Приватний заклад вищої освіти

Одеський технологічний університет «ШАГ»

Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота

«Інформаційна система поширення та актуалізації програмних ресурсів (за зразком Steam)(Frontend)»

на здобуття бакалаврського рівня вищої освіти

зі спеціальності «122 Комп’ютерні науки»

Розробники:

№ з/п	ПІБ	Роль
1	Ринкова Ніка	Розробка UI/UX - дизайну
2	Лінчевский Борислав	Розробка (backend)
3	Тарасенко Степан	Розробка (frontend)
4	Павлова Анастасія	DevOps
5	Олейник Катерина	Розробка UI/UX - дизайну

Ментори:

№ з/п	ПІБ, посада	Напрямок
1	Черкезян Крістіне Арутюнівна	Розробка ПЗ
2	Ночовкіна Інна	Графічний дизайн
3	Зайцев Вадим	DevOps

Автор звіту

_____ Тарасенко Степан Сергійович

ЗМІСТ

Анотація

УДК 004.9

Д-30
Тарасенко С. С. Платформа онлайн-маркетплейсу ігрових додатків: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності “122 Комп’ютерні науки”. - Одеса: ОТУШ, 2024 -13 с.

Робота присвячена аналізу, розробці та тестуванню онлайн-маркетплейсу ігрових додатків. Головна функцією цієї платформи полягає в наданні користувачам можливості купувати продукти, розміщені на маркетплейсі їхніми власниками, з можливістю для кожного користувача виступати як у ролі клієнта, так і у ролі власника продукту. Другорядною функцією онлайн-платформи є надання можливості користувачам додавати інших клієнтів у список "друзів", об'єднуватися в спільноти та обмінюватися повідомленнями за допомогою чатів. Платформа поділяється на велику кількість частин, кожна з яких має свою функціональність. Основними функціями цих частин є:

- Авторизація
- Автентифікація
- Реєстрація
- Додавання продуктів
- Додавання користувачів у “друзі”
- Об’єднання у спільноти
- Додавання категорій для продуктів
- Додавання медіа файлів
- Додавання коментарів до медіа файлів
- Додавання коментарів до сторінки користувача
- Додавання продуктів у список “бажаних”

- Додавання продуктів до користувацьких колекцій

Релевантність проекту аналізується через реалізацію актуальних функцій, які реалізовані розробниками задля спрощення процесу розміщення продуктів на маркетплейсі та придбання цих продуктів користувачами. Зручність у використанні сервісу також у способі реєстрації через підтвердження акаунту через повідомлення на пошту. Найбільшу кількість уваги було приділено організації архітектури бази даних та організації пошуку продуктів та додавання їх в користувацькі колекції. Це дозволяє максимально швидко знаходити продукти на сервісі та додавати куплені продукти в користувацькі колекції, що також полегшує організацію цих продуктів.

Актуальність сервісу також ґрунтується на впровадженні надійної системи безпеки для захисту особистих персональних даних. В цю систему входить шифрування вхідних та вихідних персональних даних, таких як пароль, також впровадження системи JWT та покладання його в cookie сайту, а сам JWT існує тільки одну годину, після вичерпання котрої цей JWT треба перегенерувати. Сам JWT шифрується за алгоритмом HmacSha256. Пароль шифрується за алгоритмом SHA384. Також впроваджена система ідентифікації елементів в базі даних за допомогою Guid, що дає змогу уникнути небажаних запитів з використанням SQL-ін’єкцій.

Об’єктом дослідження у рамках даної роботи виступає алгоритм пошуку продуктів з використанням категорій, котрі були вказані в продукті розробником, що спрощує пошук елементів на маркетплейсі. У якості задач дослідження було встановлено наступне:

- проаналізувати предметну область для організації архітектури бази даних на базі MySQL
- проаналізувати різні варіанти реалізації взаємодії клієнтської та серверної частини за допомогою Single Page Application
- розробити програмний інтерфейс (API), впровадивши безпечне отримання вхідних даних тільки від авторизованого користувача, використовувавши JWT за допомогою протокола HTTP
- розробити алгоритм отримання запитів з клієнтської частини на серверну та надсилання відповідей з серверної частини на клієнтську
- реалізувати взаємодію серверної частини з базою даних, використовуючи Entity Framework та додавання

міграцій(Migrations) для внесення змін до архітектури бази даних
- впровадити в серверну частину засоби захисту для забезпечення цілісності користувацьких даних

Методи дослідження, що використані для досягнення поставлених задач, утворені за допомогою методів тестування програмного забезпечення, гнучких методологій розробки, моделювання архітектури за допомогою принципів SOLID та теорії баз даних.

Окрему уваги треба звернути на те, що командна робота була організована за допомогою Agile-методологій, це дозволило нам гнучко розподілити обов'язки між членами команди та найбільш ефективно реалізовувати поставлені задачі, та мати постійний зворотній зв'язок з менторами для того, щоб редагувати код та виправляти помилки у виконаній роботі. Щоб використовувати цю методологію, наша команда використовувала багатоплатформну систему управління проектами Trello, що дозволило нам організувати та впровадити Agile методології, зробити роботу команди більш організованою, контрольованою та максимально ефективною. Також у процесі розробки back-end та front-end розробники користувались методологією екстремального програмування, що дозволило швидше впроваджувати певний функціонал, котрий вимагав додаткової реалізації. Для прискорення розробки також використовували методологію CI\CD, що дозволило всім учасникам команди реалізувати непреривну інтеграцію, пришвидшити пошук дефектів, підвищити продуктивність розробки та найбільш швидко випускати білди проекту.

Також для розробки програмного забезпечення команда розробників використовувала систему контролю версій Git, та інструмент, котрий її реалізує - GitLab. Це забезпечило нам більш гнучкий обмін актуальними версіями проекту і дало можливість скасовувати внесені зміни у разі необхідності. Також це підвищило безпеку та цілісність проекту, так як доступ до GitLab можливий виключно з використанням стороннього VPN "Tailscale" з особистою конфігурацією.

При складанні звіту використано _ посилань на електронні джерела інформації

Преамбула

Проект розробляється за зразком відомої платформи Steam, яка вже зарекомендувала себе як ефективний інструмент для поширення і оновлення програмних продуктів. Перед початком розробки було проведено детальний аналіз існуючих платформ, таких як Steam, Epic Games Launcher, GOG, для визначення основних функціональних вимог та

кращих практик. Було розглянуто цільову аудиторію, їхні потреби та очікування від системи.

Як фронтенд розробник, я відповідав за створення інтерфейсу користувача з урахуванням сучасних тенденцій у веб-дизайні. Основною технологією для реалізації проекту обрано Single Page Application (SPA), що дозволяє забезпечити швидкий та інтерактивний користувацький досвід. Для побудови інтерфейсу використано бібліотеку React, яка забезпечує високу продуктивність та гнучкість у розробці компонентів.

