



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение
высшего образования*

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных
технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1

по дисциплине

«Тестирование и верификация программного обеспечения»

**Тема: «ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА
МЕТОДОМ «ЧЕРНОГО ЯЩИКА»»**

Выполнил студент группы ИКБО-13-23

Попов А.В.
Котов А.А.
Молчанов А.
Русаков М.
Иванов В.

Принял

Нефор _____

Практическая работа выполнена

«__»_____2025 г.

(подпись студента)

«Зачтено»

«__»_____2025 г.

(подпись руководителя)

Москва 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Основания для разработки.....	6
2 Назначение разработки	6
3. Требования к программе.....	7
3.1 Функциональные требования	7
3.2 Требования к надёжности	8
3.3 Условия эксплуатации.....	8
3.4 Требования к совместимости.....	9
4. Требования к интерфейсу	9
4.1 Общие требования.....	9
4.2 Основные элементы интерфейса.....	9
4.3 Взаимодействие с элементами	10
4.4 Удобство и доступность.....	10
5. Критерии приёмки	10
6. Требования к документации	11
7. Порядок контроля и приёмки	11
8. Этапы и сроки разработки	12
ПОЛЬЗОВАТЕЛЬСКАЯ ДОКУМЕНТАЦИЯ.....	13
1 Авторизация и регистрация	13
2 Дашборд.....	14
3 Управление складом.....	15
4 Аналитика.....	18
5 Общие элементы интерфейса	18
Описание ошибок системы.....	19

Ошибка регистрации	19
Ошибка отображения заполнения складов	20
Ошибка добавления товара.....	20
Ошибка удаления всех товаров со складов.....	21
Ошибка в окне «Аналитика»	21
Ошибка сброса фильтров.....	22
Ошибка добавления новых товаров (LocalStorage).....	23
Ошибка переполнения складов	24
Техническое задание и документация программного продукта команды Стрипуха_Club	25
Анализ документации продукта команды Стрипуха_Club	32
Анализ ошибок проекта	38
ЗАКЛЮЧЕНИЕ.....	46

ВВЕДЕНИЕ

Прослушивание музыки — это часть повседневной жизни, и современным пользователям необходим удобный инструмент, который позволяет не только находить любимые треки, но и создавать собственные музыкальные коллекции.

Цель разрабатываемого сервиса — предоставить современный веб- и мобильный интерфейс для комфортного прослушивания музыки и персонализации музыкального опыта. Приложение будет использоваться широкой аудиторией — от меломанов до тех, кто ищет музыку для отдыха, работы или спорта.

Сервис	позволит	пользователям:
--------	----------	----------------

- | | | |
|--|--|--|
| | <ul style="list-style-type: none">искать и воспроизводить миллионы треков из разных жанров;создавать, редактировать и удалять персональные плейлисты;делиться своими подборками с друзьями и подписчиками;получать рекомендации на основе вкусов и истории прослушиваний;отслеживать новинки и популярные чарты. | |
|--|--|--|

Ожидаемым результатом внедрения сервиса станет повышение удобства и качества музыкального досуга пользователей, расширение их музыкального кругозора и создание индивидуального пространства для любимой музыки.

1 Основания для разработки

Разработка веб-приложения для прослушивания музыки продиктована растущей потребностью пользователей в удобных и персонализированных сервисах, позволяющих формировать собственное музыкальное пространство. В условиях современного цифрового мира разрозненные музыкальные библиотеки и офлайн-хранение треков не обеспечивают должного уровня доступности, персонализации и удобства.

Основаниями для разработки являются:

- потребность в централизованном доступе к музыке из разных жанров и источников;
- необходимость создания и управления персональными плейлистами;
- требования к персонализации музыкального опыта и рекомендациям на основе интересов пользователя;
- отсутствие у многих сервисов удобного и интуитивного интерфейса для организации музыкальных коллекций.

В качестве ориентиров при подготовке проекта использованы:

- современные практики разработки веб- и мобильных приложений в сфере цифровых медиа;
- рекомендации по UX/UI-дизайну для музыкальных стриминговых сервисов.

Эти факторы и выявленные потребности пользователей стали основанием для начала разработки музыкального сервиса нового поколения.

2 Назначение разработки

Цель разработки — создание веб-приложения для удобного прослушивания музыки и персонализации музыкального опыта пользователей.

Задачи, решаемые сервисом:

- предоставление доступа к широкой музыкальной библиотеке;
- возможность создания, редактирования и удаления персональных плейлистов;
- рекомендации треков на основе вкусов и истории прослушивания;
- снижение трудоёмкости поиска подходящей музыки за счёт умных алгоритмов;
- обеспечение удобного и безопасного доступа к сервису с разных устройств.

3 Требования к программе

3.1 Функциональные требования

Цель продукта — предоставить удобный интерфейс для прослушивания музыки, создания персональных плейлистов и анализа предпочтений пользователей.

Основные функции:

1. Управление треками:

- поиск и воспроизведение музыки по названию, исполнителю или жанру;
- добавление треков в плейлисты;
- возможность добавления в «избранное»;
- отображение списка треков в виде таблицы или карточек.

2. Управление плейлистами:

- создание нового плейлиста с указанием названия и описания;
- редактирование и удаление плейлистов;
- просмотр списка всех плейлистов;
- переход к деталям выбранного плейлиста (состав треков, длительность).

3. Аналитика и рекомендации:

- отображение статистики прослушиваний (любимые жанры, исполнители, треки);
- рекомендации на основе истории прослушивания;
- построение графиков и диаграмм активности пользователя (например, популярное время прослушивания).

4. Интерфейс:

- удобная навигация между разделами (треки, плейлисты, рекомендации, аналитика);
- адаптивная верстка для ПК и мобильных устройств;
- уведомления о новых релизах, подборках и плейлистах.

5. Авторизация и регистрация пользователей:

- форма входа по e-mail и паролю, валидация введенных данных;
- возможность регистрации нового пользователя;
- сохранение токена авторизации в localStorage для безопасного доступа.

Требования к надёжности

- система должна обеспечивать сохранность данных при сбоях браузера или сети;
- все изменения (добавление, редактирование, удаление) должны фиксироваться в базе данных;
- в случае ошибок при работе с сервером пользователь должен получать уведомление с возможностью повторной операции;
- должна быть реализована защита от потери данных при одновременной работе нескольких пользователей.

3.2 Условия эксплуатации

- веб-приложение должно корректно работать в современных браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) последних версий;
- минимальное разрешение экрана для корректного отображения интерфейса — 1280×720;
- поддержка работы на настольных ПК, ноутбуках, планшетах и смартфонах;
- рекомендуемое количество одновременно работающих пользователей — до 100.

3.3 Требования к совместимости

- клиентская часть должна быть совместима с REST API сервера;
- поддержка формата обмена данными JSON (для взаимодействия с сервером) и CSV (для выгрузки статистики и аналитики прослушиваний);
- возможность интеграции с внешними сервисами (например, социальные сети или системы рекомендаций) через API;
- корректная работа во всех популярных ОС: Windows, macOS, Linux, Android, iOS.

4. Требования к интерфейсу

4.1 Общие требования

- Интерфейс должен быть адаптивным, с поддержкой настольных и мобильных устройств.
 - Основные разделы: «Музыка», «Плейлисты», «Рекомендации», «Аналитика», «Профиль» (для пользователя/администратора).
- Использование цветовой схемы с яркими, но не перегруженными оттенками:
 - тёмно-фиолетовый (#2d1e4a) — основной цвет интерфейса;
 - фиолетовый (#5a189a) и бирюзовый (#00b4d8) — акценты и кнопки;

- салатовый (#90e0ef) и голубой (#caf0f8) — индикаторы активности/онлайн-статуса;
- оранжевый/красный — выделение ошибок или системных уведомлений.

