Создать платформу, которая упрощает проверку программного кода, автоматизирует взаимодействие между учениками и преподавателями, способствует повышению качества обучения и экономит время преподавателей.

**Цели проекта (Первая версия)**

1. Автоматизация проверки программного кода (с акцентом на Python).
2. Организация хранения и управления архивом проверенных работ.
3. Поддержка трёх ролей пользователей: ученики, преподаватели и разработчики.
4. Упрощение работы преподавателей через управление группами, заданиями и проверкой кода.
5. Предоставление обратной связи ученикам для улучшения их знаний и навыков.

**Функционал системы в первой версии**

**Для учеников**

1. Присоединение к учебным группам через уникальный пароль, выданный преподавателем.
2. Получение заданий:
   * Групповых (выдаются всем членам группы).
   * Индивидуальных (назначаются вручную или автоматически).
3. Загрузка решений через интерфейс Telegram.
4. Получение обратной связи:
   * Результаты синтаксической проверки.
   * Результаты выполнения программы на заданных тестах.
   * Рекомендации по доработке.
5. Получение дополнительных заданий:
   * При незачёте или пропуске срока выполнения.
   * Новое задание без дедлайна или требование исправить исходный код.

**Для преподавателей**

1. Управление учебными группами:
   * Создание и удаление групп.
   * Выдача пароля для присоединения учеников.
   * Добавление и удаление участников.
2. Управление заданиями:
   * Создание папок с заданиями.
   * Ручное или автоматическое распределение заданий.
   * Установка сроков выполнения.
3. Проверка загруженных решений:
   * Синтаксическая проверка.
   * Автоматическое тестирование программ на заранее созданных тестах.
   * Анализ на плагиат (сравнение с локальным архивом).
   * Возможность зачесть или отправить работу на доработку.
4. Дополнительные функции:
   * Назначение повторных заданий.
   * Установка ограничений на повторную загрузку решения.
   * Контроль прогресса учеников через аналитические отчёты.

**Для разработчиков (администраторов системы)**

1. Управление пользователями:
   * Назначение ролей (ученик, преподаватель, администратор).
   * Управление организациями и группами.
2. Диагностика системы:
   * Просмотр логов работы.
   * Выявление и устранение ошибок.

**Особенности первой версии**

1. **Анализ плагиата без ограничений:**
   * Сравнение загруженных работ с архивом проверенных заданий.
   * Неограниченное число проверок на плагиат для всех пользователей.
2. **Структурированный архив данных:**
   * Все данные сохраняются в формате:  
     Организация → Группа → Ученик → Задание → Результат.
3. **Тестирование программ:**
   * Преподаватели могут создавать тесты для проверки правильности выполнения заданий.
   * Система автоматически выполняет код на тестах и возвращает результаты.
4. **Гибкость:**
   * Настройки дедлайнов, ограничений на повторную сдачу, а также создание индивидуальных и групповых заданий.

**Примеры использования**

1. **Школа:**
   * Преподаватель создаёт группу "10Б", загружает задания по теме "Циклы".
   * Ученики решают задания, загружают решения через Telegram, которые проверяются на синтаксис, выполнение тестов и плагиат.
   * Преподаватель получает отчёты, может отправить задания на доработку или зачесть работу.
2. **Университет:**
   * Студенты получают индивидуальные задания.
   * Система автоматически проверяет код на заданных тестах и возвращает результаты.
   * При незачёте студенты получают новые задания или исправляют старые.
3. **Частный образовательный центр:**
   * Преподаватель создаёт папку "Основы Python", добавляет задания и назначает их ученикам.
   * Система проверяет выполнение заданий, предоставляет подробные отчёты о прогрессе учеников.

**Финансовая модель**

**Бесплатная версия:**

1. Хранилище до 1 ГБ.
2. Поддержка Python.
3. Основной функционал проверки заданий и управления группами.

**Платные тарифы:**

1. **Базовый (5000 руб./мес):**
   * Хранилище до 10 ГБ.
   * Расширенные текстовые отчёты.
2. **Премиум (7000 руб./мес):**
   * Поддержка C, C++, Java.
   * Хранилище до 50 ГБ.
   * Подробная аналитика успеваемости.