У проекті ми вирішили використовувати React з кількох ключових причин. По-перше, React забезпечує високу продуктивність завдяки своєму віртуальному DOM, що дозволяє ефективно оновлювати та рендерити компоненти. По-друге, компонентна архітектура React сприяє повторному використанню коду та полегшує підтримку і розширення проекту. Крім того, великий вибір бібліотек і інструментів, робить розробку більш гнучкою та ефективною. React також має широку підтримку спільноти та відмінну документацію, що значно полегшує навчання і вирішення проблем.

Верстка сторінок здійснювалася за допомогою CSS-бібліотеки Tailwind, що дозволило швидко створювати адаптивні та стильні інтерфейси. Для розробки елементів дизайн-системи було використано підхід атомарного дизайну, що включає створення базових стилів і компонентів, які можна легко використовувати та модифікувати.

Результатом нашої роботи стала інтуїтивно зрозуміла та функціональна система, яка дозволяє користувачам зручно взаємодіяти з програмними ресурсами, отримувати актуальну інформацію та залишати відгуки.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

Інформаційна система поширення та актуалізації програмних ресурсів

(за прикладом сервісу Steam)

Загальний опис.

Програмне забезпечення (ПЗ) призначене для створення мережевої системи яка дозволяє шукати, встановлювати, оновлювати та оцінювати програмні ресурси. Функції системи мають забезпечити повний життєвий

цикл управління контентом: створення, актуалізація, модифікація та знищення, а також встановлення скорингових параметрів (оцінок) наявного контенту. Головна особливість системи полягає у тому, що наявні у неї інформаційні одиниці мають бути захищені авторським правом у відповідності до умов, що їх висуває власник ПЗ.

Призначення:

ПЗ орієнтується на створення онлайн системи управління інформаційними одиницями на основі аналізу їх поточної версії. Визначальною функцією системи є можливість встановлення обраного ПЗ та контролю його актуального стану. ПЗ створюється за прикладом сервісу Steam, у той же час має складатись за принципом універсальності, мати змогу змінити цільове призначення.

Вимоги до функціоналу. Обов'язкова частина

Головна сторінка

Стартовий інтерфейс ПЗ (головна сторінка або активність) має за мету популяризацію системи. На ньому має бути забезпечено відображення популярних показників як-то загальна кількість користувачів та інформаційних одиниць, вибірка з найбільш популярного контенту та нових дописів, зовнішня реклама. Значний акцент має бути зроблений на детальному пошуку контенту за принципом таргетування. Головна сторінка обмежує функції системи для незареєстрованих користувачів лише переглядом та пошуком, інші функції надаються у разі авторизації користувача.

Реєстрація користувачів:

Користувач ПЗ має змогу зареєструватись як за допомогою авторизаційних даних (логіну/телефону/E-mail/паролю), так і за допомогою облікових записів популярних соціальних мереж (Google, Facebook, тощо).

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон та електронна пошта. На тестовому етапі, за умови

ускладнення замовлення послуг телефонії чи СМС, можливе використання лише електронної пошти. Засіб оперативного контактування при реєстрації підлягає верифікації через відповідь на тестовий виклик чи надісланий лист, або введення коду, пересланого через повідомлення довільного типу. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

За умови що користувач забув пароль, ЗОК має бути використаний для його відновлення. Схема відновлення паролю жорстко не регламентується, рекомендовано вживання тимчасових паролів (one time passwords, OTP), але допускаються й інші схеми.

Також передбачається наявність у системі окремих користувачів з додатковими правами: модераторів та адміністраторів. Реєстрація кореневого адміністратора системи має бути реалізоване при інсталяції системи, наступним користувачам змінюють права наявні адміністратори.

Користувачі системи поділяються на надавачів та споживачів послуг. При реєстрації надавача послуги вимагається додаткова перевірка його прав та ліцензій на можливість надання послуг (розміщення ПЗ) чи ведення електронної комерції. На тестовому етапі достатньо лише факту надсилання користувачеві запиту на відповідні документи, надісланого через ЗОК.

Для користувачів усіх категорій вимагається формальна валідація усіх реєстраційних даних (відповідність загальновживаним або власним шаблонам).

Особисті кабінети користувачів:

Особисті кабінети різних груп користувачів - надавачів та споживачів послуг - мають забезпечувати різну функціональність. Дозволяється на початковому етапі (до впровадження механізмів перевірки ліцензій надавачів послуг) функції надавача послуг делегувати користувачам з підвищеним статусом керування (адміністраторам) через що кабінет надавача послуги може виконуватись як панель адміністрування контента.

Особистий кабінет надавача послуги має забезпечити необхідні засоби для

Управління даними та метаданими (пропозицій та їх описів, цільових груп, тощо)

Створення, редагування, видалення (блокування), перегляд розміщених пропозицій.

Перегляд поточної активності щодо власних пропозицій (коментарів, оцінок, історії завантажень)

Контактування з користувачами-споживачами, що розміщують замовлення на контент, за допомогою ЗОК або через внутрішні засоби системи (чат або мессенджер)

Особистий кабінет споживача послуги має забезпечити необхідні засоби для

Встановлення належності до цільових груп (вибір із заданого переліку)

Пошуку контенту інших користувачів за метаданими, групування та фільтрування результатів за різними ознаками, наявними у знайденому контенті, а також за цільовою групою. Для об'ємних результатів має бути забезпечена пагінація.

Контактування з іншими користувачами-надавачами послуг, що розміщують пропозиції, за допомогою ЗОК або через внутрішні засоби системи (чат або мессенджер)

Також засобами особистих кабінетів має бути передбачено можливість

Зміни персональних даних, зокрема, контактних засобів. У разі зміни важливих даних, що раніше проходили верифікацію, вимагається їх повторна верифікація.

Зворотного зв'язку - встановлення власної оцінки (лайк або бал) для спожитої або наданої послуги, написання відгуку (коментаря) або рекомендації за фактом взаємодії.

Вимоги до функціоналу. Опціональна частина

Модерація

Для забезпечення дотримання загальноприйнятих норм спілкування, а також з метою недопущення публікування контенту, який містить заборонені компоненти (ненормативні висловлювання, заклики до протиправних дії, приниження чи дискримінація, тощо), усі інформаційні одиниці, а також створені користувачами метадані (маркери) мають в обов'язковому порядку пройти погодження (модерацію). Право надати

погодження належить кореневому адміністратору системи, а також модераторам. При погодженні інформаційна одиниця має зберігати ідентифікатор користувача, який надав погодження.

