Milestone -3

Appendix -1

UI and functionality feedback (P1 functions only)

Overview: This section involves a review of the P1 features, focusing on core functionality, UI layout, usability, and performance. The instructor will check if each feature performs as intended and adheres to previous feedback. Key UI aspects to evaluate include layout clarity, navigation flow, and responsiveness. The goal is to ensure a seamless and intuitive user experience.

Key Features to Test:

1. **Grouping Functionality**
   - **Description**: Allows users to create, join, or manage groups based on industry, skills, or job roles, enabling networking and collaboration. Users can engage with peers, mentors, or recruiters in focused groups for shared interests or career goals.
   - **UI/Functionality Points**: Ensure group creation is intuitive, group names and descriptions are clear, and that users can easily invite others or join groups. Validate notifications for group messages and posts.
2. **Live News Updates**
   - **Description**: Provides users with real-time news related to career development, industry trends, and job markets. This feature helps users stay updated with relevant information, enhancing their readiness for the job market.
   - **UI/Functionality Points**: Check that the live news feed updates automatically without page reloads and displays headlines clearly. Confirm that the content is organized, relevant, and easy to read across devices.
3. **ChatBot**
   - **Description**: A conversational AI tool to assist users with job details, platform navigation, and career guidance. It enables users to ask questions and get immediate responses, enhancing engagement and support.
   - **UI/Functionality Points**: Test the chatbot for clear, understandable responses, and check if it's readily accessible across pages. Ensure it effectively addresses common user queries and provides relevant guidance.

General UI and Performance Checks:

- **Layout and Flow**: Assess each feature's layout and whether it follows a logical flow. Verify that users can easily navigate between sections (e.g., from Grouping to Live News to ChatBot).
- **Responsiveness**: Ensure all features adjust properly across various devices (desktop, tablet, mobile).
- **Feedback Handling**: Record any bugs, layout misalignments, or performance issues. Capture the instructor's feedback on feature clarity, functionality, and adherence to UI mockups.

This feedback will guide the team in refining each feature and finalizing the P1 set for the next project phase.

P1 Candidate Features (for M3 Demo)

1. **User Authentication and Profile Setup**
    - **Description**: Provides secure login, signup, and profile creation for different user roles (Student, Recruiter, Company). Each user can set up their profile with information such as skills, job interests, and resume.
    - **Details**:
        - Supports various roles with tailored permissions (e.g., students can apply to jobs but not create them).
        - Profile setup includes fields for experience, education, skills, and a resume upload feature.
        - Allows users to customize settings like notifications and privacy preferences.
    - **Demo Focus**: Test smooth signup and login, and ensure profile fields populate correctly.
2. **Job Search and Bookmarking**

- **Description**: Enables users to search for jobs using filters such as location, job type, skills, and remote work options. Users can bookmark jobs for later review.
- **Details**:
  - Filter options like job title, location, remote options, and industry.
  - Bookmarking feature for saving jobs to a dedicated list for easy access.
  - Results should be paginated to handle large data sets and optimize load times.
- **Demo Focus**: Demonstrate search functionality and bookmarking; confirm filters work accurately.

3. **Grouping Functionality**
   - **Description**: Users can create and join groups based on shared interests, skills, or industries to foster networking and collaboration.
   - **Details**:
     - Users can create groups with unique names, descriptions, and tags (e.g., "Marketing Professionals" or "Software Engineers").
     - Members can post messages, react to posts, and share job opportunities within the group.
     - Group creators have administrative rights to manage membership, moderate content, and delete groups if needed.
   - **Demo Focus**: Show group creation, joining, and interaction capabilities; check notification system for group updates.
4. **Live News Updates**
   - **Description**: Delivers real-time news updates related to job markets, industry trends, and career development, keeping users informed on relevant topics.
   - **Details**:
     - Integrates a news feed that automatically refreshes, displaying key news headlines and brief summaries.

        - Users can click on headlines for expanded details, or filter news by categories (e.g., “Tech,” “Healthcare,” “Finance”).
        - Ensures a responsive, user-friendly display across devices.
    - **Demo Focus**: Test automatic updates, filtering options, and mobile view.

