Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и облачные технологии**

Направление подготовки **09.03.03 Прикладная информатика**

К У Р С О В О Й П Р О Е К Т

Тема: Разработка прототипа веб-приложения для поиска напарников для видеоигр

Обучающийся: Марченко Вадим Александрович, К3141

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время многие геймеры предпочитают играть не в одиночку, а вместе, находя единомышленников в разных уголках сети. Однако обычные способы поиска напарников – форумы, социальные сети или случайные лобби – не всегда удобны: там сложно понять, насколько человек подходит по уровню навыков, интересам и времени, когда он играет. Именно поэтому возникла идея создать веб-приложение, где любой игрок сможет быстро найти подходящего партнёра для совместных игровых сессий.

В рамках этой курсовой работы был спроектирован и разработан прототип сервиса, позволяющего пользователям регистрироваться, указывать свои игровые предпочтения и находить людей, с которыми будет интересно и комфортно играть. Для этого предусмотрены фильтры (по игре, времени активности, навыкам и другим параметрам), а также базовые инструменты взаимодействия (приглашения, система уведомлений). Особое внимание уделено тому, чтобы приложение было простым в использовании и предоставляло необходимую гибкость в поиске.

Для достижения поставленной цели необходимо определить круг задач, связанных с проектированием интерфейса, созданием архитектуры базы данных, интеграцией фронтенда и бекенда, а также тестированием всех ключевых сценариев пользовательского взаимодействия. Данный проект выполнялся в формате командной работы, где роли распределялись между дизайнером, фронтенд-разработчиками, бекенд-разработчиками и руководителем проекта. В процессе разработки мы исследовали опыт существующих платформ, анализировали сильные и слабые стороны аналогичных решений и стремились найти баланс между функциональностью, простотой и удобством использования.

Основная цель проекта – создать работоспособную основу, которую можно развивать и совершенствовать, добавляя новые функции вроде

рейтинга игроков, отзывов и более тонких фильтров. В отчёте описаны ключевые этапы проекта: анализ существующих решений, проектирование интерфейса, реализация функционала и тестирование прототипа в реальных условиях.

1 Описание проекта

Разрабатываемый проект представляет собой прототип веб-приложения, призванного облегчить процесс поиска напарников для совместной игры в различные онлайн-тайтлы. Идея данного приложения основывается на наблюдении, что многие игроки испытывают трудности с подбором команды или партнёров, с которыми было бы комфортно и интересно играть. Цель приложения – предоставить инструментарий, позволяющий максимально точно отбирать подходящих кандидатов по набору заранее определённых критериев.

По своей структуре проект включает несколько ключевых модулей. В первую очередь это система пользовательских профилей, где каждый игрок может указать информацию о себе, свои предпочтения и пожелания к совместной игре. Второй важный модуль – система поиска и фильтрации, позволяющая задавать широкий набор параметров (игра, жанр, уровень навыков, время активности, язык общения, регион, платформа и так далее). Также реализуется функциональность по отправке и приёму заявок, позволяющая инициировать контакт между игроками. Ещё одним не менее важным элементом является система рейтингов и отзывов, помогающая формировать репутацию пользователей, чтобы упрощать выбор надёжных и доброжелательных партнёров.

Архитектурно проект состоит из клиентской части (фронтенд), созданной на базе библиотеки React [1], и серверной части (Rails и Rust [2][3]). Rails выступает в роли основного фреймворка, обеспечивающего REST API, взаимодействие с базой данных, аутентификацию и авторизацию. Rust используется для высокопроизводительных микросервисов или отдельных задач, которые предполагают повышенные требования к быстродействию. Подобная комбинация технологий была выбрана с учётом интересов команды и стремления обеспечить достаточную гибкость и масштабируемость на будущее.

2 Цель и задачи

Главная цель данной курсовой работы заключается в создании действующего прототипа веб-приложения, который позволит геймерам находить напарников для совместной игры в режиме онлайн. Для достижения этой цели были определены следующие задачи:

1 Провести анализ аналогичных решений, уже существующих на рынке, включая специализированные платформы, боты в мессенджерах и тематические сообщества. Определить их преимущества и недостатки, а также выяснить, какие функции наиболее востребованы у самих пользователей.