Привілейовані функції

З метою покращення економічної ефективності проєкту передбачається наявність додаткових функцій системи, що доступні лише за умови платної підписки (або внутрішніх досягнень) користувачів. До таких функцій можна віднести окремі групи контенту, що є доступними тільки для привілейованих користувачів, відображення ЗОК інших користувачів (для прямого контактування), розширені засоби фільтрації та групування. Також можливе встановлення обмеження щодо кількості опублікованих одиниць контенту або замовлень.

Архітектура

ПЗ має бути спроектоване за клієнт-сервеною розподіленою архітектурою з чітко відокремленими структурними вузлами системи:

Mockup - візуальна модель інтерфейсу користувача із зазначенням зв'язків і переходів між різними частинами інтерфейсу.

Backend - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.

Web App - сайт або застосунок, доступний з веб-простору, що забезпечує візуальний інтерфейс користувачів усіх категорій

Mobile App (опціонально) - застосунок, що встановлюється на мобільних пристроях (смартфон, планшет). Дана частина виконується за наявності у команді проєкту трьох і більше програмістів.

Усі модулі ПЗ (окрім візуальної моделі Mockup) повинні обов'язково включати до свого складу засоби

тестування функціональності та модульні тести,

розгортання на новому пристрої (інсталяції) із встановленням початкових таблиць у БД та кореневого користувача-адміністратора

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на кшталт:

root

Backend

Web

Mobile

Mockup (вихідні файли або посилання на зовнішній проєкт)

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Backend - .NET або Java, або NodeJS технології з хмарним розміщенням (Azure, AWS, Google Cloud),

Web - серверні технології (ASP, Razor Pages, JSP) або React чи Angular

Mobile - Java, або Kotlin, або Flutter, або React Native

Mockup - визначається ментором з дизайну

Команда проєкту

Для виконання поставлених завдань передбачається робота команди розробників окремо за кожним з модулів проєкту.

Мінімальна команда проєкту - два учасники, у такому складі один учасник виконує модуль Backend та програмну частину Web, другий - модель Mockup та розміткову частину Web.

Рекомендована команда проєкту - три учасники, у такій команді кожен з учасників відповідає за одну з частин: Mockup, Backend, Web.

Дозволяється включення до команди більшої кількості учасників, у такому разі модуль Mobile App набуває обов'язкової реалізації. Також модуль Mockup може бути поділений на моделі Web та Mobile частин.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проєкту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носієві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

Автореферат містить описову частину, у якій розкриваються основні концепти проєкту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

Опис розробки

Розробку проекту було розпочато з дослідження найпопулярніших сервісів таких як «Steam», «Epic Games Launcher», «GOG», тощо. Були виявлені слабкі місця самого «Steam», створена карта емпатії, щоб краще зрозуміти цільову аудиторію (ЦА) проекту, та виявити їх потреби.

Цільова аудиторія проекту

1) Демографічні характеристики
   a) Стать:
      i) Чоловіки становлять переважну більшість цільової аудиторії (89%).
      ii) Жінки складають 10% аудиторії.
      iii) Інші гендерні ідентичності представлені меншою мірою
   b) Віковий склад:
      i) Основна вікова група – 20-29 років, яка становить 31% аудиторії.
      ii) Група 30-39 років також значна, складаючи 27%.

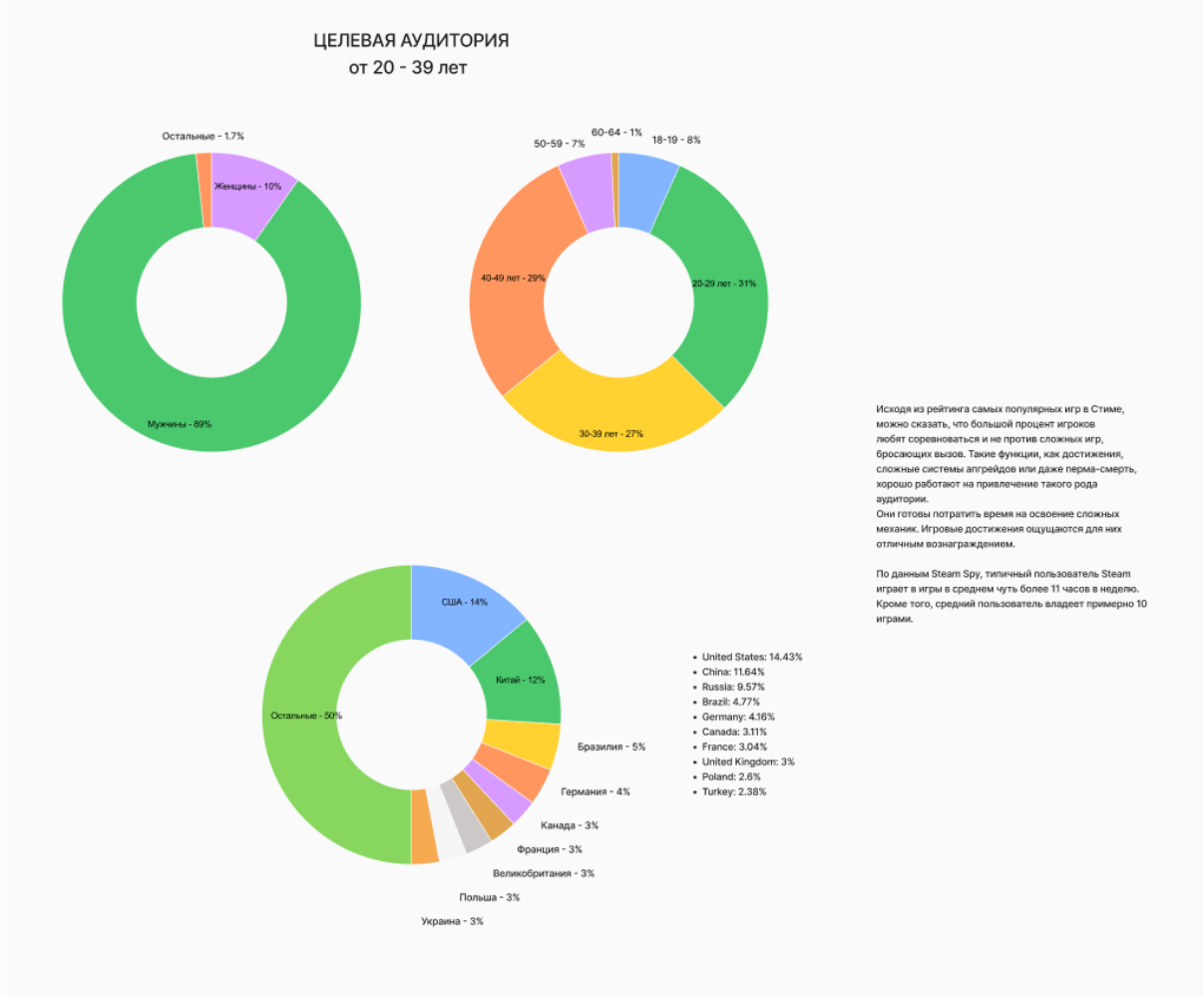
iii) Вікова група 40-49 років становить 29%.
iv) Вікові групи 18-19 років і 50-59 років складають 8% і 7% відповідно.
v) Групи 60-64 років і старші 65 років складають 1% кожна.

c) Географічне розташування:
i) Основна частина аудиторії знаходиться в США (14%).
ii) Китай (12%), Бразилія (5%) та Німеччина (4%) також є значними ринками.
iii) Інші країни, такі як Канада, Франція, Великобританія, Польща, Україна, кожна представляє 3% аудиторії.
iv) Решта світу складає 50% аудиторії.

