МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ С РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМОЙ ФИЛЬМОВ НА ОСНОВЕ АЛГОРИТМОВ KNN И SVD И НЕЙРОСЕТЕВОГО ПОИСКА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Егорова Арсения Алексеевича

Научный руководитель
доцент, к. ф.-м. н. _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

Актуальность темы. В условиях стремительного роста объемов цифрового контента пользователи сталкиваются с проблемой выбора фильмов, соответствующих их вкусам и предпочтениям. Современные стриминговые платформы предлагают тысячи наименований, однако без эффективной системы персонализированных рекомендаций процесс поиска становится трудоемким и неэффективным. В этом контексте особую значимость приобретают рекомендательные системы, способные анализировать поведение и эмоциональное состояние пользователей, а также предлагать релевантные фильмы с использованием современных технологий, таких как большие языковые модели (LLM) и OpenAI API.

Несмотря на распространенность рекомендательных алгоритмов в таких сервисах, как Netflix или Kinopoisk, существующие решения имеют ряд недостатков. Многие системы полагаются исключительно на популярность контента, игнорируя индивидуальные предпочтения. Другие, использующие коллаборативную фильтрацию, страдают от проблемы «холодного старта» — невозможности давать точные рекомендации для новых пользователей или малоизвестных фильмов. Кроме того, редкие платформы учитывают динамику изменения вкусов или эмоциональный контекст, что снижает качество рекомендаций. Интеграция с OpenAI API позволяет преодолеть эти ограничения, обеспечивая персонализированные рекомендации на основе оценок пользователя и текстовых ответов, отражающих его настроение.

Цель бакалаврской работы - разработать мобильное приложение с рекомендательной системой для подбора фильмов на основе предпочтений и эмоционального состояния пользователя с использованием OpenAI API.

Поставленная цель определила следующие задачи:

1. Изучить принципы работы рекомендательных систем (коллаборативная фильтрация, контент-базированные методы, использование LLM).
2. Проанализировать существующие решения для подбора фильмов и их ограничения.
3. Выбрать технологии для реализации мобильного приложения (React Native, Expo) и бэкенда (FastAPI, PostgreSQL, OpenAI API).
4. Разработать мобильное приложение.
5. Разработать и протестировать алгоритм рекомендаций, интегрированный

в приложение, включая нейро-поиск на основе OpenAI API.

Практическая значимость бакалаврской работы. Практическая значимость выражается в создании мобильного приложения, способного подбирать фильмы с учётом индивидуальных вкусов и эмоционального состояния, что расширяет подходы к взаимодействию пользователя с цифровыми сервисами.

Структура и объём работы. Бакалаврская работа состоит из введения, 3 разделов, заключения, списка использованных источников и 5 приложений. Общий объем работы – 67 страницы, из них 49 страниц – основное содержание, включая 8 рисунка и 2 таблицы, CD диск в качестве приложения, список использованных источников информации – 20 наименование.

Первый раздел «Теоретическое описание рекомендательных систем» посвящен анализу принципов работы современных рекомендательных систем, их классификации и методам реализации. В разделе рассматриваются ключевые алгоритмы, используемые для персонализации контента, а также их преимущества и ограничения.

Рекомендательные системы (РС) представляют собой программно- алгоритмические комплексы, предназначенные для прогнозирования предпочтений пользователей на основе анализа их поведения и исторических данных. Основная задача РС — решение проблемы информационной перегрузки путем фильтрации контента и предоставления персонализированных предложений. В работе выделены три основных подхода к построению РС:

1. **Неперсонализированные рекомендации** — основаны на общей популярности контента. Используют статистические методы (сглаженное среднее, доверительные интервалы) для минимизации проблемы "холодного старта"новых объектов.