4.2 Основные элементы интерфейса

Шапка сайта: название сервиса, логотип, поисковая строка, кнопка выхода, переход к главному меню.

Главная страница (Dashboard): персональные рекомендации, популярные треки, подборки.

Страница плейлиста: список треков с возможностью сортировки, фильтрации и поиска.

Модальные окна: для создания/редактирования плейлистов, добавления треков в коллекции.

Поисковая строка и фильтры: быстрый поиск по исполнителю, жанру, названию трека.

Панель уведомлений: оповещения о новых релизах, добавлении треков, системных сообщениях.

4.3 Взаимодействие с элементами

- Кнопки «Добавить в плейлист», «Редактировать», «Удалить» должны быть заметными и менять цвет при наведении.
- Популярные и новые треки могут выделяться цветовой подсветкой.
- Всплывающие окна не должны перекрывать основную навигацию.
- Формы регистрации и авторизации должны содержать обязательную валидацию полей (e-mail, пароль).

4.4 Удобство и доступность

- Читаемые шрифты крупного размера.
- Русский язык интерфейса по умолчанию (с возможностью смены языка).
- Поддержка пользователей с ограниченным зрением (контрастные цвета, крупные кнопки, озвучивание элементов).
- Минимизация числа действий для основных операций (например, добавление трека в плейлист — ≤ 2 шага).

5. Критерии приёмки

Программный продукт принимается заказчиком в случае успешной реализации всех функций, заявленных в техническом задании. В частности, должны быть корректно выполнены **авторизация и регистрация пользователей, поиск и воспроизведение треков, управление плейлистами и аналитические инструменты**. Все формы должны обеспечивать валидацию данных, а поиск, фильтрация и сортировка обязаны возвращать корректные результаты.

Надёжность системы подтверждается стабильной работой при одновременном использовании до ста пользователей. В случае сетевых сбоев или ошибок со стороны сервера приложение должно информировать пользователя и позволять повторить операцию без потери данных.

К интерфейсу предъявляются отдельные требования: он должен быть адаптивным и одинаково корректно отображаться на персональных

компьютерах, планшетах и смартфонах. Цветовое выделение применяется для акцентирования важной информации (например, новые релизы, рекомендации, ошибки). Все элементы управления, включая кнопки добавления треков в плейлисты, редактирования и удаления, должны быть заметными и правильно реагировать на действия пользователя. Добавление нового плейлиста или трека в коллекцию должно занимать не более двух шагов.

Кроме того, приложение должно корректно работать в последних версиях браузеров **Chrome, Firefox, Edge и Safari**, поддерживать экспорт аналитических данных (например, статистики прослушиваний) в форматах **JSON и CSV**, а среднее время отклика интерфейса при выполнении основных операций не должно превышать **двух секунд**.

6. Требования к документации

Документация к веб-приложению должна включать руководство пользователя, выполненное в формате пошаговой инструкции.

7. Порядок контроля и приёмки

Контроль и приёмка веб-приложения будут проводиться методом «чёрного ящика», предполагающим проверку функциональности без анализа внутренней структуры системы. Тестированию подлежат все ключевые сценарии: авторизация и регистрация, управление складами и товарами, контроль сроков годности, работа с аналитикой, поиск и фильтрация данных, корректное отображение интерфейса.

Приёмочные испытания включают последовательное выполнение тест-кейсов, фиксацию выявленных ошибок и их исправление. Критерием успешной приёмки считается корректная работа не менее 95% тест-кейсов и стабильное функционирование системы при одновременном использовании нескольких пользователей.

Метод «чёрного ящика» позволяет подтвердить соответствие продукта требованиям ТЗ и оценить его удобство и надёжность с точки зрения конечного пользователя.

8. Этапы и сроки разработки

Разработка веб-приложения осуществляется поэтапно в соответствии с планом-графиком. Первый этап включает анализ предметной области, формирование требований и проектирование архитектуры системы, и должен быть завершён в течение двух дней. На втором этапе выполняется разработка интерфейса и функциональных модулей, включая авторизацию, управление складами и товарами, контроль сроков годности и аналитические инструменты; этот этап планируется на четыре дня. Третий этап посвящён тестированию методом «чёрного ящика», исправлению выявленных ошибок и подготовке документации, и занимает один день. Финальный этап включает приёмку проекта заказчиком, внесение корректировок и окончательную сдачу продукта в эксплуатацию.

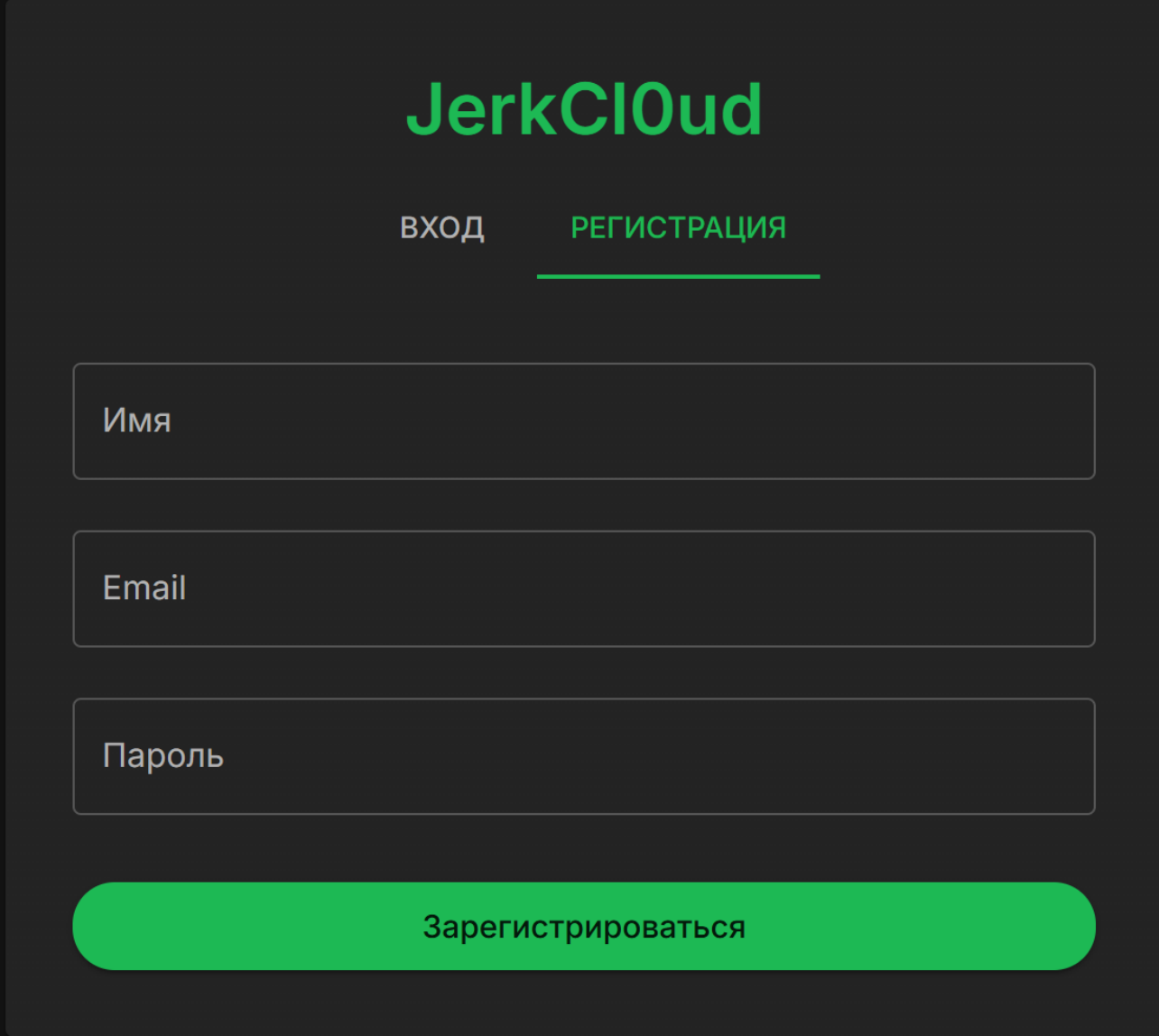
Общий срок реализации проекта составляет одну неделю, при этом последовательное выполнение этапов обеспечивает своевременное выполнение всех требований и высокое качество конечного продукта.