2 Разработать концепцию пользовательского интерфейса и навигации, согласовав её с командой и учитывая современные принципы UX/UI. На этом этапе нужно было подготовить макет в Figma и продумать, как будущие экраны взаимодействуют между собой.

3 Спроектировать структуру базы данных, чтобы хранить профили пользователей, их игровые предпочтения, настройки приватности, список заявок, а также информацию о репутации и отзывах. Здесь в качестве ориентира выступала классическая архитектура, поддерживающая реляционную модель данных с дополнительными возможностями Rails и Rust.

4 Создать фронтенд на React, придерживаясь макета и UX-решений, предложенных дизайнером. Грамотно реализовать маршрутизацию, логику форм регистрации и авторизации, а также гибкие механизмы поиска по фильтрам.

5 Реализовать на бекенде (Rails + Rust) программное обеспечение, обрабатывающее запросы от клиента, поддерживающее безопасную авторизацию и отвечающее за работу с базой данных. Предусмотреть возможность расширения функционала и высокой нагрузки в перспективе.

6 Организовать тестирование и отладку, выявить и устранить ошибки в пользовательских сценариях. Особое внимание уделить корректной передаче данных между клиентом и сервером, а также вопросу безопасности (защита от некорректных запросов, несанкционированного доступа, утечек приватных данных).

7 Подготовить и развернуть готовый прототип на общедоступном хостинге (Vercel), чтобы все члены команды, руководитель и потенциальные пользователи могли протестировать работу сервиса и дать обратную связь.

Выполнение этих семи задач было распределено по этапам и осуществлялось несколькими участниками команды, в соответствии с их профессиональным профилем. Таким образом, проект можно считать успешным, если к моменту завершения курсовой работы будет существовать доступный онлайн-прототип, реализующий ключевые пользовательские сценарии и позволяющий тестировать его функциональность.

3 Процесс разработки

Процесс разработки включал в себя несколько последовательных этапов. В самом начале, ещё до написания кода, мы приступили к анализу рынка и сбору требований. Нужно было понять, каковы основные болевые точки игроков, пользующихся уже имеющимися сервисами для поиска напарников. Для этого изучались профильные порталы, каналы на YouTube, отзывы в социальных сетях и тематических группах. Выяснилось, что большинство существующих платформ либо чрезвычайно узкоспециализированы (например, заточены строго под одну игру), либо представляют собой обобщённые чаты, в которых отсутствуют удобные механизмы сортировки и фильтрации [4]. Также нередко встречались жалобы на токсичность некоторых пользователей, поэтому продуманная система репутации и отзывов стала одним из приоритетов.

Затем был разработан макет пользовательского интерфейса. Дизайнер в команде создал визуальные наброски в Figma, включающие главную страницу, форму регистрации и авторизации, страницу профиля игрока, раздел поиска, а также интерфейс для отправки и управления заявками. Макет был тщательно согласован со всей командой, чтобы никто не оставался в неведении относительно того, как должна выглядеть и функционировать каждая страница. Этот макет служил своеобразным планом для фронтенд-разработчиков и ориентиром при вёрстке.

На стадии проектирования технической архитектуры мы приняли решение использовать Rails в качестве основного фреймворка для бекенда, так как он достаточно хорошо подходит для быстрой разработки веб-приложений и имеет встроенную систему маршрутизации и работы с базами данных. Rust был добавлен для микросервисов, отвечающих за сложные операции или те компоненты, где критична производительность и безопасность.

Когда общие детали были согласованы, команда перешла к написанию кода. Я, как фронтенд-разработчик, установил окружение React, настроил базовую структуру проекта и подключил необходимые библиотеки (Axios для HTTP-запросов [5], React Router для маршрутизации и другие вспомогательные пакеты). Одновременно бекенд-разработчики настраивали Rails-приложение, создавали модели для таблиц профилей, запросов, заявок, тестировали взаимодействие с базой данных и прописывали маршруты в соответствии с REST-архитектурой.

Параллельно шёл процесс сборки и отладки интерфейса. Дизайн из Figma переносился в адаптивную вёрстку, а каждый крупный экран разбивался на небольшие React-компоненты, чтобы обеспечить модульную структуру кода и упростить сопровождение. Форма регистрации, форма авторизации, страница поиска, страница профиля и другие части приложения тестировались в локальной среде. Регулярные митапы внутри команды позволяли оперативно обсуждать возникающие проблемы и решать, кто и как их будет устранять.