**Преимущества системы**

1. **Экономия времени преподавателей:**
   * Полная автоматизация проверки заданий.
2. **Прозрачность:**
   * Архивирование всех данных и результатов проверки.
3. **Доступность:**
   * Использование через популярный мессенджер Telegram.
4. **Масштабируемость:**
   * Возможность добавления новых функций (поддержка других языков, интеграция с LMS).

**Алгоритм работы программы (Первая версия)**

**Общие требования**

Для реализации первой версии системы потребуется:

1. **Фреймворк для бота**: Aiogram — асинхронный фреймворк для работы с Telegram API.
2. **Среда изоляции**: Docker — для безопасного выполнения пользовательского кода.
3. **База данных**: PostgreSQL — для хранения информации о пользователях, заданиях и результатах.
4. **Библиотеки Python**:
   * aiohttp — для асинхронной обработки запросов.
   * sqlalchemy — ORM для работы с базой данных.
   * pygments — для анализа и подсветки синтаксиса.

**Этапы работы программы**

**1. Инициализация бота**

1. Получение токена от BotFather.
2. Настройка webhook или polling для взаимодействия с Telegram.
3. Инициализация базовых команд: /start, /help, /create\_task, /submit, /report.
4. Подключение к базе данных (PostgreSQL) и настройка таблиц:
   * Пользователи.
   * Задания.
   * Архив проверенных решений.

**2. Регистрация пользователей**

1. Пользователь отправляет команду /start.
2. Бот предлагает выбрать роль:
   * **Ученик**:
     + Ввод пароля группы (выданного преподавателем).
     + Сохранение данных ученика в базе (ФИО, ID Telegram, группа).
   * **Учитель**:
     + Регистрация группы и генерация пароля для учеников.
     + Сохранение данных учителя и группы в базе.
3. Роль пользователя сохраняется в базе данных для дальнейшего использования.

**3. Создание и управление заданиями (для учителей)**

1. Учитель отправляет команду /create\_task:
   * Указывает название задания, описание, тестовые данные, дедлайн.
   * Загружает тесты для проверки решений (входные данные и ожидаемый результат).
2. Бот сохраняет задание в базе данных и отправляет уведомления ученикам о новом задании.

**4. Отправка решений (для учеников)**

1. Ученик отправляет команду /submit:
   * Загружает файл с решением.
   * Бот проверяет:
     + Формат файла (например, .py).
     + Соответствие заголовков заданиям.
   * Решение сохраняется в базе данных и отправляется в Docker для проверки.

**5. Проверка кода**

**Этап 1: Синтаксическая проверка**

1. Код исполняется в изолированной Docker-среде с использованием subprocess для безопасности.
2. Обрабатываются ошибки:
   * SyntaxError, IndentationError — для синтаксических проблем.
   * RuntimeError, ZeroDivisionError — для ошибок выполнения.
3. Результаты синтаксической проверки сохраняются.

**Этап 2: Выполнение тестов**

1. Код запускается с заранее загруженными тестовыми данными.
2. Сравнивается фактический вывод программы с ожидаемым результатом.
3. Для каждого теста фиксируется результат (успешно/ошибка).

**Этап 3: Проверка на плагиат**

1. Код сравнивается с архивом базы данных (поиск совпадений).
2. Вычисляется процент совпадения.
3. Результаты проверки записываются в базу.

**6. Отправка результатов**

1. **Для ученика**:
   * Результаты отправляются в Telegram:
     + Синтаксическая проверка: пройдена/не пройдена.
     + Результаты выполнения тестов.
     + Процент совпадения (анализ плагиата).
   * Если код не прошёл проверку:
     + Бот уведомляет о необходимости исправления.
     + Устанавливает дедлайн для повторной сдачи (по настройкам учителя).
2. **Для учителя**:
   * Сводный отчёт по классу:
     + Список учеников, сдавших задания.
     + Результаты тестов и анализа плагиата.
     + Ученики, которым требуется доработка.