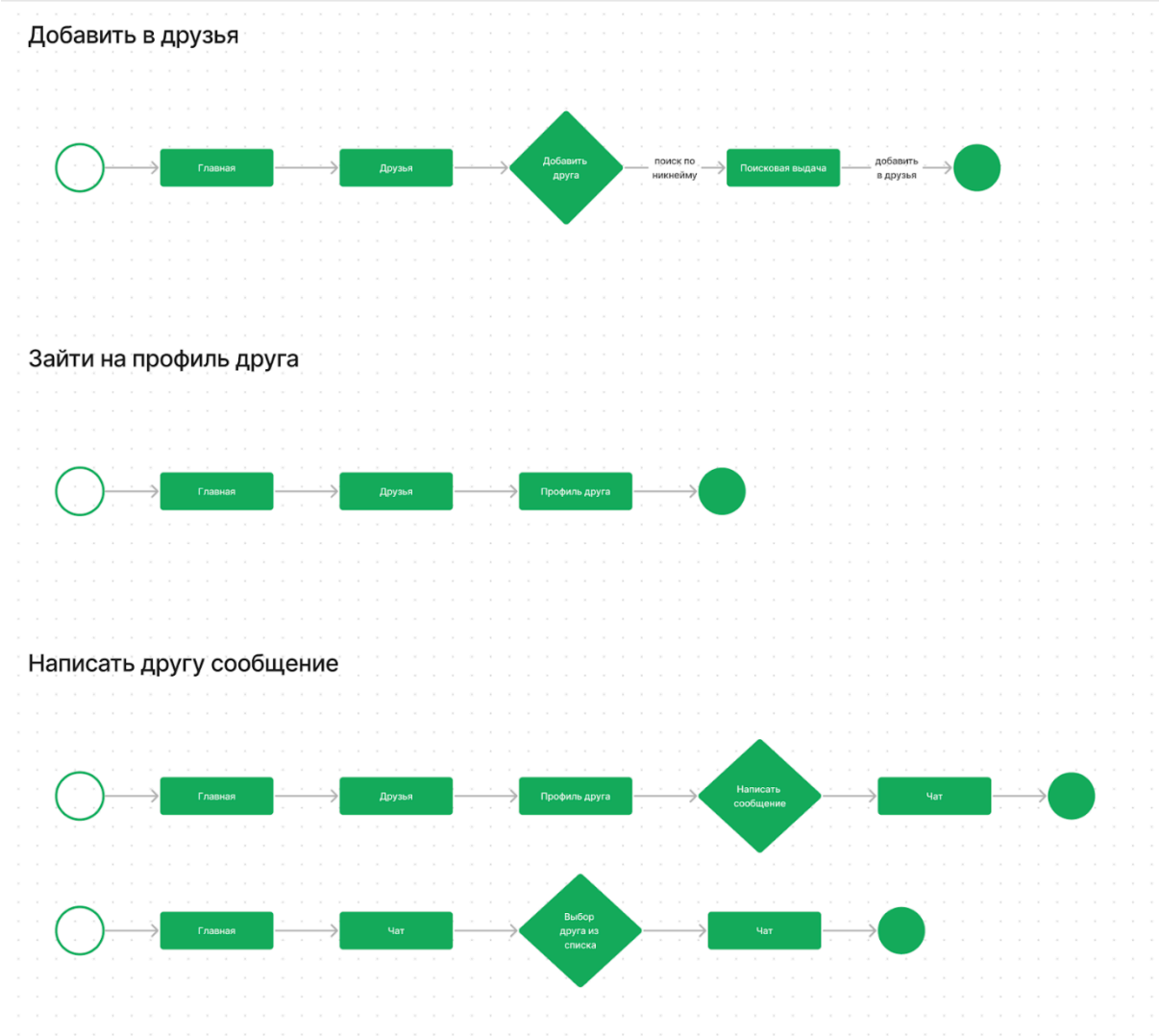
Особливості поведінки

Виходячи з рейтингу найпопулярніших ігор у Steam, можна сказати, що більшість гравців любить складні ігри, що вимагають високого рівня майстерності та стратегічного мислення. Такі функції, як досягнення, складні системи апгрейдів та навіть перманентна смерть, добре працюють для цієї цільової аудиторії. Вони готові витрачати час на освоєння складних механік, ігрові досягнення для них є важливим елементом винагороди.

Згідно з даними Steam Spy, типовий користувач Steam грає в ігри в середньому трохи більше 11 годин на тиждень. Крім того, середній користувач володіє приблизно 10 іграми.

