Техническое задание

Проект: «Клон Discord» — рабочий мессенджер с серверами и каналами

Важно: сделать максимально похожий по UX, без использования чужих логотипов/иконок/ассетов.

1. Цель работы

Разработать веб-приложение — аналог Discord: система «серверов» (сообществ), каналов (текст/голос), личных сообщений, ролей и прав доступа, обмена файлами и уведомлений. Приложение должно работать в реальном времени.

2. Модули проекта и уровень реализации

- MVP (обязательная часть): авторизация, серверы, текстовые каналы, личные сообщения, роли/права, WebSocket-чат, базовые уведомления.
- Advanced (по возможности): голосовые каналы (WebRTC), видеозвонки (опц.), стрим экрана (опц.), пуш-уведомления (опц.).

Требование «как в Discord» трактуется как: максимально похожий интерфейс и ключевые сценарии. Полный функциональный паритет с оригинальным Discord не обязателен для учебного проекта.

3. Роли и права

- Пользователь: общение, вступление на сервер по приглашению, отправка сообщений/файлов, реакции.
- Модератор: удаление сообщений, мут/бан пользователей, управление каналами (по правам).
- Владелец сервера: создание/удаление каналов, управление ролями, приглашения, настройки сервера.
- Администратор системы (опц.): управление пользователями/жалобами/логами на уровне всей платформы.

4. Основной сценарий пользователя

- Регистрация/вход.
- Просмотр списка серверов слева (как в Discord).
- Переход в сервер → выбор канала → общение в реальном времени.
- Личные сообщения (DM) между пользователями.
- Отправка файлов/картинок в чат.
- Создание приглашения (invite link) для входа на сервер.

5. Функциональные требования (MVP)

- Авторизация: регистрация, вход, выход, восстановление пароля (опц.).
- Серверы: создание сервера, редактирование названия/иконки, список участников.

- Приглашения: генерация ссылки-приглашения, вступление по ссылке, срок действия/лимит (опц.).
- Каналы: текстовые каналы (обязательно), категории каналов (опц.).
- Сообщения: отправка/получение в реальном времени (WebSocket), редактирование/удаление (по правам).
- Реакции: добавление реакции (emoji) к сообщению (минимум 5 стандартных).
- Упоминания: @user в тексте и подсветка (минимум).
- Поиск: поиск по сообщениям в текущем канале (минимум).
- Файлы: прикрепление изображений/файлов (ограничение размера).
- Статусы: онлайн/оффлайн/не беспокоить (минимум онлайн/оффлайн).

6. Голосовые каналы (Advanced, по возможности)

- Голосовые комнаты внутри сервера (подключение/отключение).
- Технология: WebRTC (допускается peer-to-peer для малых групп).
- Индикатор участников голосового канала.

Если WebRTC слишком сложно, допускается выполнить только MVP (текстовые каналы) при условии высокого качества UI/UX и стабильной работы WebSocket.

7. UI/UX требования

- Интерфейс максимально похож на Discord по расположению зон: серверы слева, каналы, чат, участники.
- Темная тема по умолчанию (как в Discord) + светлая тема (опционально).
- Адаптивность: Desktop обязателен; Mobile — упрощенный режим допускается.
- Индикация набора текста (typing indicator) — опционально.

8. Хранимые данные (MySQL, минимум)

- users: id, username, email/login, password_hash, avatar_url, status, created_at.
- servers: id, owner_id, name, icon_url, created_at.
- server_members: server_id, user_id, role_id, joined_at.
- roles: id, server_id, name, color (опц.), permissions_json.
- channels: id, server_id, type(text/voice), name, position, created_at.
- messages: id, channel_id, author_id, content, attachments_json (опц.), created_at, edited_at (опц.).
- reactions: id, message_id, user_id, emoji.
- dm_threads: id, user1_id, user2_id, created_at.
- dm_messages: id, thread_id, author_id, content, created_at.
- invites: id, server_id, token, created_by, expires_at (опц.), max_uses (опц.), uses_count.

9. Реальное время и API

- REST API для операций (серверы/каналы/профиль/роли).
- WebSocket для событий: сообщения, редактирование/удаление, реакции, статусы онлайн.
- Обязательный reconnect на фронтенде при разрыве соединения.

10. Технические требования

- Frontend: React (рекомендуется) или HTML/CSS/JS.
- Backend: Node.js (Express/Fastify) — рекомендуется для WebSocket.
- База данных: MySQL.
- Хранение файлов: локально на сервере или S3-совместимо (опционально).
- Безопасность: пароли только hash, проверка прав доступа на сервере, защита приватных каналов.
- README: установка, запуск, переменные окружения, миграции/seed.

11. Ограничения

- Запрещено использовать оригинальные ассеты Discord (логотипы, иконки, UI-ресурсы).
- Допускается визуальное сходство по структуре интерфейса и логике сценариев.
- Для учебного проекта допускается ограничение по нагрузке (например до 100–300 пользователей тестовой базы).

12. Результат работы

- Рабочее веб-приложение с серверами, текстовыми каналами и WebSocket-чатом.
- Демонстрация: 2 пользователя общаются в одном канале в реальном времени, работают роли/права.
- Приглашение по ссылке и вступление на сервер.
- Исходный код + база данных (миграции/seed).
- Опционально: голосовые каналы (WebRTC).

13. Критерии оценки

- Стабильность реального времени (WebSocket), корректная работа каналов и сообщений.
- Качество интерфейса (похожесть на Discord по UX, аккуратность).
- Роли и права (ограничения действий, приватные каналы по правам).
- Качество кода и структуры проекта, наличие README.
- Опционально (плюс): голосовые каналы, индикатор печати, пуш-уведомления.