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 сетевые технологии**

Направление подготовки (специальность) **09.03.03 Прикладная информатика**

К У Р С О В О Й П Р О Е К Т

по дисциплине «Инфокоммуникационные системы и технологии»

на тему:
Разработка мобильного приложения для микрообучения студентов по различным образовательным дисциплинам

Обучающийся Юшков Андрей Михайлович К3139

Работа сдана
Дата 05.01.2025

Санкт-Петербург 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное образование претерпевает значительные изменения под воздействием новых технологий и методов обучения. В условиях стремительного прогресса в области информационных технологий и увеличения доступности интернет-ресурсов, традиционные формы обучения постепенно уступают место более интерактивным и гибким подходам. Одним из таких подходов является использование мобильных приложений, которые позволяют студентам учиться в удобное время и в удобном месте.

В частности, для студентов Санкт-Петербургского государственного университета информационных технологий, механики и оптики (ИТМО) создание специализированной платформы для обучения может стать важным шагом к улучшению качества образовательного процесса. Немаловажную роль в образовании играет самообучение, поэтому мобильное приложение, которое интегрирует видеоуроки и тесты по различным темам, может значительно повысить уровень вовлеченности студентов, а также облегчить процесс усвоения материала.

Целью данного проекта является разработка мобильного приложения для обучения студентов ИТМО, которое будет включать в себя интерактивные видеоуроки и тесты, позволяющие пользователям эффективно осваивать учебный материал. Приложение должно обеспечить доступ к образовательным ресурсам, способствовать самопроверке знаний и предоставлять возможность обратной связи.

Для достижения указанной цели были поставлены следующие задачи:

- Подготовить и утвердить техническое задание
- Определить необходимый функционал, на основании которого создать прототип дизайна мобильного приложения.
- По результатам анализа функциональных требований и предметной области подготовить план для дальнейшей работы фронтенда и бекенда
- Разработка всей минимально необходимой части для бекенда
- Аналитика необходимой работы для фронтенда
- По макетам аналитика создать фронтенд часть, необходимую для работы приложения
- Объединить фронтенд и бекенд
- Провести тестирование продукта
- Создать презентацию для дальнейшей защиты

1 Основная часть

Суть проекта заключается в разработке мобильного приложения для микрообучения, которое позволяет пользователям быстро усваивать знания, или повторять уже пройденные темы через обучающие видео и тесты для подготовки к экзаменам и тестам, а также для увеличения их знаний в соответствующих областях. Для выполнения данных задач необходимо было сделать интуитивно понятный интерфейс и налаженную работу самого приложения, в связи с чем были использованы следующие технологии:

1) Kotlin - современный статически типизированный язык программирования, разработанный для повышения продуктивности разработчиков на платформе Java, поддерживающий функциональный и объектно-ориентированный подход.
2) .NET - платформа разработки от Microsoft, позволяющая создавать приложения для Windows, веб-сервисы и мобильные приложения с использованием различных языков программирования, таких как C# и VB.NET.
3) Figma - облачный инструмент для дизайна интерфейсов и прототипирования, позволяющий командам совместно работать над проектами в реальном времени.
4) PostgreSQL - мощная объектно-реляционная система управления базами данных с открытым исходным кодом, известная своей надежностью, расширяемостью и поддержкой сложных запросов.
5) Android Studio - официальная интегрированная среда разработки (IDE) для создания приложений под операционную систему Android, предоставляющая инструменты для написания кода, отладки и тестирования приложений.

2 Работа над проектом

Процессы работы над проектом.

В начале работы над проектом-приложением командой был составлен план разработки:

1) По желанию и способностям участников были распределены роли
2) В соответствии с ролями были распределены задачи и установлены сроки их выполнения
3) Выполнение задач
4) Доработка приложения до рабочего состояния
5) Подготовка к защите проекта
6) Командная защита проекта
7) Написание индивидуального отчета по работе

Задачи, которые мне было необходимо выполнить

Моя роль в команде была обозначена как технический писатель. В мои обязанности входило исследование и анализ технологий, необходимых для разработки приложения, написание кода для реализации отдельных функций и согласование с остальными участниками команды этих функций, а также создание документации.

