ДАТА УТВЕРЖДЕНИЯ

I ПАСПОРТ ПРОЕКТА

Наименование проекта	*Распознавание дефектов растений*
Шифр проекта (команды)	*Грядка*
Заказчик проекта	*Кафедра 806 «Вычислительная математика и программирование»*
Руководитель темы от МАИ	*Ухов Пётр Александрович*
Целевая аудитория результата проекта (кто потребитель результата проекта)	*Люди занимающиеся выращиванием разлиных культур*
Длительность проекта (даты начала и окончания)	*16.11.2024 - 30.05.2025*
РОЛИ В ПРОЕКТЕ:	ФИО
Teamlead	*Хныченко Артём Викторович*
Backend-разработчик: Python	*Ломакин Даниил Сергеевич*
Data Engineer, DevOps	*Венков Кирилл Сергеевич*
Backend-разработчик: Python	*Мустафаев Алим Рустемович*
QA Lead	*Мазепа Илья Алексеевич*
QA Assistant	*Патрикеева Ольга Вячеславовна*
Frontend-разработчик	*Васильева Екатерина Александровна*
Technical Writer	*Ляшко Зоя Николаевна*
Lead ML-Engineer	*Амиров Гайсар Ринатович*
Data Analyst / Assistant ML-Engineer	*Шестаков Константин Русланович*
Дата создания первой версии паспорта проекта	*15.12.2024*

Ссылки на ресурсы проекта

Ссылка на GitHub	https://github.com/khnychenkoav/PlantDiseaseDetector
Ссылка на трекер задач	https://confusion-crane-3bf.notion.site/1b87615558e98002a82ac48bd12dd30d?v=1b87615558e981b99dd7000cd235eb6a

II ОПИСАНИЕ ПРОЕКТА

Образ результата	*Веб-сайт для определения заболевания растения по изображению, предложение возможных причин и лечения.*
Цель проекта	*Обучить модель, которая сможет выделять из изображения растения конкретные признаки и по этим признакам распознавать тип заболевания.*
Задачи проекта	
Разработка Backend-сервисов	
1	*С помощью FastAPI настроить маршруты.*
2	*Разработать модуль обработки изображений:* • *Обеспечивает безопасную загрузку изображений.* • *Валидация формата изображения.* • *Использует предобученную модель для классификации заболеваний растений.* • *Подключение к ML модели через REST API или напрямую.*
3	*Разработать модуль рекомендаций:* • *На основе базы знаний связывает выявленные заболевания с описаниями и рекомендациями.* • *Возможность обновления базы знаний.*
4	*Разработать систему управления пользователями:* • *Регистрация и авторизация пользователей.* • *Возможность просмотра истории загрузок.*
Разработка Frontend-интерфейса	
5	*Создать пользовательский интерфейс с использованием React.js для отображения результата анализа в удобном интерфейсе.*
6	*Создать систему управление пользователями:* • *Регистрация и авторизация.* • *Просмотр истории загрузок и результатов.*
Анализ данных и машинное обучение	
7	*Реализовать предобработку изображений для обучения модели.*
8	*Обучение модели для анализа изображений растений:* • *Классификация заболеваний растений по*

	изображениям. • *Предсказание заболеваний.*
9	*Интеграция с Backend:* • *Предоставление API для получения результатов анализа.* • *Логирование и мониторинг вызовов модели.*
10	*Оптимизация модели:* • *Уменьшение размера модели для ускорения работы.* • *Возможность переобучения на новых данных.*
Хранение данных	
11	*PostgreSQL для хранения информации о пользователях, результатах анализа и рекомендациях.*
Обеспечение качества и тестирование	
12	*Автоматизировать тестирование API с использованием Postman, PyTest или Karate для проверки корректности работы сервисов.*
13	*При необходимости провести нагрузочное тестирование с помощью Apache JMeter для оценки производительности системы.*
Документирование и мониторинг	
14	*Создать и поддерживать актуальную документацию API с использованием OpenAPI/Swagger для облегчения интеграции и поддержки.*
15	*Поддерживать документацию проекта в течение всей разработки, а также создать пользовательский гайд по использованию*
Результат проекта (конкретный итоговый результат проекта)	*Работающий веб-сервис: пользователь может загрузить изображение растения и получить результаты анализа с описанием заболеваний и рекомендациями по лечению.* *Высокоточная модель машинного обучения: предобученная модель, адаптированная для классификации наиболее распространённых заболеваний растений, точность классификации достигает 90%+ на тестовом наборе данных.*
Ограничения и допущения, которые имеют или могут оказать существенное влияние на	*Ограничение: Временной бюджет.* *Допущение: опредления заболевания для одной*

