**Сайт**

Национальный исследовательский университет   
«Высшая школа экономики»

Лицей

Индивидуальная выпускная работа

Отчёт о проекте

**Система автоматизированного тестирования**

*Выполнил Мороз Александр Константинович*

Консультант:

Купцов Александр Александрович

Москва 2024

*Название проекта*: Система Автоматизированного Тестирования

*Проблемное поле*: Продукт решает ряд недостатков, которые на данном этапе имеют другие платформы. Недостатки существующих систем и пути их решения в проекте:

1. Ограниченные возможности существующих систем по автоматизации оценки и анализа результатов

Ручной труд: Преподаватели часто вынуждены вручную обрабатывать результаты тестирования, что занимает много времени и увеличивает вероятность ошибок. В настоящий момент оценка знаний участника возможна как с помощью автоматизированной системы (средствами самого сервиса), так и интеграция через API с внешними сервисами.

Сложность анализа: Существующие системы могут не предоставить удобных инструментов для анализа результатов, таких как графики, отчеты или сравнительная статистика. Это затрудняет преподавателям выявление слабых мест в знаниях студентов. На текущем этапе данный функционал в систему не интегрирован, но позволяется использование API для решения данной задачи сторонними сервисами.

Ограниченные типы оценивания: Многие платформы предлагают лишь базовые форматы тестирования (например, выбор ответа), что не позволяет оценивать более сложные навыки и знания. Пользователю предоставлена возможность дать развернутый ответ, который будет оценен вручную через средства визуализации сервиса или за счет интеграции через API с внешним сервисом.

2. Отсутствие гибкости в настройке тестов и интеграции с другими образовательными инструментами

Невозможность кастомизации: Преподаватели часто сталкиваются с жесткими шаблонами тестов, которые не позволяют адаптировать задания под конкретные учебные цели или особенности группы студентов. Частично решили задачу, дав возможность добавлять развернутый ответ на вопрос, пока без возможности добавлять к ответу дополнительные материалы в виде файлов различного формата и без возможности использования специальной верстки для составления химических, физических, математических и иных видов формул.

Проблемы интеграции: Существующие системы часто не поддерживают интеграцию с другими образовательными инструментами (например, LMS, платформами для онлайн-обучения, системами управления контентом), что ограничивает возможности использования различных ресурсов. Это не позволяет пользователю использовать нужный ему сервис. Предоставленный API позволяет провести интеграцию с любыми другими платформами.

Неудовлетворительные API: Для разработчиков, стремящихся создать собственные инструменты или расширения, часто отсутствуют качественные API, что усложняет процесс интеграции. В настоящий момент при реализации проекта был разработан базовый API создания и оценивания тестов, регистрации пользователей, сбора статистики, данный набор методов может сколько угодно расширяться как в ширину, так и в глубину.

3. Нехватка функционала для получения детализированной статистики по результатам тестирования

Отсутствие глубокого анализа: Системы могут не предоставлять детализированные отчеты по результатам тестирования, такие как анализ по категориям знаний, времени выполнения заданий и т.д. Педагог не сможет понять какой материал ученики усвоили лучше и на чем стоит акцентировать внимание. Реализованный API позволяет сервисам обращаться к результатам и анализировать их, что позволяет решить обозначенную проблему.

Недостаточная визуализация данных: Преподаватели могут не иметь доступа к визуализированным данным, что затрудняет интерпретацию результатов и принятие решений на основе этих данных. Визуализация данных может быть достигнута за счет получения данных через API и визуализации в сторонних системах.

Сложности с отслеживанием прогресса: Без возможности отслеживания динамики успеваемости студентов на протяжении времени преподаватели не могут корректировать свои методы преподавания и адаптировать учебный процесс. При необходимости оценивания динамики ученика, могут быть использованы сторонние сервисы с использованием нашего API.