5. **ChatBot**
    - **Description**: A conversational AI assistant that helps users with navigation, job search queries, and career guidance.
    - **Details**:
        - Accessible from all pages, providing quick responses to common queries such as “How to apply for jobs?” or “How do I join a group?”
        - Uses natural language processing to improve response quality and personalization.
        - Capable of redirecting users to appropriate pages or suggesting actions based on their queries.
    - **Demo Focus**: Show chatbot’s responsiveness and relevance of responses; test its ability to guide users effectively.
6. **Job Application Tracking**
    - **Description**: Allows users to track the status of their job applications and receive updates on application progress.
    - **Details**:
        - Tracks application status (e.g., “Applied,” “Under Review,” “Interview Scheduled”) with visual indicators on the user’s dashboard.
        - Users receive notifications when there is a change in application status.
        - Provides easy access to a history of applications for users to review past applications and related notes.
    - **Demo Focus**: Demonstrate status changes, notification accuracy, and dashboard integration.

Milestone -3

Instructions for Demo and Finalization

1. **Verbal Explanation**:
    - Provide a brief overview of each feature, its purpose, and how it enhances the user experience. Highlight any significant challenges or improvements made since M2.
2. **Review and Feedback Capture**:
    - Capture instructor feedback on functionality, UI flow, and responsiveness. Note any recommended changes or improvements for each feature.
3. **Commitment Decision**:
    - Based on feedback, finalize the list of P1 features. Commit only to the features that are feasible and impactful, as they cannot be changed after M3. Prioritize features that significantly enhance user engagement and align with project goals.
4. **Implementation by M5**:
    - Ensure the finalized features are thoroughly tested, fully implemented, and meet the quality standards by Milestone 5.

This structured list will ensure a smooth demo process, allow you to gather valuable feedback, and set a clear focus for the remaining milestones.

High-Level UML Class Diagram

This diagram outlines the core classes and their relationships, focusing on primary entities and an OO approach. Key components
include **User**, **Job**, **Group**, **Company**, **Recruiter**, **News**, and **Chatbot**.

Class Overview

1. **User**:
    - Attributes: user_id, firstName, lastName, email, password, userType, userSkills, userExperience, userEducation, bookmarkedJobs, groups, notifications
    - Methods: signUp(), login(), logout(), updateProfile(), bookmarkJob(), joinGroup()
2. **Job**:

- Attributes: job_id, title, description, location, requirements, salary, type, postedBy
- Methods: searchJobs(), apply(), bookmark()

3. **Group**:
   - Attributes: group_id, groupName, description, tags, members, posts
   - Methods: createGroup(), joinGroup(), postMessage(), addMember()
4. **Company**:
   - Attributes: company_id, name, industry, location, size, description, jobOpenings
   - Methods: postJob(), updateProfile()
5. **Recruiter**:
   - Attributes: recruiter_id, name, company, email, specialty, jobListings
   - Methods: manageJobListing(), contactCandidate()
6. **News**:
   - Attributes: news_id, title, category, timestamp, url
   - Methods: fetchLatest(), filterNews()
7. **Chatbot**:
   - Attributes: chatbot_id, context, userInput, response
   - Methods: processQuery(), generateResponse(), redirectToFeature()

These classes are structured to support **User Authentication, Job Management, Grouping, News Display, and Chatbot Interaction**.

High-Level Sequence Diagrams

Below are sequence diagrams for key functionalities. Each diagram describes the flow between client, backend, and database for core user actions.

1. User Authentication (Sign-Up & Log-In)

- **Actors**: User, Client (Front-end), AuthController, User Model, Database

- **Flow**:
    1. User signs up/logs in.
    2. Client sends credentials to AuthController.
    3. AuthController verifies and processes user data.
    4. If valid, AuthController hashes the password (for sign-up) or retrieves JWT (for login).
    5. A success or failure response is returned to the client.

2. Profile Management (View & Update)

- **Actors**: User, Client (Front-end), ProfileController, User Model, Database
- **Flow**:
    1. User requests to view/update their profile.
    2. ProfileController receives the request, retrieves, and/or updates the data.
    3. Updated data is saved in the database.
    4. Confirmation response is sent to the client.