результат проекта	*конкретной культуры.*
Необходимые ресурсы для выполнения проекта (компетенции исполнителей, материальные ресурсы и др.)	*Компетенции: Глубокое понимание машинного обучения, особенно в области компьютерного зрения* *Материальные ресурсы: Высокопроизводительные серверы с поддержкой GPU для обучения и тестирования моделей, наборы данных для обучения моделей* *Технические ресурсы: Инструменты мониторинга и логирования, доступ к инструментам тестирования и анализа производительности модели.*
Риски проекта	*Высокая вероятность ошибок при распознавании схожих заболеваний или редких случаев.* *Высокие затраты на GPU и облачные вычисления, особенно на этапе обучения и тестирования.* *Возможные сбои в работе REST API при высокой нагрузке.*

III КОМАНДА ПРОЕКТА

ФИО	Роль	Компетенция	Задача проекта
Хныченко Артём Викторович	Teamlead	1. Работа с инструментами контейнеризации (Docker). 2. Настройка DevOps-практик (CI/CD, GitHub Actions). 3. Документирование API с использованием OpenAPI/Swagger.	1. Выбор технологий: использование Docker для контейнеризации, Kubernetes для оркестрации, Kafka для передачи данных в реальном времени. 2. Внедрение инструментов мониторинга (Prometheus и Grafana) для оценки производительности компонентов. 3. Контроль использования OpenAPI/Swagger для документирования API.
Ломакин Даниил Сергеевич	Backend-разработчик: Python	1. Разработка REST API с использованием FastAPI.	1. Разработка REST API через FastAPI для передачи данных в ML. 2. Разработка модуля обработки изображений 3. Создания unit-тестов для всех маршрутов
Венков Кирилл Сергеевич	Data Engineer, DevOps	1. Настройка CI/CD процессов. 2. Развертывание серверной инфраструктуры. 3. Обеспечение масштабируемости и надежности системы.	1. Настроить контейнеризацию приложений (Docker). 2. Настроить автоматическое развертывание и мониторинг серверов. 3. Оптимизировать производительность системы.
Мустафаев Алим Рустемович	Backend-разработчик: Python	1. Разработка высокопроизводительных серверов и API. 2. Разработка REST API с использованием FastAPI. 3. Работа с базой данных PostgreSQL	1. Разработать модуль рекомендаций, подключить базу данных PostrgeSQL, используя ORM SQLAlchemy, настройка миграций через Alembic 2. Разработать модуль для авторизации и регистрации пользователей, используя JWT.
Мазепа Илья Алексеевич	QA Lead	1. Разработка тест-кейсов и плана тестирования. 2. Проведение	1. Составить план тестирования API и фронтенда. 2. Обеспечить контроль

		функционального и регрессионного тестирования.	качества системы.
Патрикеева Ольга Вячеславовна	QA Engineer	1. Поддержка тестирования под руководством QA-лида. 2. Проведение тестирования отдельных модулей системы.	1. Проверить работу ML-модели в интеграции с backend. 2. Участвовать в тестировании фронтенда.
Васильева Екатерина Александровна	Frontend-разработчик	1. Разработка пользовательского интерфейса. 2. Интеграция с API backend-части. 3. Обеспечение адаптивности и удобства использования	1. Реализовать загрузку изображений с клиента. 2. Настроить отображение информации о диагнозах и рекомендациях. 3. Тестировать и оптимизировать UI.
Ляшко Зоя Николаевна	Technical Writer	1. Создание и поддержка документации для проекта. 2. Подготовка инструкций для пользователей и технического описания системы.	1. Составить техническую документацию для API. 2. Написать руководство по установке и использованию системы.
Амиров Гайсар Ринатович	Lead ML-Engineer	1. Выбор подходящей архитектуры модели и ее обучение. 2. Анализ датасета и его подготовка. 3. Интеграция ML-модели с backend-частью.	1. Провести анализ датасета и подготовить данные. 2. Разработать модель для классификации болезней растений. 3. Настроить тестирование и валидацию модели.
Шестаков Константин Русланович	Data Analyst / Assistant ML-	1. Подготовка и анализ данных под руководством лидера.	1. Подготовить и аннотировать дополнительные данные. 2. Помочь с визуализацией результатов работы модели.

	Engineer	2. Участие в разработке и оптимизации модели.	