2. **Content-Based фильтрация** — анализирует характеристики объектов (например, жанры фильмов) и сопоставляет их с профилем пользователя. Ключевым методом является TF-IDF (Term Frequency-Inverse Document Frequency) для оценки значимости признаков:

$$w_{x,y} = \mathrm{tf}_{x,y} \times \log \frac{N}{\mathrm{df}_x}$$

3. **Коллаборативная фильтрация** — использует сходство между пользователями или объектами. В работе детально рассмотрены:

— Алгоритм K ближайших соседей (KNN) с косинусной мерой сходства

— Матричная факторизация (SVD) для выявления латентных факторов

Особое внимание уделено сравнительному анализу алгоритмов. Модель SVD продемонстрировала лучшую точность (RMSE 0.77 против 0.95 у KNN) благодаря способности выявлять скрытые зависимости в данных. Однако KNN сохраняет значение для интерпретируемости рекомендаций.

В разделе также проанализированы практические аспекты:

— Проблема "информационного пузыря"и методы ее минимизации
— Оптимизация вычислительной сложности для больших матриц предпочтений
— Интеграция с мобильными платформами через REST API

Теоретический анализ позволил обосновать выбор гибридного подхода, сочетающего SVD для точности и KNN для интерпретируемости. Определены критерии оценки качества рекомендаций (RMSE, MAE, R^2), использованные в практической части работы. Рассмотренные методы составили основу для разработки мобильного приложения с нейросетевым поиском на базе OpenAI API.

Второй раздел бакалаврской работы посвящён выбору и использованию программных и технических средств, обеспечивших реализацию мобильного приложения с рекомендательной системой. Для достижения поставленных задач потребовалась интеграция широкого спектра технологий: от средств интерфейсной разработки до серверных фреймворков и библиотек машинного обучения. Все компоненты были выбраны с учётом актуальности, надёжности, совместимости и доступности средств для последующей поддержки и расширения проекта.

Основой клиентской части стало использование React Native — популярного фреймворка для кроссплатформенной мобильной разработки, позволяющего создавать приложения под Android и iOS с использованием JavaScript или TypeScript. Он обеспечивает доступ к нативным компонентам мобильных платформ и позволяет организовать гибкую структуру пользовательского интерфейса. Применение React Native позволило сосредоточиться на логике приложения без необходимости разрабатывать отдельные версии под каждую платформу. В рамках проекта использовался TypeScript как надстройка над JavaScript, предоставляющая статическую типизацию, что повысило надёжность и читаемость кода.

Для ускорения процесса разработки и тестирования применялась платформа Expo. Expo предоставляет набор готовых инструментов и библиотек, упрощающих работу с камерой, хранилищем, навигацией и сетью. Также Expo позволил использовать приложение Expo Go — официальный инструмент, позволяющий тестировать приложение в реальном времени на физическом устройстве без необходимости сборки APK. Это значительно сократило цикл разработки и обеспечило быструю проверку новых функций.

Для построения интерфейса использовался встроенный StyleSheet API, предоставляющий декларативный способ стилизации компонентов. Макеты экранов включали: формы входа и регистрации, карточки фильмов, интерфейс оценки, список рекомендаций, а также анкету для нейросетевого поиска. Использование модульных компонентов обеспечило переиспользуемость и лёгкость доработок. Навигация между экранами реализована с помощью библиотеки react-navigation, предоставляющей стек-маршруты и удобный способ управления переходами.

Взаимодействие клиента с сервером осуществлялось посредством библиотеки axios — мощного HTTP-клиента для выполнения сетевых запросов. Axios позволил выполнять авторизованные запросы, обрабатывать ответы и ошибки, а также централизовать настройку маршрутов и заголовков.

Серверная часть системы была разработана с использованием FastAPI — современного Python-фреймворка для построения асинхронных REST API. FastAPI сочетает высокую производительность, понятный синтаксис и автоматическую генерацию документации. Асинхронность позволила обрабатывать несколько запросов одновременно, что критично для масштабируемости. FastAPI был использован для реализации всех маршрутов сервера: регистрация, авторизация, получение рекомендаций, сохранение оценок, обработка данных от анкеты и взаимодействие с языковой моделью.