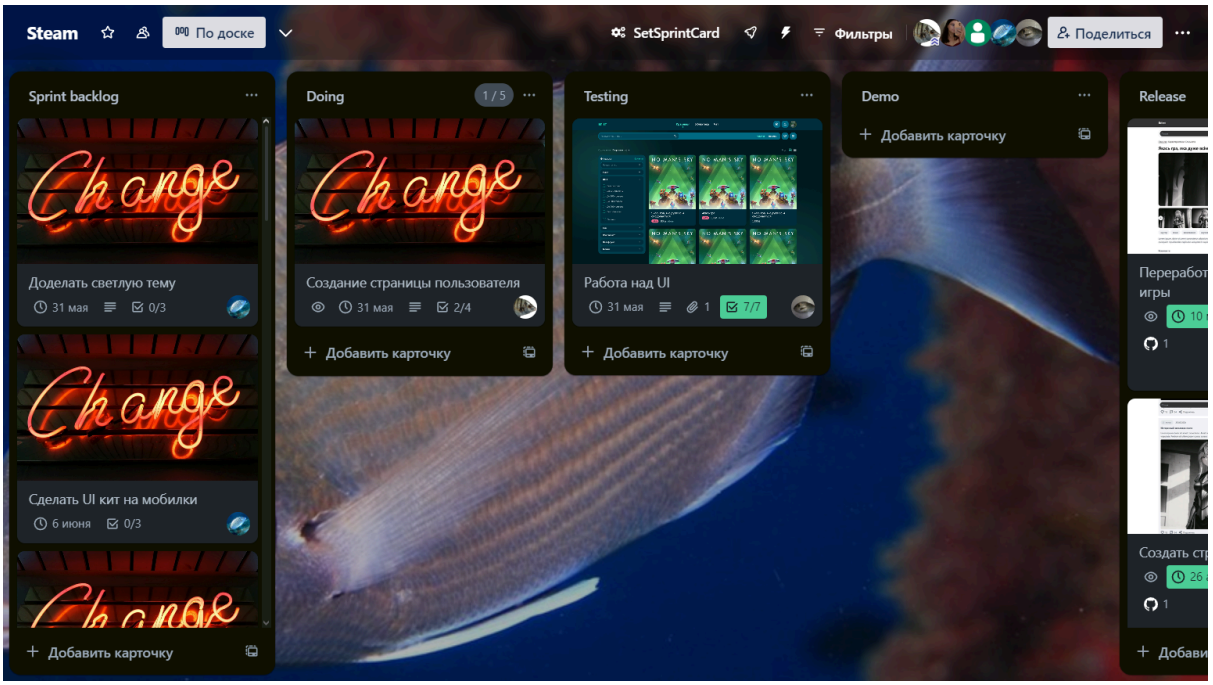

ЦЕЛЕВАЯ АУДИТОРИЯ
от 20 - 39 лет
Остальные - 1,7%
Женщины - 10%
Мужчины - 89%
60-64 - 1%
50-59 - 7%
18-19 - 8%
40-49 лет - 26%
20-29 лет - 31%
30-39 лет - 27%
Исходя из рейтинга самых популярных игр в Стиме, можно сказать, что большой процент игроков любят соревноваться и не против сложных игр, бросающих вызов. Такие функции, как достижения, сложные системы апгрейдов или даже перма-смерть, хорошо работают на привлечение такого рода аудитории.
Они готовы потратить время на освоение сложных механик. Игровые достижения ощущаются для них отличным вознаграждением.
По данным Steam Spy, типичный пользователь Steam играет в игры в среднем чуть более 11 часов в неделю. Кроме того, средний пользователь владеет примерно 10 играми.
США - 14%
Китай - 12%
Остальные - 50%
Бразилия - 5%
Германия - 4%
Канада - 3%
Франция - 3%
Великобритания - 3%
Польша - 3%
Украина - 3%
United States: 14.43%
China: 11.64%
Russia: 9.57%
Brazil: 4.77%
Germany: 4.16%
Canada: 3.11%
France: 3.04%
United Kingdom: 3%
Poland: 2.6%
Turkey: 2.38%

User Flow



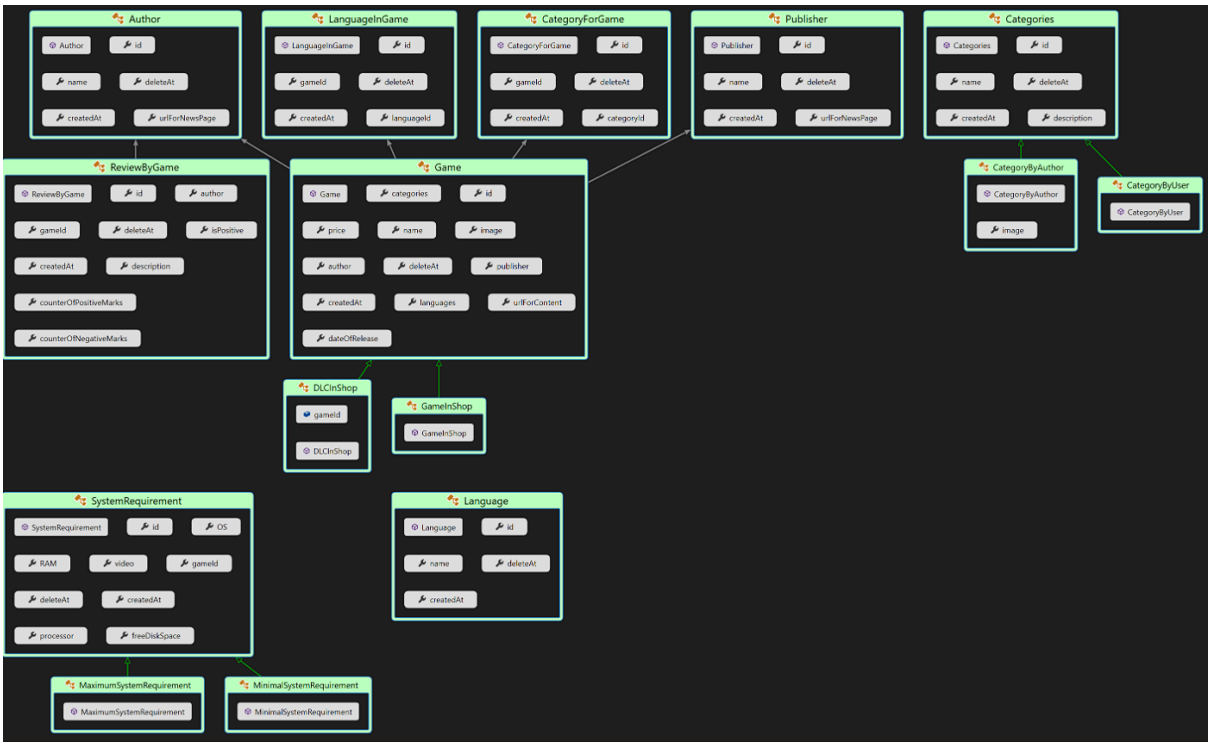
Управління задачами

Розробка вимагає злагодженої командної роботи та чіткої організації процесів. На початковому етапі розробки ми стикнулися зі складнощами в організації робочого процесу між членами команди. Для вирішення цієї проблеми було створено Trello проект, де були розроблені різні правила та кнопки для автоматизації процесу опису виконаної роботи для кожного учасника. Це дозволило уникнути необхідності створювати і описувати завдання з нуля кожного тижня вручну, що значно підвищило ефективність роботи команди.