**7. Хранение данных**

1. Загруженные решения и отчёты хранятся в облаке (например, AWS S3 или локальном хранилище).
2. Метаданные (задания, результаты проверок) сохраняются в PostgreSQL.
3. Архив решений структурируется:
   * Организация → Группа → Ученик → Задание → Результат.

**8. Формирование отчетов и аналитики**

1. Учитель запрашивает отчёт через команду /report:
   * Форматы: PDF или Excel.
   * Данные: прогресс учеников, успешность выполнения заданий, анализ по группе.
2. Бот генерирует отчёт и отправляет его преподавателю.

**Сценарий работы**

1. Учитель создаёт задание через /create\_task.
2. Ученики подключаются к группе через пароль и получают задание.
3. Ученик отправляет решение через /submit.
4. Бот выполняет:
   * Синтаксическую проверку.
   * Тестирование программы.
   * Анализ плагиата.
5. Результаты проверок отправляются ученику и учителю.
6. Учитель получает сводный отчёт о выполнении задания.

**1. Настройка PostgreSQL на виртуальной машине**

**Шаг 1: Разрешите подключение к базе данных из других устройств**

PostgreSQL по умолчанию настроен принимать соединения только с локальной машины (той, на которой он установлен). Чтобы разрешить подключения извне:

* Найдите файл конфигурации PostgreSQL (обычно он называется postgresql.conf).
* Найдите параметр, отвечающий за адреса, с которых разрешены подключения (обычно listen\_addresses) и укажите, что PostgreSQL может принимать запросы от любых устройств.

**Шаг 2: Укажите правила для доступа к базе**

Другой важный файл конфигурации — это файл, в котором указаны правила подключения (часто называется pg\_hba.conf). В этот файл нужно добавить запись, разрешающую доступ от IP-адреса основной ОС. Если вы хотите, чтобы доступ был максимально открытым, можно временно разрешить соединения с любого устройства.

**Шаг 3: Перезапустите PostgreSQL**

После внесения изменений в конфигурационные файлы нужно перезапустить PostgreSQL, чтобы изменения вступили в силу.

**2. Настройка виртуальной машины**

**Шаг 1: Узнайте IP-адрес виртуальной машины**

Для подключения из основной ОС нужно знать IP-адрес ВМ. Узнайте его через сетевые настройки. Это адрес, по которому основная ОС будет обращаться к базе данных.

**Шаг 2: Проверьте настройки брандмауэра**

Если на ВМ настроена система защиты (например, фаервол), убедитесь, что порт, который использует PostgreSQL (обычно 5432), открыт для подключения. Это позволит основной ОС отправлять запросы к базе данных на ВМ.

**3. Подключение к базе данных с основной ОС**

На основной операционной системе вы будете подключаться к базе данных на ВМ через специальную библиотеку или инструмент:

* Укажите IP-адрес виртуальной машины, порт подключения (5432), имя базы данных, имя пользователя и пароль для доступа.
* Проверьте соединение, чтобы убедиться, что ваша ОС может подключиться к базе на ВМ.

**4. Проверка соединения**

1. Убедитесь, что PostgreSQL на ВМ запущен.
2. Проверьте, что основные ОС может обратиться к базе данных.
3. Если всё настроено верно, ваша программа (чат-бот) должна без проблем подключиться к базе на виртуальной машине.

**1. Подготовительный этап (1-2 недели)**

**Цели:** Определение целей, задач, технологий, проектирование архитектуры системы.

**Задачи:**

1. Формулировка целей проекта и приоритетных функций:
   * Реализация 3 ролей пользователей: **ученик**, **преподаватель**, **администратор**.
   * Взаимодействие пользователей через Telegram-бота.
   * Хранение данных в базе данных SQLite.
   * Возможность проверки кода, включая проверку на антиплагиат.
2. Исследование потребностей целевой аудитории:
   * Определение сценариев использования для каждой роли.
3. Выбор технологий:
   * Python для серверной части.
   * Aiogram для создания Telegram-бота.
   * SQLite для хранения данных.
   * Дополнительные библиотеки: для проверки кода, обработки ошибок, построения API.
4. Создание архитектуры системы:
   * Проектирование модульной структуры (бот, бизнес-логика, работа с базой данных).
   * Определение структуры базы данных для хранения пользователей, групп, заданий, решений и ролей.
5. Регистрация Telegram-бота через BotFather.
6. Настройка рабочего окружения:
   * Установка необходимых инструментов и библиотек (Python, SQLite, Aiogram).
   * Подготовка среды разработки (Git, виртуальное окружение).

