**Пояснювальна записка  
до курсової роботи**

на тему: Система підтримки діяльності онлайн кінотеатру

КПІ.ІП-0107.045450.02.81

Київ – 2022

Зміст

[Перелік умовних позначень 4](#_Toc115992741)

[Вступ 5](#_Toc115992742)

[1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6](#_Toc115992743)

[1.1 Загальні положення 6](#_Toc115992744)

[1.2 Змістовний опис і аналіз предметної області 6](#_Toc115992745)

[1.3 Аналіз існуючих технологій та успішних IT-проєктів 6](#_Toc115992746)

[1.3.1 Аналіз відомих алгоритмічних та технічних рішень 6](#_Toc115992747)

[1.3.2 Аналіз допоміжних програмних засобів та засобів розробки 7](#_Toc115992748)

[1.3.3 Аналіз відомих програмних продуктів 7](#_Toc115992749)

[1.4 Аналіз вимог до програмного забезпечення 7](#_Toc115992750)

[1.4.1 Розроблення функціональних вимог 8](#_Toc115992751)

[1.4.2 Розроблення нефункціональних вимог 9](#_Toc115992752)

[1.5 Постановка задачі 9](#_Toc115992753)

[Висновки до розділу 10](#_Toc115992754)

[2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 11](#_Toc115992755)

[2.1 Моделювання та аналіз програмного забезпечення 11](#_Toc115992756)

[2.2 Архітектура програмного забезпечення 11](#_Toc115992757)

[2.3 Конструювання програмного забезпечення 11](#_Toc115992758)

[2.4 Аналіз безпеки даних 13](#_Toc115992759)

[Висновки до розділу 13](#_Toc115992760)

[3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 14](#_Toc115992761)

[3.1 Аналіз якості ПЗ 14](#_Toc115992762)

[3.2 Опис процесів тестування 14](#_Toc115992763)

[3.3 Опис контрольного прикладу 15](#_Toc115992764)

[Висновки до розділу 15](#_Toc115992765)

[4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 16](#_Toc115992766)

[4.1 Розгортання програмного забезпечення 16](#_Toc115992767)

[4.2 Підтримка програмного забезпечення 16](#_Toc115992768)

[Висновки до розділу 17](#_Toc115992769)

[ВИСНОВКИ 18](#_Toc115992770)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc115992771)

Перелік умовних позначень

|  |  |  |
| --- | --- | --- |
| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| API | – | Application programming interface, прикладний програмний Інтерфейс |
| SDK | – | Software development kit |
| IT | – | Інформаційні технології |
| ER | – | Entity-Relation diagram |
| OC | – | Операційна система. |
| БД | – | База даних. |
| TMDB |  |  |
| 2FA |  |  |
| API |  |  |
| MFA |  |  |
| JS |  |  |
| DOM |  |  |

Вступ

<У вступі стисло викладають:

актуальність роботи та підстави для її виконання; світові тенденції розв’язання поставлених проблем і/або завдань; оцінку сучасного стану об’єкта розробки, розкриваючи практично розв’язані завдання провідними науковими установами та організаціями, а також провідними вченими й фахівцями певної галузі; можливі сфери застосування. 1 стр. чи більше>

# АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальні положення

Онлайн-кінотеатри стали невід'ємною частиною сучасного життя пропонуючи користувачам можливість дивитися фільми і телепрограми з будь-якого пристрою з доступом до інтернету. Онлайн-кінотеатри використовують різні технології для забезпечення інтерактивного перегляду. Загалом, такі системи можуть пропонувати різноманітні функції, серед яких можна виділити наступні:

1. Перегляд фільмів за категоріями або жанрами;
2. Оцінка фільмів;
3. Пошук конкретного фільму для перегляду;
4. Коментування фільмів;
5. Формування персональних списків фільмів

Отже, наша система підтримки діяльності онлайн кінотеатру має забезпечити реалізацію цих функцій.

Оскільки система має надавати користувачам можливість переглядати фільми, а також утримувати інформацію про користувачів, їх збережені фільми та іншу додаткову інформацію, то доцільним буде використання БД. Для зберігання даних буде використана база даних Firestore.

Firestore – це NoSQL-база даних, яка надає високодоступні та масштабовані можливості зберігання даних[1]. Firestore використовує хмарну інфраструктуру, що дозволяє легко масштабувати та забезпечувати високу доступність даних. Дані організовані у вигляді документів у колекціях, що дозволяє ефективно взаємодіяти з ними та отримувати швидкий доступ до необхідної інформації.

