МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н. И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

Кафедра математического обеспечения и суперкомпьютерных технологий

УЧЕБНЫЙ КУРС

«Проектирование и архитектура программных систем»

для подготовки по направлению 09.03.04 «Программная инженерия»

СТРУКТУРА ПРОЕКТА

«QA-система на основе ИИ»

Выполнили студенты группы 3822Б1ПР1
Ворошилов Виталий Александрович, Крылов Михаил Георгиевич,
Моисеев Артём Владимирович, Морозов Егор Алексеевич,
Рамс Сергей Никитич

Проверил к.т.н., доцент
Лебедев Илья Генадьевич

Нижний Новгород
2025

Содержание

1. Рамки проекта

1.1. Матрица компромиссов проекта

- Ресурсы:
  - 1 руководитель/аналитик
  - 2 бекенд-разработчика
  - 1 ML-разработчик
  - 1 веб-разработчик
- Время: 3 месяца (октябрь-декабрь)
- Возможности:
  - поиск ответов из текста по вопросу (QA-система)
  - взаимодействие через веб-интерфейс
  - взаимодействие через Telegram-бот
  - логгирование для анализа и повышения качества обслуживания
  - поддержка русского и английского языков
- Матрица компромиссов:

Таблица 1.1.1 – Матрица компромиссов

	Фиксируется	**Согласовывается**	**Принимается**
Ресурсы	X		
Время		X	
Возможности			X

1.2. Вехи проекта

1. Завершение анализа требований и проектирования архитектуры
   - Утверждена концепция и рамки проекта
   - Подготовлены данные для обучения модели
   - Определена архитектура системы
2. Обучение модели
   - Поиск датасета
   - Обучение и тестирование
   - Валидация и тюнинг
3. Разработка серверной части, веб-приложения и Telegram-бота
4. Комплексное тестирование и демонстрация решения

1.3. Сметы проекта

Таблица 1.3.1 – Сметы проекта

	Требуемое количество ресурса	**Тариф на вид ресурса**	**Общая стоимость ресурса**
Руководитель-аналитик	1	40 рабочих часов	40 рабочих часов
Бекенд-разработчик	2	40 рабочих часов	80 рабочих часов
ML-разработчик	1	40 рабочих часов	40 рабочих часов
Веб-разработчик	1	40 рабочих часов	40 рабочих часов

Общая стоимость: 200 рабочих часов.

1.4. План-график проекта

Таблица 1.4.1 – План-график проекта

	Сентябрь 2025	**Октябрь 2025**	**Ноябрь 2025**	**Декабрь 2025**
Анализ требований и проектирование архитектуры	+	+		
Разработка модели		+		
Разработка бекенда, веб-приложения и Telegram-бота		+	+	
Комплексное тестирование и демонстрация решения				+

2. Роли и ответственности

2.1. Знания, умения и навыки

- Руководитель/аналитик:
  - Управление проектами, анализ требований, планирование сроков и ресурсов
  - Формулирование целей и критериев успеха, координация работы команды
  - Контроль качества результатов
- Бекенд-разработчик:
  - Go, Python, gRPC, Docker
  - Разработка и тестирование REST API
  - Интеграция ML-модели, логирование и мониторинг
- ML-разработчик:
  - Python, PyTorch, Transformers, Datasets
  - Fine-tuning и валидация языковых моделей (QA)
  - Работа с GPU, контейнеризация, анализ метрик (F1, EM)
- Веб-разработчик:
  - HTML, CSS, JavaScript, Vue.js
  - Создание адаптивных интерфейсов и интеграция с REST API
  - Оптимизация UX/UI и производительности
- Общие компетенции:
  - Владение Git, командная работа, ответственность
  - Знание английского языка для чтения документации

2.2. Структура команды

- руководитель/аналитик: Рамс Сергей Никитич
- 2 бекенд-разработчика: Ворошилов Виталий Александрович, Морозов Егор Алексеевич
- ML-разработчик: Моисеев Артём Владимирович
- веб-разработчик: Крылов Михаил Георгиевич

3. Протоколы проекта

3.1. Управление конфигурацией

1. Методы и средства
   - Код: Git (с размещением на GitHub) с ветвлением (GitFlow), Code Review и CI/CD
   - Документация: Git
   - Среда разработки: Docker для контейнеризации
   - Изменения: GitHub Issues для отслеживания запросов на изменение
2. Процесс запроса и принятия изменений
   - Запрос: создание Issue в GitHub с описанием и обоснованием
   - Анализ: оценка воздействия менеджером
   - Утверждение: решение утверждается командой
   - Реализация: Разработка в отдельной ветке в Git с обязательным код-ревью
   - Верификация: Тестирование в CI/CD и закрытие Issue после успешного внедрения
3. Роли и ответственность
   - Менеджер проекта: управляет процессом, возглавляет утверждение изменений
   - Разработчики: следуют процессу ветвления и код-ревью
   - QA: проверяют изменения в тестовых средах
4. Требования к системе контроля версий (Git)
   - Защита веток: прямые пуши в master-ветку запрещены
   - Code Review: обязательный Pull Request с минимум одним апрувером
   - CI-интеграция: автоматическая сборка и тесты перед слиянием
   - Структура: четкая модель ветвления (например, GitFlow) и понятные названия коммитов