**2. Разработка основного функционала (4-6 недель)**

**Цели:** Реализация основных функций платформы и их интеграция с Telegram-ботом.

**2.1. Регистрация пользователей (1 неделя)**

**Задачи:**

1. Реализовать систему регистрации и авторизации пользователей через Telegram-бот.
2. Разработать три роли пользователей:
   * **Администратор** — управляет пользователями (назначает роли), контролирует группы, задания, и систему.
   * **Преподаватель** — создаёт группы, добавляет задания, отслеживает выполнение.
   * **Ученик** — присоединяется к группам, выполняет задания, получает обратную связь.
3. Валидация данных при регистрации (например, проверка на уникальность имени пользователя).
4. Возможность администратора управлять ролями пользователей через бот.
5. Добавить возможность администраторам просматривать всех зарегистрированных пользователей и их роли.

**2.2. Создание и управление группами (1 неделя)**

**Задачи:**

1. Реализовать функцию создания групп для преподавателей.
2. Добавить возможность ученикам присоединяться к группам по уникальному коду/паролю, который выдаёт преподаватель.
3. Реализовать управление группами:
   * Просмотр списка групп.
   * Удаление групп (только преподавателем или администратором).
4. Связать систему управления группами с Telegram-ботом (команды для создания и управления группами).

**2.3. Создание заданий (1 неделя)**

**Задачи:**

1. Реализовать возможность преподавателям добавлять задания в группы, которые они создают.
2. Добавить функционал управления заданиями:
   * Просмотр заданий в конкретной группе.
   * Удаление заданий.
3. Указать дедлайн для каждого задания, который будет отображаться ученикам.
4. Интеграция с Telegram-ботом:
   * Команды для создания, удаления и просмотра заданий.

**2.4. Выполнение заданий и проверка кода (1 неделя)**

**Задачи:**

1. Реализовать функционал загрузки учениками решений заданий через Telegram-бота.
2. Создать систему проверки кода:
   * Синтаксическая проверка.
   * Проверка выполнения программы на заранее заданных тестах.
3. Отображение результатов проверки (принято/ошибка/результат тестов).
4. Добавить базовую обработку ошибок в решениях (например, при неправильной загрузке кода).

**2.5. Система антиплагиата (2 недели)**

**Задачи:**

1. Исследовать и выбрать подход для проверки на плагиат:
   * Использование готовых инструментов (например, MOSS, JPlag) или разработка базового алгоритма.
2. Интеграция проверки на плагиат с системой:
   * После синтаксической проверки решение проверяется на схожесть с другими решениями в базе.
3. Создать отчёты по плагиату для преподавателей:
   * Показывать процент совпадения и список подозрительных решений.
4. Отправка уведомлений через Telegram-бот ученикам с результатами проверки на плагиат.

**2.6. Обратная связь для учеников (1 неделя)**

**Задачи:**

1. Отправка ученикам результатов проверки кода через Telegram-бота:
   * Принято/ошибка.
   * Процент совпадения по результатам антиплагиата.
2. Добавить систему рекомендаций:
   * Например, сообщение о том, какие ошибки были найдены в решении.

**3. Тестирование и отладка (2 недели)**

**Цели:** Проверка работоспособности всех функций и их интеграция.

**3.1. Юнит-тестирование (1 неделя)**

**Задачи:**

1. Написать тесты для всех ключевых функций системы:
   * Регистрация пользователей и управление ролями.
   * Создание групп и заданий.
   * Проверка кода и результаты антиплагиата.
2. Тестирование взаимодействия Telegram-бота с базой данных.

