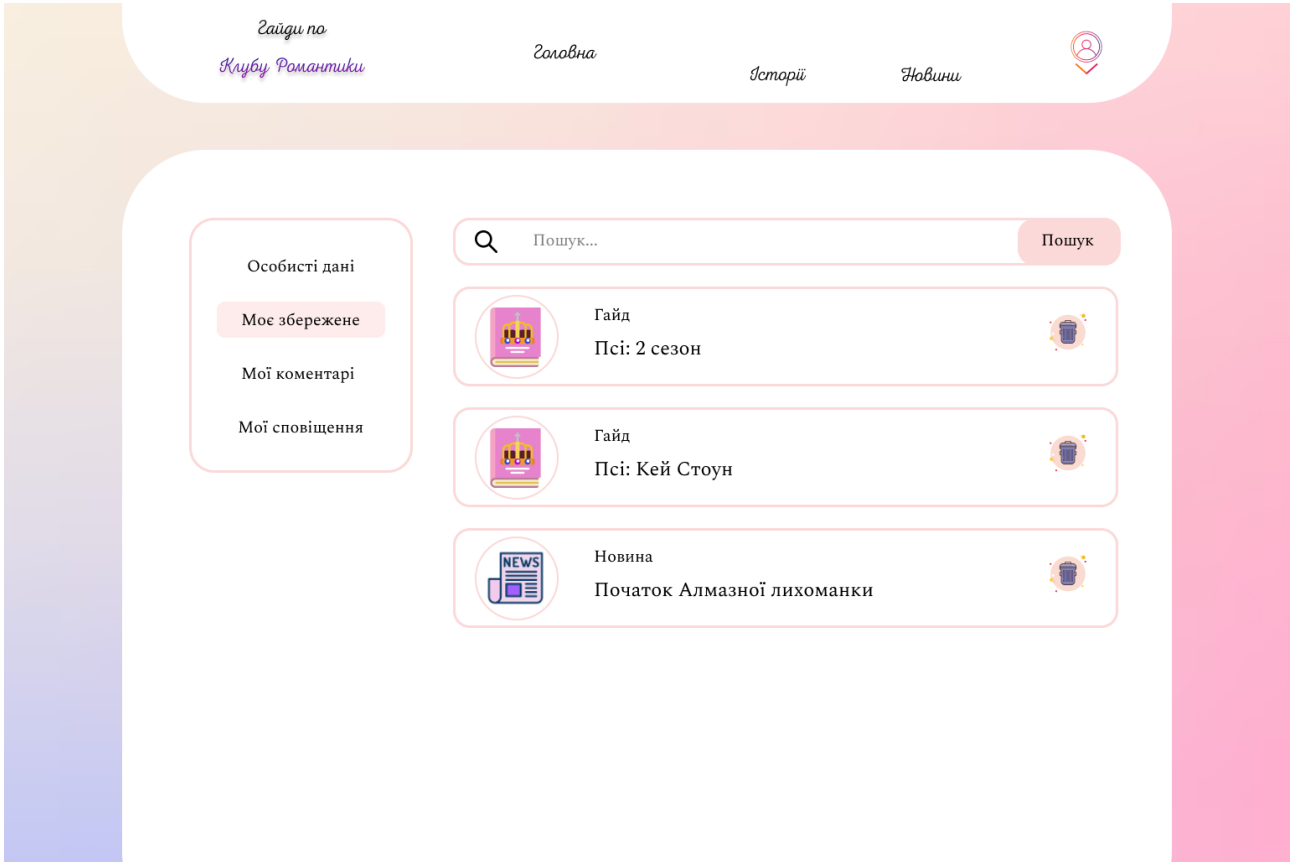

Рис. 2.26. Список збереженого користувача

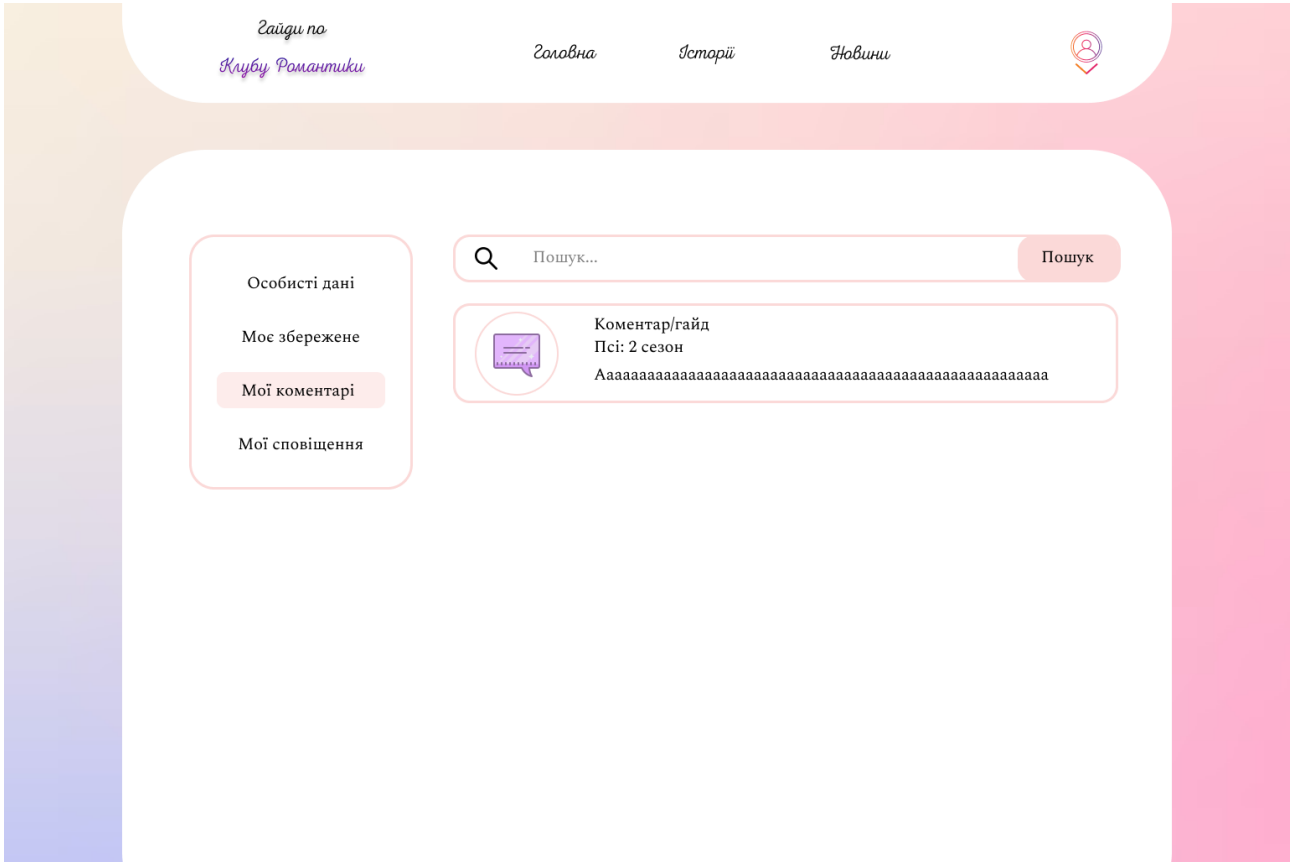

Рис. 2.27. Список коментарів користувача

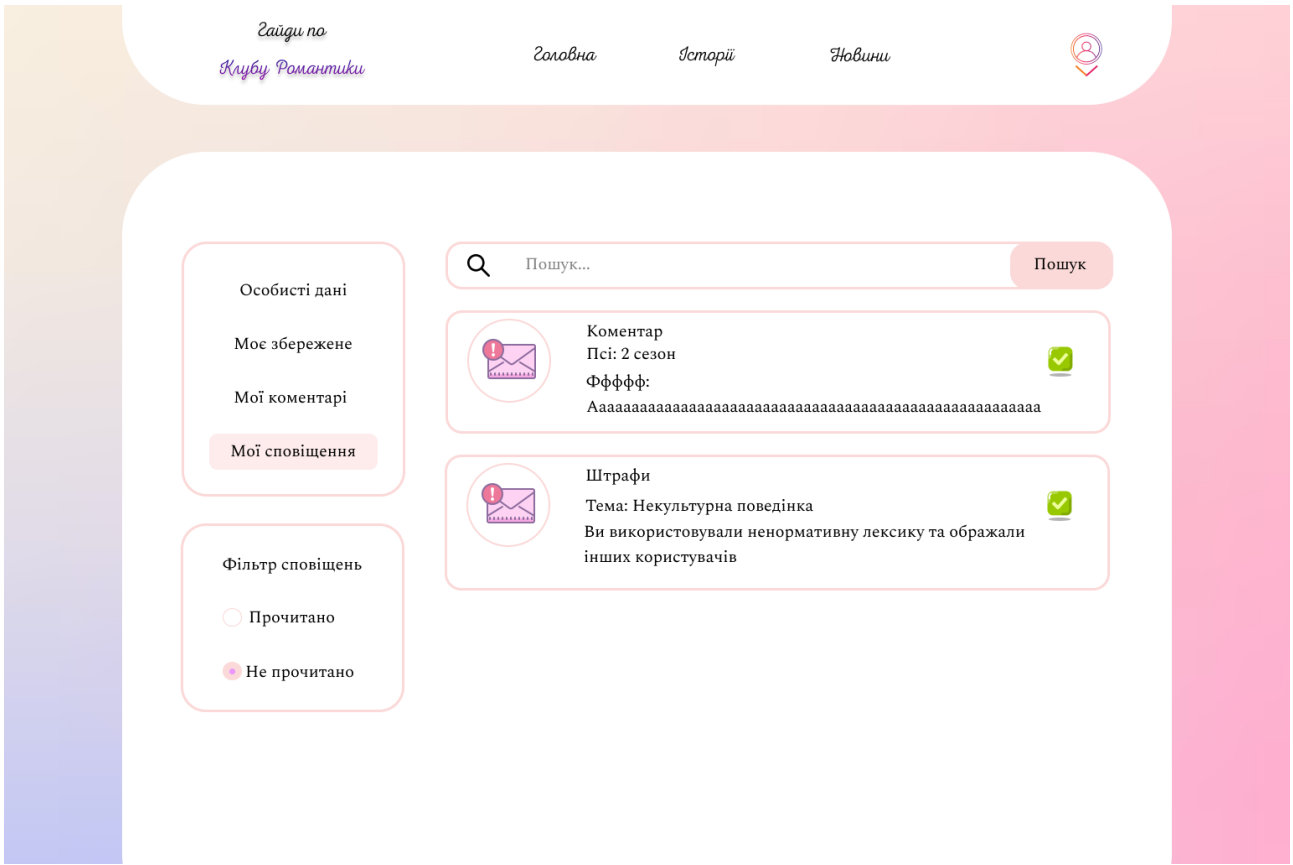

Рис. 2.28. Список сповіщень користувача

Якщо авторизований користувач є адміністратором, то вигляд особистого кабінету має деякі розбіжності. Тому функціонал буде наступний:

- особисті дані: ця сторінка майже нічим не відрізняється від сторінки кабінету користувача. По бажанню, дані можна змінити. Крім цього є можливість завантажити зображення аватару. Статистика відсутня, а також відсутня кнопка видалення акаунту;

- управління акаунтами: тут відображається список акаунтів, які збережні у базі даних системи. Адміністратор бачить основну інформацію, включаючи кількість штрафів. Також він може оштрафувати користувача або видалити його акаунт із системи. Також можна здійснити пошук конкретного акаунту;

- управління коментарями: нічим не відрізняється від управління акаунтами, лише тим, що тут відображається список усіх коментарів, які збережні у базі даних. Адміністратор також може оштрафувати користувача або видалити його коментар із системи. Також можна здійснити пошук конкретного коментаря;

- управління контентом: тут відображаються усі сторінки гайдами та новинами, які збережні у базі даних системі. Адміністратор може видалити дані, натиснувши підтвердження на спливаючому вікні, редагувати вже наявні дані, а також додавати нові. Також можна здійснити пошук конкретної сторінки;

- мої сповіщення: тут знаходиться список сповіщень. Який не відрізняється за виглядом від сповіщень звичайного користувача. Адміністратору приходять сповіщення про скарги, а також запити на видалення акаунтів. Також можна здійснити пошук конкретного сповіщення.

Зображення наведених вище функцій можна побачити на рисунках 2.29 - 2.34.

