Негосударственное аккредитованное некоммерческое частное образовательное учреждение высшего образования
«Академия маркетинга и социально-информационных технологий – ИМСИТ»
(г. Краснодар)

Академический колледж

Факультет цифровых технологий и кибербезопасности

ОТЧЕТ

О ПРОХОЖДЕНИИ СЕМЕСТРОВОЙ ПРАКТИКИ
МДК.01.02 Поддержка и тестирование программных модулей

Специальность 09.02.07 Информационные системы и программирование на базе Академического колледжа НАН ЧОУ ВО Академия ИМСИТ, г. Краснодар

составил обучающийся Розевика Артём Сергеевич

4 курса, группы 22-СПО-ИСиП-03

Руководители практики:

От академии _____ Кочура А.Н.

Отчет защищен с оценкой _____

«____» _____ 2025 г.

Краснодар

СОДЕРЖАНИЕ

1 Разработка технического задания

1.1 Область применения и назначение

Веб-приложение «Система записи на приём к врачу» предназначено для автоматизации процессов планирования, редактирования и управления медицинскими приёмами.
Система используется пациентами, врачами и администраторами медицинских учреждений для организации работы расписания и онлайн-записи.

Приложение позволяет:

- пользователям (пациентам) – записываться на приём к выбранному врачу в доступное время;
- администраторам – управлять врачами, их специализациями и расписанием приёмов;
- обеспечивать централизованное хранение данных о расписаниях и записях в реляционной базе данных.

Система разрабатывается для эксплуатации в локальной сети учреждения и через интернет.

1.2 Термины и определения

– пациент – пользователь системы, осуществляющий запись на приём;

– администратор – пользователь, управляющий данными врачей и расписанием;

– расписание врача – перечень доступных временных слотов для записи на приём;

– CRUD-операции – операции создания, чтения, редактирования и удаления данных.

1.3 Нормативные ссылки

– ГОСТ 19.201–78 – Техническое задание. Требования к содержанию и оформлению;

– ГОСТ 19.402–78 – Описание программы;

– ГОСТ 19.301–79 – Программа и методика испытаний;

– ГОСТ 19.106–78 – Общие требования к оформлению документов;

– ISO/IEC 25010:2011 – Модель качества программного обеспечения.

1.4 Технические требования

1.4.1 Функциональные требования

Таблица 1 – Функции требования

№	Функция	Описание	Приоритет
1	Регистрация пользователей	Возможность регистрации пациентов с проверкой данных	Обязательное
2	Аутентификация	Авторизация пользователей с проверкой логина и пароля	Обязательное
3	Просмотр расписания	Отображение списка врачей и информации о специализации	Обязательное
4	Запись на приём	Создание записи с проверкой доступности времени	Обязательное
5	Управление врачам	CRUD-операции для администраторов	Обязательное
6	Просмотр своих записей	RUD-операции для администраторов (добавление, редактирование, удаление)	Желательно
7	Админ-панель	Возможность пользователя просматривать свои активные записи	Обязательное

1.4.2 Нефункциональные требования

Производительность

– время отклика сервера при 50 одновременных запросах не более 500 мс. Для обеспечения высокой производительности будут использоваться такие подходы, как оптимизация запросов к базе данных и эффективная работа с SQLite.

Надёжность

– система должна быть устойчива к ошибкам ввода и сбоям. В случае сбоев сервер должен корректно обрабатывать ошибки и возвращать соответствующие HTTP-статусы. Время восстановления после сбоя не должно превышать 5 минут, с использованием корректной обработки исключений и валидации данных.

Безопасность

– хранение паролей в зашифрованном виде с использованием алгоритмов хеширования (bcrypt.js);

– валидация данных на клиентской и серверной сторонах (например, использование библиотек для валидации Express);

– авторизация с использованием JWT-токенов для обеспечения безопасности сессий;

– защита API endpoints с помощью middleware аутентификации.

Масштабируемость

– система должна быть масштабируемой для интеграции с дополнительными модулями в будущем, а также предусматривать возможность миграции на другие СУБД при увеличении нагрузки.