Вывод: Существующие образовательные платформы не удовлетворяют потребности пользователей в области оценки знаний из-за ограниченной функциональности, недостаточной гибкости и отсутствия интеграции с другими инструментами. Это создает значительные препятствия для эффективного обучения и оценки, что подчеркивает необходимость разработки более интегрированных и адаптивных решений в области образовательных технологий. В ходе разработки проекта был создан API позволяющий провести интеграцию со сторонними сервисами и получить все недостающие аналитики.

Носитель проблемы: Пользователь сторонней/сторонних систем, в которых его возможности ограничены и функционал уже существующих продуктов не является избыточным для решения проблем, с которыми он сталкивается.

Выбор проблемы: Данная проблема чрезвычайно актуально в наша время, многие преподаватели дают тесты ученикам для выполнения на онлайн платформах таких как Online Test Pad и т.д. Сервис Online Test Pad имеет ряд значительных недостатков, такие как: нажатие клавиши Enter может завершить тестирование, отсутствие учёта человеческого фактора (т.е. небольшая опечатка, например: лишний пробел, точка или запятая. Тест сразу же считает это неправильным ответом, и педагог не может вручную добавить баллы), проблема отсутствия статистики (педагог не может понять какой средний балл у учеников за тест если не произведет эти вычисления вручную). Учитывая вышеперечисленные недостатки уже существующих сервисов, появилась идея проекта.

*История работы над проектом*: Тема проекта была определена сразу, исходя из актуальных потребностей в сфере образования и анализа существующих решений.

Этап 1: Поэтапное планирование

1)Постановка целей (1-3 дня)

2)Разработка структуры базы данных (5-8 дней)

3) Разработка документации для API (7-10 дней)

4)Разработка web-сервиса (2-3 месяца)

5)Разработка web-интерфейса (1-1,5 месяц)

6)Развёртывание web-сервиса на виртуальном сервере (1-2 дня)

Этап 2: Исследование и анализ

На начальном этапе был проведен глубокий анализ существующих систем автоматизированного тестирования, выявлены их преимущества и недостатки. Особое внимание было уделено функциональности, гибкости настройки, интеграции с другими платформами и аналитическим возможностям. Был проведен опрос ([результаты](https://docs.google.com/spreadsheets/d/1Ff0Ay7-MBfTftIWehmD8IEUCccJoTxf4bmyD7hQn9xM/edit?resourcekey=&gid=30682252#gid=30682252)), благодаря которому удалось более точно определить актуальность проекта.

Этап 3: Разработка структуры базы данных

1)Выбор технологий для реализации веб-сервиса

В данном разделе обоснован выбор технологий и инструментов для разработки веб-сервиса на основе FastAPI, который будет предоставлять пользователям возможность регистрации, создания и прохождения тестов.

За короткое время было невозможно сконцентрироваться как на бэкэнде так и фронтэнде, поэтому упор был сделан на бэкэнд.

1.1. Бэкенд

1) FastAPI: Выбранный фреймворк для создания API благодаря высокой производительности, поддержке асинхронного программирования и автоматической генерации документации (Swagger) а также быстрой скорости разработки и доступности документации на сам фреймворк. Django был признан как тяжеловесный и неподходящий для быстрого старта, Flusk требует огромного количества библиотек для превращения в некое подобие Fastapi. FastAPI уже готовый вариант, который позволяет легко стартовать. Основной идея проекта была предоставить API, потому FastAPI подходил нам наилучшим образом.

2)SQLite: Легковесная реляционная база данных, подходящая для хранения данных о пользователях и тестах. Не требует установки отдельного сервера, что упрощает развертывание и тестирование продукта в условиях приближенных к реальным. На начальном этапе не ожидается большого потока пользователей, тестировщиками выступит ограниченный круг лиц, поэтому функционала SQLite будет достаточно для проведения тестирования. В дальнейшем планируется заменить SQLite на PostgreSQL как более производительную СУБД.

1.2. Фронтэнд

Разработка собственного веб-интерфейса была вынужденной мерой и преследовала цель продемонстрировать возможности предоставленного API, а также преследовала цель познакомиться с разработкой веб-интерфейсов для понимания как создается фронтэнд.