Для хранения и управления данными применялась СУБД PostgreSQL — реляционная база данных с поддержкой транзакций, внешних ключей и индексов. Она обеспечила высокую надёжность, гибкую схему и масштабируемость. Работа с БД на уровне Python кода осуществлялась через SQLAlchemy — объектно-реляционное отображение, позволяющее описывать таблицы как классы и работать с ними на уровне Python-объектов. Это позволило избежать ошибок, связанных с "сырыми"SQL-запросами, и улучшило читаемость кода.

Центральное место в рекомендательной системе занимали алгоритмы машинного обучения. Для построения и тестирования моделей использовалась библиотека Surprise, предназначенная специально для реализации рекомендательных алгоритмов. Были обучены и внедрены две модели:

- KNN (k ближайших соседей) — для поиска похожих пользователей;
- SVD (Singular Value Decomposition) — для разложения матрицы оценок и выявления скрытых факторов предпочтений.

Модели обучались на расширенном датасете, включающем записи из MovieLens и оценки, полученные от пользователей приложения. Предварительная обработка данных осуществлялась с помощью библиотеки pandas, а визуализация результатов — через matplotlib. Для оценки моделей использовались метрики RMSE, MAE и коэффициент детерминации R^2. По результатам тестирования SVD показал более высокую точность (RMSE 0.77), чем KNN (RMSE 0.95), что обусловило выбор SVD в качестве основной модели для генерации рекомендаций.

Дополнительно в систему был интегрирован модуль нейросетевого поиска с использованием OpenAI API. На стороне клиента пользователь заполнял анкету, а на сервере формировался prompt-запрос к языковой модели. Для взаимодействия с API использовалась официальная Python-библиотека openai, обеспечивающая отправку текстовых запросов и обработку ответов. В ответ возвращался список фильмов, подобранных на основе описания состояния пользователя. Эта часть системы позволила создавать рекомендации без предварительной истории оценок, что особенно важно для новых пользователей и в условиях «холодного старта».

Безопасность и стабильность API обеспечивались валидацией данных через Pydantic, а пароли хранились в зашифрованном виде с использованием библиотеки bcrypt. Все конфиденциальные данные, включая ключи API, были вынесены в .env-файл и загружались через модуль dotenv, что соответствует практике безопасной разработки.

Таким образом, реализация программной части проекта потребовала комплексного подхода к выбору средств разработки. Использованные технологии — от React Native и FastAPI до PostgreSQL, Surprise и OpenAI — были выбраны на основе их эффективности, гибкости и способности работать в связке. Это обеспечило успешную реализацию рекомендательной системы, способной

учитывать как числовые оценки, так и эмоциональное состояние пользователя.

Третий раздел работы посвящён поэтапной реализации мобильного приложения, охватывая все стадии — от проектирования структуры пользовательского интерфейса и настройки навигации до интеграции с серверной частью, обученными моделями и внешними API. Основное внимание в процессе разработки уделялось модульности, удобству пользователя, стабильности работы приложения и соответствию современным требованиям к мобильным интерфейсам.

На первом этапе была развернута рабочая среда. Для реализации клиентской части использовалась связка React Native + Expo. React Native позволил писать единый код для Android, а Expo значительно упростил сборку, запуск и тестирование. Среда разработки была настроена с помощью Visual Studio Code, что дало возможность использовать мощные расширения для React и TypeScript. Для запуска приложения на реальном устройстве использовалось официальное мобильное приложение Expo Go, через которое можно было в режиме реального времени тестировать любые изменения.