ПОЛЬЗОВАТЕЛЬСКАЯ ДОКУМЕНТАЦИЯ

1 Авторизация и регистрация

Шаг 1. При открытии приложения пользователь видит стартовую страницу с формой авторизации.

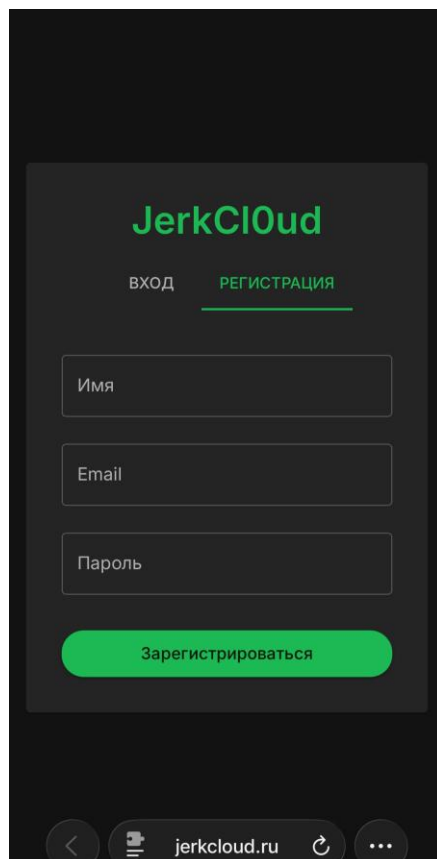
Шаг 2. Введите e-mail и пароль. Формат данных проверяется автоматически, при ошибке система выводит подсказку.

The image shows a dark-themed user interface for a service called "JerkCloud". At the top, the logo "JerkCloud" is displayed in a bright green font. Below the logo, there are two links: "ВХОД" (Login) and "РЕГИСТРАЦИЯ" (Registration). The "РЕГИСТРАЦИЯ" link is highlighted with a green underline. Underneath the links, there are three input fields for registration: "Имя" (Name), "Email", and "Пароль" (Password). At the bottom of the form is a large, rounded green button with the text "Зарегистрироваться" (Register).

Шаг 3. Если аккаунта ещё нет, нажмите кнопку «Регистрация» и заполните форму, указав e-mail, пароль и имя пользователя.

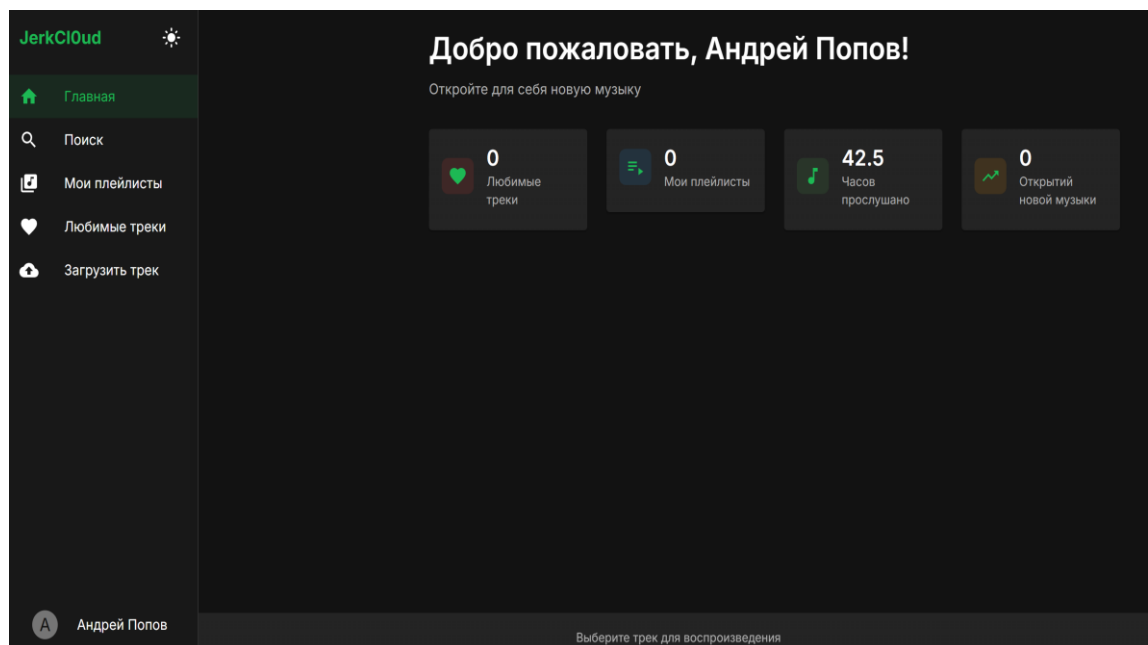
Шаг 4. Для ознакомления можно войти через демо-аккаунт.

Шаг 5. После успешного входа данные сохраняются в localStorage, чтобы не вводить их повторно при следующем открытии сайта.



2 Дашборд

Шаг 1. После авторизации открывается главная страница (Dashboard).



Шаг 2. На экране отображается меню и с информацией об аккаунте.

Шаг 3. Для создания плейлиста необходимо зайти в мои плейлисты и нажать на кнопку создать плей лист.

Шаг 4. Для поиска в меню перейти в поиск и выбираешь либо из предложенных либо при помощи строки поиска.

Поиск музыки

🔍 Поиск треков, исполнителей, альбомов...

Жанры:

Rock

Pop

Jazz

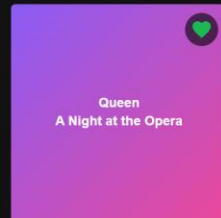
Electronic

Hip-Hop

Classical

Country

Найдено треков: 17



Bohemian Rha...

Queen

A Night at the Opera



просроченый ...