IV ЗАДАЧИ ПРОЕКТА (ОЦЕНКА ПО ВРЕМЕНИ), ПРОГРЕСС

№	Название задачи	Подзадачи	Время (в часах)
Этап 1: Настройка окружения и базовая архитектура (2 недели)			
Сводка по этапу:			
Статус этапа: Выполнено Всего задач: 14 Backend: 3 задачи, 40 часов Frontend: 3 задачи, 30 часов ML: 2 задачи, 20 часов QA: 2 задачи, 20 часов DevOps: 2 задачи, 20 часов Документация: 2 задачи, 10 часов В процессе: 0 Завершено: 14 Заблокировано: 0			
Backend			
1	Инициализация проекта FastAPI и структура	1. Создать структуру проекта: main.py, routes/, models/. 2. Настроить конфигурацию приложения (config.py). 3. Реализовать базовые роуты для проверки API.	12
2	Интеграция базы данных PostgreSQL	1. Подключить SQLAlchemy. 2. Создать таблицы для пользователей и болезней. 3. Настроить миграции через Alembic.	16
3	Документация API через Swagger/OpenAPI	1. Автоматически сгенерировать документацию для базовых роутов. 2. Описать поля для моделей данных.	12
Frontend			
4	Инициализация проекта React	1. Настроить проект. 2. Создать базовый каркас приложения: App.js, навигация.	8
5	Дизайн интерфейса для загрузки изображений	1. Разработать макет формы загрузки. 2. Реализовать начальный компонент формы загрузки.	12
6	Интеграция с API	1. Подключить библиотеку Axios для запросов. 2. Реализовать тестовый запрос к backend (GET).	10

ML			
7	Настройка окружения для разработки модели	1. Создать структуру для проекта. 2. Установить зависимости (TensorFlow, PyTorch).	8
8	Анализ доступного датасета	1. Исследовать и визуализировать данные. 2. Разработать план предобработки изображений.	12
QA			
9	Подготовка тест-кейсов для базовой проверки API	1. Написать тест-кейсы для проверки ответа API. 2. Проверить работу Swagger-документации.	10
10	Тестирование фронтенда	1. Проверить корректность и адаптивность каркаса интерфейса. 2. Передать баги в разработку.	10
DevOps			
11	Настройка Docker для backend и frontend	1. Написать Dockerfile для backend. 2. Написать Dockerfile для frontend.	12
12	Настройка docker-compose	1. Настроить docker-compose.yml для разработки и тестирования.	8
Документация			
13	Описание архитектуры проекта	1. Создать диаграммы взаимодействия компонентов. 2. Создание README для backend и frontend	10
Этап 2: Разработка API анализа изображений и базовой модели (2 недели)			
Сводка по этапу:			
Статус этапа: Выполнено Всего задач: 13 Backend: 3 задачи, 40 часов Frontend: 3 задачи, 40 часов ML: 2 задачи, 30 часов QA: 2 задачи, 20 часов DevOps: 1 задача, 10 часов Документация: 2 задачи, 10 часов В процессе: 0 Завершено: 12 Заблокировано: 1			

Backend			
14	Разработка API для загрузки изображений	1. Создать роут для загрузки изображений.	16
15	Организация хранилища изображений	1. Сохранять загруженные изображения с уникальными идентификаторами.	12
16	Подготовка API для анализа изображений	1. Реализовать предварительный вызов ML-модели через FastAPI. 2. Обрабатывать и возвращать результаты анализа.	12
Frontend			
17	Форма загрузки изображений	1. Завершить разработку интерфейса для загрузки.	16
18	Отображение результатов анализа	1. Реализовать компонент для визуализации результатов. 2. Подключить данные через Axios.	12
19	Начальная интеграция карты огорода (заблокировано)	1. Разработать прототип интерфейса карты. 2. Подключить тестовые данные.	12
ML			
20	Обучение базовой модели	1. Реализовать предобработку изображений 2. Обучить модель классификации болезней.	20
21	Экспорт модели	1. Преобразовать обученную модель в удобный формат.	10
QA			
23	Тестирование API загрузки изображений	1. Освоить написание тестов для модели. 2. Проверить корректность сохранения.	10
24	Проверка формы загрузки на фронтенде	1. Проверить корректность работы формы. 2. Задокументировать результаты тестирования.	10
DevOps			
25	Настройка CI/CD для backend	1. Начало автоматизации сборки и деплоя backend. 2. Настройка тестового окружения для API.	10
Документация			