Решение поставленных задач

В ходе работы над проектом перед мной стояло несколько задач: Поскольку наша платформа для микрообучения включает в себя обучающие видео мне предстояло найти наиболее оптимальный способ реализации трансляции видео

В ходе изучения статей я выделил несколько технологий:

1. Video on Demand (VOD):

   Описание: Позволяет пользователям просматривать заранее загруженные видео в любое время.

Технологии: Используются стриминговые серверы и протоколы, такие как HLS (HTTP Live Streaming) или MPEG-DASH.

Примеры приложений: Netflix, YouTube, Vimeo.

2. Мобильные приложения с поддержкой VOD:

   Описание: Разработка мобильных приложений, которые позволяют загружать видео на сервер и затем воспроизводить их по запросу.

   Технологии: Использование фреймворков, таких как React Native или Flutter, для создания кроссплатформенных приложений.

   Примеры: Приложения для онлайн-курсов, такие как Udemy или Coursera.

3. Системы управления контентом (CMS):

   Описание: Использование CMS, таких как WordPress с плагинами для видео, которые позволяют загружать и управлять видео-контентом.

   Примеры плагинов: WP Video Lightbox, Presto Player.

4. SaaS-платформы:

   Описание: Использование готовых SaaS-решений для хостинга и стриминга видео, которые предоставляют API для интеграции с мобильными приложениями.

   Примеры: Vimeo, Brightcove, JW Player.

5. Облачные решения:

   Описание: Хранение видео в облачных сервисах с последующим воспроизведением через мобильное приложение.

   Технологии: Использование AWS S3 для хранения и AWS CloudFront для доставки контента.

Примеры: Приложения, использующие облачные решения для хранения и потоковой передачи.

6. Потоковая передача через P2P:

   Описание: Использование технологий P2P для передачи видео, что может снизить нагрузку на сервер.

   Технологии: WebRTC для передачи видео между пользователями.

   Примеры: Приложения, использующие P2P для обмена видео.

7. Интеграция с социальными сетями:

   Описание: Использование API социальных сетей для интеграции с мобильным приложением, позволяя пользователям загружать и пересматривать видео.

   Примеры: Instagram, Facebook, TikTok.

8. Кастомные видеоплееры:

   Описание: Разработка кастомных видеоплееров для мобильных приложений, которые могут воспроизводить заранее загруженные видео.

   Технологии: Использование библиотек, таких как ExoPlayer для Android или AVPlayer для iOS.

В результате я пришел к выводу что оптимально будет реализовать трансляцию видео передаваемое с сервера на устройство при RTMP (Real Time Messaging Protocol) в сочетании с простым потоковым сервером, таким как Nginx (с модулем nginx-rtmp). Однако в ходе работы выяснилось, что команда не имеет достаточно времени для реализации данной технологии, в результате чего было принято решение реализовать трансляцию видео с помощью youtube-плеера внутри андроид приложения

Далее мне было необходимо реализовать код для подсчета процента выполнения заданий по курсу для этого я использовал язык с, вот пошаговое объяснение выполнения данной задачи

Заголовочные файлы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
```

#include <stdio.h>: Подключает стандартную библиотеку ввода-вывода, которая позволяет использовать функции для работы с консолью, такие как printf и scanf.

#include <stdlib.h>: Подключает стандартную библиотеку, которая включает функции для работы с памятью, преобразованиями и другими утилитами.

#include <string.h>: Подключает библиотеку для работы со строками, позволяющую использовать функции для копирования, сравнения и манипуляции строками.

#include <stdbool.h>: Подключает библиотеку, которая позволяет использовать тип данных bool для логических значений (true и false).

Определение констант

```
#define max_courses 10
#define max_lessons 50
#define max_users 100
```

Эти строки определяют максимальные размеры для курсов, уроков и пользователей в системе. Это позволяет легко изменять размеры в одном месте, если потребуется.

Определение структур

```
typedef struct {
```

```
    int lesson_id;
    char lesson_title[100];
    bool completed;
} lesson;
```

typedef struct {...} lesson;: Определяет структуру lesson, которая представляет собой урок.

int lesson_id;: Идентификатор урока.

char lesson_title[100];: Массив символов для хранения названия урока (до 99 символов + 1 для завершающего нуля).

bool completed;: Логическая переменная, указывающая, завершен ли урок.