После того как мы получили минимально работоспособную версию приложения, начался более целенаправленный процесс тестирования. Тестировались базовые пользовательские сценарии: регистрация, вход в систему, заполнение профиля, сохранение изменений, смена пароля, поиск других игроков по различным фильтрам, отправка и получение заявок. Особое внимание уделялось корректности отображения данных на фронтенде и надлежащей обработке ошибок (например, если пользователь не указал обязательные поля или передал некорректную информацию).

Завершающим шагом разработки стало развёртывание прототипа. Мы выбрали хостинг Vercel для клиентской части, так как он существенно упрощает выкладку React-приложений. Была настроена автоматическая сборка, при которой после каждого пуша в основную ветку репозитория происходило пересобирание фронтенда и обновлялся сайт на тестовом

домене В результате мы смогли предоставить рабочую версию прототипа, доступную по ссылке для всех желающих.

4 Моя роль в проекте

Моя роль в проекте заключалась в том, чтобы, опираясь на дизайн-макеты и общее техническое задание, создать полноценную фронтенд-часть приложения. Это означало, что я отвечал за разработку интерфейса, взаимодействие компонентов, маршрутизацию между страницами, а также интеграцию с бекендом через HTTP-запросы.

Во-первых, я осуществил базовую настройку React-приложения, инициализировал необходимые зависимости и создал каркас проекта, включающий в себя структуру директорий, скрипты сборки и конфигурационные файлы. Затем я разбил макеты Figma на отдельные модули и начал поэтапно их реализовывать. Например, отдельные страницы (главная, регистрация, авторизация, профиль, поиск, список заявок) оформлялись как «pages», а повторно используемые элементы (шапка сайта, форма входа, карточка профиля, фильтр поиска) выносились в «components».

Во-вторых, значительное внимание уделялось управлению состоянием в React. Поскольку в приложении предусмотрено хранение информации о пользователе (например, авторизован он или нет, какие параметры фильтрации он выбрал, какие заявки отправил), необходимо было использовать либо React Context, либо отдельную библиотеку для стейт-менеджмента. Мы выбрали упрощённый подход на основе Redux + Redux Toolkit. Я также реализовал механизм передачи токена аутентификации в заголовках запросов Axios, чтобы Rails мог корректно идентифицировать пользователя.

В-третьих, я занимался адаптивной вёрсткой. Дизайн изначально был спроектирован как responsive, поэтому нужно было проверить, корректно ли всё отображается при различных разрешениях экрана. Это включало медиазапросы в CSS, тестирование на мобильных устройствах и планшетах,

а также проверку, не ломается ли макет при слишком большом или слишком маленьком разрешении.

Наконец, я тесно взаимодействовал с командой бекенда, чтобы сформировать единый API-контракт. Когда на стороне Rails добавлялся новый эндпоинт (например, для получения списка подходящих напарников на основе фильтров), я оперативно вносил нужные изменения в Axios-запросы и обновлял соответствующие компоненты React. При возникновении проблем с CORS или форматом сериализации совместно искали решения, позволяющие продолжать интеграцию.

5 Анализ проделанной работы

В ходе реализации проекта я приобрёл значительный опыт, связанный с построением и интеграцией многоуровневых веб-приложений. Тем не менее, ряд моментов вызвал ощутимые сложности и потребовал дополнительных усилий.

Прежде всего, что настройка окружения и совместная работа React, Rails и Rust оказалась трудоёмкой. Хотя Rails отлично справляется со своей задачей в качестве фреймворка для быстрого прототипирования, при взаимодействии с Rust приходилось учитывать особенности сборки и совместимости разных версий библиотек. Rust использовался для отдельных узкоспециализированных модулей, и настройка их бесшовной работы в составе Rails-приложения требовала более детальной проработки, чем если бы всё писалось только на Rails.

