**1. Введение**

**1.1. Назначение**  
Разработать Telegram-бота для приема заказов от пользователей с возможностью загрузки файлов, назначения исполнителей и отслеживания статусов.

**1.2. Цель**  
Автоматизация процесса создания, обработки и управления заказами через Telegram.

**2. Требования к системе**

**2.1. Функциональные требования**

* **Регистрация пользователей**:
  + Разделение на роли: клиенты, исполнители, администраторы.
* **Создание заказа**:
  + Выбор предмета, типа работы, описания.
  + Загрузка до 10 файлов (до 20 МБ каждый).
* **Управление заказами**:
  + Просмотр списка заказов.
  + Принятие заказов исполнителями.
  + Отправка результата клиенту.
* **Администрирование**:
  + Назначение ролей.
  + Просмотр статистики.

**2.2. Нефункциональные требования**

* **Производительность**:
  + Поддержка до 1000 активных пользователей.
  + Время отклика ≤ 2 сек.
* **Безопасность**:
  + Валидация входящих файлов.
  + Ограничение доступа к командам по ролям.
* **Хранение данных**:
  + Состояния бота сохраняются между перезапусками (Redis/JSON).
  + Файлы заказов хранятся локально в zip-архивах.

**3. Технологический стек**

* **Язык**: Python 3.10+
* **Фреймворки**:
  + aiogram (для Telegram API)
  + aiohttp (для асинхронных запросов)
* **Хранение состояний**:
  + RedisStorage (продакшен) / JSONStorage (разработка)
* **База данных**: SQLite/PostgreSQL (для пользователей и заказов)
* **Дополнительно**:
  + APScheduler (для планировщика задач)
  + python-archive (для работы с zip-файлами)

**4. Архитектура**

**4.1. Компоненты**

* **Core**:
  + main.py – точка входа, инициализация бота.
* **Handlers**:
  + user\_handlers.py – команды пользователей.
  + order\_handlers.py – логика заказов.
  + admin\_handlers.py – админ-панель.
* **Services**:
  + order\_service.py – CRUD для заказов.
  + file\_service.py – управление файлами.
* **Storage**:
  + RedisStorage – для состояний FSM.
  + JSONStorage – для разработки.

**4.2. Схема взаимодействия**

Пользователь → Telegram Bot → Dispatcher → Handlers → Services → DB/Storage

**5. Пояснительная записка к коду**

**5.1. Структура проекта**

/bot

│── /handlers # Обработчики команд

│── /services # Бизнес-логика

│── /logs # Логи

│── /states # Состояния FSM

│── /utils # Логирование, клавиатуры

│── main.py # Запуск бота

**5.2. Ключевые механизмы**

* **FSM (Finite State Machine)**:
  + Состояния заказа: waiting\_for\_subject → waiting\_for\_files → completed.
  + Данные хранятся в Redis/JSON.
* **Работа с файлами**:
  + Файлы временно сохраняются в temp\_files (в рамках состояния), затем упаковываются в zip.
  + Каждый заказ имеет уникальную папку: orders/{order\_id}\_{user\_id}.
* **Безопасность**:
  + Проверка MIME-типов файлов.
  + Лимит размера (20 МБ).

**5.3. Структура БД.**

