**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**Проект**: Narxoz Social  
**Заказчик**: Университет Narxoz  
**Разработчики**: Лихачев Артем, Джетиберген Мадияр, Мукашев Марлен

**1. Общие сведения**

**1.1 Название проекта**  
Narxoz Social

**1.2 Краткое описание**  
Narxoz Social — это внутренняя социальная сеть университета Narxoz, разработанная для студентов, преподавателей, организаций (кафедр, клубов, отделов) и администраторов с целью заменить внешние мессенджеры (Telegram, WhatsApp) и минимизировать риски коррупционных действий. Сеть позволяет создавать посты, чаты (включая групповые), подписываться друг на друга, лайкать и комментировать публикации, а также управлять событиями (ивентами) с рассылкой уведомлений подписчикам.

**1.3 Цель проекта**

1. Обеспечить безопасную и прозрачную коммуникацию внутри университета, исключив необходимость использовать сторонние мессенджеры.
2. Создать единую платформу, в которой можно публиковать учебные, административные, развлекательные посты, а также проводить обсуждения в режиме реального времени.
3. Обеспечить инструменты для быстрой модерации и контроля со стороны университета (админов, модераторов).
4. Добавить функционал событий (ивентов) с календарём и уведомлениями для подписчиков.

**1.4 Основания и предпосылки**

* В университете Narxoz запрещены сторонние чаты преподавателей и студентов для снижения коррупционных рисков.
* Необходима централизованная система, к которой руководство университета имеет доступ при необходимости контроля или расследования.
* Запрос от кафедр и организаций университета на возможность планирования мероприятий и уведомления аудитории.
* Логины (учётные записи) генерируются админом:
  + **Студент**: S######## (8 цифр после буквы S, пример: S22016275).
  + **Преподаватель**: F######## (8 цифр после буквы F).

**2. Задачи проекта**

1. **Создать веб-приложение** (монолитную структуру), в котором будут реализованы:
   * Авторизация по JWT-токенам (регистрация/логин).
   * Публикация постов, комментариев, лайков.
   * Личные чаты и групповые чаты (с использованием WebSocket).
   * Подписки на пользователей и на организации.
   * Система событий (ивентов) с возможностью подписки, отображением в календаре и рассылкой уведомлений.
2. **Обеспечить разграничение ролей**: студент, преподаватель, организация, модератор, админ.
3. **Обеспечить безопасность**: доступ только для зарегистрированных вуза; **нельзя** чтобы пользователь сам регистрировался. Учетные записи создаёт только админ.
4. **Реализовать механику модерации**: возможность удалять посты/комментарии и просматривать чаты при необходимости.
5. **Реализовать хранение данных** в PostgreSQL (для основного функционала) и в MongoDB (для чатов).
6. **Настроить отправку уведомлений** (через Celery, Redis для кэширования) и управление нагрузкой (Nginx как прокси и балансировщик).
7. **Обеспечить тестирование**: юнит-тесты (Pytest для бэкенда), e2e-тесты (Cypress для фронтенда), непрерывная интеграция по возможности.

**3. Функциональные требования**

**3.1 Регистрация и авторизация**

**Регистрация**:

* Осуществляется **только админом**.
* Админ вручную (или полуавтоматически) генерирует логин в формате S######## (для студента) или F######## (для преподавателя) или O######## (для организаций).
* Регистрация новых пользователей: через бэкенд (Django), генерируется JWT-токен.
* У каждого пользователя в профиле должны быть поля email (для сброса пароля и уведомлений) и nickname (уникальное имя, по которому можно находить профиль).
* При создании учётной записи сохраняется хэш пароля; он не хранится в открытом виде.

**Авторизация:**

 JWT-токен: пользователь вводит логин (S22016275, Fxxxxx...) и пароль,

получает JWT.

 При необходимости предусмотрен **сброс пароля**: пользователю отправляется письмо на email со ссылкой (или кодом) для сброса парол

 Токен передаётся в заголовках при запросах к API.

**3.2 Роли и права**

* **Студент**: может читать ленту, публиковать посты, комментировать, лайкать, отправлять личные сообщения и вступать в группы.
* **Преподаватель**: помимо функционала студента, может создавать группы (групповые чаты) и выкладывать учебные материалы.
* **Организация**: аналогично преподавателю, но специализируется на анонсах, мероприятиях, может формировать группы со студентами.
* **Модератор**: контролирует контент (посты, комментарии, ивенты), имеет доступ для удаления или редактирования.
* **Админ**: полный доступ, назначение ролей, просмотр любых чатов при подозрении на коррупцию, блокировка пользователей.