При всём этом, проект принёс много положительных результатов. Мы смогли практически применять знания о REST-архитектуре, о разделении ответственности между клиентом и сервером, о безопасной передаче данных и защите от несанкционированного доступа. Я освоил более глубоко механизмы роутинга в React, научился оперативнее верстать интерфейсы по макетам Figma, а также улучшил понимание того, как организовывать работу с формами и валидацией пользовательских данных.

В целом, можно сказать, что основные цели были достигнуты, а возникавшие трудности стали дополнительным стимулом для изучения смежных областей и поиска оптимальных решений.

6 Взаимодействие с командой и руководителем

Данный проект реализовывался в формате командной работы, и эффективность взаимодействия между участниками напрямую влияла на качество и скорость разработки. Наш состав включал в себя дизайнера, двух фронтенд-разработчиков, бекенд-разработчика (Rails), специалиста, занимающегося Rust-модулями, а также руководителя проекта.

Общение происходило через мессенджер и регулярные видеозвонки. Мы проводили короткие звонки, на которых каждый рассказывал о прогрессе, возникающих проблемах и потребностях. В конце недели организовывали демо, где показывали, что уже сделано, и планировали ближайшие задачи. В качестве инструмента для постановки и отслеживания задач применялась доска Odoo, где создавались карточки по каждому важному функциональному блоку: «регистрация», «авторизация», «страница поиска», «система уведомлений», «профиль пользователя» и так далее.

Моё взаимодействие с дизайном происходило напрямую. Дизайнер предоставлял макеты в Figma, я брал из них стили, шрифты, цвета, расположение элементов и стремился в точности воспроизвести вид, задуманный на макете. На рисунке 1 изображён интерфейс Figma с дизайном проекта.