1)JavaScript: Основной язык для веб-разработки, обеспечивающий интерактивность интерфейса. JavaScript это язык с одним из самых низких порогов кода и для старта во фронтэнде он подходит наилучшим образом. В идеале если проект будет дальше разрабатываться, то JavaScript будет заменен на TypeScript в комплекте с одним из современных фронтэнд фреймворков на подобии react, angular, vue или их аналогов.

2)CSS: Простой и мощный язык стилей, обеспечивающий гибкость, кроссбраузерную совместимость и возможность быстрой интеграции с фреймворками для создания привлекательных веб-интерфейсов. Было принято решение в начале использования CSS препроцессоров оценить возможности самого CSS и сложность его применения к подобным проектам.

1.3. Структура Базы Данных(PK – primary\_key; FK – ForeignKey)

1) Структура сессии пользователя при прохождении теста

"user\_quiz\_session"

Id-сессии, Integer, PK

Id-пользователя, Integer, FK(users.id)

Id-теста, Integer, FK(quizes.quiz\_id)

Id-вопросов в сессии, String

Время начала сессии, DateTime

Время окончания сессии, DateTime, nullable

Время затраченное на сессию, Integer

Время обновления сессии, DateTime, nullable

Результат сессии, Integer, nullable

Дополнительная попытка, Integer, FK (user\_extra\_try.id), nullable

Номер вопроса в сессии, Integer

Кол-во пройденных вопросов, Integer, default=0

Данная таблица наиболее интересна, что именно из-за неё БД теряет нормальную форму, нарушается атомарность за счет поля id-вопросов в сессии, которое хранит в себе сразу несколько значений для id-шников вопросов. Id-шники представлены в виде списка и сконвертированы в JSON строку. Дополнительный минус подобного решения заключается в нарушении целостности при удалении вопроса id вопроса остается в строке id-вопросов в сессии, база данных сама не контролирует целостность данных этой колонки. Плюсы подобного решения: повышение производительности работы базы данных, при большом объёме вопросов и большом объёме сессий производить выборку вопросов становится довольно дорого по времени и вычислительным ресурсам, поэтому при проектировании БД были приняты следующие упрощения: после того как автор закончил формирование теста он крайне редко будет удалять вопрос из своего теста следовательно проблема целостности имеет для данного проекта минимальное значение. Выгодой для нас становится возможность определить следующий вопрос, который нужно показать пользователю за один запрос к БД.

2)Структура пользователя

"users"

Id-пользователя, Integer, PK

Электронная почта, String, unique

Пароль, String

3)Структура создателя

"quiz\_creators"

Id-создателя, Integer, PK

Id-пользователя, Integer, FK (users.id)

4)Структура теста

"quizes"

Id-теста, Integer, PK

Дата создания, String

Id-создателя, Integer, FK(quiz\_creators.id)

Время на тест, Integer, nullable

Количество вопросов, Integer, default=0

Количество вопросов для завершения, Integer

Имя теста, String

Описание теста, String, nullable

Количество попыток на тест, Integer, nullable

Показывать результаты, Boolean, default=False

Тип оценки теста, Integer, тип оценки теста in (1, 2, 3, 4), nullable=True

Возможность переключения между вопросами, Boolean, default=False

Количество простых вопросов, Integer, nullable, default=None

Количество средних вопросов, Integer, nullable, default=None

Количество сложных вопросов, Integer, nullable, default=None

Возможность изменения ответа, Boolean, default=False

5)Структура вопроса теста

"quiz\_questions"

Id-вопроса, Integer, PK

Текст вопроса, String

Описание вопроса, String, nullable

Время на вопрос, Integer, nullable

Баллы за вопрос, Integer, nullable

Тип вопроса, Integer, тип вопроса in (1, 2, 3)

Id-теста, Integer, FK (quizes.quiz\_id)

Номер вопроса, Integer, nullable

Сложность вопроса, Integer, сложность вопроса in (1, 2, 3), default=1

Подсказка на вопрос, String, nullable

6)Структура ответов на вопрос теста

"question\_answers"