После базовой инициализации проекта с помощью команды expo init началась реализация ключевых экранов. Первоочередной задачей стало создание экрана авторизации и регистрации. На нём реализованы формы ввода логина, почты и пароля, подключена проверка корректности ввода, добавлена индикация ошибок. Для обмена данными с сервером использовалась библиотека axios, а состояние форм обрабатывалось с помощью хуков useState и useEffect. Затем в проекте был реализован экран оценивания фильмов. Пользователь мог просматривать список доступных фильмов и выставлять оценки в диапазоне от 1 до 5. Эти данные отправлялись через POST-запрос на сервер, где сохранялись в базе данных PostgreSQL. Каждая карточка фильма содержала название, жанры и описание, а также элемент управления для выставления оценки. Особое внимание было уделено тому, чтобы после отправки оценки происходило обновление интерфейса без перезагрузки экрана, что обеспечивало ощущение плавного и «живого» взаимодействия.

Далее был реализован экран персональных рекомендаций. После авторизации пользователь мог перейти на этот экран, где отображался список фильмов, отобранных по модели SVD или KNN. Запрос на сервер отправлялся по уникальному идентификатору пользователя, и в ответ приходил массив фильмов с

их описанием. Здесь применялась логика условного рендера: если пользователь ранее не выставлял оценки, приложение информировало его о необходимости оценить хотя бы несколько фильмов для запуска модели.

Одним из важнейших этапов разработки стал модуль нейросетевого поиска, построенный на основе интеграции с OpenAI API. Для этого в приложении был создан отдельный экран с интерактивной анкетой. Пользователь заполнял четыре открытых текстовых вопроса, касающиеся его текущего настроения, желаемого впечатления от фильма, личных интересов и планов на день. После нажатия кнопки «Получить рекомендации» все введённые данные собирались и отправлялись на сервер в виде JSON. На серверной стороне формировался структурированный prompt-запрос, который передавался в языковую модель. Ответ в формате JSON содержал список фильмов с пояснением причин, по которым они были выбраны. Эти фильмы отображались в виде вертикального списка, оформленного в виде карточек.

Каждый экран проекта был оформлен с помощью встроенного API StyleSheet.creat
что обеспечивало единый стиль для всех компонентов. Основное оформление включало адаптивную типографику, выравнивание по центру, использование цветовых акцентов и стандартных системных шрифтов. Для всех элементов были предусмотрены отступы и fallback-сценарии на случай ошибок. Приложение поддерживает как вертикальную, так и горизонтальную ориентацию.

Навигация между экранами была организована с использованием библиотеки react-navigation/native и стек-навигации. Все маршруты были логически структурированы: после авторизации пользователь попадал на главный экран, откуда имел доступ к рекомендациям, анкете и своему профилю. Обработка маршрутов была реализована через expo-router, обеспечив чистый и модульный подход к навигации.

На всём протяжении разработки применялись принципы модульности и переиспользуемости кода. Каждый логический блок (авторизация, оценка, рекомендации, нейропоиск) был оформлен как отдельный компонент или экран. Это упростило как отладку, так и масштабирование проекта в будущем. Кроме того, активно использовались хуки React (useState, useEffect, useContext), что обеспечило управление состоянием приложения без необходимости использования дополнительных библиотек.

Особое внимание уделялось обработке исключений и устойчивости при-

ложения. Все сетевые запросы имели try/catch-блоки, ошибки логировались и отображались в виде предупреждений пользователю. В случае потери соединения с сервером пользователь получал сообщение об ошибке, а интерфейс корректно возвращался в исходное состояние.

На заключительном этапе был выполнен визуальный аудит приложения. Были протестированы все пользовательские сценарии: от регистрации до получения рекомендаций через нейропоиск. Приложение проходило тестирование на реальном устройстве Android в разных условиях подключения (Wi-Fi, 4G, offline). На этом этапе также были сделаны финальные правки по позиционированию элементов, текстовкам и мелким визуальным недочётам.

Завершающим шагом стало визуальное оформление результатов разработки. Были сделаны скриншоты всех основных экранов: формы входа, экрана фильмов с оценками, блока персональных рекомендаций и анкеты для нейропоиска. Скриншоты демонстрируют внешний вид приложения и были включены в приложение к дипломной работе как подтверждение успешной реализации всех этапов.