**3.3 Публикация постов и лента**

* Пользователи публикуют посты (текст, изображения).
* Поддержка лайков и комментариев.
* Возможность подписываться на пользователей/организации, чтобы видеть их посты в приоритетной (или персональной) ленте.
* Поле nickname упрощает поиск пользователей и упоминание их в постах (при необходимости).

**3.4 Личные и групповые чаты**

* **Личные сообщения** (1-на-1): отправка через WebSocket или REST (в зависимости от архитектуры, но для обновлений в реальном времени лучше WebSocket).
* **Групповые чаты** (создаются преподавателями, организациями): несколько участников, общий чат, также на WebSocket.
* **MongoDB** используется для хранения сообщений в чатах (эффективно для быстрого доступа и больших объёмов).

**3.5 Уведомления и события (Celery, Redis)**

* События (ивенты), которые создают организации или преподаватели.
* Подписчики получают уведомления о новом событии, а также напоминания (с помощью Celery + Redis для очередей и кэширования).
* Redis может использоваться для краткосрочного кэширования/хранения данных, Celery — для фоновых задач (рассылка уведомлений).
* email пользователя может использоваться для отправки оповещений (настройки рассылки по желанию).

**3.6 Модерация и администрирование**

* **Модератор**: панель, где можно видеть последние жалобы или контент, помеченный системой. Удаление/редактирование.
* **Админ**: доступ к созданию пользователей, изменению ролей, просмотру всех чатов и статистике.

**3.7 Тестирование**

* **Pytest** (или встроенный тестовый фреймворк Django) для юнит-тестов бэкенда.
* **Cypress** для интеграционных/конечных (e2e) тестов фронтенда.
* Проверка JWT-авторизации, ролевой модели, функционала чатов, уведомлений, корректности API.

**4. Дизайн и пользовательский интерфейс**

 **Стиль**: современный минималистичный дизайн (Material UI, Ant Design, либо собственная стилизация).

 **Навигация**: «Лента», «Сообщения», «Группы», «Мероприятия/Календарь», «Профиль».

 **Профиль**: поля nickname, email, avatar, role, full\_name. Возможность редактирования avatar и некоторых личных данных.

 **Адаптивность**: корректное отображение на мобильных устройствах.

 **Прототипы**: (User Flow, CJM в Figma).

**5. Архитектура и Backend**

1. **Django (Python)** как основной фреймворк.
   * **Django REST Framework** для API (эндпоинты для авторизации, постов, комментариев, событий и т.д.).
   * **JWT** (Simple JWT или аналог) для аутентификации.
2. **PostgreSQL** как основная СУБД: хранение данных о пользователях, ролях, постах, комментариях, ивентах.
3. **MongoDB** для хранения сообщений чатов (быстрый доступ, удобное масштабирование).
4. **Nginx** в роли прокси-сервера и балансировщика нагрузки (SSL-терминация, маршрутизация к Django и другим сервисам).
5. **Celery** для фоновых задач: рассылка уведомлений, плановые задачи (напоминания о событиях).
6. **Redis** для кэширования, очередей Celery.
7. **WebSocket** (через Django Channels или отдельный сервис) для реалтайм-чатов.
8. **Docker, docker-compose**: упрощённое развёртывание (образы для Django, Redis, Postgres, MongoDB, Nginx).

**6. Frontend**

1. **React** (JavaScript/TypeScript) с компонентной архитектурой.
2. **Webpack** для сборки, возможно использование CRA (Create React App).
3. **Поддержка WebSocket** на фронте (например, socket.io или нативный WebSocket API) для чатов и уведомлений в реальном времени.
4. **React Router** для маршрутизации (если нужно несколько страниц).
5. **Cypress** для e2e-тестирования пользовательских сценариев.

**7. Интеграции и дополнительные модули**

1. **Календарь событий**: интеграция с модулями календаря (можно использовать любую готовую React-библиотеку).
2. **Уведомления**:
   * Email-уведомления (опционально, возможно через SMTP).
   * Push-уведомления (если потребуется мобильная версия, React Native).
3. **Админ-панель**:
   * Django Admin (по умолчанию) или кастомная панель в React для управления пользователями, контентом.