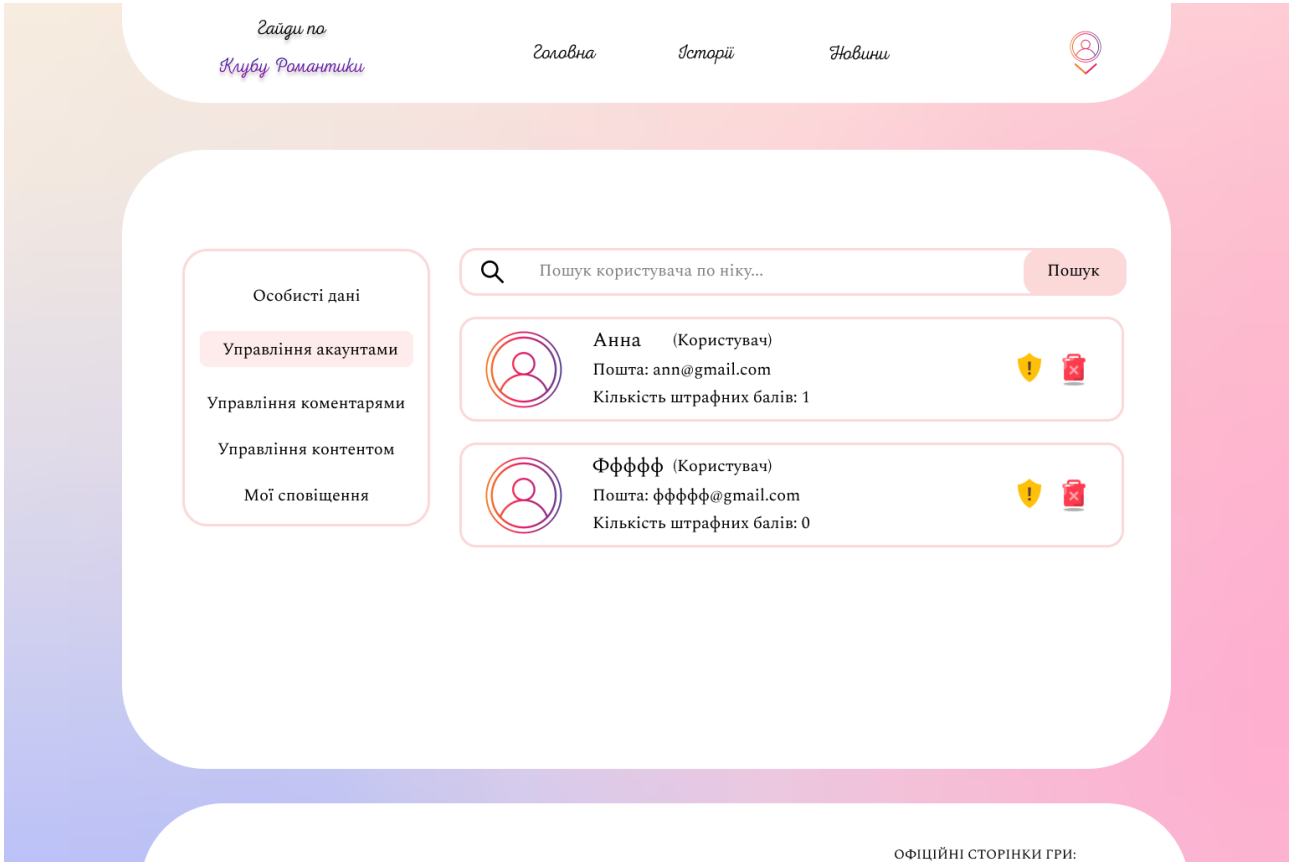

Рис. 2.29. Управління акаунтами для адміністратора

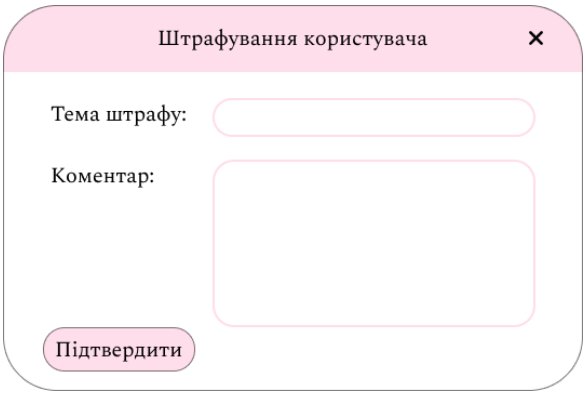

Рис. 2.30. Форма для штрафування користувача

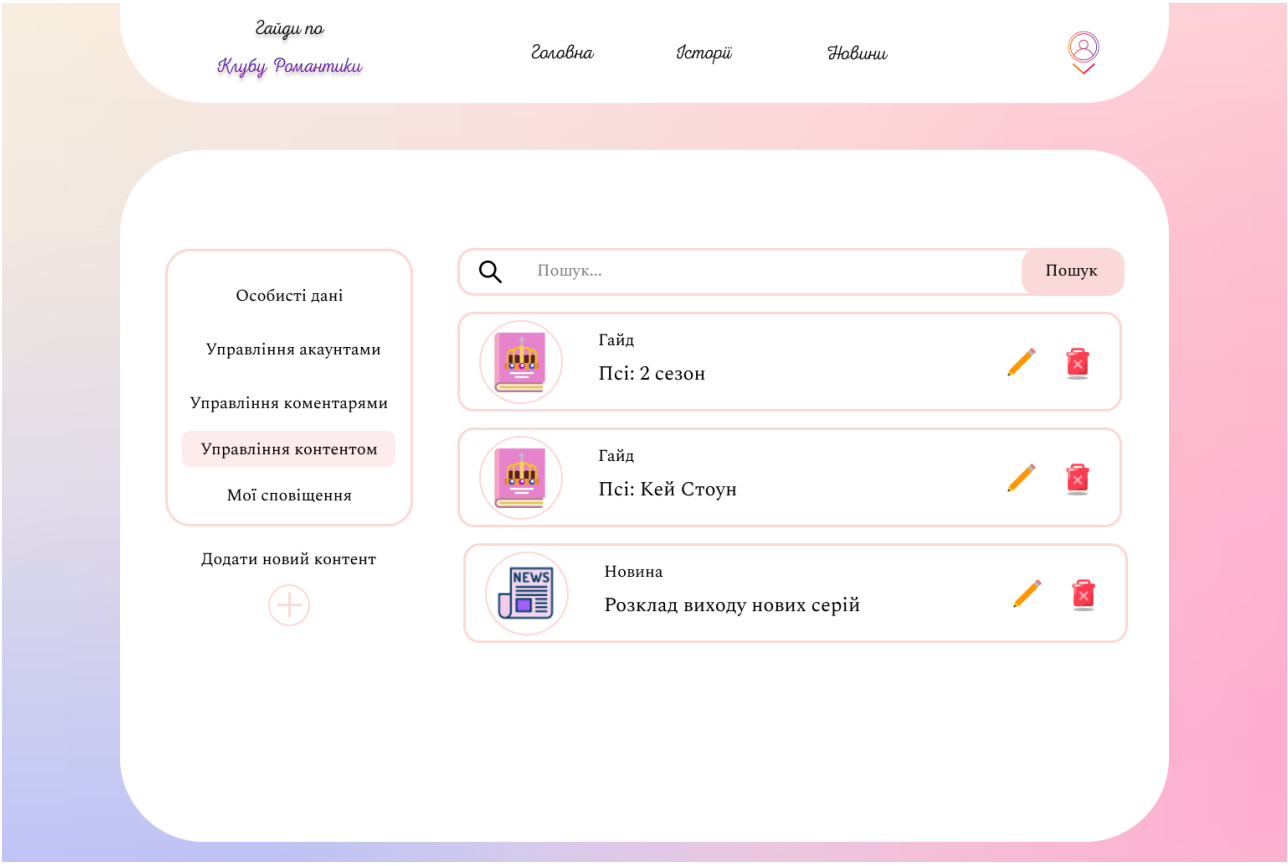

Рис. 2.31. Управління контентом для адміністратора

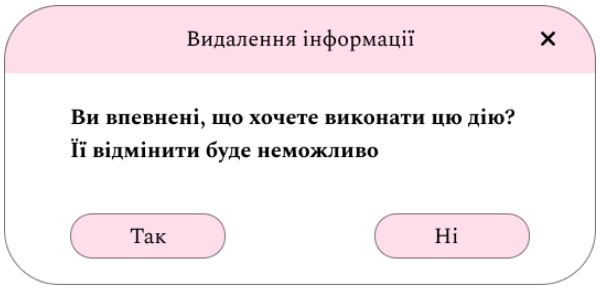

Рис. 2.32. Підтвердження видалення

Рис. 2.33. Вибір типу при додаванні нового контенту

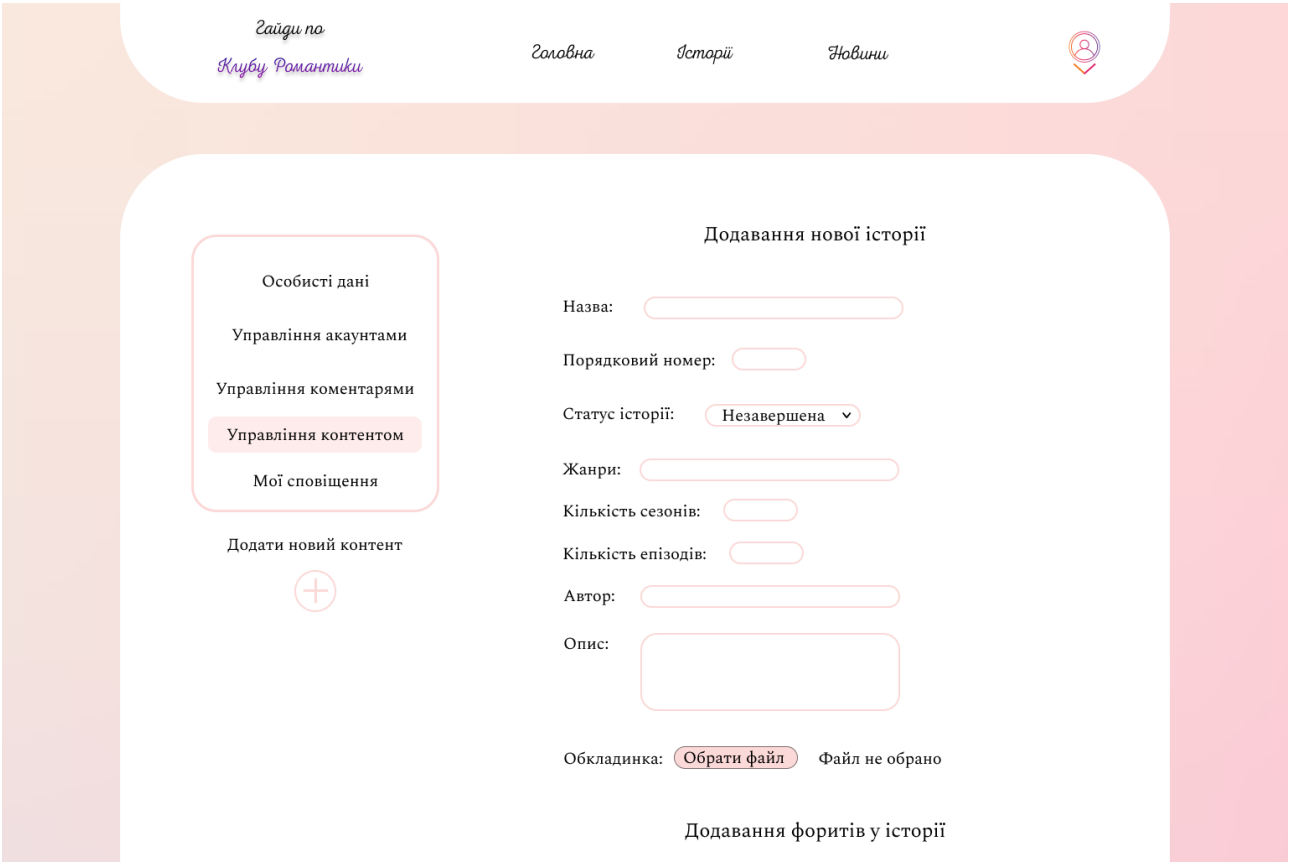

Рис. 2.34. Форма для додавання нового контенту

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Вхідні дані:

- передбачуване число операторів інформаційної системи: 3500;
- коефіцієнт складності програми: 1,4;
- коефіцієнт корекції програми в ході її розробки: 0,1;
- годинна заробітна плата програміста: 208 грн/год;

Станом на початок 2023 року за даними сайту dou.ua [20] заробітна плата розробника, що поєднує у собі навички у роботі з ReactJS та Java/Spring знаходиться в межах 300$ та 2400$. Таким чином, середня заробітна плата розробника складає 1350$. На початок червня 2023 року курс валют складає 36,57 грн. долар США. Отже, середня зарплата в гривнях за місяць дорівнює приблизно 49370 грн. При стандартному графіку (176 годин/місяць) заробітна плата за годину буде становити приблизно 280 грн.

- коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі: 1,25;
- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності: 1,2;
- вартість машино-години ЕОМ: 9 грн/год.

Для даної системи необхідною є якісна та безперервна робота сервера, зберігання та підтримки великої кількості даних на сервері. Отже, також необхідна оренда ноутбука, тому вартість даної оренди на місяць буде становити приблизно 1600 грн. [21]. При стандартному графіку (176 годин/місяць) вартість машино-години ЕОМ за годину роботи буде становити 9 грн.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p), \quad (3.2)$$

де q – передбачуване число операторів (3500);

C – коефіцієнт складності програми (1,4);

p – коефіцієнт корекції програми в ході її розробки (0,1).

За формулою (3,2) умовне число операторів в програмі становить:

$$Q = 3500 * 1{,}4 * (1 + 0{,}1) = 5390 \quad (3.3)$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q*B}{(75..85)*k}, \quad (3.4)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,25);

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. За відсутністю стажу роботи та наявністю освіти дорівнює 1,2.

Таким чином за формулою (3.4) отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{5390*1,25}{85*1,2} = 66,1, \text{ людино-годин}, \quad (3.5)$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)*k}, \quad (3.6)$$

де Q – умовне число операторів програми (5390);

k – коефіцієнт кваліфікації програміста (1,2).

Згідно формулі (3.6), отримаємо:

$$t_a = \frac{5390}{25*1,2} = 179,7, \text{ людино-годин}, \quad (3.7)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)*k}, \quad (3.8)$$

Маючи формулу (3.8) витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{5390}{25*1,2} = 179,7, \text{ людино-годин,} \quad (3.9)$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)*k}, \quad (3.10)$$

Витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання за формулою (3.10):

$$t_{\text{отл}} = \frac{5390}{5*1,2} = 898,3, \text{ людино-годин,} \quad (3.11)$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1,5 * t_{\text{отл}}, \quad (3.12)$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.12):

$$t_{\text{отл}}^{k} = 1,5 * 898,3 = 1347,5, \text{ людино-годин,} \quad (3.13)$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \quad (3.14)$$

де $t_{\text{др}}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{\text{др}} = \frac{Q}{(15..20)*k}, \quad (3.15)$$

$t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0{,}75 * t_{\text{др}}, \quad (3.16)$$

Маючи формули (3.15), (3.16) та (3.14) обчислюємо витрати праці на документацію:

$$t_{\text{др}} = \frac{5390}{20*1{,}2} = 224{,}6, \text{ людино-годин}, \quad (3.17)$$

$$t_{\text{до}} = 0{,}75 * 224{,}6 = 168{,}5, \text{ людино-годин}, \quad (3.18)$$

$$t_{\text{д}} = 224{,}6 + 168{,}5 = 393{,}05, \text{ людино-годин}, \quad (3.19)$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 66{,}1 + 179{,}7 + 179{,}7 + 898{,}3 + 393{,}05 = 1\,766{,}85, \text{ людино-годин}. \quad (3.20)$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ КПО включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}. \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * С_{ПР}, \text{ грн}, \quad (3.22)$$

де t – загальна трудомісткість, людино-годин (1 766,85);

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата програміста становить 208 грн / год, за формулою (3.22) отримуємо:

$$З_{ЗП} = 1\,766{,}85 * 208 = 367\,505, \text{ грн}, \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} * С_{Мч}, \text{ грн}, \quad (3.24)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год (898,3 год);

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год (9 грн/год).

Підставивши в формулу (3.24) відповідні значення, обчислюємо вартість необхідного для налагодження машинного часу:

$$З_{МВ} = 898{,}3 * 9 = 8\,085, \text{ грн}, \quad (3.25)$$

Звідси, за формулою (3.21), витрати на створення програмного продукту:

$$К_{ПО} = 367\,505 + 8\,085 = 375\,590, \text{ грн}, \quad (3.26)$$

Очікуваний період створення програмного застосунку:

$$T = \frac{t}{B_k * F_p}, \text{ міс}, \quad (3.27)$$

де B_k – число виконавців (1);

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

За формулою 3.27 очікуваний період створення програмного забезпечення:

$$T = \frac{1766{,}85}{1*176} \approx 10{,}04 \text{ міс.} \quad (3.28)$$

Висновки: За отриманими результати видно, що розробка даної системи буде дорівнювати 10,04 місяцям з урахуванням стандартного робочого часу: 40-годинного робочого дня та 176-годиного робочого тижня. Очікуванні витрати, які будуть потрібні для розробки даного продукту будуть наступними: 375 590 грн.

ВИСНОВКИ

Основною метою даної роботи було розробити інформаційну систему для підтримки гравців, а конкретно, для фанатів візуальних новел, з використанням технологій ReactJS та Java/Spring.

У процесі був створений вебсайт, який відповідає поставленим на початку функціональним вимогам, і відповідно надає користувачам зручний та інтуїтивно-зрозумілий інтерфейс, широкий набір можливостей, який дозволяє ефективно та швидко орієнтуватися на сайті та знаходити потрібну інформацію.

Актуальність даного продукту має практичне значення для гравців візуальних новел, оскільки створена інформаційна система надає їм доступ до кращих проходжень та рекомендацій, які сприятимуть поліпшенню геймплею та загальному досвіду гри. Крім того, система може бути використана розробниками візуальних новел як засіб просування та популяризації своїх продуктів.

У розробці були використанні наступні технології:

- ReactJS, бібліотека, яка дозволила розробити гнучкий та привабливий інтерфейс для користувачів. Компонентний підхід ReactJS сприяв швидкому розгортанню і зручному управлінню різними частинами системи;
- Java/Spring, фреймворк, який дозволив розробити серверну частину додатку, а також забезпечив зручну модульну архітектуру та підтримку вебсервісів, що сприяло зручній взаємодії між клієнтською та серверною частинами системи.

Окрім цього у кваліфікаційній роботі було визначено вартість створення даного додатка (375 590 грн.) та розраховано час, необхідний для його розробки (приблизно 10 місяців).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sales growth of graphic novels from 2018 to 2020. URL: https://www.statista.com/statistics/1091347/sales-growth-graphic-novels-us/ (дата звернення: 17.05.2023).

2. Історія відеоігор. URL: https://uk.wikipedia.org/wiki/%D0%86%D1%81%D1%82%D0%BE%D1%80%D1%96%D1%8F_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80 (дата звернення: 17.05.2023).

3. Жанри відеоігор. URL: https://uk.wikipedia.org/wiki/%D0%96%D0%B0%D0%BD%D1%80%D0%B8_%D0%B2%D1%96%D0%B4%D0%B5%D0%BE%D1%96%D0%B3%D0%BE%D1%80 (дата звернення: 17.05.2023).

4. Візуальна новела. URL: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BD%D0%BE%D0%B2%D0%B5%D0%BB%D0%B0 (дата звернення: 17.05.2023).

5. ReactJS Tutorial. URL: https://www.tutorialspoint.com/reactjs/index.htm (дата звернення: 20.05.2023).

6. Рендеринг елементів. URL: https://uk.legacy.reactjs.org/docs/rendering-elements.html (дата звернення: 20.05.2023).

7. ReactJS – JSX. URL: https://www.tutorialspoint.com/reactjs/reactjs_jsx.htm (дата звернення: 20.05.2023).

8. Banks Alex, Porcello Eve. Learning React: functional web development with React and Redux. O'Reilly Media, Inc., 2017. 350p.

9. React Router. URL: https://www.w3schools.com/react/react_router.asp (дата звернення: 20.05.2023).

10. Ganatra Sagar. React Router Quick Start Guide: Routing in React Applications Made Easy. Packt Publishing Ltd, 2018. 156p.