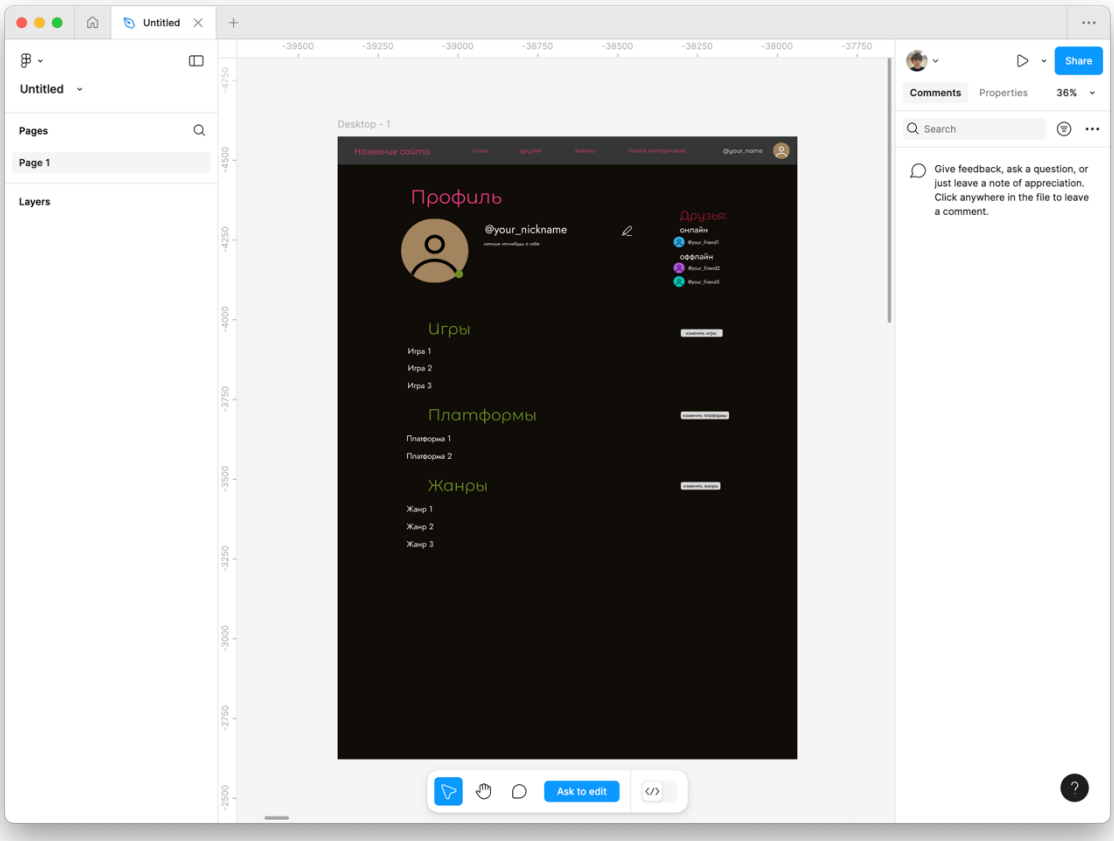

Рисунок 1 – Интерфейс Figma с дизайном веб-приложения

Также я активно общался с бекенд-разработчиком, который создавал эндпоинты на Rails. Мы согласовывали формат запроса и ответа, структуру JSON, имена полей и возможные коды ошибок. При необходимости я сообщал, какие именно данные требуются фронтенду для корректного отображения. Если со стороны Rails находились ошибки или баги, я отправлял скриншоты, чтобы вместе искать решение.

Руководитель проекта вносил существенный вклад в организацию взаимодействия: ставил приоритеты, помогал решать спорные технические вопросы, контролировал сроки выполнения этапов, а также давал регулярную обратную связь. Он оценивал дизайн и функциональность, проверял, насколько наше решение соответствует поставленной цели. По итогам каждого спринта (недели или двух) мы получали список замечаний и пожеланий, которые вносили в бэклог для будущих доработок.

ЗАКЛЮЧЕНИЕ

Подводя итоги курсового проекта «Разработка прототипа веб-приложения для поиска напарников для видеоигр», можно сделать вывод, что поставленные цели в целом были достигнуты. Удалось создать и протестировать приложение, отвечающее базовым потребностям современных игроков, желающих найти себе партнёров для сетевых сессий. Проект продемонстрировал работоспособность идеи, согласно которой возможно объединить в одном сервисе гибкие механизмы поиска, репутационную систему и удобный интерфейс, позволяющий экономить время и повышать качество совместного игрового опыта.

В ходе работы был проведён анализ аналогичных решений, что позволило выявить ключевые факторы, влияющие на востребованность такого сервиса. К ним относятся удобство, простота использования, наличие продвинутых фильтров и адекватная система рейтингов и отзывов. Разработка велась поэтапно: от анализа и дизайна до тестирования и развёртывания прототипа. Технологический стек включил в себя React на стороне фронтенда, Rails и Rust на стороне сервера, что дало определённую гибкость и перспективы на расширение функционала в будущем.

Были сформулированы и решены задачи по проектированию базы данных, настройке клиент-серверного взаимодействия, организации системы аутентификации и уведомлений. Взаимодействие с командой и руководителем сыграло важную роль в том, что проект постоянно получал обратную связь и корректировки. Несмотря на возникавшие трудности с совместимостью, нехваткой времени и некоторой сложностью интеграции нескольких языков и фреймворков, все ключевые модули были реализованы, а основная функциональность – поисковая система, личные профили и заявки – заработала стабильно.

Мой личный вклад заключался прежде всего в создании фронтенд-части: настройке React, вёрстке интерфейса на базе макетов Figma, интеграции Axios и настройке маршрутизации, а также тесном взаимодействии с бекенд-разработчиками для согласования API. В результате получилось клиентское приложение, способное обмениваться данными с сервером, корректно обрабатывать ответы и наглядно отображать информацию о профилях и заявках.

Проект на текущем этапе можно считать базовым прототипом, который уже сейчас решает основную задачу – упрощает поиск напарников и предлагает структуру для дальнейшего развития рейтинговых и репутационных механизмов. При желании сервис можно дополнить новыми функциями, улучшить дизайн, увеличить производительность и стабильность. В перспективе есть все предпосылки для того, чтобы превратить его в полноценное коммерческое решение или открытый продукт для игрового сообщества.

