### **Платформа для помощи студентам с гибким посещением**

#### **Описание проекта**

Данная платформа предназначена для студентов с гибким графиком посещения занятий. Она помогает организовать расписание, взаимодействовать с преподавателями, отслеживать оценки, задолженности и получать своевременные уведомления.

#### **Основные функции**

1. **Личный кабинет студента**
   * Просмотр персональных данных (имя, номер студенческого билета, группа).
   * Отображение текущего расписания.
   * История посещений и выполненных заданий.
   * Получение уведомлений об изменениях в расписании, задолженностях и оценках.
2. **Управление расписанием**
   * **Для студентов:**
     + Просмотр доступных занятий.
     + Запись на занятия в удобное время.
     + Уведомления о добавленных или отменённых занятиях.
   * **Для преподавателей:**
     + Добавление новых занятий.
     + Установка расписания консультаций.
     + Контроль посещаемости студентов.
3. **Запись на консультации**
   * Возможность студента выбрать удобное время для консультации у преподавателя.
   * Ограничение количества записей на одно время.
   * Уведомление преподавателя о новых записях.
4. **Отслеживание задолженностей**
   * Отображение списка задолженностей по дисциплинам.
   * Интеграция с системой оценки для отображения необработанных или незачтённых работ.
5. **Оценки**
   * Просмотр текущих и итоговых оценок по дисциплинам.
   * Доступ преподавателей для внесения или изменения оценок.
   * Уведомления студентам о новых оценках.
6. **Роли и доступ**
   * **Роли пользователей:**
     + Студенты: доступ к расписанию, записи на занятия и консультации.
     + Преподаватели: управление расписанием, внесение оценок, создание уведомлений.
     + Администраторы: управление пользователями и системой.
   * Настройка прав доступа для каждой роли.
7. **Уведомления**
   * Отправка уведомлений через email и мессенджеры (Telegram, WhatsApp):
     + Изменения в расписании.
     + Напоминания о консультациях и заданиях.
     + Новые оценки и задолженности.
8. **Дополнительные функции**
   * Интеграция с API учебных заведений для автоматической синхронизации данных.
   * Экспорт расписания в календарь Google или Outlook.
   * Тёмная/светлая тема интерфейса.

#### **Технологический стек**

1. **Frontend**:
   * **React.js** или **Vue.js**: для создания динамичного и отзывчивого интерфейса.
   * **CSS Framework**: TailwindCSS или Material-UI для стилизации.
2. **Backend**:
   * **Spring Boot** (Java): надёжная и производительная архитектура для реализации REST API.
   * **Node.js** (опционально): для уведомлений в реальном времени (websockets).
3. **База данных**:
   * **PostgreSQL**: реляционная база данных для хранения информации о студентах, занятиях и расписании.
4. **Интеграции**:
   * **Email**: SMTP (например, SendGrid, Mailgun).
   * **Мессенджеры**: Telegram Bot API, Twilio для WhatsApp.
5. **Развёртывание**:
   * **Docker**: для контейнеризации приложения.
   * **Heroku/AWS/GCP**: для хостинга.

#### **Архитектура**

1. **Frontend:**
   * Модуль личного кабинета.
   * Модуль расписания.
   * Модуль уведомлений.
2. **Backend:**
   * Сервис управления пользователями (студенты, преподаватели).
   * Сервис расписания и записи на занятия.
   * Сервис уведомлений.
   * Сервис аналитики (статистика посещаемости, задолженности).
3. **База данных:**
   * Таблицы:
     + Пользователи (id, роль, имя, email).
     + Занятия (id, название, время, место).
     + Оценки (id, студент\_id, предмет\_id, оценка).
     + Задолженности (id, студент\_id, предмет\_id, дата).

#### **Пользовательские сценарии**

1. **Студент:**
   * Входит в личный кабинет.
   * Просматривает расписание и записывается на занятия.
   * Проверяет свои задолженности и оценки.
   * Записывается на консультацию.
2. **Преподаватель:**
   * Входит в личный кабинет.
   * Создаёт занятия и устанавливает время консультаций.
   * Вносит оценки и отмечает посещаемость.
3. **Администратор:**
   * Управляет пользователями и расписанием.
   * Настраивает систему уведомлений.

#### **Пример интерфейса**

1. **Главная страница студента**:
   * Блок расписания (текущие занятия и доступные консультации).
   * Блок уведомлений (предстоящие события и изменения).
   * Блок задолженностей и оценок.
2. **Панель управления преподавателя**:
   * Календарь для управления расписанием.
   * Список студентов с возможностью добавления оценок.
3. **Уведомления**:
   * Централизованный модуль с настройками (включение/выключение типов уведомлений).