Id-ответа, Integer, PK

Текст ответа, String

Id-вопроса, Integer, FK (quiz\_questions.id)

Количество баллов за ответ, Integer, nullable

Номер ответа, Integer, nullable

Правильный ответ, Boolean

7)Структура ответа на вопрос с выбором ответа

"user\_choice\_answers"

Id-ответа, Integer, PK

Id-сессии, Integer, FK (user\_quiz\_session.id)

Id-вопроса, Integer, FK (quiz\_questions.id)

Id-ответа, Integer, FK (question\_answers.id)

Время ответа, DateTime

Баллы за ответ, Integer, default=0

Правильный ответ, Boolean

8)Структура ответа на вопрос со свободным ответом

"user\_text\_answer"

Id-ответа, Integer, PK

Id-сессии, Integer, FK (user\_quiz\_session.id)

Id-вопроса, Integer, FK (quiz\_questions.id)

Id-ответа, Integer, FK (question\_answers.id)

Время ответа, DateTime

Баллы за ответ, Integer, default=0

Ответ, String

Правильный ответ, Boolean, nullable

9)Структура запроса на прохождение теста сверх лимита попыток

"user\_extra\_try"

Id-запроса, Integer, PK

Id-пользователя, Integer, FK (users.id)

Id-теста, Integer, FK (quizes.quiz\_id)

Время создания, DateTime

Одобрение, Boolean, default=None, nullable

Количество попыток, Integer, default=1

На данном этапе разработки проекта эта структура не используется, но планируется её дальнейшее применение при модернизации проекта.

10)Структура проверки ответа на вопрос в сессии

'user\_quiz\_questions'

Id-ответа на вопрос в сессии, Integer, PK

Id-сессии, Integer, FK (users.id)

Id-вопроса, Integer, FK (quiz\_questions.id)

Ответ есть, Boolean, default=False

11)Структура переподключения сессии

'session\_reconnect'

Id-переподключения, Integer, PK

Id-сессии, Integer, FK (user\_quiz\_session.id)

Id-вопроса, Integer, FK (quiz\_questions.id), nullable

Продолжительность теста, Integer

Время переподключения, DateTime

На данном этапе разработки проекта эта структура не используется, но планируется её дальнейшее применение при модернизации проекта.

Этап 4: Разработка схемы обмена данными клиент-сервер

1) AddUserRequest

Электронная почта, EmailStr

Пароль, str

2) LogInUserRequest

Электронная почта, EmailStr

Пароль, str

3) UpdateUserRequest

Электронная почта, EmailStr

Новый пароль, str

Старый пароль, str

4) AddQuizRequest

Имя теста, str

Длительность теста, int or None

Id пользователя, int or None

Id создателя, int or None

Количество вопросов для завершения, int

Описание теста, str or None

Количество попыток на тест, int or None

Показывать результаты после прохождения, bool

Тип оценки теста, int

Переключение между вопросами, bool

Возможность повторного ответа на вопрос, bool

Количество простых вопросов, int or None

Количество средних вопросов, int or None

Количество сложных вопросов, int or None

5) UpdateQuizRequest

Имя теста, str

Длительность теста, int or None

Количество вопросов для завершения, int

Описание теста, str or None

Количество попыток на тест, int or None

Показывать результаты после прохождения, bool

Тип оценки теста, int

Переключение между вопросами, bool

Возможность повторного ответа на вопрос, bool

Количество простых вопросов, int or None

Количество средних вопросов, int or None

Количество сложных вопросов, int or None

6) AddQuizQuestion

Текст вопроса, str

Описание вопроса, str or None

Время на вопрос, int or None

Баллы за вопрос, int or None

Тип вопроса, int

Id теста, int

Номер вопроса, int or None

Сложность вопроса, int or None

Подсказка на вопрос, str or None

7) UpdateQuizQuestion

Текст вопроса, str

Описание вопроса, str or None

Время на вопрос, int or None

Баллы за вопрос, int or None

Номер вопроса, int or None

Сложность вопроса, int or None

Подсказка на вопрос, str or None

8) AddQuestionAnswer

Текст ответа, str

Id вопроса, int

Баллы за ответ, int or None

Номер ответа, int or None

Правильный ответ, bool

9) UserAnswer

Список ответов, List[int]