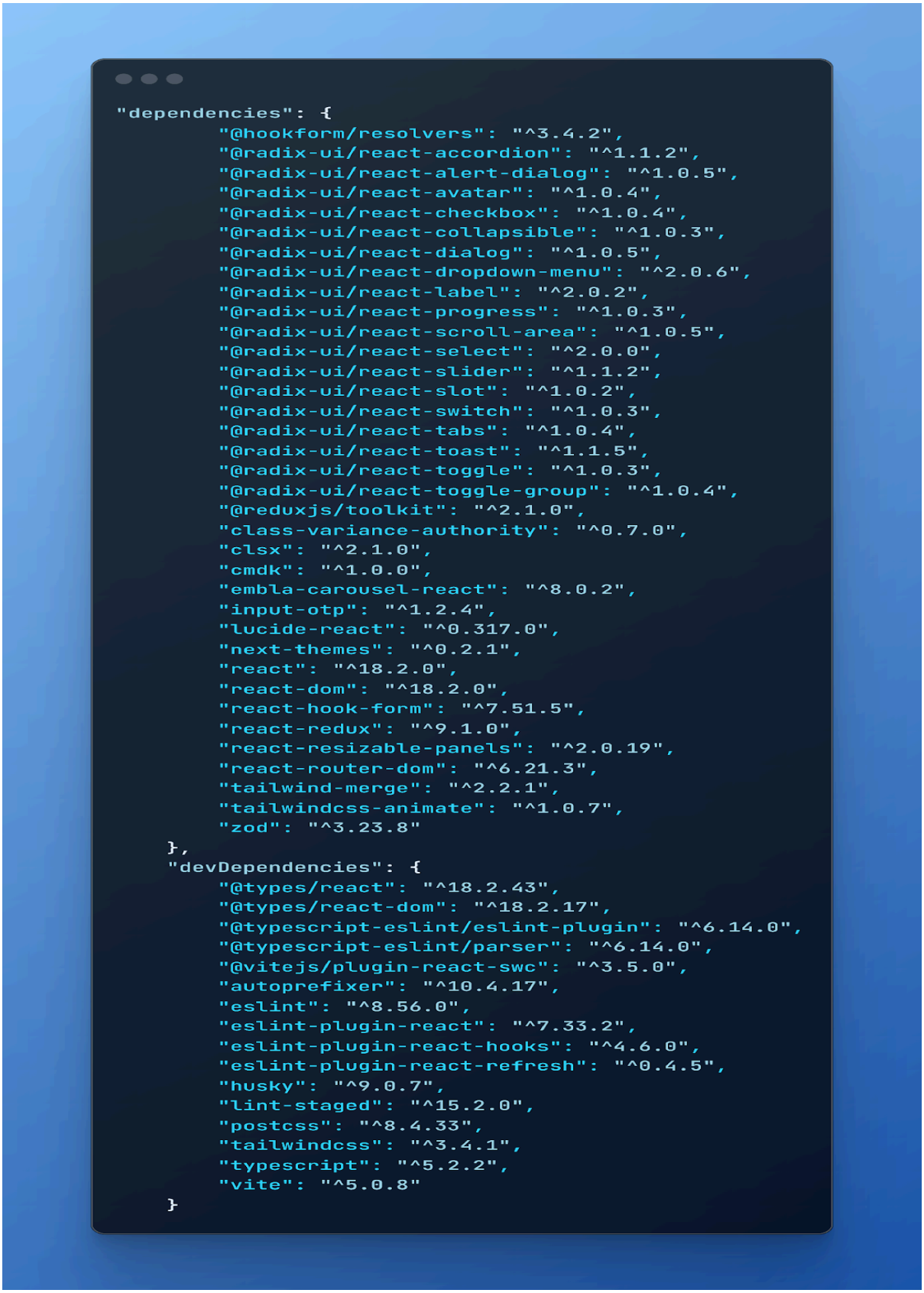


Entity

Поки ще не були готові жодні макети від дизайнерів, було здійснено проектування класів і сутностей на стороні бэкенду. Це дозволило нам визначити основну архітектуру системи та забезпечити стабільну базу для подальшої роботи.



На основі вимог до системи були обрані основні фреймворки та бібліотеки для розробки сайту. Використані залежності включають:



```
"dependencies": {
        "@hookform/resolvers": "^3.4.2",
        "@radix-ui/react-accordion": "^1.1.2",
        "@radix-ui/react-alert-dialog": "^1.0.5",
        "@radix-ui/react-avatar": "^1.0.4",
        "@radix-ui/react-checkbox": "^1.0.4",
        "@radix-ui/react-collapsible": "^1.0.3",
        "@radix-ui/react-dialog": "^1.0.5",
        "@radix-ui/react-dropdown-menu": "^2.0.6",
        "@radix-ui/react-label": "^2.0.2",
        "@radix-ui/react-progress": "^1.0.3",
        "@radix-ui/react-scroll-area": "^1.0.5",
        "@radix-ui/react-select": "^2.0.0",
        "@radix-ui/react-slider": "^1.1.2",
        "@radix-ui/react-slot": "^1.0.2",
        "@radix-ui/react-switch": "^1.0.3",
        "@radix-ui/react-tabs": "^1.0.4",
        "@radix-ui/react-toast": "^1.1.5",
        "@radix-ui/react-toggle": "^1.0.3",
        "@radix-ui/react-toggle-group": "^1.0.4",
        "@reduxjs/toolkit": "^2.1.0",
        "class-variance-authority": "^0.7.0",
        "clsx": "^2.1.0",
        "cmdk": "^1.0.0",
        "embla-carousel-react": "^8.0.2",
        "input-otp": "^1.2.4",
        "lucide-react": "^0.317.0",
        "next-themes": "^0.2.1",
        "react": "^18.2.0",
        "react-dom": "^18.2.0",
        "react-hook-form": "^7.51.5",
        "react-redux": "^9.1.0",
        "react-resizable-panels": "^2.0.19",
        "react-router-dom": "^6.21.3",
        "tailwind-merge": "^2.2.1",
        "tailwindcss-animate": "^1.0.7",
        "zod": "^3.23.8"
    },
    "devDependencies": {
        "@types/react": "^18.2.43",
        "@types/react-dom": "^18.2.17",
        "@typescript-eslint/eslint-plugin": "^6.14.0",
        "@typescript-eslint/parser": "^6.14.0",
        "@vitejs/plugin-react-swc": "^3.5.0",
        "autoprefixer": "^10.4.17",
        "eslint": "^8.56.0",
        "eslint-plugin-react": "^7.33.2",
        "eslint-plugin-react-hooks": "^4.6.0",
        "eslint-plugin-react-refresh": "^0.4.5",
        "husky": "^9.0.7",
        "lint-staged": "^15.2.0",
        "postcss": "^8.4.33",
        "tailwindcss": "^3.4.1",
        "typescript": "^5.2.2",
        "vite": "^5.0.8"
    }
```

Після того як з'явилися макети від дизайнерів, ми розпочали створення React TSX компонентів, які відображали загальне розташування елементів та їх зовнішній вигляд. Ці компоненти були спроектовані таким чином, щоб відповідати отриманим макетам, забезпечуючи точне відтворення візуальних елементів на сторінці. Ми використовували технологію JSX для поєднання HTML та JavaScript, що дозволило нам створити динамічні та інтерактивні інтерфейси.