Також Firestore підтримує режим реального часу, що дозволяє миттєво отримувати зміни у даних без необхідності оновлення сторінок чи додатків. База даних дозволяє додавати нові, видаляти старі та редагувати поля до існуючих документів без перерви в роботі системи через запити.

Звичайно, застосунок, що має можливість аутентифікації користувачів має надавати надійний функціонал, що забезпечить безпеку даних. Отже, для створення особистих облікових записів у системі підтримки діяльності онлайн кінотеатру буде використана система Firebase Auth.

Firebase Auth — це система аутентифікації, яка надає різні методи аутентифікації[2]. В нашому випадку оберемо аутентифікацію через електрону пошту та пароль. Цей метод є найпоширенішим і забезпечує достатній рівень безпеки.

Firebase Auth забезпечує безпеку даних користувачів за допомогою наступних заходів:

1. Хешування паролів: паролі користувачів хешуються перед їхнім зберіганням у базі даних[3]. Це ускладнює відновлення паролів у разі їхнього втрати;
2. Двофакторна аутентифікація (2FA): 2FA надає додатковий рівень безпеки, вимагаючи від користувача ввести код, який надсилається на його пристрій під час входу в систему[4];
3. Безпека даних на рівні сервера: Firebase Auth реалізує ряд заходів безпеки на рівні сервера, таких як фільтрація вхідних даних та перевірка безпеки API[5].

Для розробки веб-застосунку системи підтримки діяльності онлайн кінотеатру буде використана технологія React.

React - це бібліотека для створення інтерфейсів користувача, яка була розроблена Facebook[6]. Вона дозволяє розробникам створювати великі веб-додатки, які використовують дані, які можуть змінюватися з часом, без перезавантаження сторінки. Це робить його ідеальним для створення сучасних односторінкових веб-додатків[7].

React має ряд переваг, які роблять його придатним для розробки веб-застосунків для онлайн кінотеатру:

* Компонентність: React дозволяє розробляти інтерфейс користувача з використанням компонентів. Це спрощує розробку та обслуговування коду;
* Декларативність: React-код є декларативним, що означає, що розробник описує, що він хоче побачити на екрані, а не те, як це досягти. Це робить код більш зрозумілим і читабельним;
* Ефективність: React використовує техніку, відому як Virtual DOM, для ефективного рендерингу інтерфейсу користувача. Це дозволяє React швидко та плавно оновлювати інтерфейс користувача при зміні даних.

Для реалізації функціональності системи підтримки діяльності онлайн кінотеатру буде використана React-бібліотека React Router. Вона дозволяє розробляти веб-застосунки з динамічними маршрутами[8]. React Router використовує техніку, відому як компоненти маршрутизації, для визначення маршрутів у веб-застосунку. Компоненти маршрутизації відповідають за відображення вмісту сторінки, яка відповідає певному маршруту.

## Змістовний опис і аналіз предметної області

На сучасному етапі розвитку ІТ-технологій індустрія розваг, зокрема сфера онлайн кінотеатрів, є однією з перспективних та швидкорозвиваючих галузей. Вона відзначається стрімким зростанням популярності та поширенням сервісів для перегляду відео контенту в Інтернеті.

Існує багато платформ для перегляду та прокату фільмів, які пропонують користувачам широкий спектр контенту, зокрема фільми, серіали, документальні фільми та телешоу. Однак ці платформи також мають низку недоліків, які можуть негативно вплинути на користувацький досвід.

На сьогоднішній день, використання ІТ-технологій у сфері онлайн кінотеатрів стало нормою. Програмне забезпечення, яке використовується для підтримки діяльності онлайн кінотеатру, включає в себе різні компоненти, такі як системи управління контентом, аутентифікація користувачів, зберігання та обробка даних.

Однак, незважаючи на широке використання цих технологій, вони мають й певні недоліки. Наприклад, багато систем не забезпечують достатнього рівня безпеки даних, що може призвести до витоку приватної інформації користувачів. Крім того, деякі системи можуть бути складними у використанні, що знижує задоволеність користувачів.

Також, однією з ключових вимог сучасних глядачів є забезпечення зручного та комфортного середовища для перегляду фільмів. Недостатня функціональність та низька інтерактивність деяких існуючих онлайн кінотеатрів може призводити до негативного досвіду використання.