11. Documentation – Tailwind CSS. URL: https://v2.tailwindcss.com/docs (дата звернення: 20.05.2023).

12. The Pros and Cons of Tailwind CSS. URL: https://www.webdesignerdepot.com/2021/09/the-pros-and-cons-of-tailwind-css/ (дата звернення: 20.05.2023).

13. Java. URL: https://uk.wikipedia.org/wiki/Java (дата звернення: 20.05.2023).

14. Spring Framework Documentation. URL: https://docs.spring.io/spring-framework/reference/ (дата звернення: 25.05.2023).

15. MariaBD. URL: https://en.wikipedia.org/wiki/MariaDB (дата звернення: 25.05.2023).

16. Federico Razzoli. Mastering MariaDB. Packt Publishing Ltd, 2014. 384p.

17. Figma. URL: https://en.wikipedia.org/wiki/Figma_(software) (дата звернення: 30.05.2023).

18. Visual Studio Code. URL: https://en.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 30.05.2023).

19. HeidiSQL. URL: https://en.wikipedia.org/wiki/HeidiSQL (дата звернення: 30.05.2023).

20. Зарплати українських розробників – зима 2023. URL: https://dou.ua/lenta/articles/salary-report-devs-winter-2023/ (дата звернення: 05.06.2023).

21. Оренда потужного ноутбука. URL: https://prom.ua/ua/p1541921868-arenda-moschnogo-noutbuka.html (дата звернення: 05.06.2023).

ДОДАТОК А

КОД ПРОГРАМИ

Код front-end частини додатку:

index.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <link rel="icon" type="image/svg+xml" href="/src/assets/img/favicon_pack/A diamond and a cup of hot tea.png" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>GuidesRC</title>
</head>

<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>

</html>
```

index.css:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

* {
  margin: 0;
  padding: 0;
  @apply box-border;
}

html, body, #root {
  height: 100%;
}

body {
  font-family: "Spectral";
  @apply bg-globalBg bg-no-repeat bg-cover;
}

@font-face {
```

```
  font-family: "nexaBold";
  src: url("/src/assets/font/NexaScript/nexascript_bold.ttf");
 }

 @font-face {
  font-family: "nexaHeavy";
  src: url("/src/assets/font/NexaScript/nexascript_heavy.ttf");
 }

 @font-face {
  font-family: "nexaLight";
  src: url("/src/assets/font/NexaScript/nexascript_light.ttf");
 }

 @font-face {
  font-family: "nexaRegular";
  src: url("/src/assets/font/NexaScript/nexascript_regular.ttf");
 }

 @font-face {
  font-family: "nexaThin";
  src: url("/src/assets/font/NexaScript/nexascript_thin.ttf");
 }
```

main.jsx:

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { RouterProvider } from 'react-router-dom'
import './index.css'
import router from './router'

ReactDOM.createRoot(document.getElementById('root')).render(
  <RouterProvider router={router} />
)
```

router.jsx:

```
import { Provider } from "react-redux";
import { Outlet, Route, createBrowserRouter, createRoutesFromElements } from "react-router-dom";
import { AdminAccount, UserAccount } from "./pages/account";
import Layout from "./pages/common/Comon";
import { Home } from "./pages/home";
import Login from "./pages/login/Login";
import SignUp from "./pages/login/SignUp";
import { News } from "./pages/news";
import { Stories, StoryDetails, NewsDetails, StoryGuides } from "./pages/stories";
```

```
import store from "./store";

export default createBrowserRouter(createRoutesFromElements(
  <Route element={<Provider store={store} children={<Outlet />} />}>
    <Route path="/" element={<Layout />}>
      <Route index element={<Home />} />
      <Route path="stories" element={<Stories />} />
      <Route path="stories/:id" element={<StoryDetails />} />
      <Route path="stories/:id/seasonNum" element={<StoryGuides />} />
      <Route path="stories/:id/character" element={<StoryCharacter />} />
      <Route path="news" element={<News />} />
      <Route path="news/:id" element={<NewsDetails />} />
      <Route path="account-admin" element={<AdminAccount />} />
      <Route path="account-user" element={<UserAccount />} />
    </Route>
    <Route path="/login" element={<Login />} />
    <Route path="/signup" element={<SignUp />} />
  </Route>
))
```

index.js:

```
import { configureStore } from "@reduxjs/toolkit";
import { userReducers } from "../pages/auth/store";

export default configureStore({
  reducer: {
    user: userReducers
  }
})
```

userSlice.js:

```
import { createSlice } from "@reduxjs/toolkit";
import userAvatar from "../../../assets/img/background.svg"

const userSlice = createSlice({
  name: "user",
  initialState: {
    data: {
      name: "",
      mail: "",
      password: "",
      avatar: userAvatar
    },
    isAuth: true,
    isAdmin: false
  },
  reducers: {
    setAuth(state, action) {
```

```
            state.isAuth = action.payload;
        },
        toggleAuth(state, action) {
            state.isAuth = !state.isAuth;
        },
        setAdmin(state, action) {
            state.isAdmin = action.payload
        },
        setAvatar(state, action) {
            state.data.avatar = action.payload
        }
    }
})

export default userSlice;
```

Login.jsx:

```
import { useDispatch } from "react-redux";
import { useNavigate } from "react-router-dom";
import { setAdmin, setAuth } from "../auth/store";
import { BgLogin, IconDiamonds, IconTea } from "./assets";

const Login = () => {
    const dispatch = useDispatch();
    const navigate = useNavigate();

    const handleSubmit = () => {
        navigate("/");
    }

    return (
        <div className="flex items-center justify-center h-screen">
            <div className="flex flex-row text-xl font-nexaThin bg-white rounded-[50px] max-w-screen-sm aspect-w-82 overflow-hidden">
                <div className="w-1/2 flex flex-col select-none">
                    <div className="flex flex-row justify-between px-10 py-5 text-purple-500">
                        <button
                            onClick={() => navigate("/signup")}
                            className="hover:rounded-xl p-2 hover:bg-purple-100">
                            Реєстрація
                        </button>
                        <h1 className="font-nexaRegular p-2">Авторизація</h1>
                    </div>
                    <form onSubmit={handleSubmit} className="flex flex-col h-full mt-5">
                        <div className="flex flex-row overflow-hidden px-5 py-5">
                            <div  justify-center w-20 border-y-2 border-purple-200 rounded-l-xl bg-gradient-to-b from-purple-200 px-3">
                                Пошта
                            </div>
                            <input
```

```
                        type="email"
                        placeholder="rClub@gmail.com"
                        className="w-full px-2 border-2 border-purple-200 rounded-r-xl
focus:outline-none" />
                    </div>
                    <div className="flex flex-row overflow-hidden px-5">
                      <div className="flex justify-center w-20 border-y-2 border-purple-200
rounded-l-xl bg-gradient-to-b from-purple-200 px-3">
                        Пароль
                      </div>
                      <input
                        type="password"
                        placeholder="********"
                        className="w-full px-2 border-2 border-purple-200 rounded-r-xl
focus:outline-none" />
                    </div>
                    <div className="flex flex-row items-center justify-center mt-auto">
                      <img
                        src={IconDiamonds}
                        style={{ transform: "rotate(-20deg)" }}
                        className="w-12 h-12 mx-2" />
                      <img
                        src={IconTea}
                        style={{ transform: "rotate(20deg)" }}
                        className="w-12 h-12 mx-2" />
                    </div>
                    <button className="mt-auto py-5 bg-gradient-to-b from-purple-200
hover:from-purple-300">Авторизуватися</button>
                  </form>
                </div>
                <div className="w-1/2">
                  <img src={BgLogin} className="w-full h-full" />
                </div>
              </div>
            </div>
        )
    }

    export default Login
```

Signup.jsx:

```
    import { useDispatch } from "react-redux";
    import { useNavigate } from "react-router-dom";
    import { setAuth } from "../auth/store";
    import { BgLogin, IconDiamonds, IconTea } from "./assets";

    const SignUp = () => {
      const dispatch = useDispatch();
      const navigate = useNavigate();

      const handleSubmit = () => {
```

```
        navigate("/");
      }

      return (
        <div className="flex items-center justify-center h-screen">
          <div className="flex flex-row text-xl font-nexaThin bg-white rounded-[50px] max-w-screen-sm aspect-w-82 overflow-hidden">
            <div className="w-1/2">
              <img src={BgLogin} className="w-full h-full" />
            </div>
            <div className="w-1/2 flex flex-col select-none">
              <div className="flex flex-row justify-between px-10 py-5 text-purple-500">
                <h1 className="font-nexaRegular p-2">Реєстрація</h1>
                <button
                  onClick={() => navigate("/login")}
                  className="hover:rounded-xl p-2 hover:bg-purple-100">
                  Авторизація
                </button>
              </div>
              <form onSubmit={handleSubmit} className="flex flex-col h-full mt-5">
                <div className="flex flex-row overflow-hidden px-5">
                  <div className="flex justify-center w-20 border-y-2 border-purple-200 rounded-l-xl bg-gradient-to-b from-purple-200 px-10">
                    Нік
                  </div>
                  <input
                    type="text"
                    placeholder="ВашНік..."
                    className="w-full px-2 border-2 border-purple-200 rounded-r-xl focus:outline-none" />
                </div>
                <div className="flex flex-row overflow-hidden px-5 py-5">
                  <div className="flex justify-center w-20 border-y-2 border-purple-200 rounded-l-xl bg-gradient-to-b from-purple-200 px-3">
                    Пошта
                  </div>
                  <input
                    type="email"
                    placeholder="rClub@gmail.com"
                    className="w-full px-2 border-2 border-purple-200 rounded-r-xl focus:outline-none" />
                </div>
                <div className="flex flex-row overflow-hidden px-5">
                  <div className="flex justify-center w-20 border-y-2 border-purple-200 rounded-l-xl bg-gradient-to-b from-purple-200 px-3">
                    Пароль
                  </div>
                  <input
                    type="password"
                    placeholder="********"
                    className="w-full px-2 border-2 border-purple-200 rounded-r-xl focus:outline-none" />
```

```
                        </div>
                        <div className="flex flex-row items-center justify-center mt-auto">
                          <img
                            src={IconDiamonds}
                            style={{ transform: "rotate(-20deg)" }}
                            className="w-12 h-12 mx-2" />
                          <img
                            src={IconTea}
                            style={{ transform: "rotate(20deg)" }}
                            className="w-12 h-12 mx-2" />
                        </div>
                        <button className="mt-auto py-5 bg-gradient-to-b from-purple-200
hover:from-purple-300">Зареєструватися</button>
                      </form>
                    </div>
                  </div>
                </div>
              )
            }

            export default SignUp;
```

Common.jsx:

```
            import { Outlet } from "react-router-dom";
            import { Footer, Header } from "./components";

            const Layout = () => {
              return (
                <div className="flex flex-col max-w-screen-lg m-auto">
                  <Header />
                  <Outlet />
                  <Footer />
                </div>
              )
            }

            export default Layout;
```

Footer.jsx:

```
            import { FacebookIcon, InstagramIcon, LogoText, RedditIcon, TelegramIcon, WebSiteIcon
} from "../assets";

            const Footer = () => {
              return (
                <footer className="flex flex-row items-center justify-around bg-white mt-8 p-10
space-x-5 rounded-t-[90px] text-sm uppercase">
                  <div className="w-1/3 flex flex-row items-center">
                    <div className="w-1/2 flex flex-col space-y-4">
                      <label>Головна</label>
```

```
                    <label>Історії</label>
                    <label>Новини</label>
                  </div>
                  <div className="w-1/2 flex flex-col space-y-4">
                    <label>Правила сайту</label>
                    <label>Служба підтримки</label>
                    <label>Політика конфіденційності</label>
                    <label>Угода користувача</label>
                  </div>
                </div>
                <div className=" w-1/3 flex flex-col items-center space-y-10 text-center">
                  <LogoText />
                  <label>© Copyright 2023 Гайди по клубу романтики – Всі права захищені</label>
                </div>
                <div className=" w-1/3 flex flex-col items-center text-center space-y-12">
                  <div className="space-y-5">
                    <label>офіційні сторінки гри:</label>
                    <div className="flex flex-row items-center space-x-5">
                      <WebSiteIcon className="w-8 h-8" />
                      <TelegramIcon className="w-8 h-8" />
                      <InstagramIcon className="w-8 h-8" />
                      <FacebookIcon className="w-8 h-8" />
                      <RedditIcon className="w-8 h-8" />
                    </div>
                  </div>
                  <label>Всі авторські права на логотипи, ігрові зображення, назви історій – належать
                    компанії Your Story Interactive</label>
                </div>
              </footer>
            );
          }

          export default Footer;
```

Header.jsx:

```
          import React, { useState } from "react";
          import { useSelector } from "react-redux";
          import { useNavigate } from "react-router-dom";
          import { ArrowAvatar, ArrowStory, IconAvatar, LogoText } from "../assets";
          import DropdownMenu from "./DropDownMenu";

          const Header = () => {
            const navigate = useNavigate();

            const isAuth = useSelector((state) => state.user.isAuth);
            const isAdmin = useSelector((state) => state.user.isAdmin);

            const [isOpen, setOpen] = useState(false);
```

```
    const toggleMenuOpen = () => {
      setOpen(!isOpen);
    };

    const handleAvatarClick = () => {
      navigate(isAdmin ? "/account-admin" : "/account-user");
    }

    return (
      <header className="flex flex-row items-center justify-around bg-white px-5 rounded-
b-[90px] font-nexaThin text-xl">
        <LogoText className="h-28 w-40" />
        <label
          onClick={() => navigate("/")}
          className="cursor-pointer w-auto p-2 hover:rounded-xl hover:bg-purple-100">
          Головна
        </label>
        <div className="flex flex-row items-center p-2 space-x-3 cursor-pointer w-auto
hover:rounded-xl hover:bg-purple-100">
          <label onClick={() => navigate("/stories")}>Історії</label>
          <ArrowStory className="h-4 w-4" />
        </div>
        <label
          onClick={() => navigate("/news")}
          className="cursor-pointer w-auto p-2 hover:rounded-xl hover:bg-purple-100">
          Новини
        </label>
        <div
          className="flex flex-col items-center cursor-pointer relative"
          onClick={handleAvatarClick}
          onMouseDown={toggleMenuOpen}
          onMouseLeave={toggleMenuOpen}>
          <img src={IconAvatar} className="h-10 w-10" />
          <ArrowAvatar className="h-5 w-5" />
          {isOpen && (
            <DropdownMenu
              navigate={navigate}
              buttons={[
                { label: "Авторизація", route: "/login" },
                { label: "Реєстрація", route: "/signup" },
              ]}
            />
          )}
        </div>
      </header>
    );
  };

  export default Header;
```

DropDownMenu.jsx:

```
import React from "react";

const DropdownMenu = ({ buttons, navigate }) => {
  const handleButtonClick = (route) => {
    navigate(route);
  }

  return (
    <div className="absolute mt-24 w-48 bg-white rounded-[50px] shadow-xl divide-y-2 divide-slate-200 overflow-hidden">
      {buttons.map((button, index) => (
        <button
          key={index}
          onClick={() => handleButtonClick(button.route)}
          className="p-2 text-lg hover:bg-gray-100 w-full text-center">
          {button.label}
        </button>
      ))}
    </div>
  );
};

export default DropdownMenu;
```

Carousel.jsx:

```
import React from 'react';
import { Carousel } from 'react-responsive-carousel';
import 'react-responsive-carousel/lib/styles/carousel.min.css';
import { images, imagesData } from '../../common/components/ImagesStory';
import { ArrLeft, ArrRight } from '../assets';

const CarouselStory = () => {
  const ArrPrev = (onClickHandler, hasPrev) =>
    hasPrev && (
      <button onClick={onClickHandler} className="absolute left-0 top-1/2 transform -translate-y-1/2 z-10">
        <ArrLeft className="h-12 w-12" />
      </button>
    );

  const ArrNext = (onClickHandler, hasNext) =>
    hasNext && (
      <button onClick={onClickHandler} className="absolute right-0 top-1/2 transform -translate-y-1/2">
        <ArrRight className="h-12 w-12" />
      </button>
    );

  return (
    <Carousel
      showArrows={true}
```

```
            showStatus={false}
            showIndicators={false}
            centerMode={true}
            infiniteLoop={true}
            showThumbs={false}
            centerSlidePercentage={100 / 3}
            renderArrowNext={ArrNext}
            renderArrowPrev={ArrPrev}>
            {images.map(imagesData)}
          </Carousel>
      );
    };
    export default CarouselStory;
```

Home.jsx:

```
    import { Line } from "../common/assets";
    import { DiamandHomeIcon, TeaHomeIcon } from "./assets";
    import CarouselStory from "./components/Carousel";

    const Home = () => {
      return (
        <div className="flex flex-col items-center bg-white mt-8 p-10 rounded-[90px] space-y-7">
          <label className="text-xl">Romance Club</label>
          <label>Ласкаво просимо до нашого сайту, присвяченого грі “Romance Club” - захоплюючого світу, де ти сам керуєш долею головних
            героїв. Неважливо, які у тебе причини - брак часу або просто складність у проходженні, - тепер тобі більше не потрібно
            хвилюватися про марне витрачання коштовних діамантів, сумніватися у правильності свого вибору або головувати над тим,
            який шлях чи сторону обрати. Усе це вже зроблено за тебе!<p className="mt-5" />На нашому сайті ти знайдеш найважливішу
            та найактуальнішу інформацію про улюблену гру. Представляємо тобі корисні посібники, детальні проходження історій, широкий
            вибір романтичних напрямків та багато іншого. Ми прагнемо зробити твій ігровий процес якомога приємнішим!
          </label>
          <Line className="h-10 w-full" />
          <label className="text-xl">Останні оновлення серед історій!</label>
          <CarouselStory />
          <Line className="h-10 w-full" />
          <label className="text-xl">Інформація про ігрові ресурси та їх накопичення</label>
          <div className=" flex flex-row space-x-8">
            <div className="flex flex-col justify-center items-center w-1/2 text-center shadow-2xl rounded-[50px] p-8 space-y-5">
              <DiamandHomeIcon className="" />
              <label>Алмази - головна валюта для відкривання додаткових можливостей в грі, будь це персонаж, наряд або додаткова сюжетна
                лінія. Їх ніколи не буває багато. Їх можна отримати за допомогою щоденної нагороди, яка збільшується з кожним новим входом
```

```
                        у гру. Чи купити у самій грі або у офіційному магазині.</label>
                    </div>
                    <div className="flex flex-col justify-center items-center w-1/2 text-center shadow-2xl rounded-[50px] p-8 space-y-5">
                        <TeaHomeIcon />
                        <label>Чай - ще одна валюта крім Алмазів, що дозволяє відкривати епізоди історій. Одна Чашечка відкриває один обраний
                            епізод історії, який ви можете прочитати до кінця. Максимальна кількість чашок: 3. Через деякий час,
                            втрачений Чай з'являється знову. Чай можна купити у самій грі або у офіційному магазині.</label>
                    </div>
                </div>
                <Line className="h-10 w-full" />
                <label className="text-xl">Останні новини!</label>
                <div className="grid grid-cols-2 gap-10 select-none">
                    {latestNews.map((item, index) => (
                        <div key={index} className="flex flex-row rounded-[50px] shadow-xl overflow-hidden cursor-pointer">
                            <div className="w-1/2">
                                <img src={item.src} className="h-full w-full" />
                            </div>
                            <div className="w=1/2 flex flex-col px-8 justify-between">
                                <label className="text-center text-lg m-auto">{item.title}</label>
                                <label className="text-right text-gray-500 ml-auto">{item.created}</label>
                            </div>
                        </div>
                    ))}
                </div>
            </div>
        );
    }

    export default Home;
```

AccountUser.jsx:

```
    import { useState } from "react";
    import { Comments, Notifications, PersonalData, SavedData, Search } from "./components";
    import { Link } from "react-router-dom";

    const UserAccount = () => {
        const [activeButton, setActiveButton] = useState("personalData");

        const handleButtonClick = (button) => {
            setActiveButton(button);
        };

        const buttons = [
            { id: "personalData", label: "Особисті дані" },
            { id: "savedData", label: "Моє збережене" },
```

```
            { id: "comments", label: "Мої коментарі" },
            { id: "notifications", label: "Мої сповіщення" },
        ];

        const renderButtons = () => {
            return buttons.map((button) => (
                <button
                    key={button.id}
                    className={`${activeButton === button.id ? "bg-pink-100 px-6 py-1 w-40 rounded-xl" : "px-6 py-1 w-40"}`}
                    onClick={() => handleButtonClick(button.id)}
                >
                    {button.label}
                </button>
            ));
        };

        const renderContent = () => {
            switch (activeButton) {
                case "personalData":
                    return <PersonalData />;
                case "savedData":
                    return <SavedData />;
                case "comments":
                    return <Comments />;
                case "notifications":
                    return <Notifications />;
                default:
                    return null;
            }
        };

        return (
            <div className="flex flex-row items-center bg-white mt-8 p-10 rounded-[90px] space-y-7">
                <div className="w-1/3 flex flex-col items-center mb-auto">
                    <div className="flex flex-col items-center justify-around h-52 w-56 py-5 px-8 border-4 border-pink-100 rounded-3xl">
                        {renderButtons()}
                        <Link to={"/stories/3"}>
                        <label>Rkfw</label>
                        </Link>
                    </div>
                </div>
                <div className="w-2/3">
                    {renderContent()}
                </div>
            </div>
        );
    }

    export default UserAccount;
```

AccountAdmin.jsx:

```
const AdminAccount = () => {
  const [activeButton, setActiveButton] = useState("personalData");

  const handleButtonClick = (button) => {
    setActiveButton(button);
  };

  const buttons = [
    { id: "personalData", label: "Особисті дані" },
    { id: "savedData", label: "Управління акаунтами" },
    { id: "comments", label: "Управління коментарями" },
    { id: "comments", label: "Управління контентом" },
    { id: "notifications", label: "Мої сповіщення" },
  ];

  const renderButtons = () => {
    return buttons.map((button) => (
      <button
        key={button.id}
        className={`${activeButton === button.id ? "bg-pink-100 px-6 py-1 w-40 rounded-xl" : "px-6 py-1 w-40"}`}
        onClick={() => handleButtonClick(button.id)}
      >
        {button.label}
      </button>
    ));
  };

  const renderContent = () => {
    switch (activeButton) {
      case "personalData":
        return <PersonalData />;
      case "accounts":
        return <Accounts />;
      case "comments":
        return <Comments />;
      case "content":
        return <Content />;
      case "notifications":
        return <Notifications />;
      default:
        return null;
    }
  };

  return (
    <div className="flex flex-row items-center bg-white mt-8 p-10 rounded-[90px] space-y-7">
      <div className="w-1/3 flex flex-col items-center mb-auto">
```

```
                <div className="flex flex-col items-center justify-around h-52 w-56 py-5 px-8 border-4 border-pink-100 rounded-3xl">
                    {renderButtons()}
                    <Link to={"/stories/3"}>
                      <label>Rkfw</label>
                    </Link>
                  </div>
                </div>
                <div className="w-2/3">
                  {renderContent()}
                </div>
              </div>
            );
          }

          export default AdminAccount;
```

Код back-end частини додатку:

WebApp:

```
package ua.web.app;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class WebApp {
 public static void main(String[] args) {
          SpringApplication.run(WebApp.class, args);
 }
}
```

KeyConfig:

```
package ua.web.app.config;
import java.security.Key;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import io.jsonwebtoken.io.Decoders;
import io.jsonwebtoken.security.Keys;

@Configuration
public class KeyConfig {

 @Bean
 public PasswordEncoder passwordEncoder() {
          return new BCryptPasswordEncoder(12);
 }
```

```
	@Bean
	public Key tokenKey(Environment environment) {
		String defaultKey =
"siobewbr3w839bczdgt55b4c88cn353kc3hch78k0kcllmlemaibnnpa";
		String secretKey = environment.getProperty("jwt.key.token", defaultKey);
	   return Keys.hmacShaKeyFor(Decoders.BASE64.decode(secretKey));
	}
	}
```

WebConfig:

```
	package ua.friendly.talk.config;
	import org.springframework.context.annotation.Configuration;
	import org.springframework.web.servlet.config.annotation.CorsRegistry;
	import org.springframework.web.servlet.config.annotation.EnableWebMvc;
	import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

	@EnableWebMvc
	@Configuration
	public class WebConfig implements WebMvcConfigurer {

	 @Override
	 public void addCorsMappings(CorsRegistry registry) {
		registry.addMapping("/**").allowedMethods("*");
	 }

	}
```

User:

```
	package ua.web.app.model;
	import javax.persistence.Column;
	import javax.persistence.Entity;
	import javax.persistence.EnumType;
	import javax.persistence.Enumerated;
	import javax.persistence.GeneratedValue;
	import javax.persistence.Id;