Однак, коли прийшов час інтегрувати дані з бекенду на фронтенд, виявилося, що наші компоненти не були готові для відображення динамічної кількості інформації. Спочатку ми створювали компоненти зі статичним вмістом, базуючись на передбачених макетах. Це стало проблемою, коли ми почали працювати з реальними даними, які могли змінюватися в кількості та змісті.

Для вирішення цієї проблеми ми майже повністю переробили компоненти, зберігши лише загальну структуру. Було впроваджено використання стану компонентів (State) та властивостей (Props) для передачі даних та їх оновлення.

Після завершення переробки та створення нових компонентів, ми реалізували систему аутентифікації користувачів. Це включало створення реєстраційної та авторизаційної форм. Ми також забезпечили збереження сесій користувачів строком на місяць за допомогою JWT (JSON Web Tokens) та Cookies, що підвищило рівень безпеки та зручності для користувачів.

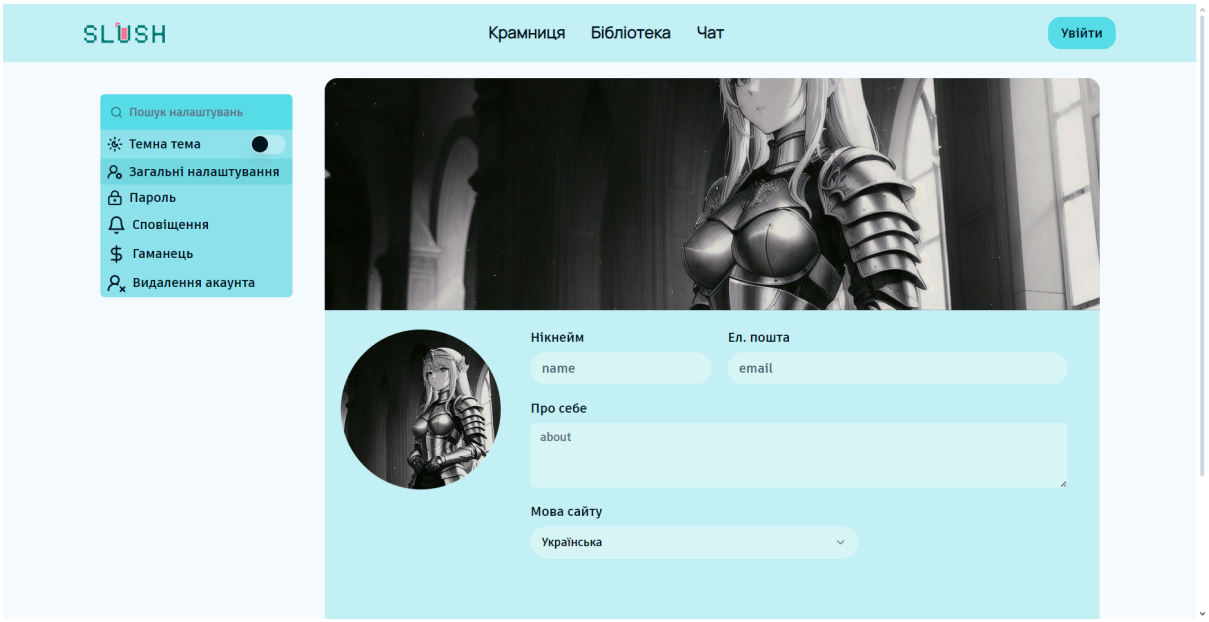
Після тестування системи аутентифікації ми написали fetch запити для всіх компонентів, щоб отримати інформацію, необхідну для їх роботи. Це включало запити до різних кінцевих точок API для отримання даних про продукти, користувачів, коментарі та інший контент. Ми використовували fetch запити для обробки HTTP запитів, що спрощувало процес інтеграції з бекендом.

Також були налаштовані всі компоненти для збереження контенту від користувача, такого як пости, скріншоти тощо. Для цього ми реалізували

функціональність завантаження файлів та обробки форм, що дозволило користувачам легко взаємодіяти з системою і зберігати свій контент.

Після успішного зв'язування фронтенду та бекенду ми реалізували налаштування користувача. Це включало створення сторінки профілю, де користувачі могли оновлювати свої особисті дані, змінювати налаштування безпеки та керувати своїми покупками. Для цього ми використовували React Context API для зберігання і оновлення інформації про користувача в режимі реального часу.

Коли основні функціональні сторінки, такі як головна сторінка магазину, сторінка картки товару, сторінка спільноти (форум), бібліотека куплених користувачем товарів, чат, сторінка користувача та налаштування користувача були реалізовані, ми приступили до внесення фінальних штрихів. Це включало налаштування компонентів усіх основних кольорів, скруглення елементів та інших стилістичних деталей, передбачених UI kit. UI kit був розроблений в останню чергу, тому всі стилі та компоненти з нього були додані на фронтенд в самому кінці розробки. Це забезпечило узгоджений і привабливий зовнішній вигляд усієї системи.



Таким чином, ми створили інтуїтивно зрозумілу та функціональну систему, яка відповідає сучасним стандартам та вимогам користувачів, забезпечуючи надійний інструмент для поширення та актуалізації програмних ресурсів.

Реалізовані можливості проекту:

Реєстрація користувача:

Реалізовано можливість користувача зареєструватися та увійти до системи. Після введення даних, користувач повинен ввести код, який надходить на пошту.

Головна сторінка:

- Відображення всіх ігор різних категорій та жанрів динамічно.
- Перехід на сторінку конкретної гри при натисканні на неї.

Сторінка гри:

- Детальне описання гри.
- Перегляд скріншотів та відео контенту з гри.
- Читання відгуків користувачів.
- Купівля гри.
- Додавання гри в обране.
- Подання скарги на гру.
- Перегляд мінімальних системних вимог.

Вкладка спільноти гри:

- Читання та написання постів про гру.
- Перегляд скріншотів та відео, завантажених іншими користувачами.
- Читання гайдів та новин від розробників.

Бібліотека куплених ігор:

- Перегляд та завантаження куплених ігор.
- Сортування ігор.
- Читання контенту та новин від розробників та спільноти.

Профіль користувача:

- Перегляд профілю конкретного користувача.
- Перегляд контенту, опублікованого користувачем.
- Додавання користувача у друзі.
- Надсилання повідомлень у чаті.

Сторінка налаштувань користувача:

- Налаштування особистої інформації.
- Зміна пароля.
- Поповнення балансу.

Система аутентифікації:

- Реалізація аутентифікації користувачів.
- Збереження сесій користувачів терміном на годину.

Збереження та обробка контенту:

- Збереження контенту від користувачів (пости, скріншоти тощо).
- Реалізація fetch-запитів для отримання інформації для компонентів.

Фінальні налаштування:

- Налаштування основних кольорів, скруглення елементів та інші аспекти дизайну відповідно до UI kit.