Текстовый ответ, str or None

10) Points

Баллы, int

11) UserAnswerChoice

Id сессии, int

Id вопроса, int

Список ответов, List[int]

12) UserAnswerText

Id сессии, int

Id вопроса, int

Текстовый ответ, str

13) UserUpdateAnswerText

Текстовый ответ, str

14) UserUpdateAnswerChoice

Список ответов, List[int]

15) RefreshSession

Id сессии, int

Время отключения, int or None

Этап 5: Функции обработки запросов

Были разработаны следующие функции обработки запросов:

1)Пользователь

1) Добавление пользователя

2) Вход пользователя в систему

3) Получение всех пользователей

4) Получение пользователя по id

5) Получение пользователя по email

6) Изменение пользователя

2)Тест

1) Добавление теста

2) Удаление теста

3) Изменение теста

4) Получение списка всех тестов

5) Получение теста по id

6) Получение списка вопросов на тест

7)Получение списка тестов по id пользователя

3)Вопрос к тесту

1) Добавление вопроса

2) Удаление вопроса

3) Изменение вопроса

4) Получение вопроса по id

5) Получение списка вариантов ответа на вопрос

4)Ответ на вопрос к тесту

1) Добавление варианта ответа на вопрос

2) Удаление варианта ответа на вопрос

3) Изменение варианта ответа на вопрос

4) Получение варианта ответа по id

5)Ответ пользователя на вопрос к тесту

1) Добавление выбранного варианта ответа на вопрос

2) Изменение выбора ответа

3) Удаление выбора ответа

4) Добавление текстового ответа

5) Изменение текстового ответа

6) Удаление текстового ответа

6)Сессия теста

1) Начало сессии

2) Получение следующего вопроса сессии

3) Окончание сессии

4) Сохранение ответов

5) Получение результатов пользователя

6) Получение вопросов в сессии для отображения результатов

7) Получение вопросов в сессии для отображения статистики

8) Добавление баллов

9) Получение сессии по id

10) Получение средних результатов и количества попыток на тест

11) Получение всех сессий на тест по id пользователя

Этап 6: Тестирование бэкэнда

Для данного этапа был использован инструмент для тестирования и автоматизации работы с API, Postman. С его помощью появилась возможность отправлять запросы и получать ответы без необходимости писать логику обращения к серверу вручную. В Postman был создан открытый [workspace](https://app.getpostman.com/join-team?invite_code=4059215423f35b9afd8f6670cf7dea5d), в котором вы можете просмотреть начальные запросы. Не все возможные модели запросов были внесены в Postman, потому что часть из них была создана во время написания фронтэнда. [Полный список функций обработки запроса](http://185.42.14.235:8000/docs)

Этап 7: Написание фронтэнда

Разработан интерактивный фронтэнд, обеспечивающий взаимодействие пользователя с сервером. Реализованы обработчики событий, отправка/получение данных с помощью fetch, отображение информации и валидация данных.

Этап 8: Стилизация фронтэнда

Стилизация улучшила разработанный фронтэнд, сделав его не только цветным, но и более удобным для прочтения и взаимодействия.

Этап 9: Форматирование кода

На финальном этапе весь код на языке программирования python был приведен к единому формату. С помощью Flake8 были найдены и исправлены все ошибки и недочеты в оформлении кода.