	@Entity
	public class User {

	 @Id
	 @GeneratedValue
	 private Long id;

	 @Column(nullable = false, unique = true)
	 private String name;

	 @Column(nullable = false)
	 private String mail;

	 @Column(nullable = false)
```

```
private String password;

@Column(nullable = true)
private String avatar;

@Column(nullable = false)
private bit iaAuth;

@Column(nullable = false)
private bit iaAdmin;

@Column(nullable = false)
@Enumerated(EnumType.STRING)
private UserRole role;

@Column(nullable = false)
private int penalties;

public User() {
        role=UserRole.USER;
        name = "";
        mail = "";
        password = "";
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getMail() {
        return mail;
}

public void setMail(String mail) {
        this. mail = mail;
}

public String getPassword() {
        return password;
}

public void setPassword(String password) {
        this.password = password;
}

public String getAvatar() {
        return avatar;
}
```

```
public void setAvatar(String avatar) {
        this.avatar = avatar;
}

public String getIsAuth() {
        return isAuth;
}

public void setIsAuth(bit isAuth) {
        this.isAuth = iaAuth;
}

public String getIsAdmin() {
        return isAmin;
}

public void setPenalties(int penalties) {
        this. penalties = penalties;
}

public String getPenalties() {
        return penalties;
}

public UserRole getRole() {
        return role;
}

public void setRole(UserRole role) {
        this.role = role;
}

public Long getId() {
        return id;
}


}
```

UserRole:

```
package ua.friendly.talk.model;

public enum UserRole {

USER, ADMIN;
}
```

UserController:

```
package ua.web.app.controller;
```

```
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import ua.web.app.exception.UserFriendlyException;
import ua.web.app.service.UserService;

@RestController
public class UserController {

 private @Autowired UserService userS;

 @GetMapping("/registration")
 public Object registration(HttpServletRequest request) {
          var account = userS.createUser(request);
          var token = userS.generateToken(account);
          return ResponseEntity.of(token);
 }

 @GetMapping("/login")
 public Object login(HttpServletRequest request) {
          var account = userS.loginUser(request);
          var token = userS.generateToken(account);
          return ResponseEntity.of(token);
 }

 @GetMapping("/validation")
 public Object validation(HttpServletRequest request) {
          var account = userS.validateUser(request);
          return ResponseEntity.of(account);
 }

@GetMapping("/users")
public Object getUsers() {
   var userList = userS.getAllUsers();
   return ResponseEntity.ok(userList);
}

 @ExceptionHandler(UserFriendlyException.class)
 public Object handler(UserFriendlyException ex) {
          return ResponseEntity.status(HttpStatus.NOT_FOUND).body(ex.getMessage());
 }

 @ExceptionHandler(NullPointerException.class)
 public Object handler(NullPointerException ex) {
          return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Not correct data");
 }
```

```
}
```

Comments:

```
package ua.web.app.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class User {

 @Id
 @GeneratedValue
 private Long id;

 @Column(nullable = true)
 private Long id_parent;

 @Column(nullable = false)
 private String comment;

 @Column(nullable = false)
 private Date createdAt;

 @Column(nullable = true)
 private int rating;

 @Column(nullable = false)
 private String path;

 public User() {
          id_parent = "";
          comment = "";
          createdAt = "";
          rating = "";
         path = "";
 }

 public Long getIdParent() {
          return id_parent;
 }

 public String getCommentl() {
          return comment;
 }

 public String getCreatedAt() {
          return createdAt;
```

```
}

public String getRating() {
        return rating;
}

public void setRating(String rating) {
        this.rating = rating;
}

public String getPath() {
        return path;
}

public Long getId() {
        return id;
}


}
```

CommentsController:

```
package ua.web.app.controller;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import ua.web.app.model.Comment;
import ua.web.app.model.User;
import ua.web.app.service.CommentService;
import ua.web.app.service.UserService;

@RestController
@RequestMapping("/comment")
public class CommentController {

   private @Autowired UserService userService;
   private @Autowired CommentService commentService;

   @GetMapping("/create")
   public Object createComment(HttpServletRequest request) {
      var user = userService.validateUser(request);
      var comment = commentService.createComment(request);
      return ResponseEntity.status(HttpStatus.CREATED).body(comment);
   }

   @GetMapping("/delete")
   public Object deleteComment(HttpServletRequest request) {
```

```
        var user = userService.validateUser(request);
        var comment = commentService.deleteComment(request);
        return ResponseEntity.status(HttpStatus.OK).body(comment);
    }

    @GetMapping("/list")
    public Object getComments(HttpServletRequest request) {
        var user = userService.validateUser(request);
        var messageId = Long.parseLong(request.getParameter("messageId"));
        var comments = commentService.getCommentsForMessage(messageId);
        return ResponseEntity.status(HttpStatus.OK).body(comments);
    }

    @GetMapping("/info")
    public Object getCommentInfo(HttpServletRequest request) {
        var commentId = Long.parseLong(request.getParameter("commentId"));
        var comment = commentService.getCommentById(commentId);
        return ResponseEntity.status(HttpStatus.OK).body(comment);
    }
}
```

Stories:

```
package ua.web.app.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Stories {

    @Id
    @GeneratedValue
    private Long id;

    @Column(nullable = false, unique = true)
    private String title;

    @Column(nullable = false, unique = true)
    private int numStory;

    @Column(nullable = false)
    private String status;

    @Column(nullable = true)
    private String genres;

    @Column(nullable = true)
    private Integer seasons;

    @Column(nullable = true)
```

```
private Integer episodes;

@Column(nullable = true)
private String author;

@Column(nullable = true)
private String description;

@Column(nullable = true)
private String cover;

public Stories() {
}

public Long getId() {
   return id;
}

public String getTitle() {
   return title;
}

public void setTitle(String title) {
   this.title = title;
}

public int getNumStory() {
   return numStory;
}

public void setNumStory(int numStory) {
   this.numStory = numStory;
}

public String getStatus() {
   return status;
}

public void setStatus(String status) {
   this.status = status;
}

public String getGenres() {
   return genres;
}

public void setGenres(String genres) {
   this.genres = genres;
}

public Integer getSeasons() {
   return seasons;
```

```
    }

    public void setSeasons(Integer seasons) {
        this.seasons = seasons;
    }

    public Integer getEpisodes() {
        return episodes;
    }

    public void setEpisodes(Integer episodes) {
        this.episodes = episodes;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getCover() {
        return cover;
    }

    public void setCover(String cover) {
        this.cover = cover;
    }
}
```

StoryController:

```
package ua.web.app.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import ua.web.app.model.Stories;
import ua.web.app.service.StoryService;

@RestController
@RequestMapping("/story")
public class StoryController {
```

```
    private final StoryService storyService;

    @Autowired
    public StoryController(StoryService storyService) {
        this.storyService = storyService;
    }

    @PostMapping("/create")
    public ResponseEntity<Long> createStory(@RequestBody Stories story) {
        Long id = storyService.createStory(story);
        return ResponseEntity.status(HttpStatus.CREATED).body(id);
    }

    @GetMapping("/get/{id}")
    public ResponseEntity<Stories> getStory(@PathVariable Long id) {
        Stories story = storyService.getStory(id);
        if (story != null) {
            return ResponseEntity.status(HttpStatus.OK).body(story);
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }

    @PutMapping("/update/{id}")
    public ResponseEntity<Void> updateStory(@PathVariable Long id, @RequestBody
Stories updatedStory) {
        boolean isUpdated = storyService.updateStory(id, updatedStory);
        if (isUpdated) {
            return ResponseEntity.status(HttpStatus.OK).build();
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<Void> deleteStory(@PathVariable Long id) {
        boolean isDeleted = storyService.deleteStory(id);
        if (isDeleted) {
            return ResponseEntity.status(HttpStatus.OK).build();
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }
}
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка інформаційної системи підтримки ігроків з використанням технологій ReactJS та Java/Spring»
студентки групи 122-19-4 Пітіченко Антоніни Миколаївни

Керівник економічного розділу
професор каф. ПЕП та ПУ, к.е.н **О. Г. Вагонова**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
ПітіченкоДиплом.doc	Пояснювальна записка до кваліфікаційної роботи у форматі Word.
ПітіченкоДиплом.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
ПітіченкоДиплом.zip	Архів, який містить код програми.
Презентація	
ПітіченкоДиплом.pptx	Презентація кваліфікаційної роботи