Совместимость

– приложение должно корректно работать в современных браузерах (Chrome, Firefox, Edge);

– поддержка ОС Windows и Linux;

– использование SQLite как встроенной системы управления базами данных;

– кроссплатформенность благодаря использованию Node.js.

1.5Внешние интерфейсы

GUI:

– веб-интерфейс с навигацией по разделам: "Главная", "Врачи", "Мои записи", "Администрирование";

– используется Express.static для обслуживания статических файлов (HTML, CSS, JS).

API:

– REST API для взаимодействия клиентской и серверной частей;

– пример эндпоинта создания записи:

POST /api/appointments

```json
{
	"doctor_id": 5,
	"appointment_date": "2025-03-10",
	"appointment_time": "11:00"
}
```

Форматы данных:

– JSON – для передачи данных, записи будут создаваться через POST-запросы, а информация будет храниться в созданном файле через SQLite.

1.6 Требования к документированию и сопровождению

В комплект документации должны входить:

– техническое задание (ТЗ) – для описания требований и функциональности системы;

– описание программы, включая структуру, архитектуру и алгоритмы. Программа должна следовать архитектуре Model-View-Controller (MVC), где серверная часть отвечает за обработку запросов, работу с базой данных и представление данных;

– программа и методика испытаний / План тестирования, включая описание того, как будет проводиться юнит-тестирование для серверной части (с использованием Jest или Mocha) и тестирование API;

– протоколы испытаний и отчёт о тестировании;

– руководство пользователя и Руководство администратора – для обеспечения правильной работы конечных пользователей и администраторов;

– руководство по установке системы (включая настройку Node.js и инициализацию базы данных SQLite);

– реестр версий и изменений – для отслеживания изменений в проекте.

Документация должна быть оформлена по ГОСТ 19.*, сохраняться в папке docs/ в форматах .docx и .pdf.

Архитектура системы следует паттерну Model-View-Controller (MVC):

– модели для работы с данными (User, Doctor, Appointment);

– контроллеры для обработки бизнес-логики;

– маршруты для обработки HTTP-запросов;

– middleware для аутентификации и обработки ошибок.

1.7 Критерии приёмки

Программа считается принятой, если выполнены следующие условия:

– реализованы все обязательные функциональные требования (регистрация пользователей, запись на прием, управление врачами и расписанием);

– приложение корректно работает при вводе валидных и невалидных данных. В случае ошибок система должна возвращать подробные сообщения

об ошибках с соответствующими HTTP-статусами (например, 400 – для нев лидных данных);

– все тест-кейсы из плана испытаний выполнены успешно, без критических дефектов. Юнит-тесты для серверной части и тестирование REST API должны быть проведены и результат должен быть удовлетворительным;

– приложение работает в пределах установленной производительности, время отклика при 50 одновременных запросах не более 200 мс;

– приложение должно быть устойчиво к сбоям и иметь систему восстановления в случае падений.

1.8 Смета и сроки выполнения

Таблица 2 – сроки выполнения

Этап	Содержание	Срок	Результат
1	Сбор и анализ требований	1 неделя	Утверждённое ТЗ
2	Проектирование архитектуры и БД	1 неделя	ER-диаграммы, UML
3	Разработка и модульное тестирование	1 недели	Исходный код, unit-тесты
4	Интеграция и системное тестирование	1 неделя	Отчёт о тестировании
5	Подготовка документации и сдача проекта	0,5 недели	Репозиторий и PDF-документы

Общая продолжительность: около 4 недель.

1.9 Приложения

Пример экранных форм пользовательского интерфейса

– включает эскизы интерфейсов для пользователей (пациентов, врачей, администраторов).

Фрагменты JSON-запросов и ответов

– примеры API-запросов для записи на прием, получения списка врачей, изменения расписания.

ER-диаграмма структуры базы данных

– диаграмма, отражающая связи между таблицами для хранения данных о пользователях, врачах, расписаниях и записях.

Пример протокола тестирования

– примерные тесты для проверки функционала API, валидации данных и корректности работы с базой данных.

Схема взаимодействия модулей (диаграмма компонентов UML)

– модульная диаграмма, показывающая взаимодействие клиентской и серверной частей приложения, а также работу с базой данных.