**8. Требования к производительности и безопасности**

1. **Производительность**:
   * Система должна выдерживать параллельную работу нескольких сотен пользователей (минимум).
   * Оптимизировать запросы к PostgreSQL, использовать индексы.
   * Для чатов (MongoDB) — горизонтальная масштабируемость при росте нагрузки.
2. **Безопасность**:
   * Использовать HTTPS (SSL-сертификат).
   * Сторонние пользователи без аккаунтов Narxoz не должны иметь доступ к закрытым разделам.
   * JWT-токены должны иметь срок действия; предусмотреть механизмы рефреша и логаута.
   * Регулярно обновлять Docker-образы (Django, Redis, Nginx и т.д.) и проверять уязвимости.
   * Сброс пароля — через email (ссылка/код).

**9. План работ (примерный)**

1. **Подготовка окружения**: настройка Docker, docker-compose, базовая структура Django + React.
2. **Реализация бэкенда**:
   * Модели (PostgreSQL + MongoDB), API (Django REST), аутентификация (JWT).
   * Подключение Celery + Redis, настройка Nginx.
3. **Реализация фронтенда**:
   * Базовые страницы (лента, профиль, сообщения, группы, календарь событий).
   * Интеграция с API, роутинг, верстка и дизайн.
4. **WebSocket**: чаты в реальном времени (личные и групповые).
5. **Модерация**: интерфейс для модератора, механизмы удаления и блокировки.
6. **Тестирование**:
   * Бэкенд (Pytest, покрытие основных сценариев).
   * Фронтенд (Cypress для e2e, проверка логина, постов, чатов).
7. **Документация**: финальное описание API, пользовательский гид.
8. **Презентация** (диплом): демонстрация основных сценариев работы.

**10. Требования к дизайну и документации**

1. **Дизайн**: использовать стиль, согласованный с брендбуком университета (цвета Narxoz, логотип и т.д.), либо современный минималистичный UI.
2. **Документация**:
   * Краткое руководство для пользователей (как создавать посты, чаты, события).
   * Техническая документация по запуску (Docker, настройки окружения), структуре БД, API-эндпоинты.
3. **Языковая поддержка**: на MVP-этапе — русский/английский интерфейс (по приоритетам заказчика).

**11. Критерии приёмки**

1. **Работоспособность ключевых модулей**: лента, чаты (включая групповые), подписки и события.
2. **Регистрация**: пользователь **не** может зарегистрироваться сам, только админ. Проверить логику выдачи логина S########/F########.
3. **Правильное разграничение ролей**: админ, модератор, студент, преподаватель, организация.
4. **Защита и безопасность**: все запросы (REST, WebSocket) с проверкой JWT, недопустимость доступа без авторизации.
5. **Удобство использования**: интуитивная навигация, корректная адаптивная вёрстка.
6. **Модульное и интеграционное тестирование**: нет фатальных ошибок при запуске и использовании.
7. **Документация**: наличие инструкции по установке и развертыванию.

**12. Формат сдачи и дополнительные условия**

1. **Сдача результата**: репозиторий (GitHub) с кодом (backend + frontend), docker-compose-файл для развёртывания, краткая инструкция (README).
2. **Демонстрация**: на дипломной защите запуск приложения, демонстрация основных сценариев (регистрация, лента, чат, создание события, модерация контента).
3. **Сроки**: финальная версия MVP — к дате защиты диплома (указать конкретные сроки по календарному плану).

**Примерная структура каталога проекта**

/project-root

├─ backend/

│ ├─ narxoz\_social/ (Django project)

│ ├─ requirements.txt

│ ├─ ...

├─ frontend/

│ ├─ src/

│ ├─ package.json

│ ├─ ...

├─ docker-compose.yml

├─ Dockerfiles (для backend, frontend, nginx)

├─ README.md

└─ ...

**Приложения (при необходимости)**

* [Ссылка на Figma](https://www.figma.com/board/Zblz3wERhNTpRKf5wO0m3Y/narxoz-flow?node-id=0-1&p=f) (User Flow, CJM, макеты интерфейсов).
* [Изображение архитектуры](https://miro.com/app/board/uXjVK0gHryU=/) — монолит (Django), PostgreSQL, MongoDB, Redis, Celery, Nginx, React.

**Заключение**

Данное **Техническое задание** описывает основные цели, задачи, функциональность и архитектурные решения для реализации MVP-проекта «Narxoz Social». При успешном выполнении ТЗ будет создана внутренняя соцсеть с чатами, постами, ивентами и системой ролей, соответствующая требованиям университета Narxoz и обеспечивающая прозрачный механизм коммуникации, безопасный с точки зрения контроля и предотвращения коррупции.