Інтеграція функцій взаємодії користувачів, таких як коментування, оцінювання та обмін рекомендаціями, допоможе створити сприятливе комунікаційне середовище серед глядачів.

Розробка системи персоналізації контенту та зручного управління списку улюблених фільмів дозволить кожному користувачеві створити унікальний досвід перегляду.

В рамках цієї курсової роботи, ми обрали шлях розробки веб-застосунку для онлайн кінотеатру “Mediatoria”, який не тільки дозволяє користувачам переглядати фільми онлайн, але й взаємодіяти з системою через коментарі та оцінки. Наша мета - створити зручне середовище та інтерфейс для користувачів, де вони зможуть переглядати, оцінювати, коментувати та зберігати фільми з різних пристроїв. Ми використовуємо React для розробки фронтенду, Firebase та Firestore для аутентифікації та утримання даних, а також TMDB для завантаження фільмів. Це дозволяє нам вдосконалити процес використання знань предметної області в програмному забезпеченні.

## Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку для підтримки діяльності онлайн-кінотеатрів «Mediatoria». Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

### Аналіз відомих алгоритмічних та технічних рішень

Реєстрація користувачів – ключовий етап створення будь-якої системи, і вибір правильного методу грає критичну роль у забезпеченні безпеки, ефективності та зручності для користувача застосунка. В ході створення системи необхідно визначити методи для реєстрації, керування власною системою, використання сторонніх сервісів та інші. Отже, спробуємо визначити підходити для вирішення задач.

Існує кілька різних методів розв'язання задачі реєстрації користувачів. Найпоширенішими з них є:

* Використання локальної бази даних;
* Використання сторонньої служби аутентифікації.

Використання локальної бази даних дозволяє зберігати облікові записи користувачів у локальній базі даних програми. Цей спосіб простий в реалізації, але має ряд недоліків. Зокрема, він не дозволяє масштабувати додаток, оскільки база даних може бути перевантажена великою кількістю користувачів.

Використання стороннього сервісу автентифікації, в свою чергу, зберігає облікові записи користувачів у сторонньому сервісі автентифікації, такому як Google, Facebook або Twitter. Цей метод дозволяє масштабувати додаток, оскільки навантаження розподіляється між різними серверами стороннього сервісу.

Firebase Auth - це сторонній сервіс автентифікації, що надається компанією Google. Firebase Auth пропонує широкий спектр функцій, які дозволяють легко реалізувати реєстрацію користувачів у веб-додатках. До переваг Firebase Auth можна віднести:

* Економія часу на розробку методів аутентифікації;
* Отримання детальної аналітики та демографічної інформації про користувачів;
* Інтеграція з іншими сервісами Firebase, що полегшує спільне використання сервісів;
* Використання таких стандартів, як OAuth 2.0 та OpenID Connect, які можна легко інтегрувати з власним бекендом[9];
* Захист даних користувачів шляхом надання безпечних методів аутентифікації.

Задача зберігання даних користувача є однією з найголовніших у веб-додатках. У нашому випадку нам потрібно зберігати такі дані користувача, як: ім'я, електронна пошта, дата реєстрації, список збережених фільмів, що дозволяє зберігати інформацію про користувачів, таку як їхні імена, паролі, адреси електронної пошти, коментарі та інші дані.

Існує кілька різних методів розв'язання задачі збереження даних користувачів. Найпоширенішими з них є:

* Використання локальної бази даних;
* Використання сторонньої бази даних;
* Використання хмарної бази даних.

Локальна база даних надає можливість зберігати дані користувачів у локальній базі даних додатку. Цей метод простий в роботі, але має ряд недоліків. Зокрема, він не дозволяє масштабувати додаток, оскільки база даних може бути перевантажена великою кількістю користувачів.

При використанні методу сторонньої бази даних дані користувачів розміщуються в сторонній базі даних (MySQL[10], PostgreSQL[11] або MongoDB[12]). Цей метод вже забезпечує масштабованість, оскільки навантаження розподіляється між різними незалежними серверами баз даних.

Хмарні бази даних надають можливість зберігати всі необхідні дані в хмарі (Google Cloud Platform[13], Amazon Web Services[14] або Microsoft Azure[15]). Вони використовуються для структурованого зберігання інформації та надають потужні можливості запитів для пошуку та маніпулювання даними. Завдяки своїй зручності, стабільності та функціональності вони мають перевагу над іншими методами, що використовуються.