*Целевая аудитория*: Преподаватели (учителя, профессора, тьюторы и т.д.), студенты (школьники, учащиеся, аспиранты т.д.) в процессе обучения. [Отзывы](https://docs.google.com/spreadsheets/d/1IrCUL2N4r1c9qAkLHuFbbe-74laOpYBs77y8a2gcUD4/edit?resourcekey=&gid=184338167#gid=184338167)

*Описание продукта*: Web-сервис с API и web интерфейсом. Для проведения тестирования web-сервис развёрнут на виртуальной машине в сети интернет, доступ к сервису открытый, любой пользователь может пройти регистрацию и пользоваться возможностями сервиса. Функционал сервиса доступный через web-интерфейс и api - создание, каталогизация и поиск тестов, также прохождение тестирования, выставление оценок и просмотр статистики. Статистика имеет 2 вида: статистика участника теста и статистика создателя теста. Любой участник может быть как создателем, так и участником теста. Были реализованы следующие пользовательские сценарии:

1)Регистрация пользователя

1. Пользователь вводит адрес электронной почты, с которым он будет заходить в систему.

2. Пользователь вводит пароль, с которым он будет заходить в систему.

3. Если выбранный адрес электронной почты уже используется в системе, то пользователю сообщается об этом и предлагается войти.

4. Если пароль содержит менее 8 символов, или не содержит спец. символов система сообщает, что пароль должен быть более 8 символов и пользователь должен придумать новый пароль.

5. Если введённый адрес электронной почты не соответствует формату, то система выводит сообщение об ошибке и просит ввести адрес ещё раз.

6. Если все введённые данные соответствуют требованиям регистрации, то система подтверждает успешную регистрацию.

2)Вход в систему пользователя

1. Пользователь вводит адрес электронной почты.

2. Пользователь вводит пароль.

3. Если адрес электронной почты или пароль неверны, система сообщает об ошибке и предлагает повторить ввод.

4. Если данные введены правильно, пользователь переходит в главное меню системы.

3)Создание теста

1. В главном меню пользователь выбирает опцию "Мои тесты".

2. Пользователь выбирает опцию "Создать тест".

3. Пользователь вводит название теста.

4. Пользователь задает описание теста (по желанию).

5. Пользователь задает количество вопросов, которое будет в его тесте.

6. Пользователь задает время на прохождение теста (по желанию).

7. Пользователь задает количество попыток на прохождение теста (по желанию).

8. Пользователь выбирает показывать ли человек, который проходит тест результаты после выполнения теста.

9. Пользователь выбирает будет ли человек, который проходит тест иметь возможность возвращаться к предыдущим вопросам.

10. Пользователь выбирает будет ли человек, который проходит тест иметь возможность изменить ответ к предыдущим вопросам.

11. Пользователь задает количество простых вопросов, которые будут показаны при прохождении теста(по желанию).

12. Пользователь задает количество вопросов средней сложности, которые будут показаны при прохождении теста(по желанию).

13. Пользователь задает количество сложных вопросов, которые будут показаны при прохождении теста(по желанию).

14. После завершения ввода данных система сохраняет тест если все данные введены корректно, иначе возвращает пользователю ошибку.

4)Добавление вопроса к тесту

1. В главном меню пользователь выбирает опцию "Мои тесты".

2. Пользователь выбирает опцию "Список тестов".

3. Пользователь выбирает конкретный тест, к которому хочет добавить вопрос.

4. Пользователь выбирает опцию "Вопросы к тесту".

5. Пользователь выбирает опцию "Добавить вопрос".

6. Пользователь вводит текст вопроса.

7. Пользователь вводит описание вопроса(по желанию).

8. Пользователь вводит время выделенное на вопрос(по желанию).

9. Пользователь вводит количество баллов за правильный ответ на вопрос(по желанию).

10. Пользователь выбирает тип вопрос(вопрос с выбором одного варианта ответа, вопрос с выбором нескольких вариантов ответа, вопрос с свободным ответом).

11. Пользователь вводит номер вопроса(по желанию).

12. Пользователь выбирает сложность вопроса(по желанию).

13. Пользователь делает возможность просмотреть подсказку на вопрос(по желанию).

14. После завершения ввода данных система сохраняет вопрос если все данные введены корректно, иначе возвращает пользователю ошибку.

5)Добавление ответа к вопросу

1. В главном меню пользователь выбирает опцию "Мои тесты".

2. Пользователь выбирает опцию "Список тестов".

3. Пользователь выбирает конкретный тест, к которому хочет добавить вопрос.

