
Build Prototype- FULL-STACK Python API Developer

Overview
Build a simplified real-time pair-programming web application. Two users should be able to join the same room, edit code at the same time, and see each other's changes instantly. The system should also provide an AI-style autocomplete suggestion (this can be mocked and does not need to be real AI).

Expected Time: 6–10 hours
Stack Required:
Backend: Python, FastAPI, WebSockets
Frontend: React, TypeScript, Redux Toolkit (Minimal Implementation works)

---

1. Core Requirements

A. Room Creation & Joining

- Users can create a new room (generate a room ID).
- Users can join an existing room via a URL such as /room/.
- No authentication is required.

B. Real-Time Collaborative Coding

- Implement WebSockets so that two users in the same room share a code editor.
- When one user types, the other should see updates immediately.
- A simple syncing approach is fine (last-write wins or basic diffing).
- In-memory storage for each room's code state is acceptable.

C. AI Autocomplete (Mocked)

- Provide a POST endpoint /autocomplete that accepts:
  { code: "…", cursorPosition: number, language: "python" }
- Return a simple mocked suggestion (static, rule-based, etc.).
- On the frontend, when the user stops typing for ~600ms, call this endpoint and show the suggestion in the editor.

---

2. Backend Requirements (FastAPI)

- REST endpoints:
  • POST /rooms → returns { roomId }
  • POST /autocomplete → returns a mocked suggestion
- WebSocket endpoint:
  • /ws/ for real-time code updates
- Maintain room state in database (Postgres)
- Code should be organized in a clean project structure (routers, services, etc.).

---

3. Minimal Frontend Requirements (React + TypeScript + Redux) (Optional- Folks who would like to Demo it Via basic Brower or Postman is also welcome)

---

4. Deliverables

A. A Git repository containing:

- /backend folder (FastAPI project)

B. A README explaining:

- How to run both services
- Architecture and design choices
- What you would improve with more time
- Any limitations

C. (Optional) A deployed demo link is welcome but not required.

---

5. Evaluation Criteria

- Backend structure and clarity
- WebSocket implementation quality
- Code readability and maintainability

- Functionality of real-time collaboration
- Attention to detail in the README
- Optional improvements or enhancements