діджей мустафа

Jjerk



Вор

Mr.Credo

Вор



Медляк

Mr.Credo

Медляк



Bohemian Rhapsody

Queen



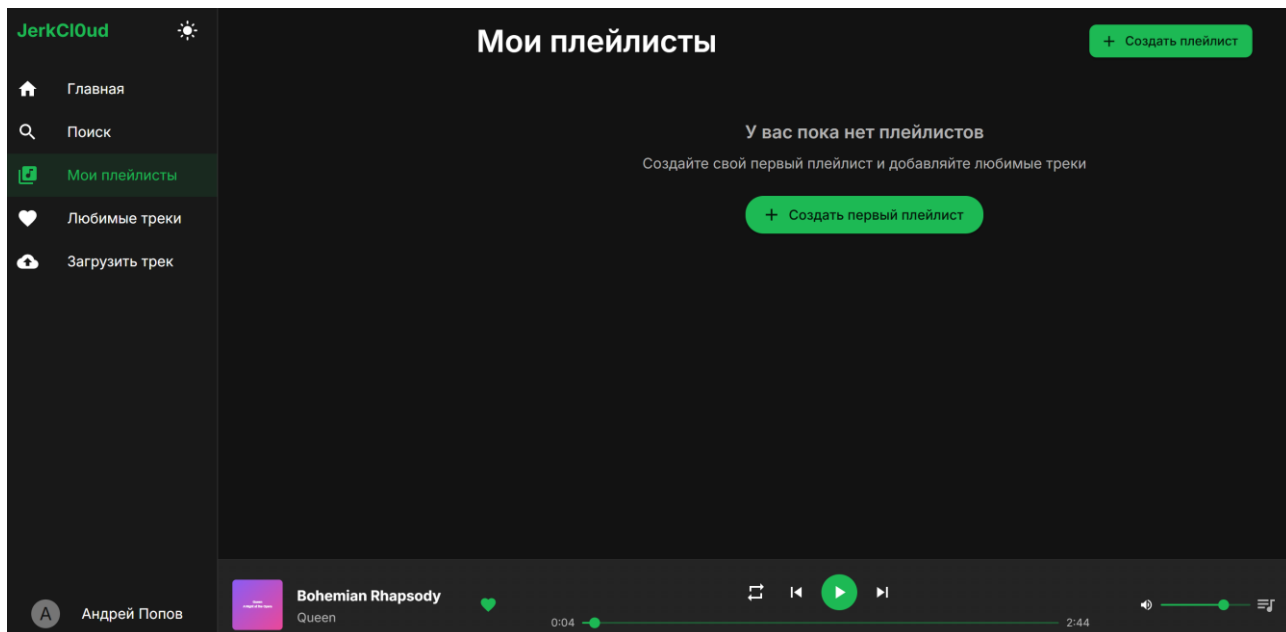
0:04



2:44

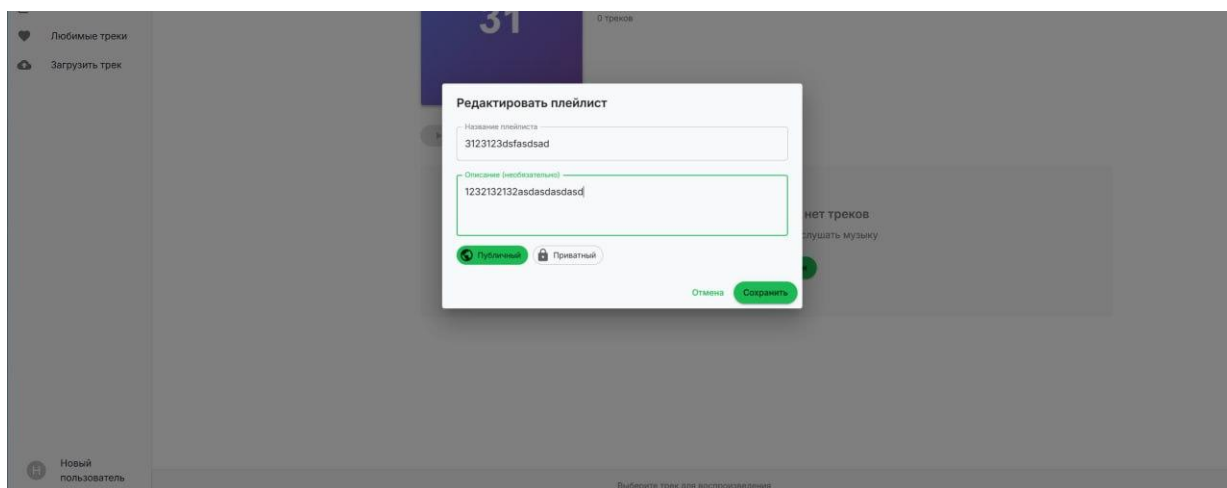
Шаг 5. Для перехода внутрь конкретного плейлиста нажмите на его карточку.

3 Управление плейлистом

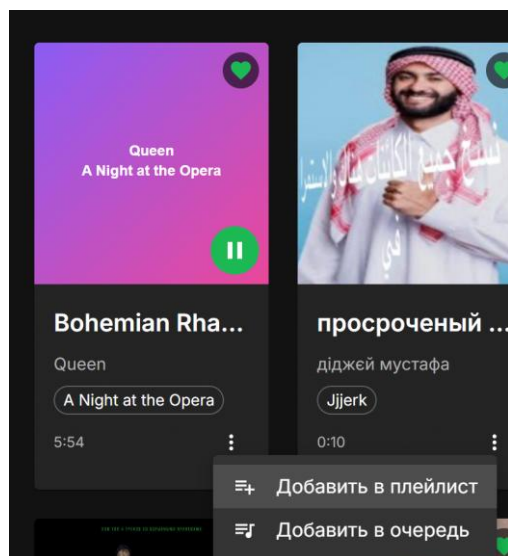


Шаг 1. В верхней части страницы отображается карточка с данными о альбоме.

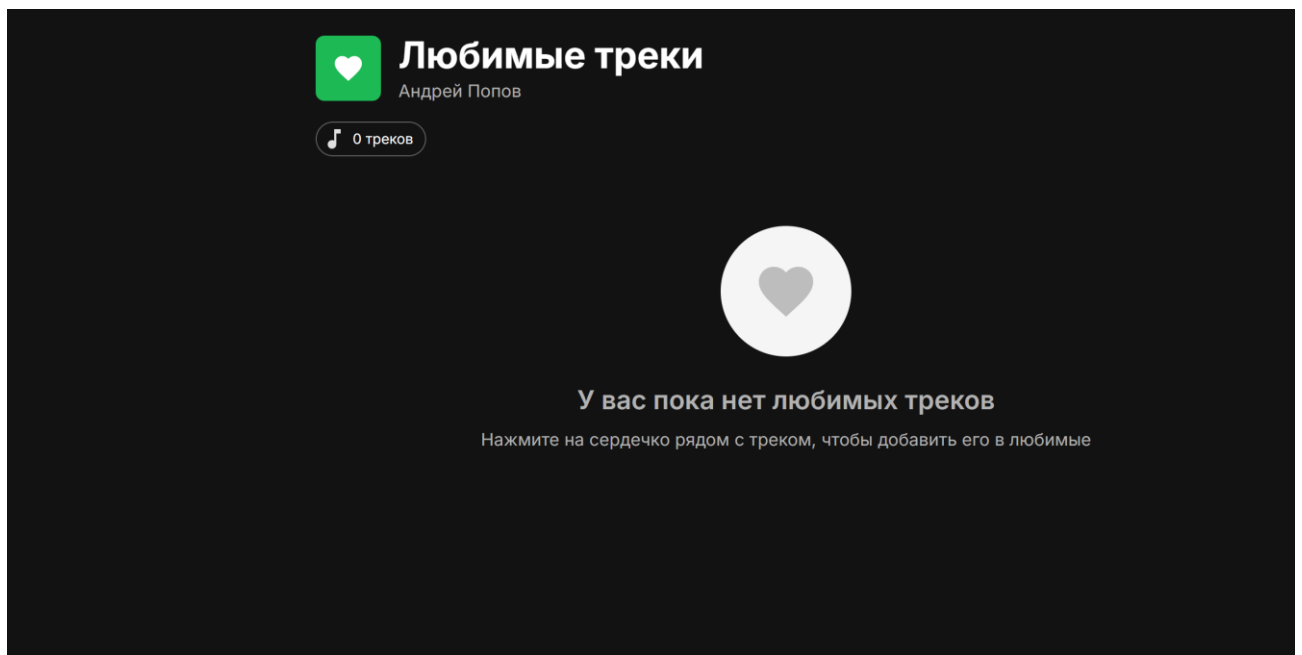
Шаг 2. При нажатии на плейлист можно его редактировать .