```
typedef struct {
    int course_id;
    char course_name[100];
    lesson lessons[max_lessons];
    int total_lessons;
} course;
```

typedef struct {...} course;: Определяет структуру course, представляющую курс.

int course_id;: Идентификатор курса.

char course_name[100];: Название курса.

lesson lessons[max_lessons];: Массив уроков, который содержит все уроки, входящие в курс.

int total_lessons;: Общее количество уроков в курсе.

```
typedef struct {
    int user_id;
    char username[50];
    bool completed_courses[max_courses];
    int total_courses;
```

} user;

typedef struct {...} user;: Определяет структуру user, представляющую пользователя.

int user_id;: Идентификатор пользователя.

char username[50];: Имя пользователя.

bool completed_courses[max_courses];: Массив, указывающий, завершены ли курсы пользователем (по одному на каждый курс).

int total_courses;: Общее количество курсов, в которых зарегистрирован пользователь.

```
typedef struct {
    user users[max_users];
    course courses[max_courses];
    int total_users;
    int total_courses;
    bool completed_tasks[max_users][max_courses]; // Двумерный массив для отслеживания выполнения заданий
} learningplatform;
```

typedef struct {...} learningplatform;: Определяет структуру learningplatform, представляющую платформу для обучения.

user users[max_users];: Массив пользователей.

course courses[max_courses];: Массив курсов.

int total_users;: Общее количество пользователей на платформе.

int total_courses;: Общее количество курсов на платформе.

bool completed_tasks[max_users][max_courses];: Двумерный массив, где строки представляют пользователей, а столбцы — курсы. Каждая ячейка указывает, завершил ли пользователь задание в соответствующем курсе.

Функция добавления пользователя

void adduser(learningplatform *platform, const char *username) {

Определяет функцию adduser, которая принимает указатель на структуру learningplatform и строку username.

if (platform->total_users < max_users) {

Проверяет, не превышает ли текущее количество пользователей максимальное количество.

user *user = &platform->users[platform->total_users++];

Если место есть, добавляет нового пользователя в массив users и увеличивает total_users.

user->user_id = platform->total_users;

Присваивает пользователю идентификатор, равный текущему количеству пользователей.

strncpy(user->username, username, sizeof(user->username) - 1);

user->username[sizeof(user->username) - 1] = '\0';

Копирует имя пользователя в массив username, гарантируя, что оно не превышает размер массива и добавляет завершающий ноль.

user->total_courses = 0;

memset(user->completed_courses, 0, sizeof(user->completed_courses));

Инициализирует количество курсов пользователя в 0 и обнуляет массив completed_courses.

memset(platform->completed_tasks[platform->total_users - 1], 0, sizeof(platform->completed_tasks[0])); // Инициализация

Инициализирует строку в двумерном массиве completed_tasks для нового пользователя значениями false.

} else {

printf("Максимальное количество пользователей.\n");

}

}

Если максимальное количество пользователей уже достигнуто, выводит сообщение об ошибке.

Функция добавления курса

void addcourse(learningplatform *platform, const char *course_name, int total_lessons) {

Определяет функцию addcourse, которая принимает указатель на структуру learningplatform, название курса и общее количество уроков.

if (platform->total_courses < max_courses && total_lessons <= max_lessons && total_lessons > 0) {

Проверяет, можно ли добавить новый курс (не превышает ли общее количество курсов максимальное количество и корректно ли указано количество уроков).

course *course = &platform->courses[platform->total_courses++];

Если место есть, добавляет новый курс в массив courses и увеличивает total_courses.

course->course_id = platform->total_courses;

Присваивает курсу идентификатор, равный текущему количеству курсов.

strncpy(course->course_name, course_name, sizeof(course->course_name) - 1);

course->course_name[sizeof(course->course_name) - 1] = '\0';

Копирует название курса в массив course_name, гарантируя, что оно не превышает размер массива и добавляет завершающий ноль.

course->total_lessons = total_lessons;

Устанавливает общее количество уроков в курсе.

for (int i = 0; i < total_lessons; i++) {

course->lessons[i].lesson_id = i + 1;

sprintf(course->lessons[i].lesson_title, "Урок %d", i + 1);

course->lessons[i].completed = false;

```
        }
```
В цикле инициализирует каждый урок в курсе, устанавливая его идентификатор, название и статус завершения (не завершен).
```
    } else {
        printf("Ошибка: недопустимое количество уроков.\n");
    }
}
```
Если есть ошибка при добавлении курса, выводит сообщение об ошибке.

Функция завершения урока