Firestore - це хмарна база даних NoSQL від Google. Вона надає гнучке, масштабоване сховище даних для мобільних, веб- та серверних додатків. Серед найважливіших переваг інструменту можна відзначити наступні[16]:

* Асинхронні запити;
* Взаємодія з базою даних за допомогою простих запитів, які можна використовувати для виконання складних NoSQL запитів;
* Забезпечення синхронізації в режимі офлайн, що допомагає користувачам взаємодіяти з застосунком за відсутності мережевого з'єднання;
* Автоматичне горизонтальне масштабування відповідно до навантаження[17];
* Використання протоколу HTTPS для шифрування передачі даних та вбудованих механізмів безпеки, які визначають правила захисту від спроб несанкціонованого доступу.

Ще однією проблемою є наповнення застосунку контентом. Оскільки фільмотека має бути різноманітною та актуальною, потрібно вирішити, як отримувати дані про фільми.

Одним з методів є власноручне додавання фільмів. В цьому випадку, адміністратори будуть вручну додавати та оновлювати інформацію про фільми. Цей метод часом використовується в невеликих проектах, проте він потребує багато часу та є неефективний для великих обсягів даних. Отже, необхідно отримувати інформацію з окремих публічних джерел.

Наш застосунок буде використовувати TMDB API[18]. Отримання даних з БД, у такому випадку, здійснюється через використання API. API TMDB є вирішенням вищеописаної проблеми та також має додаткові переваги[19]:

* Велика кількість даних;
* Надання простого у використанні інтерфейсу, що дозволяє розробникам легко отримувати дані про фільми;
* Розроблений для масштабування, що дозволяє йому обробляти великі обсяги запитів;
* Розширені функції пошуку фільмів за різними критеріями (жанр, рік виходу фільмів).

HTTP-запити - це основний спосіб взаємодії веб-додатків із зовнішніми ресурсами, такими як сервери API, веб-сайти та бази даних. Існує багато методів та технологій здійснення HTTP запитів у JavaScript, проте ми зазначимо найактуальніші[20].

Першим зазначимо доволі старий метод – XMLHttpRequest. Цей інтерфейс є вбудованим та відомим багатьом розробникам. Однак його API є менш зручним та потребує багато коду для реалізації навіть простих запитів.

Іншим сучасним методом є Fetch API, що був введений у стандарті HTML 2015. Він підтримує проміси та забезпечує гнучкий API для виконання запитів до серверів та роботи з відповідями[21]. Можна було б спокійно обрати Fetch API, проте він все ж має більш обмежені функції в порівнянні з Axios та не є доступним у всіх браузерах старих версій[22].

Axios – це бібліотека JavaScript для виконання HTTP-запитів з браузера або Node.js. Вона надає простий у використанні API і підтримує різні функції, включаючи запити потокового передавання. Має наступні переваги[23]:

* Використовує Promise API, що дозволяє використовувати async/await, щоб зробити код здійснення запитів більш зрозумілим та синхронним[24];
* Працює на різних браузерах, забезпечуючи однакову поведінку;
* Автоматичне перетворення даних JSON;
* Інтерцептори HTTP – здатність перехоплювати HTTP-запити для дослідження або зміни HTTP-запитів від вашого додатку до сервера або навпаки (реєстрація, аутентифікація, повторний запит);
* Підтримка запитів потокового передавання, що дозволяє обробляти дані по мірі їх надходження, а не чекати отримання всієї відповіді[25].

Для здійснення аутентифікації у JavaScript існують різні методи:

* Використання локальної БД, що є простим рішенням у реалізації, проте воно не дозволяє масштабувати застосунок через перевантаженість системи та сама система може бути вразливою до атак;
* Використання сторонньої служби БД (Google, Facebook), що дозволяє масштабувати застосунок, оскільки навантаження розподіляється між окремими серверами. Надійність гарантується службою.
* Використання власної сторонньої служби. Вона забезпечує повний контроль над процесом підтвердження особи, але потребує найбільших зусиль у реалізації та підтримці.