Шаг 3.Чтобы добавить трек в плейлист, нужно навестить на карточку трека, нажать на три точки и добавить в плейлист).



Шаг 4. Также трек можно добавить в избранное, нажав на сердечко, если оно стало зеленым, значит трек добавлен в избранное



Шаг 5. Для удаления товара используйте кнопку «Удалить» и подтвердите действие в модальном окне.

4 Добавить трек на сайт ДОБАВИТЬ НОВЫЙ ТРЕК


Заполните информацию о треке и загрузите необходимые файлы

🎵 Основная информация

<input type="text" value="Название трека *"/>	<input type="text" value="Исполнитель *"/>
<input type="text" value="Альбом"/>	<input type="text" value="Жанр"/>
<input type="text" value="Длительность (мм:сс или секунды)"/> <small>Введите в формате мм:сс или количество секунд</small>	<input type="text" value="Дата релиза"/> 30.09.2025

☁ Загрузка файлов

Аудиофайл *




Загрузить аудио

Поддерживаются форматы: MP3, WAV, FLAC, OGG

Перетащите файл сюда или нажмите для выбора

Обложка (необязательно)



Загрузить обложку

Поддерживаются форматы: JPG, PNG, WEBP

Перетащите файл сюда или нажмите для выбора

Шаг 1. Заполнить форму от треке.

Шаг 2. Загрузить сам аудиофайл и обложку трека.

Шаг 3. После заполнения всех полей нажать на кнопку Добавить трек или отмена, если передумали.

.

5 Общие элементы интерфейса

- В боковой панели расположены логотип, имя пользователя и кнопка выхода.
- Боковое меню содержит ссылки на основные разделы: «Главная», «Поиск», «Мои плейлисты», «Любимые треки», «Загрузить трек». Активный раздел подсвечивается.

Описание ошибок системы

Ошибка регистрации

Ошибка 1. Обновление имени пользователя

- Описание: После изменения имени пользователя интерфейс показывает уведомление о успешном обновлении, однако новое имя не отображается до перезагрузки страницы.
- Ожидаемое поведение: Имя пользователя должно обновляться мгновенно во всех разделах интерфейса после подтверждения изменений.
- Фактическое поведение: Имя остаётся старым до перезагрузки страницы.
- Шаги воспроизведения:
 1. Перейти в профиль пользователя.
 2. Изменить имя и сохранить изменения.
 3. Обратит внимание на отображаемое имя без перезагрузки страницы.
- Критичность: Средняя — нарушает UX, но данные корректно сохраняются в базе.

Ошибка 2. Несоответствие лайка в поиске и в «Любимых»

- Описание: Трек, отмеченный как лайкнутый в поиске, не всегда корректно отображается в разделе «Любимые».
- Ожидаемое поведение: Любой трек, помеченный лайком, должен автоматически появляться в «Любимых».
- Фактическое поведение: Лайкнутый трек может не отображаться в «Любимых».
- Шаги воспроизведения:
 1. Найти трек через поиск.
 2. Отметить его как лайк.
 3. Перейти в «Любимые» и проверить наличие трека.
- Критичность: Средняя — влияет на корректность отображения пользовательских предпочтений.

Ошибка 3. Некорректное время прослушивания треков

- Описание: Время прослушивания треков не обновляется в зависимости от фактического прослушанного времени (замокано/фиксировано).
- Ожидаемое поведение: Время прослушивания должно соответствовать реальному количеству проигранных секунд трека.
- Фактическое поведение: Время прослушивания остаётся неизменным.
- Шаги воспроизведения:
 1. Запустить трек.
 2. Прослушать часть трека.
 3. Проверить обновление времени прослушивания.
- Критичность: Средняя — влияет на статистику и рекомендации.

Ошибка 4. Обновление названия и описания плейлиста

- Описание: Обновление названия и описания плейлиста не работает — изменения не сохраняются и не отображаются в интерфейсе.

- Ожидаемое поведение: После редактирования название и описание плейлиста должны обновляться мгновенно.
- Фактическое поведение: Изменения не применяются ни сразу, ни после перезагрузки страницы.
- Шаги воспроизведения:
 1. Открыть плейлист для редактирования.
 2. Изменить название или описание.
 3. Сохранить изменения и проверить результат.
- Критичность: Высокая — блокирует возможность персонализации плейлистов.

Ошибка 5. Лайк в плеере сбрасывает проигрывание

- Описание: Если нажать кнопку «Лайк» при воспроизведении трека, проигрывание сбрасывается и трек начинается с начала.
- Ожидаемое поведение: Лайк не должен прерывать текущий трек — воспроизведение продолжается с текущей позиции.
- Фактическое поведение: Трек начинает проигрываться с нуля после лайка.
- Шаги воспроизведения:
 1. Запустить трек в плеере.
 2. Нажать «Лайк» во время проигрывания.
 3. Обратить внимание на сброс времени воспроизведения.
- Критичность: Средняя — мешает комфортному прослушиванию.

Техническое задание и документация программного продукта команды Стрипуха_Club

Введение

JerkCl0ud — полнофункциональный музыкальный сервис, позволяющий пользователям прослушивать музыку, создавать и управлять плейлистами, отмечать треки «лайками», а также получать персонализированные рекомендации. Приложение обеспечивает удобный интерфейс для поиска музыки, управления коллекциями, просмотра истории прослушиваний и анализа предпочтений.

Тестирование будет проводиться методом «чёрного ящика», проверяя корректность работы на основе пользовательских сценариев и получаемых результатов без доступа к исходному коду.

2. Основания для разработки

Разработка JerkCl0ud проводится как учебный фуллстек-проект и преследует три цели:

Практическая: создание работающего прототипа музыкального сервиса с полным функционалом.

Обучающая: моделирование стандартного функционала современных стриминговых сервисов.

Контроль качества: проверка стабильности, надёжности и UX через тестирование пользовательских сценариев.

3. Назначение разработки

Приложение решает следующие задачи:

Автоматизация прослушивания музыки: предоставление пользователю удобного инструмента для поиска и воспроизведения треков.

Управление плейлистами: создание, редактирование и удаление плейлистов, добавление треков и организация коллекций.

Персонализация: рекомендации треков на основе истории прослушиваний, лайков и предпочтений пользователя.

Аналитика: просмотр статистики прослушивания, популярности треков и эффективности плейлистов.

4. Требования к приложению

4.1 Функциональные требования

1) Пользователь и профиль

Регистрация: создание учётной записи с email и паролем, проверка корректности данных, блокировка повторной регистрации с одним email.

Авторизация: вход по учётным данным, автоматический выход после бездействия (сессии/JWT).

Управление профилем: редактирование имени, email, пароля и персональных настроек.

Удаление профиля: возможность безвозвратного удаления через настройки.

2) Поиск и треки

Поиск по названию, исполнителю или жанру.

Отображение результатов с возможностью лайка, добавления в плейлист и предпрослушивания.

Состояние лайка должно быть синхронизировано между поиском, плейлистами и «Любимыми».

3) Плейлисты

Создание, редактирование названия и описания, удаление.

Добавление/удаление треков.

Обновления названия и описания должны отображаться мгновенно.

4) Плеер

Воспроизведение треков с возможностью паузы, перемотки, лайка.

Лайк не должен прерывать проигрывание трека.

Время прослушивания должно корректно обновляться в статистике.

5) Аналитика и рекомендации

Просмотр статистики прослушивания треков и плейлистов.

Персональные рекомендации на основе истории прослушиваний и лайков.

Экспорт данных в форматах JSON/CSV.

6) Служебные функции

Обработка ошибок и понятные сообщения пользователю.

Логирование ключевых действий: регистрация, вход, лайки, создание плейлистов.