26	API-документация для загрузки изображений	1. Описание эндпоинта загрузки и его параметров.	6
27	Руководство по работе с моделью ONNX	1. Описание шагов экспорта и тестирования модели.	4
Этап 3: Функционал рекомендаций (2 недели)			
Сводка по этапу:			
Статус этапа: В процессе Всего задач: 11 Backend: 2 задачи, 40 часов Frontend: 2 задачи, 40 часов ML: 2 задачи, 30 часов QA: 2 задачи, 20 часов DevOps: 1 задача, 10 часов Документация: 2 задачи, 10 часов В процессе: 7 Завершено: 4 Заблокировано: 0			
Backend			
28	Модуль рекомендаций	1. Создать таблицу данных о болезнях и рекомендациях. 2. Реализовать API для получения рекомендаций.	16
29	Доработка API и базы данных	1. Разработать оставшиеся роуты. 2. Доработать хранение данных в PostgreSQL, оптимизировать миграции.	24
Frontend			
30	Интерфейс рекомендаций	1. Создать компонент для отображения рекомендаций. 2. Подключить данные через API.	16
31	Интеграция	1. Завершение полной интеграции с backend.	24
ML			
32	Улучшение производительности модели	1. Оптимизация модели для более быстрого предсказания. 2. Тестирование на реальных данных.	16
33	Добавление новых данных в датасет	1. Аннотация новых изображений. 2. Обучение модели на расширенном датасете.	14
QA			

34	Тестирование рекомендаций	1. Проверить корректность получения рекомендаций из API. 2. Проверить отображение рекомендаций на фронтенде.	10
35	Тестирование карты огорода	1. Проверить функционал редактирования и сохранения. 2. Задокументировать баги и передать в разработку.	10
DevOps			
36	Настройка мониторинга	1. Оценить необходимость внедрения систем мониторинга.	10
Документация			
37	Описание функционала рекомендаций	1. Подробное описание API и структуры данных.	6
38	Инструкция для работы с картой огорода	1. Руководство для пользователей по управлению схемой.	4
Этап 4: Завершение и тестирование (2 недели)			
Сводка по этапу:			
Статус этапа: Не начат Всего задач: 16 Backend: 4 задачи, 40 часов Frontend: 3 задачи, 40 часов ML: 2 задачи, 20 часов QA: 3 задачи, 40 часов DevOps: 2 задача, 20 часов Документация: 2 задачи, 20 часов В процессе: 0 Завершено: 0 Заблокировано: 0			
Backend			
39	Тестирование и отладка API	1. Написать unit-тесты для всех маршрутов. 2. Провести нагрузочное тестирование API при необходимости.	16

40	Оптимизация производительности и безопасности	1. Проверить производительность запросов к базе данных. 2. Настроить ограничение размера запросов, убедиться в защите от атак.	12
41	Реализация простого логирования	1. Добавить логирование запросов и ошибок через встроенный logging модуль Python. 2. Настроить ротацию логов (ежедневное обновление файлов).	8
42	Интеграция финальной версии ML-модели	1. Обновить модель на сервере. 2. Провести финальное тестирование интеграции.	4
Frontend			
43	Тестирование и отладка интерфейса	1. Провести ручное тестирование всех компонентов интерфейса. 2. Исправить выявленные баги и несоответствия. 3. Убедиться, что интерфейс адаптивен для мобильных устройств.	20
44	Оптимизация загрузки данных	1. Настроить кэширование данных при необходимости. 2. Проверить корректную работу кэширования и удаления устаревших данных.	12
45	Финальная стилизация интерфейса	1. Привести стили компонентов к единому виду. 2. Убедиться, что цвета и шрифты соответствуют дизайну.	8
ML			
46	Финальная валидация модели	1. Проверить точность модели на тестовом наборе данных. 2. Исправить или улучшить предсказания, если требуется.	12
47	Обновление документации по модели	1. Описать все изменения и особенности работы финальной модели. 2. Подготовить короткую справку для разработчиков backend.	8
QA			

48	Финальное тестирование API	1. Провести полное функциональное тестирование всех маршрутов. 2. Убедиться в отсутствии уязвимостей.	16
49	Тестирование интеграции frontend и backend	1. Проверить корректность передачи данных между компонентами. 2. Убедиться, что ответы API корректно отображаются в интерфейсе.	16
50	Тестирование UX	1. Проверить, насколько интуитивен и понятен интерфейс. 2. Составить список предложений для улучшения.	8
DevOps			
51	Завершение настройки Docker	1. Создать production-ready Docker-контейнеры для backend и frontend. 2. Проверить сборку и развертывание на тестовом сервере.	10
52	Развертывание системы на сервере	1. Настроить сервер для размещения системы. 2. Проверить доступность API и интерфейса.	10
Документация			
53	Финализация технической документации	1. Описать работу API, архитектуру и структуру базы данных. 2. Создать руководство по развертыванию для разработчиков.	12
54	Подготовка пользовательского руководства	1. Описать процесс работы с интерфейсом. 2. Подготовить FAQ для пользователей.	8
ИТОГО ПЛАНИРУЕМОЕ ВРЕМЯ НА ПРОЕКТ :			**530**