```
void completelesson(learningplatform *platform, int user_id, int course_id, int lesson_id) {
```
Определяет функцию completelesson, которая принимает указатель на структуру learningplatform, идентификатор пользователя, идентификатор курса и идентификатор урока.
```
    if (user_id < 1 || user_id > platform->total_users || course_id < 1 || course_id > platform->total_courses || lesson_id < 1 || lesson_id > platform->courses[course_id - 1].total_lessons) {
```
Проверяет, что все переданные идентификаторы находятся в допустимых пределах.
```
        printf("Ошибка: неверные данные.\n");
        return;
    }
```
Если данные неверные, выводит сообщение об ошибке и завершает выполнение функции.
```
    course *course = &platform->courses[course_id - 1];
    lesson *lesson = &course->lessons[lesson_id - 1];
```
Получает указатели на курс и урок, которые пользователь завершает.

```
    lesson->completed = true;
            platform->completed_tasks[user_id - 1][course_id - 1] = true; //
```
Обновляем статус выполнения задания

Устанавливает статус урока как завершенный и обновляет соответствующую ячейку в двумерном массиве completed_tasks.

```
    bool all_lessons_completed = true;
    for (int i = 0; i < course->total_lessons; i++) {
        if (!course->lessons[i].completed) {
            all_lessons_completed = false;
            break;
        }
    }
```
Проверяет, завершены ли все уроки в курсе. Если есть хотя бы один незавершенный урок, устанавливает

флаг all_lessons_completed в false.

```
    if (all_lessons_completed) {
        platform->users[user_id - 1].completed_courses[course_id - 1] = true;
    }
}
```
Если все уроки завершены, обновляет статус завершения курса для пользователя.

Функция вычисления завершенности курса

```
float calculatecoursecompletion(const course *course) {
```
Определяет функцию calculatecoursecompletion, которая принимает указатель на курс и возвращает процент завершенности.

```
    int completed_lessons = 0;
    for (int i = 0; i < course->total_lessons; i++) {
        if (course->lessons[i].completed) {
            completed_lessons++;
```

```
        }
    }
```
Считает количество завершенных уроков в курсе.
```
    if (course->total_lessons == 0) return 0.0f;
    return (float)completed_lessons / course->total_lessons * 100;
}
```
Если в курсе нет уроков, возвращает 0. В противном случае вычисляет и возвращает процент завершенных уроков.

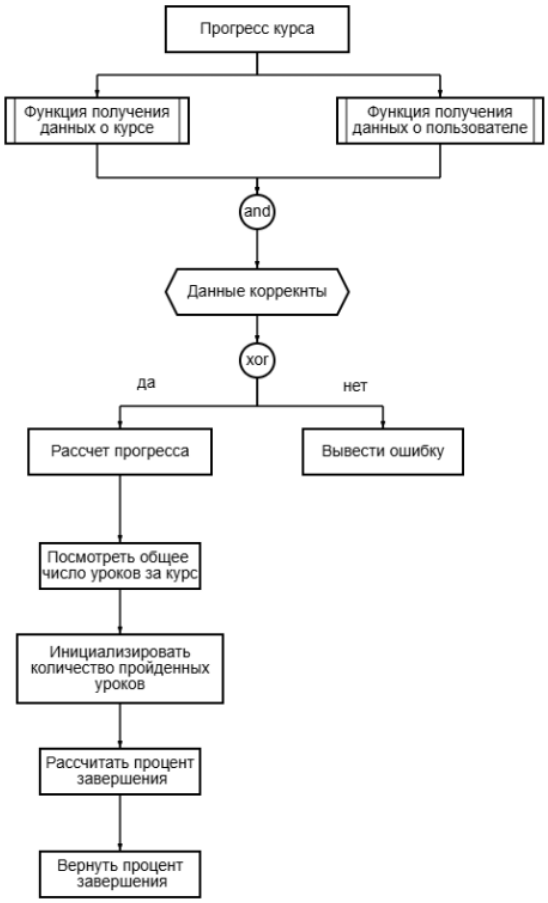
Функция отображения прогресса пользователя
```
void displayprogress(const learningplatform *platform, int user_id) {
```
Определяет функцию displayprogress, которая принимает указатель на структуру learningplatform и идентификатор пользователя.
```
    if (user_id < 1 || user_id > platform->total_users) {
        printf("Ошибка: неверный номер пользователя.\n");
        return;
    }
```
Проверяет, что идентификатор пользователя находится в допустимых пределах. Если нет, выводит сообщение об ошибке и завершает выполнение функции.
```
    const user *user = &platform->users[user_id - 1];
    printf("Прогресс пользователя '%s':\n", user->username);
```
Получает указатель на пользователя и выводит его имя.
```
    for (int i = 0; i < platform->total_courses; ++i) {
        float completion_percentage = calculatecoursecompletion(&platform-
>courses[i]);
        printf("Курс '%s' - завершен на %.2f%%\n", platform-
>courses[i].course_name, completion_percentage);
    }
}
```