4.2 Надёжность

Целостность данных: изменения профиля, лайки и плейлисты должны синхронизироваться на всех устройствах.

Доступность: 99% времени в учебной среде.

Устойчивость к сбоям: защита критичных операций от потери данных при сетевых ошибках.

4.3 Эксплуатация

Браузеры: Chrome, Firefox, Safari, Edge последних версий.

Сетевое окружение: работа при задержках до 300 мс и потере пакетов до 2%.

4.4 Совместимость

Адаптивный интерфейс: корректное отображение на мобильных и настольных устройствах, Full HD и выше.

Touch-интерфейс: удобное управление интерактивными элементами на мобильных устройствах.

5. Интерфейс

Навигация: единая шапка с «Поиск», «Плейлисты», «Любимые», «Аналитика», «Профиль»/«Войти».

Карточка трека: изображение обложки, название, исполнитель, кнопки лайка и добавления в плейлист.

Плейлист: список треков с возможностью редактирования, удаления, изменения порядка.

Сообщения: ошибки и успехи — заметные и привязанные к действиям

пользователя.

Адаптивность: интерактивные элементы $\geq 44 \times 44$ пикселя, без перекрытий.

6. Критерии приёмки

Тестирование: $\geq 95\%$ тест-кейсов (позитивные и негативные сценарии).

Корректность плейлистов: все операции с плейлистами выполняются без ошибок, обновления отображаются мгновенно.

Безопасность сессий: автоматическое завершение сессий, предотвращение дублирования действий.

Кросс-платформенность: корректная работа на мобильных (360×640) и настольных ($\geq 1366 \times 768$) устройствах.

Если хочешь, я могу подготовить раздел «Ошибки и баги JerkCloud», где будут описаны все твои проблемы с лайками, плейлистами и временем прослушивания в формате готового отчёта для QA. Это сразу вписывается в ТЗ/документацию.

Хочешь, чтобы я это сделал?

Требования к документации

Документация проекта

Для запуска и использования JerkCloud должна быть предоставлена следующая документация:

1. README для пользователя:

Краткое описание основных функций для конечного пользователя:

- регистрация и авторизация;
- поиск и прослушивание треков;
- создание, редактирование и удаление плейлистов;
- управление «лайками» и «Любимыми» треками;
- редактирование профиля;
- просмотр статистики и рекомендаций.

2. DOCS для разработчика:

Подробное техническое руководство по развёртыванию проекта локально или на тестовом стенде:

- Приложение запускается из **main.tsx**, где **App** обернут в Zustand для управления состоянием и BrowserRouter для маршрутизации.
- Главная страница содержит **Header**, **Hero-блок** и список популярных треков из **useTracksStore**.
- Страница авторизации поддерживает вход и регистрацию, хранит **JWT** в cookies и сторе, выводит уведомления через **Toast**.
- Плейлисты реализованы через **usePlaylistsStore** с компонентами **PlaylistItem**, поддержкой редактирования, добавления/удаления треков.
- Страница трека подгружает данные по ID и отображает описание, альбом, исполнителя, обложку и статистику прослушивания.
- Компонент плеера управляет воспроизведением, лайками, перемоткой и отображением времени прослушивания.

2. Порядок контроля и приёмки

Приёмка включает следующие этапы тестирования:

1. Функциональное тестирование:

Полная проверка всех ключевых сценариев пользователя:

- регистрация, вход, выход;
- поиск и воспроизведение треков;
- создание, редактирование и удаление плейлистов;
- добавление треков в «Любимые»;
- редактирование профиля;
- просмотр аналитики и персональных рекомендаций.

2. Негативное тестирование:

Проверка обработки ошибок:

- ввод некорректного email или слабого пароля;
- отправка пустых обязательных полей;
- попытка регистрации с уже существующим email;
- действия с истёкшей сессией;
- попытка добавить трек, который уже находится в плейлисте.

3. UI/UX-тестирование:

Проверка адаптивности интерфейса, доступности элементов с клавиатуры, корректного отображения уведомлений и системных сообщений.

3. Этапы и сроки разработки

Этап 1: Аналитика и уточнение требований — 1 неделя

- Детализация всех пунктов ТЗ, создание прототипов ключевых экранов (главная, поиск, плеер, плейлисты, профиль).

Этап 2: Проектирование — 1 неделя

- Разработка детальных UI/UX-потокосов;
- Проектирование и описание API-контрактов.

Этап 3: Реализация функционала — 2–3 недели

- Последовательная разработка модулей: поиск → плеер → плейлисты → лайки → профиль → аналитика.

Этап 4: Тестирование и исправление ошибок — 1–2 недели

- Функциональное, негативное и нагрузочное тестирование;
- Исправление выявленных багов и недочётов.

Этап 5: Итоговая приёмка и релиз — 1 неделя

- Финальный прогон тестов, подготовка документации, развёртывание на production-сервере.

Анализ документации продукта команды Стрипуха_Club

1. Структура и полнота документации

Плюсы:

- Документ имеет логичную структуру: Введение → Основания → Назначение → Требования → Критерии приёмки → Контроль и приёмка → Этапы и сроки разработки.
- Содержит как **функциональные**, так и **нефункциональные требования**, что обеспечивает комплексное понимание проекта.
- Чётко описаны цели приложения: автоматизация вычислений, визуализация формул, поддержка различных типов выражений.
- Указаны конкретные критерии приёмки с количественными показателями (например, рендеринг формулы ≤ 1 секунда, точность проверки совпадений $\geq 90\%$).
- Присутствует план-график разработки с разбивкой на этапы, что позволяет оценить сроки реализации проекта.

Недостатки/замечания:

- В разделе «Требования к интерфейсу» повторяются пункты из функциональных требований. Это дублирование снижает читаемость документа.
- Нет конкретных технологий или фреймворков для реализации приложения. Упомянуто только «выбор подходящих технологий», без уточнения фронтенда, бэкенда, библиотек для рендеринга формул или работы с LaTeX.
- Нет информации о возможных ограничениях платформы (браузеры, устройства, минимальные требования).
- В разделе безопасности требования очень общие: «контроль доступа», но не уточнено, какой тип авторизации используется и как защищаются данные.
- Отсутствует описание взаимодействия с базой данных и возможных форматов хранения математических выражений.
- Нет описания логики обработки ошибок и уведомлений пользователя, что важно для веб-приложений с интерактивным UI.

2. Функциональные требования

Сильные стороны:

- Покрывает ключевой функционал: ввод формул, редактирование, конвертация в LaTeX, обратное преобразование, экспорт, анализ совпадений.
- Упоминается поддержка как базовых операций, так и сложных математических структур (дроби, корни, интегралы и др.).
- Включена возможность интеграции с текстовыми документами и экспортом в .docx и .pdf.

Рекомендации по улучшению:

- Следует указать ограничения на вводимые формулы (например, максимально допустимая длина или глубина вложенности).
- Уточнить поведение приложения при некорректном или синтаксически неверном вводе формул.
- Добавить описание работы с историей формул или возможностью сохранения проектов пользователя.

3. Нефункциональные требования

Плюсы:

- Производительность: упомянут быстрый рендеринг и минимальная задержка анализа.
- Безопасность: контроль доступа.

Недостатки:

- Не указаны требования к масштабируемости и нагрузочной устойчивости.
- Не описана совместимость с различными браузерами, разрешениями экранов, устройствами (desktop/mobile).
- Нет требований к логированию действий пользователя или аудиту работы системы.

4. Критерии приёмки и тестирование

Плюсы:

- Чётко указаны количественные показатели успешности тестов ($\geq 95\%$ тест-кейсов).
- Перечислены основные функции, которые должны быть проверены: вычисления, LaTeX-конвертация, экспорт, интерфейс, безопасность.
- Присутствует раздел о модульном, интеграционном, системном и приёмочном тестировании, что полностью покрывает жизненный цикл QA.

