Для реализации такого ПО на Python можно использовать следующие инструменты:

**Графический интерфейс**

Для создания GUI можно использовать:

* **PyQt6 / PySide6** – мощный фреймворк для создания профессиональных приложений.
* **Tkinter** – стандартная библиотека Python, но с ограниченными возможностями.
* **Kivy** – поддерживает кроссплатформенную разработку, но сложнее в освоении.

Лучший выбор – **PyQt6 / PySide6**, так как они обеспечивают гибкость и современный дизайн.

**Работа с чатами**

Для реализации чатов можно использовать:

* **WebSockets (модуль websockets)** – для организации общения в реальном времени.
* **Pyro4** – для создания распределенных систем.
* **SQLite + файлы** – если чаты будут работать без сети.

**Хранение данных**

* **SQLite** – если нужен встроенный вариант без сервера.
* **PostgreSQL / MySQL** – если в будущем возможна масштабируемость.
* **TinyDB** – если требуется простая JSON-ориентированная база.

**Доска заметок и задачи**

* **QGraphicsView (в PyQt6)** – можно сделать интерактивную доску.
* **SQLite или JSON-файлы** – для хранения данных.

**Напоминания и дедлайны**

* **APScheduler** – для периодических напоминаний.
* **PyQt6 timers** – если напоминания должны работать внутри приложения.

**Как двигаться дальше?**

1. **Определить архитектуру** – локальное приложение или с сервером?
2. **Выбрать основной стек** – PyQt6 + SQLite + WebSockets (например).
3. **Сделать прототип интерфейса** – чтобы понять, как всё будет выглядеть.
4. **Реализовать базовые функции** – чаты, доску, задачи, дедлайны.
5. **Добавить улучшения** – голосования, напоминания, аналитику.

Для серверного решения тебе понадобится серверная и клиентская часть.

**Технологический стек**

**Клиент (десктопное приложение)**

* **PyQt6 / PySide6** – для создания интерфейса.
* **WebSockets (модуль websockets)** – для работы с чатом в реальном времени.
* **SQLite** – если нужна локальная база на клиенте для кеширования данных.
* **Requests / HTTPX** – если данные будут запрашиваться через REST API.

**Сервер**

* **FastAPI / Flask** – для создания REST API.
* **WebSockets (websockets или FastAPI WebSockets)** – для чатов в реальном времени.
* **PostgreSQL / MySQL** – в качестве основной базы данных.
* **Celery + Redis** – для фоновых задач (например, отправка уведомлений).

**Структура проекта**

1. **Сервер** (API + WebSockets)
   * Авторизация пользователей.
   * Управление группами и участниками.
   * Чаты (общий и личные) с возможностью голосования.
   * Система задач (назначение, дедлайны, напоминания).
   * Доска заметок (CRUD-операции).
   * Хранение прогресса работы.
2. **Клиентское приложение**
   * Подключение к серверу через WebSockets и REST API.
   * Интерфейс чатов и голосований.
   * Визуализация задач и прогресса.
   * Доска заметок.
   * Уведомления о дедлайнах.

**Как двигаться дальше?**

1. Развернуть **сервер** (FastAPI + WebSockets + PostgreSQL).
2. Сделать **авторизацию** пользователей.
3. Реализовать **чаты** и **голосования**.
4. Подключить **доску заметок и задачи**.
5. Настроить **уведомления** через Celery.
6. Разработать **клиентское приложение** (PyQt6).
7. Связать клиент с сервером.

**Реализация регистрации групп и пользователей**

**1. Создание группы**

* Пользователь вводит название группы.
* Генерируется уникальный код (например, 6-значный случайный код).
* Создаётся запись в базе данных (группа + её создатель).
* Код выдаётся пользователю для приглашения других.

**2. Присоединение к группе**

* Пользователь вводит код группы.
* Проверяется существование группы с этим кодом.
* Если группа найдена, пользователь добавляется в её состав.

Ссылка на весь диалог: https://chatgpt.com/c/67a5a2ec-f84c-8012-a3a5-1ce0c4053539