```
    return 0;
}
```

Завершает выполнение программы и возвращает 0, что указывает на успешное завершение.

В ходе выполнения работы возникли работы возникли проблемы с изучениям нового для меня языка программирования, а также с разделением пользователей, данная проблема решилась созданием двумерного массива и массива числа заданий в каждом курсе

Последним заданием для меня стало создание диаграммы по данному коду



Проблемой стало незнание правильного построения таблиц, с чем мне помог мой руководитель проекта, скинув ссылки на необходимые статьи

Анализ моей работы

Я считаю что с данными мне задачами я справился, я создал рабочий код и таблицу, описывающую его работу, однако есть места, которые можно было улучшить, к ним я бы отнес то, что я должен был учесть возможности товарищей по команде и сразу предложить наиболее простой способ добавления видео в приложение, а так же при большем опыте я бы мог написать более быстрый код.

В ходе работы я научился базовому программированию на си, узнал способы передачи и трансляции видео, а так же я научился строить таблицы

3 Коммуникация в команде

Работа в команде была важной частью столь сложного проекта, в котором нужно было обсуждать работу между дизайнерами, разработчиками и аналитиками

С самого начала мы провели несколько встреч, где подробно обсудили цели проекта, распределили роли и задачи, а так же утвердили этапы работы. В ходе работы над проектом коммуникация происходила посредством совещаний два раза в неделю в платформе Zoom, моменты работы, которые касались не всего коллектива происходила в чатах Telegramm. Все активно коммуницировали предлагали новые идеи и решения.

Руководитель проекта играла ключевую роль в сплочении команды. Она четко формулировала задачи и сроки выполнения, помогала при возникновении трудностей, при просмотре моей таблицы я получил конструктивную критику, после чего исправил недостатки

Оценка работы руководителя

Руководитель грамотно формулировала задачи, поддерживала командный дух. Она проявила себя в качестве хорошего лидера с высокими управленческими и социальными навыками. Она приветствовала новые идеи

и всегда была рада обратной связи. Работа над проектом была продуктивной благодаря отличному взаимодействию с командой и поддержке руководителя. Я многому научился наблюдая за своим руководителем. В связи со всем вышеперечисленным, я считаю что мой руководитель заслуживает наивысшей оценки.

Заключение

Наша команда успешно справилась с поставленной задачей. Мы смогли создать необходимый функционал для работы приложения по микрообучению, в нем реализована функция регистрации и входа в аккаунт, просмотр видеоматериалов и просмотр данных по курсам

Весь необходимый функционал в приложении присутствует, однако от некоторых идей пришлось отказаться или упростить, например регистрация по ITMO id, трансляция видео с сервера, тесты по курсам и процедура рассчета, от этих идей отказались в виду ограничения по времени и ресурсам

Я проделал немалую работу и считаю что моя работа была полезна, однако, так как я присоединился чуть позже остальных часть моей работы не вошла в конечный результат в виду ограничения по времени, необходимое для создания приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Видео из YouTube в андроид приложении https://ru.stackoverflow.com/questions/438333/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE-%D0%B8%D0%B7-youtube-%D0%B2-%D0%B0%D0%BD%D0%B4%D1%80%D0%BE%D0%B8%D0%B4-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B8
2) Настройка nginx https://www.8host.com/blog/nastrojka-servera-video-striminga-na-nginx-rtmp/
3) Варианты воспроизведения видео https://flussonic.ru/doc/dash-protocol/
4) Правильное создание таблиц https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/epc_notation

ПРИЛОЖЕНИЕ

А Техническое задание Название проекта: Мобильное приложение для микрообучения. Цель: разработать мобильное приложение для микрообучения студентов по различным образовательным дисциплинам.