Недостатки:

- Не указаны инструменты для тестирования (например, фреймворки для модульных тестов, автоматизации UI-тестов).
- Нет описания сценариев негативного тестирования, кроме общих указаний.

5. Этапы и сроки разработки**Плюсы:**

- План-график разбит по неделям с учётом анализа, проектирования, реализации базового и расширенного функционала, тестирования и приёмки.
- Общая длительность проекта (10 недель) реалистична для учебного или небольшого коммерческого проекта.

Рекомендации по улучшению:

- Можно добавить контрольные точки (milestones) для проверки промежуточных результатов и корректировки требований.
- Уточнить распределение ресурсов, ответственных лиц, а также план резервирования времени на исправление критических багов.

6. Общая оценка**Сильные стороны документации:**

- Полная структура, охватывающая функционал, интерфейс, критерии приёмки и сроки разработки.
- Конкретные количественные показатели для приёмки и тестирования.

- Чёткое разделение функциональных и нефункциональных требований.

Области для улучшения:

1. Уточнить технологии и архитектуру (фронтенд, бэкенд, базы данных, рендеринг формул).
2. Убрать дублирование требований в разделе «интерфейс».
3. Добавить конкретику по безопасности и авторизации.
4. Определить ограничения по платформам, браузерам и устройствам.
5. Расширить сценарии негативного тестирования.
6. Уточнить обработку ошибок и уведомления пользователя.

Вывод:

Документация содержит базовый и расширенный функционал, критерии приёмки и план-график, что делает её достаточно зрелой для старта разработки. Однако для практической реализации рекомендуется добавить технические детали (стек технологий, базы данных, обработка ошибок), уточнения по безопасности и кросс-платформенной совместимости, а также конкретные сценарии негативного тестирования.

Ошибка 1. Неправильный Dockerfile для фронтенд-контейнера

```

bombatcat@work:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
81d3eb0e4701   postgres:17.2  "docker-entrypoint.s..." 19 minutes ago Up About a minute (healthy) 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp testdeploy-p
52b958fffd29c   testdeploy-frontend  "docker-entrypoint.s..." 19 minutes ago Exited (126) About a minute ago testdeploy-f
rontend-1
fa93aa9ac769   nginx:latest   "/docker-entrypoint..." 19 minutes ago Restarting (1) 3 seconds ago testdeploy-n
ginx-1
e752572224ff   api-gateway:latest  "java -jar app.jar"      About an hour ago Up About a minute 0.0.0.0:8082->8082/tcp, [::]:8082->8082/tcp testdeploy-a
pi-gateway-1
6b36deda4571   auth-service:latest  "java -jar app.jar"      About an hour ago Up About a minute 0.0.0.0:8072->8072/tcp, [::]:8072->8072/tcp testdeploy-a
uth-service-1
0a2ccebf6fe7   formula-serivce:latest  "java -jar app.jar"      About an hour ago Up About a minute 0.0.0.0:8071->8071/tcp, [::]:8071->8071/tcp testdeploy-f
ormula-serivce-1
d431658dae72   data-base-service:latest  "java -jar app.jar"      About an hour ago Up 22 seconds 0.0.0.0:8070->8070/tcp, [::]:8070->8070/tcp testdeploy-d
ata-base-service-1
ce147dce68d5   service-server:latest  "java -jar app.jar"      About an hour ago Up About a minute (health: starting) 0.0.0.0:8081->8081/tcp, [::]:8081->8081/tcp testdeploy-s
ervice-server-1
bombatcat@work:~$ docker logs 52b958fffd29c
> avis@0.0.0 dev
> vite

sh: vite: Permission denied

> avis@0.0.0 dev
> vite

sh: vite: Permission denied

> avis@0.0.0 dev
> vite

```

Рисунок N - ошибка с правами доступа

Ошибка `sh: vite: Permission denied` возникает, когда Docker пытается запустить скрипт `vite`, но операционная система контейнера не может выполнить этот файл. Чаще всего это происходит по трём причинам: во-первых, базовый образ (например, `node:18-alpine`) не содержит некоторых системных пакетов или шелл-бинарников, которые необходимы для исполнения скрипта; во-вторых, после копирования проекта в контейнер (`COPY . .`) права на файлы `node_modules/.bin/*` могли измениться, и они больше не исполняемые; в-третьих, `vite` может быть установлен локально в проекте, но путь к нему не прописан, и система не видит исполняемый файл. Для пользователя это выглядит как «ничего не сломалось на этапе сборки», но при запуске контейнера фронтенд падает с непонятной ошибкой.

Ошибка часто неочевидна, потому что `npm install` может завершиться без ошибок, а контейнер собирается успешно. На хосте проект запускается нормально, а в контейнере нет, так как права файлов или отсутствие нужных бинарников влияют только внутри контейнера. Для новичка это выглядит как загадочная «`Permission denied`», хотя на самом деле проблема связана с правами на файлы и отсутствием глобальной установки CLI.

```
# Используем официальный Node.js образ
FROM node:18-alpine

# Устанавливаем рабочую директорию
WORKDIR /app

# Копируем package.json и package-lock.json
COPY package*.json ./

# Устанавливаем зависимости
RUN npm install

# Копируем весь исходный код
COPY . .

# Указываем порт для Vite
EXPOSE 5173

# Запуск сервера разработки
CMD ["npm", "run", "dev"]
```

Рисунок N - неправильный Dockerfile

Вот правильный вариант для фронтенда с Node 18 и vite, который учитывает права на исполнение и гарантирует, что vite будет доступен:

```
# Используем Node 18
FROM node:18-alpine

# Устанавливаем рабочую директорию
WORKDIR /app

# Копируем package.json и package-lock.json
COPY package*.json ./

# Устанавливаем зависимости
RUN npm install

# Копируем весь проект
COPY . .

# Даём права на исполнение vite (если установлен локально)
RUN chmod +x ./node_modules/.bin/vite

# Или устанавливаем vite глобально
RUN npm install -g vite

# Запуск фронтенда
CMD ["npm", "run", "dev"]
```

Рисунок N - правильный Dockerfile

Ошибка 2. Хост не найден в upstream

```
bombatcat@work:~$ docker logs fa93aa9ac769
2025/09/29 15:44:12 [emerg] 1#1: host not found in upstream "api-gateway" in /etc/nginx/nginx.conf:16
nginx: [emerg] host not found in upstream "api-gateway" in /etc/nginx/nginx.conf:16
2025/09/29 15:44:12 [emerg] 1#1: host not found in upstream "api-gateway" in /etc/nginx/nginx.conf:16
nginx: [emerg] host not found in upstream "api-gateway" in /etc/nginx/nginx.conf:16
2025/09/29 15:44:13 [emerg] 1#1: host not found in upstream "api-gateway" in /etc/nginx/nginx.conf:16
```

Рисунок N - Вывод лога о ошибке

Ошибка возникает из-за того, что Nginx не может разрешить имя хоста "api-gateway" в своей конфигурации во время запуска. Это происходит, поскольку Docker Compose запускает контейнеры асинхронно, и сервис nginx может стартовать раньше, чем api-gateway станет доступен в сети. В результате встроенный DNS Docker не может timely предоставить IP-адрес для api-gateway, что приводит к сбою в upstream-блоке nginx.conf. Повторяющиеся записи в логах указывают на политику restart: always, которая заставляет nginx перезапускаться, но без решения корневой проблемы ошибка повторяется.