Таким образом, итог курсовой работы можно сформулировать следующим образом: цель достигнута, прототип готов к демонстрации и дальнейшему совершенствованию. Это позволяет считать проект состоявшимся, а полученные знания и навыки – ценным опытом для будущей профессиональной деятельности в сфере веб-разработки и кооперативных игровых сервисов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. React – A JavaScript library for building user interfaces: [Электронный ресурс]. URL: https://reactjs.org/.
2. Ruby on Rails – Официальная документация: [Электронный ресурс]. URL: https://rubyonrails.org/.
3. Rust – Документация по языку программирования: [Электронный ресурс]. URL: https://www.rust-lang.org/.
4. Traversy Media — Обучающие видеоролики по веб-разработке: [Электронный ресурс].
   URL: https://www.youtube.com/c/TraversyMedia.
5. Официальная документация Axios: [Электронный ресурс].
   URL: https://axios-http.com/docs/intro.

ПРИЛОЖЕНИЕ

Техническое задание на “Разработка прототипа веб-приложения для поиска напарников для видеоигр”

1 Общие положения

1.1 Название проекта: «Разработка прототипа веб-приложения для поиска напарников для видеоигр».

1.2 Цель (назначение): создать работающий прототип веб-приложения, который позволит пользователям находить напарников для совместных игр, используя удобные фильтры и систему репутации.

1.3 Сроки выполнения:

– Начало: 01.11.2024

– Окончание: 20.12.2024

1.4 Команда проекта:

– Исполнитель проекта (руководитель проекта): Еникеев Ринат Алиевич

– Дизайн и UX: Пластинина Мария Вячеславовна

– Backend-разработчики: Эль Хеннави Александр, Янченко Денис Сергеевич

– Frontend-разработчики: Марченко Вадим Александрович, Аксенова Алина Рустамовна

1.5 Основные этапы задач:

– Сбор требований и анализ

– Проектирование интерфейса и дизайн

– Проектирование архитектуры и настройка среды разработки

– Разработка основных модулей приложения

– Тестирование и отладка

2 Технические требования

2.1 Функциональные требования

– Создание и управление профилем: возможность регистрации нового пользователя, редактирования личных данных, изменения списка интересов и просмотра собственного статуса в системе.
– Система поиска и фильтрации: поиск напарников по играм, жанрам, навыкам, времени доступности, языку общения и другим параметрам; выдача результатов с учётом выбранных фильтров.
– Механизм отправки запросов на совместную игру: пользователи должны иметь возможность отправлять приглашения, принимать или отклонять запросы, а также видеть их статус (ожидание, принято, отклонено).
– Уведомления о запросах и приглашениях: система должна уведомлять пользователей в режиме реального времени (или при следующем входе) о поступивших заявках, сообщениях и изменениях в их профилях.
– Регистрация и авторизация пользователей: доступ к основному функционалу только после авторизации; поддержка базовых мер безопасности (хеширование паролей, защита от SQL-инъекций).

2.2 Нефункциональные требования

– Удобство использования (UI/UX): интерфейс должен быть простым, адаптивным и не перегруженным элементами; основные функции должны быть доступны без дополнительных сложных переходов.
– Лёгкость внесения обновлений: кодовая база должна поддерживать совместную разработку и интеграцию новых модулей без риска сломать существующую функциональность.
– Производительность: система должна корректно обрабатывать запросы пользователей и предоставлять результаты поиска в

разумные сроки (при проектной нагрузке не менее 1000 пользователей одновременно).

– Используемый стек технологий:

Язык разработки бекенда: Ruby (часть на Rails), высоконагруженные сервисы — Rust по необходимости.

Язык разработки фронтенда: JavaScript (React).

СУБД: PostgreSQL (хранение профилей, заявок, статистики).

– Совместимость: корректная работа в современных браузерах (Chrome, Firefox, Safari, Edge).

3 Ожидаемый результат

3.1 MVP системы, удовлетворящее описанным выше функциональным и нефункциональным требованиям, развёрнутое на сервере и доступное для тестирования конечными пользователями и членами команды.

3.2 Гибкая архитектура, позволяющая в дальнейшем безболезненно масштабировать проект, добавлять новые фильтры, модули геймификации (уровни, награды) и дополнительную аналитику (навыки игроков, эффективность команд).

3.3 Продуманный интерфейс, ориентированный на интуитивную навигацию и высокую скорость выполнения основных операций (создание профиля, фильтрация, поиск, отправка заявок).

4 *Конечная цель*: предоставить игрокам удобное средство для нахождения и приглашения напарников в различных сетевых видеоиграх, улучшая их совместный опыт и ускоряя организацию командных сессий.