3. Job Search and Bookmarking

- **Actors**: User, Client (Front-end), JobController, Job Model, Database
- **Flow**:
    1. User searches for jobs by criteria.
    2. JobController queries Job data based on filters.
    3. Matching jobs are returned to the client in a list format.
    4. User can bookmark jobs, triggering JobController to save the job to bookmarkedJobs.
    5. Success response is returned.

4. Grouping (Create & Join Group)

- **Actors**: User, Client (Front-end), GroupController, Group Model, Database
- **Flow**:
    1. User creates or joins a group via GroupController.

2. If creating, group data is saved to the database.
3. If joining, the GroupController adds the user to the members list.
4. Confirmation of group creation or joining is sent back.

5. Live News Fetching

- **Actors**: User, Client (Front-end), NewsController, News Model, External News API
- **Flow**:
    1. Client requests the latest news from NewsController.
    2. NewsController fetches relevant articles from the News API.
    3. News articles are processed and sent to the client.
    4. Articles are displayed in real-time for the user.

6. Chatbot Query Handling

- **Actors**: User, Client (Front-end), ChatbotController, Chatbot Model, Various Feature Controllers
- **Flow**:
    1. User interacts with the Chatbot by typing a query.
    2. ChatbotController processes the query and identifies the intent.
    3. If the query relates to an existing feature (e.g., job search), the Chatbot redirects to JobController.
    4. ChatbotController generates a response and sends it to the user.

Milestone -3

Project Status Overview

Current Progress:

- **Core P1 Features** (User Authentication, Profile Setup, Job Search, Grouping, News Updates, Chatbot) have been implemented and are undergoing final testing.
- **UI Refinements** based on M2 feedback are complete, including improved layout flow, responsiveness, and dark mode integration.
- **Backend & Database** setup is stable, with FastAPI managing API requests, MongoDB for data persistence, and Redis for caching and real-time notifications.

Risks

1. Schedule Risk

- **Description**: Delays in testing and integration of specific P1 features, particularly for features requiring external APIs (e.g., News API and Chatbot).
- **Resolution Plan**:
    - Prioritize critical path features like User Authentication and Job Search.
    - Implement parallel testing for non-dependent features to optimize time.
    - Assign team members to specific modules to speed up testing and debugging.

2. Team Coordination and Communication Risk

- **Description**: Potential delays or misunderstandings due to diverse roles and responsibilities across the team (front-end, back-end, and database).
- **Resolution Plan**:
    - Conduct daily stand-up meetings to track progress, clarify tasks, and address blockers.
    - Establish a shared task board (e.g., Trello or Jira) for tracking individual responsibilities and progress.
    - Set up quick check-in calls when urgent issues arise to maintain coordination.

3. Technical Risk

- **Description**: Issues with integrating third-party services (e.g., News API, Chatbot API) and potential for service downtime affecting real-time functionality.
- **Resolution Plan**:
    - Implement fallback mechanisms to handle API errors gracefully, such as cached data or preloaded resources.
    - Conduct rigorous testing on API integration endpoints and monitor for reliability.
    - Identify and document alternative APIs in case the primary service encounters disruptions.

4. Skill Gaps

- **Description**: Team members with limited experience in specific tools (e.g., WebSockets for real-time chat functionality or Redis for caching).
- **Resolution Plan**:
    - Conduct knowledge-sharing sessions or mini-workshops led by team members with expertise in these areas.
    - Assign additional resources (e.g., tutorials, documentation) to upskill members quickly.
    - Pair experienced and inexperienced team members on complex tasks to ensure faster learning and implementation.

5. Performance and Scalability Risk

- **Description**: Potential performance issues with real-time features (e.g., Chatbot, News Feed) that may affect user experience if not optimized.
- **Resolution Plan**:
    - Monitor server performance with load testing and optimize any identified bottlenecks.
    - Implement caching strategies for commonly accessed data (e.g., popular news or common Chatbot queries).
    - Adjust server configurations and database indexing for improved scalability.