Як видно з переліку найбільш гарантованим та комфортним у використанні є метод отримання послуг від сторонніх служб. Одною з таких є Firebase Auth через зручну інтеграцію за допомогою API. Також, маємо змогу отримувати детальну аналітику та налаштовувати різноманітні методи аутентифікації, включаючи адресу електронної пошти та пароль, телефонні номери, а також зовнішні провайдери, такі як Google, Facebook, або Twitter. Він автоматично обробляє усі аспекти безпеки.

Firebase Auth надає можливість використовувати мультифакторну аутентифікацію, забезпечуючи додатковий шар безпеки через використання додаткового фактору, такого як SMS-код чи аутентифікація на пристрої[26].

Окрім цього, маємо можливість легко керувати правами доступу, визначаючи рівень доступу для кожного типу користувачів та надаючи їм управління даними[27]. Оскільки маємо 3 можливих стани користувачів (неавторизований, звичайний користувач та адміністратор), то даний функціонал стане у нагоді.

Розробка фронтенду на JavaScript включає в себе різні інструменти та бібліотеки, які розробники використовують для створення привабливих та ефективних веб-застосунків. Звичайно, стандартний JS вже надає інструментарій для створення інтерфейсів[28]. Проте, він не пропонує функціонал, що зробить процес розробки легшим. У цьому контексті, були створені альтернативи до використання стандартного JS якими є:

* Бібліотека jQuery, що має більший функціонал ніж стандартна версія JS[29]. Вона була популярна в минулому для спрощення роботи з DOM та взаємодії зі стороною клієнта. Проте, із розвитком сучасних бібліотек та фреймворків, використання jQuery визнається менш ефективним.
* Angular, React та Vue – фреймворки JavaScript, які надають повний набір інструментів для розробки веб-додатків[30]. Angular та Vue схожі у використанні, проте саме Vue визначається своїм прогресивним підходом та гнучкістю. У свою чергу, React відомий за свій декларативний синтаксис та компонентний підхід.

Звичайно, що для розробки фронтенду необхідно обрати актуальний варіант з повним набором інструментів. В нашому випадку оберемо React, що додатково має наступні переваги:

1. Використання компонентної архітектури, дозволяє розробникам створювати повторно використовувані компоненти для інтерфейсу користувача;
2. Використання віртуального DOM, який дозволяє додатку ефективно оновлювати та відображати компоненти;
3. Використання моделі рендерингу на стороні сервера (SSR), яка дозволяє генерувати HTML-код на сервері. Це робить додатки швидшими за додатки, створені на інших фреймворках.

### Аналіз допоміжних програмних засобів та засобів розробки

<У пункті викладають:

опис допоміжних програмних засобів та їх порівняльний аналіз, якщо існує кілька аналогів (мова йде про сторонні бібліотеки, пакети, фреймворки, тощо). Розглядаються засоби розробки за допомогою яких можна виконати вашу розробку, робиться їх порівняльний аналіз та на його основі обираються ті, що вам підходять (маються на увазі мови програмування та IDE).>

### Аналіз відомих програмних продуктів

<У пункті викладають:

опис готових програмних продуктів за вашою предметною областю, які частково чи повністю реалізують функціонал описаний у технічному завданні. Обов’язково має бути порівняльна таблиця функціоналу та особливостей з вашою розробкою.>

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняння з аналогом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал | Дипломний проєкт(назва) | Аналог для порівняння 1 | Аналог для порівняння n | Пояснення |
|  |  |  |  |  |

## Аналіз вимог до програмного забезпечення

<У підрозділі викладають:

опис головного(-х) функціоналу розробки, діаграму варіантів використання та опис варіантів використання. Обсяг 5 стр., чи більше >

Головною функцією програмного забезпечення є …, більше функцій можна побачити на рисунку 1.3.

Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.2 - 1.17 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Гість (незареєстрований користувач) |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: пошта користувача, пароль в системі, та його повтор для підтвердження, а також чек бокс для підтвердження умов сервісу. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу. |
| Extension | В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом. |
| Post-Condition | Створення сторінки користувача, перехід на сторінку входу |

### Розроблення функціональних вимог

<У пункті викладають:

модель вимог, їх опис та матрицю трасування.>

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.4 наведено загальну модель вимог, а в таблицях 1.18 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.5.

