## **Тестовое задание «Clinic Appointments»**

Соберите минимальный, но полноценный микросервис для записи пациентов и подготовьте его так, чтобы коллега мог развернуть проект за пару минут и сразу увидеть живой API.

### **1. Что нужно сделать**

**1.1 Сервис и бизнес-логика**

• Напишите приложение на **FastAPI** или **Flask**.

• Создайте модель **Appointment** и два эндпойнта:

– POST /appointments — создать запись;

– GET /appointments/{id} — получить запись по ID.

• В базе должна быть уникальная пара doctor\_id + start\_time, чтобы один врач не принимал двух пациентов одновременно.

**1.2 База и окружение**

• Используйте **MySQL** или же **Postgres** (в последних проектах мы использовали SuperBase); опишите окружение в docker-compose.yml (сервисы api и db).

• Схема создаётся автоматически при запуске или скриптом init.sql.

**1.3 Dockerfile**

• Базовый образ python:3.12-slim.

• Обязательные строки: USER app (никакого root) и HEALTHCHECK CMD curl -f http://localhost:8000/health || exit 1.

• Секреты и DSN только через переменные окружения; положите пример .env.example.

**1.4 Качество кода**

• black, isort, flake8 — 0 ошибок.

• Напишите как минимум один юнит-тест и один интеграционный тест на pytest.

**1.5 Makefile**

• Добавьте цели make lint и make test, чтобы стиль и тесты проверялись одной командой.

**1.6 CI/CD**

• Настройте **GitHub Actions** с одним workflow: шаг **lint** → шаг **tests**. Оба шага должны проходить.

**1.7 README**

• Инструкция «clone → cp .env.example .env → docker compose up -d –build → curl» должна занимать меньше минуты.

### **2. Док-артефакты (положите рядом с кодом)**

1. **Архитектурная схема**: модули сервиса и их связи.
2. **ER-диаграмма** или текстовое описание структуры БД.
3. **Activity-диаграмма** главного сценария «пациент записывается на приём».
4. **Документ бизнес-процесса**: кто, когда и что делает в системе; роли на чтение, изменение и удаление данных в какой-нибудь доске типа Miro
5. **Отдельный раздел «Проектирование → реализация»**: докажите, что сначала спроектировали, а потом писали код (короткий порядок шагов и принятые решения).

### **3. Telegram-бот с ИИ-подбором врача**

• Опишите, как бот спрашивает пациента, ищет свободное время и врача, затем вызывает API для создания записи.

• Реализовывать код бота **не обязательно**; достаточно текстового сценария и, при желании, stub-пакета с точкой входа bot.py, чтобы показать направление мысли.

• Укажите, какую ML-/LLM-библиотеку или внешнее API вы бы использовали для «умного» подбора.

### **4. Бонус (любой один пункт даёт +5 % к итоговой оценке)**

• Типизация mypy не ниже 90 %.

• После тестов CI собирает и пушит Docker-образ.

• k8s-манифест для деплоя.

• Расширенный Makefile: команды make up, make down, make migrate и т. д.

### **5. Ниже — критерии, по которым оценивается тест. Мы используем фиксированные проценты, чтобы все кандидаты были в равных условиях.**

• Compose поднимается без ошибок — **обязательное условие**.

• Образ non-root и все секреты через env — 15 %.

• CRUD работает, уникальность соблюдена — 15 %.

• Все тесты проходят локально и в CI — 15 %.

• Код без ошибок black/isort/flake8 — 10 %.

• Структура проекта понятна — 10 %.

• Цели Makefile выполняются — 10 %.

• README короткий и точный — 5 %.

• HEALTHCHECK корректен — 5 %.

• Один бонус-пункт выполнен — +5 %.

### **7. Короткий опрос (добавьте файл answers.txt)**

• Самооценка 0-10 по: Python, FastAPI/Flask, Docker/Compose, GitHub Actions, SQL, «код с ИИ», встраивание решений в бизнес-процессы.

• Английский: да / нет.

• Как трекаете задачи: Scrum, Kanban, другое (укажите).

### **8. Как отправить готовое задание**

1. **Создайте репозиторий на GitHub**.
2. Загрузите в него всё, что требуется:  
     
    – исходный код, Dockerfile, docker-compose, Makefile, README;  
     
    – все диаграммы и документы (архитектура, ERD, Activity, Miro-ссылка (или другая доска), bot-сценарий);  
     
    – файл answers.txt с опросом.
3. Отправьте ссылку на гит в чате когда тестовое будет выполнено
4. Срок сдачи — **до понедельника 7 июля** с момента получения задания. Если нужен другой дедлайн, сразу скажите.
5. Нет возможности работать с GitHub? Сложите те же файлы в zip-архив и пришлите ссылку на Google Drive / Я.Диск.

Есть вопросы — задавайте сразу, на все отвечу!