Інтеграція фронтенду та бекенду:

- Успішне зв'язування фронтенду та бекенду.
- Реалізація fetch-запитів для всіх компонентів.

Основні функціональні сторінки:

- Головна сторінка магазину.
- Сторінка картки товару.
- Сторінка спільноти.
- Бібліотека куплених товарів.
- Чат.
- Сторінка користувача.
- Налаштування користувача.

Ці елементи забезпечують функціональність та зручність використання платформи для поширення та актуалізації програмних ресурсів за зразком Steam.

Критичний аналіз та висновки

Процес розробки інформаційної системи поширення та актуалізації програмних ресурсів (за зразком Steam) продемонстрував важливість чіткої організації робочого процесу та ефективної командної взаємодії. Використання Trello для автоматизації процесів управління завданнями дозволило команді зосередитися на технічних аспектах проекту, зменшивши адміністративне навантаження.

В ході проекту були розроблені й перероблені численні React TSX компоненти, що забезпечило гнучкість і масштабованість інтерфейсу. Реалізація системи аутентифікації підвищила безпеку користувачів, а використання State дозволило ефективно управляти станом додатку.

Налагодження fetch запитів та обробка даних з бекенду дозволили створити інтерактивний і динамічний інтерфейс. Впровадження користувацьких налаштувань та системи збереження контенту покращило користувацький досвід, що є ключовим аспектом для успіху подібних платформ.

Критичний аналіз

Процес розробки виявив кілька важливих уроків та областей для покращення:

1. **Організація робочого процесу**: Початкові труднощі в організації роботи команди підкреслили необхідність чіткої комунікації та планування. Використання Trello виявилося ефективним рішенням, але впровадження його на більш ранньому етапі могло б запобігти деяким проблемам.
2. **Проектування компонентів**: Первинне проектування React компонентів без урахування динамічної природи даних призвело до значного перевитрати часу на їх переробку. На майбутнє важливо враховувати можливість зміни обсягів даних з самого початку.
3. **Інтеграція з бекендом**: Переходячи до інтеграції з бекендом, стало очевидним, що більш тісна співпраця між фронтенд та бекенд розробниками на початкових етапах могла б уникнути багатьох проблем з передачею даних.
4. **Тестування**: Система аутентифікації потребувала ретельного тестування для забезпечення її безпеки і надійності. Додаткові ресурси, витрачені на тестування, окупилися стабільністю та безпекою кінцевого продукту.
5. **Дизайн та UX**: Остання інтеграція елементів з UI kit показала, що більш раннє залучення дизайнерів у процес могло б зменшити кількість необхідних змін в кінцевій стадії розробки.

Проект продемонстрував високу здатність команди адаптуватися до змін та вирішувати виникаючі проблеми, що є ключовим фактором для успішної реалізації складних інформаційних систем.

Використані ресурси

Література

1. "Learning React: Modern Patterns for Developing React Apps" by Alex Banks and Eve Porcello

- Видавництво: O'Reilly Media

- Опис: Глибоке занурення в основи та сучасні патерни розробки з використанням React.

2. "React – Up & Running: Building Web Applications" by Stoyan Stefanov

- Видавництво: O'Reilly Media

- Опис: Покрокове керівництво для швидкого освоєння React та розробки веб-додатків.

3. "Fullstack React: The Complete Guide to ReactJS and Friends" by Anthony Accomazzo, Nathaniel Murray, and Ari Lerner

- Видавництво: Fullstack.io

- Опис: Комплексний довідник з React, що охоплює всі аспекти розробки повноцінних додатків.

4. "TypeScript Quickly" by Yakov Fain and Anton Moiseev

- Видавництво: Manning Publications

- Опис: Детальний посібник з вивчення TypeScript, включаючи найкращі практики та реальні приклади.

5. "Pro TypeScript: Application-Scale JavaScript Development" by Steve Fenton

- Видавництво: Apress

- Опис: Глибоке дослідження TypeScript, його можливостей та використання у великих проєктах.

6. "JavaScript: The Good Parts" by Douglas Crockford

- Видавництво: O'Reilly Media

- Опис: Класичний посібник з JavaScript, що охоплює його найкращі частини та найефективніші техніки програмування.

7. "You Don't Know JS: ES6 & Beyond" by Kyle Simpson

- Видавництво: O'Reilly Media

- Опис: Глибоке вивчення нових можливостей JavaScript (ES6 і далі).

8. "Redux in Action" by Marc Garreau and Will Faurot

- Видавництво: Manning Publications

- Опис: Посібник з використання Redux у реальних проєктах, з практичними прикладами та порадами.

9. "The Road to React: Your Journey to Master Plain Yet Pragmatic React.js" by Robin Wieruch

- Видавництво: Independently Published

- Опис: Практичний підхід до вивчення React з прикладами та порадами для швидкого освоєння фреймворка.

10. "Refactoring UI" by Adam Wathan and Steve Schoger

- Видавництво: Independently Published

- Опис: Практичний посібник з покращення дизайну інтерфейсів користувача з використанням сучасних підходів та інструментів.

11. "Tailwind CSS: From Zero to Production" by Simon Knott

- Видавництво: Independently Published

- Опис: Вичерпний посібник з вивчення та використання Tailwind CSS у веб-розробці.

12. "Design Systems: A Practical Guide to Creating Design Languages for Digital Products" by Alla Kholmatova

- Видавництво: Smashing Magazine

- Опис: Керівництво з розробки дизайн-систем для цифрових продуктів, включаючи практичні приклади та рекомендації.

Документація

1. React Documentation

- URL: https://reactjs.org/docs/getting-started.html

2. TypeScript Handbook

- URL: https://www.typescriptlang.org/docs/handbook/intro.html

3. Redux Toolkit Documentation

- URL: https://redux-toolkit.js.org/introduction/getting-started

4. Next.js Documentation

- URL: https://nextjs.org/docs

5. Tailwind CSS Documentation
   - URL: https://tailwindcss.com/docs

6. Radix UI Documentation
   - URL: https://www.radix-ui.com/docs/primitives/overview/introduction

7. React Router Documentation
   - URL: https://reactrouter.com/en/main

8. Vite Documentation
   - URL: https://vitejs.dev/guide/

9. ESLint Documentation
   - URL: https://eslint.org/docs/latest/

10. Prettier Documentation
   - URL: https://prettier.io/docs/en/index.html

11. PostCSS Documentation
   - URL: https://postcss.org/documentation/

12. Husky Documentation
   - URL: https://typicode.github.io/husky/#/

Сайти

1. chatgpt.com
2. gemini.google.com
3. codeium.com
4. copilot.microsoft.com
5. v0.dev
6. ui.shadcn.com
7. developer.mozilla.org
8. nerdcave.com
9. cssgrid-generator.netlify.app
10. lucide.dev

Програмне забезпечення

1. VS Code