### **План разработки платформы для студентов с гибким посещением**

#### **1. Этапы разработки**

### **Этап 1: Подготовка**

1. **Анализ требований**
   * Уточнить все функциональные требования к системе.
   * Собрать информацию о предпочтениях пользователей (студенты, преподаватели, администрация).
2. **Выбор технологий**
   * Утвердить стек технологий: **Spring Boot** для backend, **React.js** или **Vue.js** для frontend, **PostgreSQL** для базы данных.
3. **Проектирование архитектуры**
   * Разработать схемы баз данных.
   * Спроектировать архитектуру системы (разделение на frontend и backend, модули).
4. **Среда разработки**
   * Настроить рабочую среду (IDE, Git, CI/CD).
   * Определить хостинг для тестирования (например, Heroku или локальный сервер).

### **Этап 2: Backend разработка**

#### **2.1. Подготовка базы данных**

* Создать таблицы:
  + **users** (id, роль, имя, email, пароль, и т.д.).
  + **schedule** (id, название занятия, дата, время, преподаватель\_id).
  + **grades** (id, студент\_id, предмет\_id, оценка).
  + **debts** (id, студент\_id, задолженность).
  + **notifications** (id, текст, дата\_создания, пользователь\_id).
* Настроить связи между таблицами.

#### **2.2. Реализация API**

* Настроить Spring Boot приложение с REST API.
* Реализовать основные эндпоинты:
  + **/api/users**: управление пользователями (CRUD).
  + **/api/schedule**: создание и получение расписания.
  + **/api/grades**: добавление и просмотр оценок.
  + **/api/debts**: отслеживание задолженностей.
  + **/api/notifications**: отправка и получение уведомлений.

#### **2.3. Реализация авторизации и ролей**

* Настроить JWT авторизацию.
* Реализовать роли:
  + Студент: доступ только к расписанию, консультациям и оценкам.
  + Преподаватель: доступ к расписанию, внесению оценок и консультациям.
  + Администратор: полный доступ к системе.

#### **2.4. Тестирование backend**

* Написать unit-тесты для каждого модуля.
* Настроить Postman коллекцию для проверки API.

### **Этап 3: Frontend разработка**

#### **3.1. Дизайн интерфейса**

* Создать прототипы UI:
  + Главная страница студента (расписание, уведомления, задолженности).
  + Панель преподавателя (управление расписанием, оценки студентов).
  + Административная панель (управление пользователями и системой).
* Утвердить дизайн с заказчиком.

#### **3.2. Реализация компонентов**

* Разработать главные страницы:
  + **Личный кабинет студента.**
  + **Расписание.**
  + **Задолженности и оценки.**
* Реализовать авторизацию и регистрацию.

#### **3.3. Интеграция с API**

* Настроить взаимодействие frontend с backend через REST API.
* Добавить обработку ошибок (например, истёкший токен или отсутствие доступа).

#### **3.4. Тестирование интерфейса**

* Проверить корректность отображения данных.
* Убедиться в удобстве навигации и дизайна.

### **Этап 4: Уведомления**

1. **Email уведомления**
   * Настроить SMTP сервер.
   * Реализовать отправку email при:
     + Записи на консультацию.
     + Добавлении новых оценок.
     + Напоминании о задолженностях.
2. **Интеграция с мессенджерами**
   * Настроить Telegram Bot API для отправки уведомлений.
   * Добавить поддержку WhatsApp через Twilio.

### **Этап 5: Деплой**

1. **Контейнеризация**
   * Настроить Docker для backend и базы данных.
   * Создать Docker-Compose для локального запуска.
2. **Развёртывание**
   * Выбрать хостинг:
     + AWS, GCP, или Heroku для тестирования.
   * Настроить CI/CD pipeline.
3. **Тестирование производительности**
   * Проверить скорость отклика API.
   * Провести нагрузочные тесты.

### **Этап 6: Завершение**

1. **Документация**
   * Подготовить документацию по API.
   * Написать руководство для пользователей.
2. **Обучение**
   * Провести обучение для администраторов и преподавателей.

### **Ожидаемые результаты**

* Полностью функционирующая платформа.
* Удобный интерфейс для студентов, преподавателей и администраторов.
* Возможность отправки уведомлений через email и мессенджеры.
* Безопасная авторизация и разграничение ролей.