В процессе разработки системы большое внимание было уделено качественному тестированию как backend, так и frontend части приложения. Проверка работы API осуществлялась вручную и с помощью инструмента Postman, что позволило оперативно обнаруживать и устранять ошибки на этапах создания и настройки маршрутов. Каждый из ключевых эндпоинтов был последовательно проверен: от регистрации пользователя до получения рекомендаций. При тестировании особое внимание уделялось корректности передачи и обработки данных, а также валидации входных параметров. Благодаря этому удалось добиться стабильной работы всех основных функций сервера.

На стороне клиентского приложения верификация производилась как в Android-эмуляторе, так и на реальных мобильных устройствах. Это обеспечило тестирование поведения интерфейса в различных условиях: от качества отображения элементов до времени отклика при сетевых задержках. Такой подход позволил выявить и учесть особенности работы с различными плотностями пикселей, поведения клавиатуры и рендеринга списков. Были внесены правки в стили, улучшена отзывчивость форм и оптимизированы состояния экрана.

Таким образом, третий раздел демонстрирует не только последовательную реализацию всех компонентов мобильного приложения, но и глубоко продуман-

ный подход к архитектуре, взаимодействию с сервером и созданию удобного пользовательского опыта. Проект завершился созданием стабильного, адаптивного и функционального решения, готового к дальнейшему использованию и развитию.

Произведена демонстрация работы приложения, его функционала: экран рекомендаций, экран нейро-поиска (рисунки 1-3).