Рисунок 1.4 – Модель вимог у загальному вигляді

Таблиця 1.16 – Функціональна вимога FR-1

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Система повинна надавати можливість реєстрації користувачеві шляхом введення пошти, паролю, підтвердження паролю. |

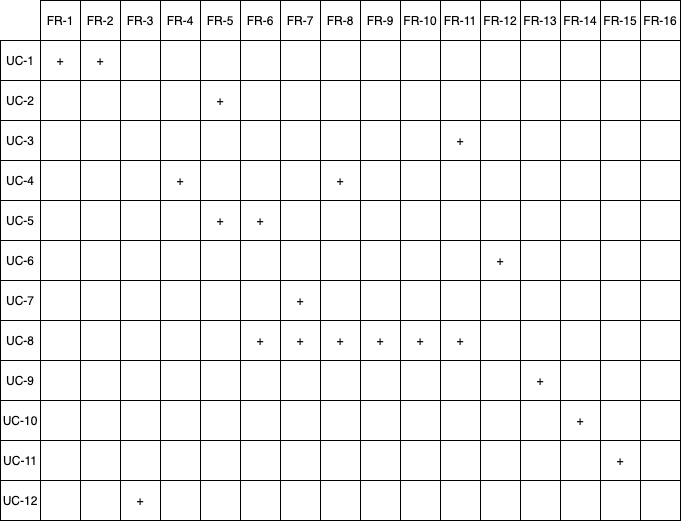


Рисунок 1.11 – Матриця трасування вимог

### Розроблення нефункціональних вимог

<У пункті викладають:

опис нефункціональних вимог.>

## Постановка задачі

<У підрозділі викладають:

що саме ви збираєтесь реалізувати. Це не зовсім вимоги і варіанти використання. Це достатньо узагальнене поняття з метою, цілями та задачами, що підлягають розв’язанню у результаті розробки програмного забезпечення на курсову роботу. Обсяг 1 сторінка, чи більше. >

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Моделювання та аналіз програмного забезпечення

<У підрозділі викладають:

опис бізнес процесів вашої розробки та їх наочне представлення за допомогою засобів моделювання (BPMN та інші). Обсяг 2 сторінки, чи більше. >

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 2.1).

Опис послідовності створення облікового запису користувача:

* користувач переходить на сторінку реєстрації;
* користувач заповнює поля реєстрації;
* якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою;

## Архітектура програмного забезпечення

<У підрозділі викладають:

втілення архітектурного паттерну для вашого ПЗ та деталізація його компонентів за необхідності. Необхідно виконати повний опис архітектури (схеми, таблиці та ін.). Обсяг 2 сторінки, чи більше.>

## Конструювання програмного забезпечення

<У підрозділі викладають:

опис оригінальних алгоритмів чи модифікацій існуючих. Опис структур даних, програмних структур та ін. Опис бази даних з представленням концептуальної, логічної чи фізичної моделі та з описом сутностей чи таблиць. Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці. Аналіз системних вимог. Обсяг 4 сторінки, чи більше.>

Бібліотека xpring, яка виконує запити в мережу блокчейн – однопоточна, тобто не може оброблювати паралельні запити. Через таку специфіку бібліотеки, довелось розробляти рішення яке б не блокувало запити з різних потоків, або інших частин коду. За основу ідеї алгоритму була взята багатопоточність самої мови програмування java. Java для вирішення проблем з потоками представляє ділянки коду які можна синхронізувати, що унеможливлює одночасний доступ до нього з різних потоків. Ця синхронізація проходить за допомогою передачі управління об’єкта-монітора. Алгоритм вирішення проблеми з доступом до однопоточної бібліотеки наведено на рисунку 2.3.

В якості системи управління базами даних використовується Postgres. База даних серверу призначена для зберігання користувачів, а також даних про їх …. Опис таблиць бази даних наведено у таблицях 2.11 - 2.14. Модель бази даних наведена на рисунку 2.12.

Таблиця 2.11 – Опис таблиці user

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| user | id | serial | ідентифікаційний номер користувача |
| email | varchar | електронна пошта користувача |
| password\_id | int | посилання на запис у таблиці password, де зберігається пароль користувача |

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.22 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва утиліти | Опис застосування |
| 1 | IntelliJ IDEA | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Postman | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 3 | MySQL Workbench | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних. |

## Аналіз безпеки даних

<У підрозділі викладають:

аналіз вразливостей ПЗ та будь-які питання пов’язані з безпекою даних. >

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

<У підрозділі викладають:

аналіз якості ПЗ за певними метриками. Обсяг 2 сторінки, чи більше.>

## Опис процесів тестування

<У підрозділі викладають:

опис процесів тестування та приклади тестів. Обсяг 4 сторінки, чи більше.>

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.3 – 3.30.