**3.2. Интеграционное тестирование (1 неделя)**

**Задачи:**

1. Проверка корректности взаимодействия всех компонентов:
   * Telegram-бот.
   * База данных.
   * Система проверки решений.
2. Тестирование реальных сценариев:
   * Ученики присоединяются к группе, выполняют задания.
   * Преподаватели создают задания, получают результаты.

**4. Развертывание на сервере (1 неделя)**

**Цели:** Подготовка инфраструктуры для работы проекта.

**Задачи:**

1. Выбор облачного провайдера (AWS, Google Cloud, DigitalOcean).
2. Настройка сервера (например, Ubuntu): установка Python, SQLite, необходимых библиотек.
3. Развертывание проекта: настройка окружения, деплой Telegram-бота.
4. Настройка автоматического запуска бота при перезагрузке сервера (например, через systemd).

**5. Запуск и мониторинг (1-2 недели)**

**Цели:** Обеспечение стабильной работы системы и улучшение функционала.

**Задачи:**

1. Настройка мониторинга сервера:
   * Использование инструментов мониторинга (например, Netdata, Prometheus).
2. Сбор обратной связи от пользователей:
   * Ученики, преподаватели и администраторы оставляют отзывы о работе системы.
3. Исправление найденных ошибок и улучшение функционала.

**6. Документация и планирование следующего этапа (1 неделя)**

**Цели:** Обеспечение удобства использования и планирование развития проекта.

**Задачи:**

1. Создание документации для пользователей:
   * Инструкция по использованию Telegram-бота для учеников, преподавателей, администраторов.
2. Создание документации для разработчиков:
   * Архитектура системы, описание базы данных.
   * Пошаговая инструкция по развертыванию проекта.
3. Планирование следующих этапов развития:
   * Поддержка других языков программирования.
   * Аналитика успеваемости учеников.
   * Расширенные отчёты для преподавателей.

**Итоговый график работы:**

1. Подготовительный этап: 1-2 недели.
2. Разработка основного функционала: 4-6 недель.
3. Тестирование и отладка: 2 недели.
4. Развертывание на сервере: 1 неделя.
5. Запуск и мониторинг: 1-2 недели.
6. Документация и планирование: 1 неделя.

**Общая продолжительность:** 10-13 недель.

**Команда для реализации проекта**

|  |  |  |
| --- | --- | --- |
| ФИО участника | Роль | Функционал |
| Тушов В.М. | Лидер (Руководитель проекта) | * Разработка чат-бота в Telegram (минимально жизнеспособный продукт, MVP). * Тестирование чат-бота для проверки работоспособности и удобства использования. * Исправление ошибок и реализация пожеланий пользователей для улучшения функционала. * Развёртывание на сервере, настройка и оптимизация работы платформы. * Обслуживание сервера, обеспечение его бесперебойной работы. * Разработка технической документации (описание архитектуры, API, принципов работы системы). |
| Вершинина Леночка(очень сильно подходит) или Ершова Уля | Заместитель руководителя проекта по обратной связи пользователей и SMM | * Обработка обратной связи от пользователей, включая сбор отзывов и предложений. * Управление группами в социальных сетях (VK, Telegram). * Ответы на сообщения сообщества в VK и комментарии в Telegram-канале. * Разработка контент-планов для публикации в социальных сетях, включая анонсы, новости и образовательные материалы. |
| Настя Татаурова или Женечка Хомутиннова | Заместитель руководителя проекта по продвижению и брендингу | * Разработка брендинга проекта (логотипы, фирменный стиль, графические элементы). * Подготовка и оформление презентаций, включая визуальные материалы, инфографику и анимации. * Создание видеоконтента, включая обучающие ролики и рекламные видео для продвижения платформы. |
| (кого-то из друзей программистов) | Заместитель руководителя проекта по технической поддержки | * Предоставление технической поддержки пользователям (ученикам, преподавателям, разработчикам). * Диагностика и устранение технических проблем в работе системы. * Управление системными настройками (пользователи, роли, организации и группы). * Анализ логов работы системы для выявления ошибок и предложений по оптимизации. |