SOFTWARE ENGINEERING

Project Plan Document

TOPIC: Alumni Network and Job Portal System

MEMBERS:
Samiksha M (PES1UG22CS513)
Sharanya G S (PES1UG22CS543)

1. Identify the Lifecycle

- Lifecycle Chosen: Agile Model
- Justification:
  - Agile allows for iterative development and frequent feedback, critical for user-driven features like profile management, job posting, and mentoring.
  - The Alumni Network and Job Portal require continuous updates based on user feedback, especially since user needs can evolve as the system is used (e.g., improved job search algorithms, notification systems, or networking features).
  - Agile's flexibility will allow the development team to adapt to changes in user requirements and system updates during each sprint, ensuring delivery of high-priority features first.

2. Tools Selection

- Planning Tool: Jira – For sprint planning, backlog management, and tracking progress through the lifecycle.
- Design Tool: Lucidchart – To create ER diagrams, wireframes, and system architecture for the Alumni Network and Job Portal.
- Version Control: Git/GitHub – To manage code revisions, collaboration among team members, and integration of different modules (e.g., authentication, job portal).
- Development Tool: Visual Studio Code (VS Code) – A lightweight yet powerful code editor for front-end and back-end development.
- Bug Tracking: Jira – Integrated with the planning tool to track issues, assign tasks, and monitor bug resolution across sprints.
- Testing Tool: Selenium – For automating functional testing of web-based modules, such as the job posting system and mentoring interface.

3. Deliverables

- Reuse Components:
  - Authentication API: Authentication methods can leverage OAuth2 for social login or reuse standard authentication libraries.
  - Job Portal API: Existing job search and post APIs can be integrated or extended to reduce development time.

- Notification Module: Reuse standard notification libraries for sending email/SMS.

- Build Components:
  - Profile Management Module: Custom-built to allow alumni and students to manage their profiles, upload documents, and control privacy settings.
  - Mentoring Module: Tailored to enable personalised mentoring requests and user communication.
  - Networking Module: Develop features for creating discussion forums, direct messaging, and connecting alumni with students.

Justification: Reusing established APIs and libraries for authentication, notifications, and job postings reduces development time and ensures robust, tested functionalities. Building custom modules allows us to address unique user requirements like privacy controls, profile management, and mentoring.

4. Work Breakdown Structure (WBS)

Level	Task	Subtask
1	Planning	Requirements gathering, project scheduling, resource allocation
2	Design	UI/UX design, Database design, System architecture
3	Development	Front-end: User interface development using HTML/CSS/JavaScript Back-end: API development (Authentication, Job Portal, Mentoring) Database Integration: MySQL database for storing user profiles, job postings, connections
4	Testing	Unit testing, Integration testing, User acceptance testing
5	Deployment	Regular updates, bug fixes, user feedback implementation

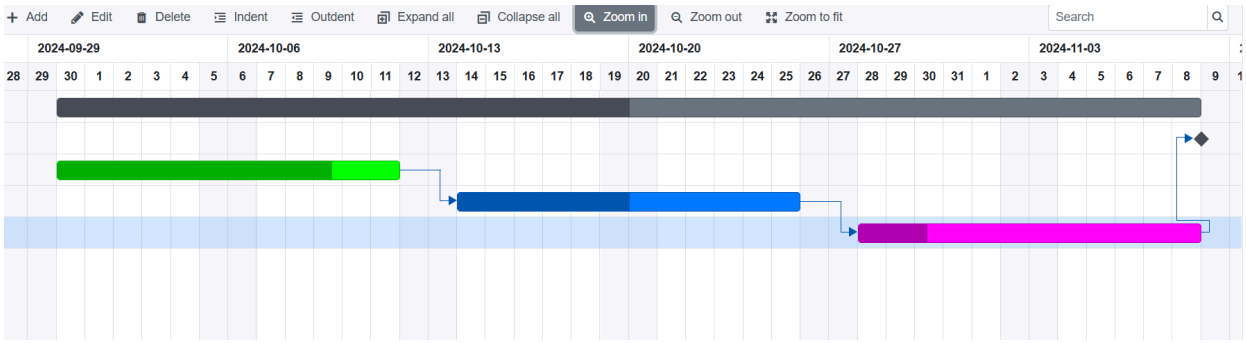
5. Effort Estimation & Gantt Chart

Adjusted Effort Estimation (in Person-Months):

- Planning: 0.5 person-month (reduced time)
- Design: 1 person-month (shared effort across UI/UX and database)
- Development: 3 person-months (focusing on essential features)
    - Split into:
        - Front-end: 1 month
        - Back-end: 1 month
        - Database: 1 month
- Testing: 1 person-month (unit and integration testing combined)
- Deployment: 0.5 person-month
- Post-deployment Support: Ongoing (can overlap with the deployment phase)

Revised Gantt Chart (for 12 Weeks):

- Week 1-2: Planning and requirements gathering.
- Week 3-4: Design phase (UI/UX, database schema).
- Week 5-9: Development phase (Front-end, back-end, database integration).
- Week 10-11: Testing phase (Unit and integration testing).
- Week 12: Deployment, post-deployment adjustments, and final delivery.



6. Coding Details

- Languages:
  - Front-end: HTML5, CSS3, JavaScript, React (for dynamic UI components).
  - Back-end: Python (Flask or Django for API development), Node.js for certain services.
  - Database: MySQL for user profile, job postings, and mentoring records.

- Version Control: Use Git with feature branches for each major module (authentication, job posting, mentoring).
- Testing:
  - Unit Testing: Each API (e.g., Authentication API) will have its own set of tests using pytest (for Python).
  - Integration Testing: Using Selenium to simulate user interactions (e.g., registering, searching for jobs).
  - Performance Testing: To ensure the system handles up to 10,000 concurrent users, simulate load using JMeter.