V РЕСУРСЫ И МАТЕРИАЛЫ ПРОЕКТА

Используемые инструменты и технологии	
Backend	Языки программирования: 1. Python. Фреймворки и библиотеки: 1. FastAPI: Создание REST API для взаимодействия между компонентами. 2. OpenAPI/Swagger: Документирование API. Инструменты DevOps: 1. Docker: Контейнеризация микросервисов.
Frontend	Языки и фреймворки: 1. React.js: Разработка пользовательского интерфейса. 2. TypeScript/JavaScript: Основной язык для разработки фронтенда. Связь с Backend: 1. REST API: Получение данных с Backend-Python.
Обработка и хранение данных	Хранилище данных: 1. PostgreSQL: Реляционная база данных для хранения изображений, информации о пользователях и аналитики.
DevOps	Инфраструктура: 1. Docker: Контейнеризация всех компонентов. CI/CD: 1. GitHub Actions: Автоматизация тестирования и развертывания.
Тестирование	Инструменты для API: 1. Postman: Проверка и автоматизация API-запросов. 2. PyTest: Автоматическое тестирование эндпоинтов и логики.
Ссылки на внешние ресурсы	
Python: https://www.python.org/ FastAPI: https://fastapi.tiangolo.com/ru/	

OpenAPI/Swagger: https://www.openapis.org/

Docker: https://www.docker.com/

React.js: https://react.dev/

TypeScript: https://www.typescriptlang.org/

JavaScript: https://developer.mozilla.org/ru/docs/Web/JavaScript

REST API: https://developer.mozilla.org/ru/docs/Glossary/REST

TensorFlow: https://www.tensorflow.org/

PyTorch: https://pytorch.org/

Pandas: https://pandas.pydata.org/

PostgreSQL: https://www.postgresql.org/

CSV: https://ru.wikipedia.org/wiki/CSV

GitHub Actions: https://docs.github.com/ru/actions

Postman: https://www.postman.com/

PyTest: https://docs.pytest.org/en/latest/

Даты посещения ссылок: 10.12.2024 – 15.03.2025

Данные

Датасеты

Тренировочные данные	Источник: NewPlantDiseasesDataset(https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset) Описание: 1. Данные изображений растений для классификации заболеваний. 2. Содержат более 70,000 изображений листьев различных культур с различными типами заболеваний и здоровыми экземплярами. 3. Изображения разделены по классам. Параметры: 1. image: RGB-изображение листа растения. 2. Размер изображения: 256x256 пикселей (можно изменять). 3. Формат файла: JPEG/PNG. Использование: 1. Обучение моделей компьютерного зрения для диагностики заболеваний растений. 2. Применение transfer learning для улучшения классификации. Формат: JPEG, PNG.

Тестовые данные	Описание: 1. Данные, подготовленные для тестирования моделей. 2. Включают изображения растений, сделанные в реальных условиях (различное освещение, фон, углы съемки). 3. Содержат больше шума и сложности по сравнению с тренировочным набором. Параметры: Те же, что в тренировочных данных, с дополнительными полями. Использование: 1. Оценка точности классификационных моделей. 2. Проверка работы модели в реальных полевых условиях. 3. Анализ ошибок и дообучение модели для повышения устойчивости. Формат: JPEG, PNG.
Алгоритмы	
Обучение модели	Метод: Transfer Learning с предварительно обученными моделями (ResNet). Процесс: 1. Замена финального слоя модели на слой с числом выходов, равным количеству классов. 2. Замораживание базовых слоев и дообучение верхних слоев на данных. 3. Полное дообучение модели на данных для адаптации к реальным условиям.
Тестирование устойчивости модели	Метод: Использование сложных данных датасета. Процесс: 1. Оценка модели на реальных изображениях из датасета. 2. Анализ устойчивости к изменению условий (освещение, фон, углы съемки). 3. Дополнительное обучение на ошибочных предсказаниях для

	повышения точности.

VI КОММЕНТАРИИ И МЫСЛИ КОМАНДЫ