Таблиця 3.3 – Тест 1.1

|  |  |
| --- | --- |
| Тест | Реєстрація користувача |
| Модуль | Реєстрація користувача |
| Номер тесту | 1.1 |
| Початковий стан системи | Користувач знаходиться на сторінці реєстрації |
| Вхідні данні | Електронна пошта, пароль, підтвердження паролю |
| Опис проведення тесту | У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 10 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ, і який не входить у топ 10000 найпопулярніших паролей, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього … |
| Очікуваний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації. |
| Фактичний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації. |

## Опис контрольного прикладу

<У підрозділі викладають:

повний опис контрольного прикладу з усіма можливими розгалуженнями та особливостями. Кроки доповнюють ілюстраціями. Не обов’язковий розділ.>

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

<У підрозділі викладають:

повний опис покрокового розгортання ПЗ. Кроки доповнюють ілюстраціями. Обсяг 2 сторінки, чи більше.>

Клієнтську і серверну частини програмного забезпечення було вирішено розгорнути на платформі Heroku. Для розгортання було використано сервіс GitHub Actions, який надає можливості для постійної інтеграції і розгортання [13].

Розгортання починається коли новий код застосунку доставляється у репозиторій у гілку main. Тоді у середовищі GitHub Actions створюється Docker image за допомогою Dockerfile, що знаходиться у проекті. Цей image розгортається у Heroku за допомогою пакету heroku-deploy. Інформацію про розгортання клієнтської і серверної частини проекту можна побачити на рисунках 4.1 і 4.2.

Рисунок 4.1 - Інформація про розгортання клієнту

## Підтримка програмного забезпечення

<У підрозділі викладають:

опис того, як буде виконуватись підтримка програмного забезпечення. Ілюстрації. Обсяг 1 сторінка, чи більше.>

Користувачі повинні мати можливість отримати нову версію консольного застосунку з кожною версією. До того ж кожна нова версія консольного застосунку повинна бути опублікована в npm. Для автоматизації цього процесу був використаний сервіс GitHub Actions.

Створення нового випуску починається, коли нова версія консольного застосунку доставляється у репозиторій у гілку main, тобто коли commit має tag формату “v\*.\*.\*.”, де замість “\*” знаходиться число. Тоді у середовищі GitHub Actions встановлюється NodeJS. Після цього для проекту встановлюються залежності і проект збирається. Bash скрипт за допомогою бібліотеки pkg генерує виконувані файли (executables) для Linux і для Windows, та пакує файл для Linux у .deb пакет. Після цього .deb пакет і файл для Windows архівуються,

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВИСНОВКИ

У висновках викладають найважливіші наукові й практичні результати роботи та наводять:

* оцінку одержаних результатів і їх відповідність сучасному рівню наукових і технічних знань;
* ступінь впровадження та можливі галузі або сфери використання результатів роботи;
* наукову, науково-технічну, соціально-економічну значущість роботи;
* доцільність продовження досліджень за відповідною тематикою тощо.

Також у висновках необхідно відобразити стан вирішення усіх поставлених в курсовій роботі задач.

В результаті виконання курсової роботи було спроєктовано …

В якості середовища розробки обрано …

У якості БД використано …

Після реалізації застосунку він був протестований на пристроях з різними версіями Android, з різними розмірами екранів щоб переконатися, що додаток акуратно відображається на різних пристроях.

Наукова новизна роботи (якщо вона є) полягає в наступному (достатньо вказати щось одне).

Вперше:

* реалізовано можливість запитів від пацієнта до лікаря;
* використано те-то, що дозволило те-то.

Модифіковано:

* те-то, що дозволило те-то.

Набуло подальший розвиток:

* те-то, що дозволило те-то.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Оформлення бібліографії ДСТУ 7.1:2006](https://drive.google.com/file/d/1VUr7fwKHOk8t1u8sh-Sv2dlEc_6OBoIS/view?usp=sharing)
2. [Оформлення бібліографії ДСТУ 8302-2015](https://drive.google.com/file/d/17RYEYVnSXolL0S8D1k85VfReYd2yC6WO/view?usp=sharing) з прикладами