Сроки выполнения:

Начало 01.11.2024

Окончание 20.12.2024

Исполнитель проекта:

Ганичева Лидия Сергеевна.

Термины и сокращения:

МП – Мобильное приложение;

ТД – Тестовые данные;

СУБД – Система управления базам данных;

ТЗ – Техническое задание;

ФТ – Функциональные требования;

Технические требования:

Разработка фронт части МП HTML (для создания базовой структуры страниц и контента), CSS (для стилизации внешнего вида) и JavaScript (для добавления интерактивности), любой другой язык программирования для верстки на усмотрение разработчика и команды.

Разработка бэк части МП Java/PHP/ Ruby/ C#/ любой другой язык программирования на усмотрение разработчика PostgreSQL/MySQL / и т. д.

Функциональные требования:

Безопасность хранения паролей, при обработке паролей от пользователей необходимо использовать хэш функцию для шифрования и хранить в БД зашифрованное поле.

Регистрация и авторизация пользователя в системе.

Перечень курсов и программ отображаются и кликабельны для пользователей.

Подсчет статистики и прогресса по курсу у каждого пользователя по всем его открытым курсам.

Этапы работы:

Подготовка к разработке МП;

Работы аналитика по реализации ФТ;

Разработка бэка МП;

Разработка фронта МП;

Администрирование инфраструктуры для работы МП;

Тестирование продукта после реализации;

Управление командой разработки;

Задачи проекта:

Проанализировать предметную область (стек). Необходимо на опыте (статьи хабр и т.д.) аналогов выяснить с какими трудностями столкнулись команды при работе с определенным стеком. По окончанию анализа

совместно с командой обсудить предлагаемый стек, зафиксировать договорённость по стеку;

Проанализировать предметную область (продукт). Необходимо провести ресечь аналогов (в части интерфейса МП), собрать пулл наилучший практик по реализации дизайна МП. Зафиксировать результаты в файле, базе знаний;

Разработать дизайн МП. Необходимо по результатам предыдущей задачи подготовить макеты для экранов МП. Зафиксировать макеты в файле, ссылке на фигму.

Подготовить постановку на витрины данных. По результатам анализа функциональных требований и предметной области подготовить постановку на необходимые витрины данных (атрибутивный состав, типы данных, констрэйны и т.д.). Постановку отобразить в отдельном файле, удобном для чтения разработчика, реализации витрин.

Создать витрины данных. Необходимо по постановке аналитика создать скрипты для витрин БД. Создать таблицы, подготовить совместно с аналитиком тестовые данные, заинсертить ТД в витрины.

Разработка бэка МП. Необходимо развернуть БД, докер. Подготовить инфраструктуру для работы команды

Разработать бэк. Разработать все необходимые процедуры для бэк части МП;

Сверстать фронт. По макетам аналитика сверстать фронт часть МП;

Объединить фронт и бэк. После верски объединить фронт и бэк приложения.

Тестирование продукта. В ходе работы проводить поэтапное тестирование продукта;

Подготовка презентации на защиту проекта. Подготовить презентацию для защиты проекта, собрать основные результаты, отобразить их, согласовать с командой и руководителем;

Минимальные функциональные требования к приложению:

Регистрация и вход в систему;

Перечень курсов и программ;

Подсчет статистики и прогресса по курсу;

Основные результаты работы:

1. Аналитика

Проведен ресечь аналогов, баз знаний, выбран стек технологий для работы команды

Проанализированы ФТ

Подготовлена Постановка на витрины данных

2. Дизайн

Проведен ресечь аналогов, выбраны лучшие примеры реализации

Подготовлены макеты для МП

3. Фронт МП

Сверстан фронт МП

Фронт и бэк МП соеденен, обмен информацией работает корректно

4. Бэк МП

Развернута необходимая инфраструктура для работы МП

Созданы витрины данных

Прописан код для работы бэка

5. Тестирование

Проведено тестирование МП, функциональность полностью реализует ФТ

Устранены дефекты разработки

6. Отчёты

Разработано МП по минимальным ФТ

Подготовлена презентация по результатам работы команды