4. Пользователь выбирает опцию "Вопросы к тесту".

5. Пользователь выбирает конкретный вопрос, к которому хочет добавить ответ.

6. Пользователь выбирает опцию "Добавить ответ".

7. Пользователь вводит текст вопроса.

8. Пользователь вводит количество баллов за ответ(по желанию).

9. Пользователь вводит номер ответа(по желанию)

10. Пользователь выбирает будет ли этот ответ являться правильным или нет.

11. После завершения ввода данных система сохраняет ответ если все данные введены корректно, иначе возвращает пользователю ошибку.

6)Начать тестирование

1. В главном меню пользователь выбирает опцию "Тесты".

2. Пользователь выбирает конкретный тест, который он хочет пройти.

3. Пользователь выбирает опцию "Начать тест".

7)Смена данных профиля

1. В главном меню пользователь выбирает опцию "Профиль".

2. Пользователь выбирает что он хочет изменить(почту или пароль).

3. При желании изменить пароль пользователь выбирает опцию "Изменить пароль".

3.1. Пользователь вводит новый пароль и затем старый.

4. При желании изменить почту пользователь выбирает опцию "Изменить почту".

4.1. Пользователь вводит новую почту.

5. После завершения ввода данных система сохраняет изменения если все данные введены коректно, иначе возвращает пользователю ошибку.

8)Ответ на вопрос с развернутым ответом

1. Пользователь видит вопрос с развернутым ответом.

2. Пользователь вводит свой ответ в текстовое поле.

3. После ввода ответа пользователь нажимает "Следующий вопрос".

4. Система сохраняет ответ и переходит к следующему вопросу.

9)Ответ на вопрос с выбором одного или нескольких вариантов ответа

1. Пользователь видит вопрос с вариантами ответов.

2. Пользователь выбирает один или несколько вариантов ответа.

3. После выбора пользователь нажимает "Следующий вопрос".

4. Система сохраняет ответ и переходит к следующему вопросу.

10)Просмотр результатов для создателя теста

1. В главном меню создатель теста выбирает опцию "Результаты".

2. Создатель выбирает опцию "Результаты прохождения моих тестов".

3. Система отображает список тестов создателя.

4. Создатель выбирает тест, по которому хочет просмотреть результаты.

5. Создатель может видеть результаты каждого пользователя и общее количество участников определенного теста.

11)Просмотр результата пользователем, прошедшим тест

1. В главном меню пользователь выбирает опцию "Результаты".

2. Пользователь выбирает опцию "Мои результаты".

3. Система показывает результаты всех тестов, которые пользователь прошел.

4. Пользователь может видеть свои результаты и статистику по каждому тесту.

*Рефлексия*: К сожалению не удалось сделать проверку email’а путем отправки на него письма с кодом подтверждения из-за нехватки времени и недостатка знаний о том, как через CRM-систему или сервисы отправки транзакционных писем направить письмо на почту пользователя. Не удалось учесть возможность прерывания сессии пользователя при прохождении теста в связи с недостатком знаний. Также не было возможности сделать профессиональный дизайн в связи с недостатком опыта и отсутствия дизайнера. В процессе были приобретены навыки по работе с FastAPI, SQLite GitHub и виртуальной машиной. Были освоены основы языков программирования JavaScript и CSS. В процессе работы с бэкэндом были усовершенствованы навыки работы с Python.

*Как можно дальше развивать проект*: На данном этапе проект имеет минимальный функционал, для работы. В дальнейшем планируется расширить функционал, добавив возможности, которых на данный момент нет в других сервисах. Также продукт имеет непрофессиональный дизайн, который не подходит для демонстрации и представления продукта. В дополнение стоит добавить интеграцию со сторонними сервисами, такими как ChatGPT для автоматизации проверки тестов. Планируется изменить структура базы данных при популяризации проекта, в связи с тем, что SQLite подходит как база данных для мини-проекта, но в больших масштабах стоит заменить её на PostgreSQL.

*Отзывы пользователей сервиса*:



