3.2. Управление изменениями

1. Процесс. Универсальный процесс для всех изменений:
   - Подача. Заполнение запроса на изменение.
   - Анализ. Оценка влияния на сроки, бюджет, содержание и качество.
   - Решение. Рассмотрение и утверждение/отклонение менеджементом.
   - Реализация. Внесение изменений в план проекта и работы после утверждения.
2. Форма запроса на изменение. Стандартная форма должна включать:
   - Инициатора и даты начала и завершения работ

- Описание изменения (причина, что менять, желаемый результат)
- Обоснование и приоритет
- Оценка выгод и рисков

3. Роли и ответственности
   - Инициатор: подает запрос
   - Команда: оценивает усилия и технические риски.
   - Руководитель проекта: координирует процесс, ведет журнал изменений, утверждает или отклоняет изменения (включает менеджера проекта, представителя заказчика, технического лидера)

4. Изменения, влияющие на контракт. Любое изменение, затрагивающее бюджет, сроки или содержание, требует официального пересмотра контракта. Используется «треугольник компромиссов»:
   - При увеличении Возможностей - пересматриваются Время или Ресурсы.
   - При сжатии Времени - увеличиваются Ресурсы или уменьшаются Возможности.

3.3. Управление внедрениями

1. Подготовка
   - Настройка окружений: создание идентичных тестовых и рабочего окружений с помощью Docker
   - Автоматизация CI/CD: настройка конвейера для автоматической сборки, тестирования и развертывания артефактов
   - Управление конфигурацией и данными: проверка конфигурации, подготовка и тестирование скриптов миграции БД
2. Пилотное внедрение
   - Цель: Проверка в реальных условиях на ограниченной группе пользователей.
   - Производится развертывание на пилотную группу, обучение и сбор обратной связи, пристальный мониторинг и поддержка.
   - Принятие решение о полном запуске зависит от результатов анализа успешности этого внедрения.
3. Финальное внедрение
   - Развертывание начинается с готовым планом отката и предварительным уведомлением пользователей о времени проведения работ.
   - Усиленный мониторинг в первые 24-48 часов.

3.4. Достижение качества проекта

- Ожидания к качеству решения:
  - Функциональность - Модель корректно отвечает на вопросы по заданному тексту; интерфейсы (веб и Telegram) поддерживают полный цикл взаимодействия.
  - Точность ответов - F1 > 0.70, EM > 0.40 на тестовой выборке SberQUAD (или аналогичной).
- Процесс проверки качества
  - Ручное тестирование веб-приложения и Telegram-бота.
  - Оценка качества модели: Запуск на выделенной тестовой выборке из SberQUAD (или её подмножества). Расчёт метрик EM и F1 с использованием официального скрипта SquAD.
  - Мониторинг в runtime: Логирование времени обработки, ошибок, статусов.
- Роли в процессе достижения качества
  - Руководитель проекта/аналитик - формулирует ожидания заказчика, утверждает критерии приемки, принимает решение о готовности релиза.
  - ML-инженер/Data Scientist - обеспечивает качество модели: дообучение, оценка метрик, интерпретируемость ответов.
  - Backend-разработчик, веб-разработчик - контроль за корректностью бэкенда, веб-интерфейса, веб-приложения и Telegram-бота.

3.5. Рабочая среда проекта

Организационные требования:

- Команда: проект реализуется небольшой распределённой командой (5 человек), работающей удалённо.
- Коммуникация: Ежедневные stand-up (15 мин, в Telegram).
- Планирование спринтов и ретроспективы - раз в 2 недели.
- Все обсуждения ведутся в Telegram-чате проекта. Каждый участник проекта обеспечивает собственное рабочее место, но с минимальными требованиями:
- Компьютер: современный ноутбук (Intel i5 / Ryzen 5 или выше, 16 ГБ ОЗУ, SSD). Для ML-инженера желательно наличие GPU (NVIDIA, 6+ ГБ VRAM) или доступ к облачному GPU (Colab Pro, Yandex DataSphere).

- Интернет: стабильное подключение > 20 Мбит/с (для загрузки моделей, деплоя, видеоконференций).
- Требования к инструментам и системам: система контроля версий Git + GitHub Среда разработки VS Code / PyCharm; Jupyter Notebook (для экспериментов с моделью).