**Паспорт проекта QuestionMark**

**Название проекта:**  
Telegram-бот "QuestionMark" для проведения опросов и сбора статистики по ответам пользователей.

**Краткое описание:**  
Бот позволяет пользователям создавать вопросы с несколькими вариантами ответов, отвечать на вопросы других, просматривать статистику по ответам, а также ставить лайки, дизлайки и жаловаться на вопросы. Реализована система ролей (незарегистрированный, зарегистрированный, модератор), хранение данных в SQLite через SQLAlchemy, логирование событий и ошибок.

**Необходимые библиотеки**

* **telebot** - для работы с Telegram API.
* **python-telegram-bot** - для асинхронного взаимодействия с Telegram (Application, MessageHandler, CommandHandler и др.).
* **sqlalchemy** - для ORM и работы с базой данных SQLite.
* **logging** - для логирования событий и ошибок.
* **datetime** - для работы с датой и временем.
* **json** - для хранения и загрузки вспомогательных данных (например, связей пользователей и вопросов).
* **os** (по необходимости) - для работы с файловой системой.
* **telegram** - для объектов Telegram (например, ReplyKeyboardMarkup).

**Общая структура кода по файлам**

**Файл 1 - main.py (DrAeyJ + Adamusas)**

**Назначение:**  
Основная логика Telegram-бота: запуск, обработка команд, регистрация пользователей, взаимодействие с вопросами, логирование.

**Основные компоненты:**

* Импорт библиотек и моделей.
* Настройка логирования (файлы логов для предупреждений и ошибок).
* Инициализация базы данных.
* Глобальные переменные для текущего режима, текущего вопроса и ответов.
* Асинхронные обработчики команд:
  + /start - запуск/перезапуск бота, регистрация пользователя, определение роли и соответствующих команд. - DrAeyJ
  + /help - вывод справки и списка доступных команд в зависимости от роли пользователя. - Adamusas
  + /get\_question - получение случайного вопроса, формирование клавиатуры для ответов. - DrAeyJ
  + /add\_question - добавление нового вопроса (реализуется отдельно). - Adamusas
  + /moderate - модерация вопросов (для модераторов). - Adamusas
* Функции для регистрации пользователя и работы с JSON-файлами связей.
* Обработка ответов на вопросы, подсчет статистики, лайков/дизлайков/жалоб.
* Обработка ошибок и исключений с логированием.

**Файл 2 - \_\_all\_models.py (DrAeyJ)**

**Назначение:**  
Описание ORM-моделей для пользователей и вопросов.

**Основные компоненты:**

* Импорт datetime и SQLAlchemy.
* Класс **User**:
  + id, telegram\_id, name, type (роль), счетчики активности, дата окончания блокировки.
* Класс **Questions**:
  + id, author\_tg\_id, content, варианты ответов (до 4), счетчики по каждому ответу, лайки, дизлайки, жалобы.
* Методы \_\_repr\_\_ для удобного вывода информации о моделях.

**Файл 3 - db\_session.py (Adamusas)**

**Назначение:**  
Инициализация и создание сессий SQLAlchemy для работы с базой данных.

**Основные компоненты:**

* Импорт SQLAlchemy.
* Создание базового класса моделей.
* Глобальная фабрика сессий.
* Функция **global\_init(db\_file)** - инициализация базы данных, создание таблиц.
* Функция **create\_session()** - получение новой сессии для работы с БД.

**Файл 4 - logger\_filter.py (DrAeyJ)**

**Назначение:**  
Расширение стандартной системы логирования Python: фильтрация сообщений по уровню логирования и расширенное форматирование исключений.

**Основные компоненты:**

* Класс **MaxLevelFilter(logging.Filter)** - фильтр, пропускающий только сообщения с уровнем не выше заданного (например, чтобы писать предупреждения в один лог, а ошибки - в другой).
* Класс **ExceptionFormatter(logging.Formatter)** - форматтер, который добавляет дополнительные отступы к трассировке исключения для лучшей читаемости в логах.
* Использование:  
  Эти классы применяются в main.py при настройке логирования, чтобы разделять логи по уровням и улучшать читаемость ошибок.