Техническое задание

Проект: Система управления тестами и викторинами (простая) — админ-панель

1. Цель работы

Разработать веб-приложение для создания и управления тестами/викторинами: администратор (преподаватель) создаёт тесты, вопросы и варианты ответов, отмечает правильные. Данные хранятся в MySQL. По тесту формируется ссылка для прохождения студентом.

2. Роли

- Преподаватель/Администратор: создаёт и редактирует тесты, вопросы и ответы.
- Студент (опционально): проходит тест по ссылке и получает результат.

3. Основной сценарий (админ-панель)

- Администратор входит в систему.
- Создаёт новый тест (название, описание, настройки).
- Добавляет вопросы теста.
- Для каждого вопроса добавляет варианты ответов и отмечает правильные.
- Сохраняет тест и получает ссылку на прохождение.

4. Функциональные требования (администратор)

- Авторизация администратора (логин/пароль).
- CRUD тестов: создать/просмотреть/редактировать/удалить тест.
- CRUD вопросов: добавить/редактировать/удалить вопрос внутри теста.
- CRUD ответов: добавить/редактировать/удалить варианты ответов.
- Возможность отметить один правильный вариант (single-choice) или несколько (multi-choice) — на выбор проекта.
- Публикация теста: тест имеет статус черновик/опубликован.
- Ссылка на прохождение: уникальный URL вида /quiz/{public_id}.

5. Требования к редактору вопросов (React UI)

- Удобный интерфейс добавления вопросов.
- Динамическое добавление/удаление вариантов ответа.
- Валидация: вопрос не может быть пустым; минимум 2 варианта ответа; должен быть выбран правильный ответ.
- Предпросмотр теста (опционально).

6. Функциональные требования (страница прохождения теста) — опционально

- Студент открывает ссылку и видит тест.
- Проходит вопросы и отправляет ответы.
- Система автоматически проверяет ответы и показывает результат: количество правильных, процент, оценка (опционально).
- Сохранение попытки прохождения (опционально): кто проходил, когда, результат.

7. Структура данных (MySQL, минимум)

- tests: id, title, description, status(draft/published), public_id, created_at, updated_at.
- questions: id, test_id, text, type(single/multi), position, created_at.
- answers: id, question_id, text, is_correct(0/1), position.
- attempts (опционально): id, test_id, user_name/anon_id, started_at, finished_at, score.
- attempt_answers (опционально): attempt_id, question_id, answer_id.

8. API и логика проверки

- Backend должен предоставлять REST API для админки (CRUD тестов/вопросов/ответов).
- Публичный endpoint для получения теста по public_id.
- Если реализована страница прохождения: endpoint для отправки ответов и расчёта результата.
- Проверка: сравнение выбранных ответов с is_correct.

9. Технические требования

- Frontend: React (админ-панель обязательно).
- Backend: Node.js или PHP.
- База: MySQL.
- Защита админки авторизацией.
- README: установка, миграции/создание таблиц, запуск frontend/backend, тестовый аккаунт админа.

10. Ограничения

- Личный кабинет студента не требуется (достаточно доступа по ссылке).
- Сложные типы вопросов (соответствие, ввод текста) — не обязательны.
- Достаточно 1–2 типов вопросов: single-choice или multi-choice.

11. Результат работы

- Рабочая админ-панель: создание теста → вопросы → ответы → публикация → ссылка.
- Демонстрация на тесте минимум из 5 вопросов.
- Опционально: страница прохождения и автоматическая проверка.
- Исходный код и база данных.

12. Критерии оценки

- Корректный CRUD тестов/вопросов/ответов в MySQL.
- Удобный React-редактор вопросов (динамические варианты, валидация).
- Стабильная работа ссылок на тест и публикации.
- Аккуратный UI и структура проекта.
- Опционально (плюс): сохранение попыток и результаты прохождения.