Второй аспект проблемы — отсутствие динамического разрешения имён в стандартной конфигурации Nginx. По умолчанию Nginx разрешает хосты только один раз при старте, и если api-gateway ещё не готов, разрешение терпит неудачу. Это усугубляется отсутствием явной зависимости в docker-compose.yml для nginx от api-gateway, а также healthcheck, который мог бы подтвердить готовность сервиса перед запуском nginx. В итоге, без этих мер, система попадает в цикл ошибок, препятствуя нормальной работе прокси.

```
nginx:
  image: nginx:latest
  ports:
    - "8080:8080"
  environment:
    API_HOST: api-gateway:8082
    FRONTEND_HOST: frontend:5173
  volumes:
    - ./nginx/nginx.conf.template:/etc/nginx/nginx.conf.template:ro
  command: /bin/bash -c "envsubst '$$API_HOST $$FRONTEND_HOST' < /etc/nginx/nginx.conf.template > /etc/nginx/nginx.conf && nginx -g
  restart: always
```

Рисунок N - неправильный отрывок кода в docker-compose.yml

Чтобы исправить ошибку "host not found in upstream 'api-gateway'" в Nginx, добавим в docker-compose.yml зависимость depends_on для сервиса nginx, указав условие service_healthy для api-gateway и service_started для frontend. Также добавим healthcheck в сервис api-gateway, чтобы проверять его готовность (например, через curl -f http://localhost:8082/actuator/health). В файле

nginx.conf.template включим директиву resolver 127.0.0.11 valid=10s; в блок upstream для динамического разрешения имён. Эти изменения обеспечат запуск nginx только после готовности api-gateway и корректное разрешение его имени.

```
services:
  nginx:
    image: nginx:latest
    ports:
      - "8080:8080"
    environment:
      API_HOST: api-gateway:8082
      FRONTEND_HOST: frontend:5173
    volumes:
      - ./nginx/nginx.conf.template:/etc/nginx/nginx.conf.template:ro
    command: /bin/bash -c "envsubst '$$API_HOST $$FRONTEND_HOST' < /etc/nginx/nginx.conf.template > /etc/nginx/nginx.conf && nginx -g"
    restart: always
    depends_on:
      api-gateway:
        condition: service_healthy
      frontend:
        condition: service_started
    api-gateway:
```

Рисунок N - исправленный код в docker-compose.yml

Ошибка 3. Сбой разрешения имени хоста в блоке upstream Nginx

```
2025/09/29 15:49:57 [emerg] 1#1: host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
nginx: [emerg] host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
2025/09/29 15:50:57 [emerg] 1#1: host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
nginx: [emerg] host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
2025/09/29 15:51:57 [emerg] 1#1: host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
nginx: [emerg] host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
2025/09/29 15:52:57 [emerg] 1#1: host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
nginx: [emerg] host not found in upstream "frontend" in /etc/nginx/nginx.conf:20
```

Рисунок N - Вывод лога о ошибке

Чтобы исправить ошибку "host not found in upstream 'api-gateway'" в Nginx, добавим в docker-compose.yml зависимость depends_on для сервиса nginx, указав условие service_healthy для api-gateway и service_started для frontend. Также добавим healthcheck в сервис api-gateway, чтобы проверять его готовность (например, через curl -f http://localhost:8082/actuator/health). В файле nginx.conf.template включим директиву resolver 127.0.0.11 valid=10s; в блок upstream для динамического разрешения имён. Эти изменения обеспечат запуск nginx только после готовности.

```

services:
  nginx:
    image: nginx:latest
    ports:
      - "8080:8080"
    environment:
      API_HOST: api-gateway:8082
      FRONTEND_HOST: frontend:5173
    volumes:
      - ./nginx/nginx.conf.template:/etc/nginx/nginx.conf.template:ro
    command: /bin/bash -c "envsubst '$$API_HOST $$FRONTEND_HOST' < /etc/nginx/nginx.conf.template > /etc/nginx/nginx.conf && nginx"
    restart: always

```

Рисунок N - неправильный отрывок кода в docker-compose.yml

В docker-compose.yml добавлена зависимость `depends_on` для сервиса `nginx` с условием `service_healthy` для `frontend` и `api-gateway`, что гарантирует запуск `nginx` только после их готовности. Для `frontend` введён `healthcheck`, проверяющий доступность порта 5173 через `curl -f http://localhost:5173`, чтобы Docker Compose дождался полной готовности сервиса. В конфигурации Nginx (файл `nginx.conf.template`) добавлена директива `resolver 127.0.0.11 valid=10s`;, позволяющая динамически обновлять IP-адреса сервисов через DNS Docker, устраняя проблему статического разрешения имён.

```

services:
  nginx:
    image: nginx:latest
    ports:
      - "8080:8080"
    environment:
      API_HOST: api-gateway:8082
      FRONTEND_HOST: frontend:5173
    volumes:
      - ./nginx/nginx.conf.template:/etc/nginx/nginx.conf.template:ro
    command: /bin/bash -c "envsubst '$$API_HOST $$FRONTEND_HOST' < /etc/nginx/nginx.conf.template > /etc/nginx/nginx.conf && nginx -g"
    restart: always
    depends_on:
      api-gateway:
        condition: service_healthy
      frontend:
        condition: service_healthy
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "5173:5173"
    volumes:
      - ./frontend:/app
      - /app/node_modules
    stdin_open: true
    tty: true
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:5173"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s

```

Рисунок N - исправленный код в docker-compose.yml

ЗАКЛЮЧЕНИЕ

Подготовленное техническое задание демонстрирует хороший уровень проработки: основные пользовательские сценарии описаны, функциональность разбита по блокам, указаны требования к удобству интерфейса, надежности и кроссплатформенности. Наличие количественных критериев, таких как ограничение времени на оформление заказа или процент успешного прохождения тестов, является сильной стороной документации, так как позволяет проверять соответствие продукта заявленным ожиданиям. Вместе с тем в документе отсутствуют важные уточнения — нет конкретных требований к сложности пароля, не описаны поиск и фильтрация в каталоге, не определён процесс оплаты, а также не предусмотрен административный функционал для управления товарами и заказами. Требования к производительности и нагрузке сформулированы недостаточно, а архитектурные детали лучше вынести в отдельный раздел. Несмотря на эти недочёты, документация можно считать хорошей основой для учебного проекта.

Реализация продукта на текущем этапе частично соответствует заявленным требованиям, но содержит значительные недоработки. В ходе тестирования выявлен ряд критических ошибок: некорректная работа кнопок смены пароля и удаления профиля, отсутствие отображения даты регистрации, проблемы с сохранением данных в личном кабинете. В корзине обнаружены ошибки бизнес-логики — дублирование товаров при добавлении, невозможность удаления позиции при уменьшении количества до нуля, некорректное обновление страницы с отображением «сырого» JSON, отсутствие очистки корзины после оформления заказа и сохранения заказа в истории. Также не работает кнопка подгрузки дополнительных товаров в каталоге. Эти дефекты затрагивают ключевые пользовательские сценарии, без которых интернет-магазин не может считаться полнофункциональным.

Таким образом, продукт лишь частично соответствует техническому заданию и пока не удовлетворяет критериям приёмки. Основные сценарии реализованы

формально, но функционируют с критическими сбоями. Чтобы довести проект до приемлемого уровня качества, необходимо устранить выявленные ошибки, доработать и расширить ТЗ (поиск и фильтрация товаров, требования к паролям, админ-функционал, имитация процесса оплаты, базовые показатели производительности), а также провести повторное тестирование с включением проверки безопасности.

В целом команда проделала серьёзную работу и создала хорошую базу для учебного проекта. Однако до итоговой приёмки и соответствия заявленным требованиям требуется существенная доработка функционала и устранение критических багов. После этого продукт сможет продемонстрировать более высокий уровень качества и соответствие заявленным ожиданиям.