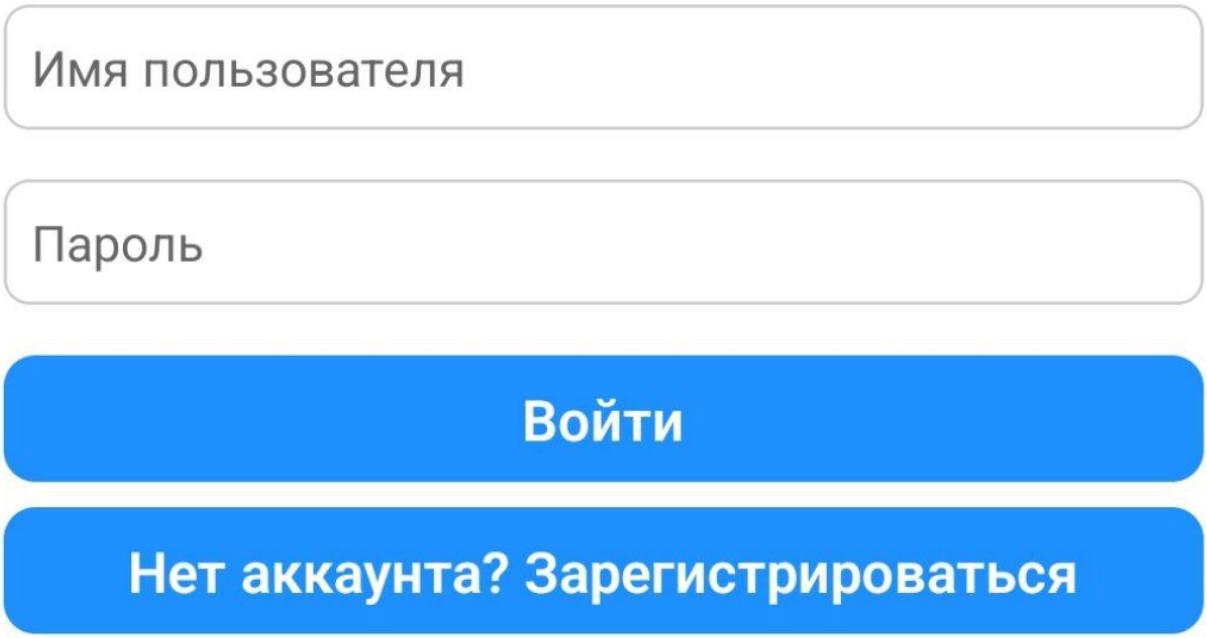

Рисунок 1 – Окно авторизации

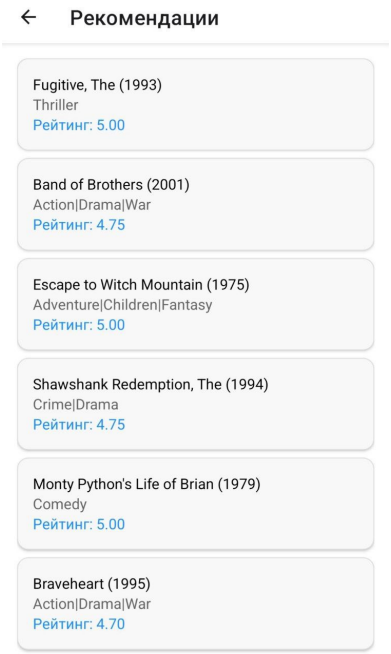

Рисунок 2 – Окно персонализированных рекомендаций

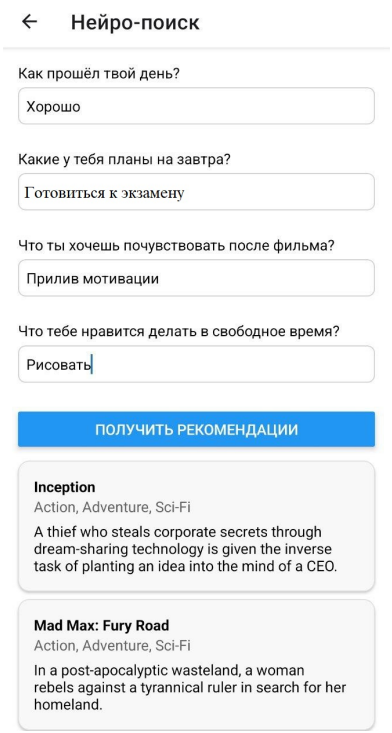

Рисунок 3 – Окно "нейро-поиска"

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной выпускной квалификационной работы было разработано мобильное приложение для подбора фильмов с системой персонализированных рекомендаций, объединяющее современные подходы машинного обучения, инструменты backend- и frontend-разработки, а также интеллектуальные возможности больших языковых моделей через OpenAI API. Созданное мобильное приложение позволяет пользователю получать рекомендации на основе истории оценок, а также воспользоваться нейро-поиском, который адаптируется под текущее эмоциональное состояние, используя текстовые ответы. Интеграция OpenAI API обеспечила гибкость системы, позволяя генерировать рекомендации даже для новых пользователей без оценок, что решает проблему «холодного старта».

Проект демонстрирует практическую реализацию сразу нескольких важных концепций: организацию клиент-серверного взаимодействия, интеграцию пользовательских данных из базы, построение рекомендательных моделей на основе алгоритмов SVD и KNN, использование OpenAI API для анализа текстовых ответов и предпочтений, а также реализацию интуитивного интерфейса на платформе React Native.

Благодаря применению FastAPI и PostgreSQL удалось обеспечить высокую производительность и масштабируемость серверной части, а библиотека Surprise позволила эффективно обучать и тестировать классические модели.

Особое внимание было уделено расширяемости системы: архитектура проекта предусматривает возможность подключения новых моделей, переход на облачные хранилища, реализацию дополнительных функций (избранное, лайки, расширенный профиль) и дальнейшую интеграцию с OpenAI API для более глубокой персонализации. Все эти элементы формируют фундамент для развития системы в сторону полноценного интеллектуального помощника по выбору контента.

Решённая в проекте задача наглядно демонстрирует потенциал применения рекомендательных систем, усиленных возможностями больших языковых моделей, в мобильной среде, где важны точность предсказаний, удобство и адаптивность интерфейса. Реализация подобного решения в рамках дипломной работы позволила углубить знания в области анализа данных, программной инженерии и работы с современными API.