Team Charter

1. Team Purpose and Objectives

Purpose

To collaboratively design and develop a job application platform that simplifies the job-hunting experience through a swipe-based interface. The product will allow users to easily browse and apply for jobs, with a focus on simplicity and speed.

Objectives

- Build a minimum viable product (MVP) that is functional and user-friendly.
- Ensure all team members contribute to both frontend and backend to gain full-stack experience.
- Practice team-based software development to prepare for real-world engineering roles post-graduation.
- Design filtering and job recommendation logic that accounts for edge cases and avoids overfiltering.
- Mimic the intuitive experience of popular apps by making job applications as simple as swiping.

2. Common Values and Standards

Core Values

- Respect: Every voice is heard and valued.
- Psychological Safety: Safe space to share ideas, ask questions, and make mistakes.
- Collaboration: We prioritize teamwork and shared success over individual glory.
- Transparency: Decisions and updates are clearly communicated.

Team Standards

- · Come prepared to meetings.
- Stay engaged with your assigned feature or task.
- Use shared tools consistently (e.g., Git, meeting notes, Slack/Discord).
- Submit updates or blockers before each meeting, even if you cannot attend.

3. Roles, Responsibilities, and Accountability

Structure

- Feature-Based Teams: Each member works on a defined feature (e.g., filtering, UI, auto-apply).
- Shared Responsibilities: Everyone touches both frontend and backend code as needed.

Roles (rotating or shared as necessary)

- Project Lead: Coordinates task delegation, sets meeting agendas, oversees progress.
- Tech Lead(s): Ensures code quality, advises on technical decisions, handles merges and pull requests.
- Scrum Master (optional): Keeps meetings on track and ensures blockers are addressed.
- Note-Taker/Communicator: Records meeting notes and shares summaries with the team.

Accountability

- Tasks must be completed by the deadline unless communicated otherwise.
- If a task is delayed, notify the group proactively and propose a plan to catch up.
- Team members should review each other's code to ensure knowledge sharing and maintain quality.

4. Conflict-Resolution Strategies

Potential Issues

- Miscommunication due to conflicting schedules.
- · Unclear ownership of tasks or responsibilities.
- Disagreements on implementation choices.
- Uneven workload distribution.

Conflict-Resolution Approach

- 1. Address Privately First: Individuals involved discuss the issue one-on-one.
- 2. Bring to Group: If unresolved, raise in a team meeting for collaborative resolution.
- 3. Mediation: If needed, a neutral team member (or TA/instructor if applicable) mediates.
- 4. Actionable Steps: Define concrete actions to resolve the issue and avoid recurrence.

5. Addressing Rule Violations

If someone is not following agreed-upon expectations (e.g., missing updates, not contributing):

- Step 1: Have a direct, respectful conversation with the individual.
- Step 2: If the issue persists, raise it with the team and define consequences (e.g., reassignment of tasks).
- Step 3: Escalate if necessary (to TA, instructor, or mentor if part of a course).

Everyone is expected to hold each other accountable in a constructive and supportive way.

6. Communication Protocols

Tools

- . GitHub for version control and issue tracking.
- · Slack/Discord for daily communication.
- · Google Docs for meeting notes and planning.

Protocols

- Share daily or weekly updates, especially if you are blocked or will miss a deadline.
- · Meeting notes should be shared in a central location after every meeting.
- Questions should be asked early rather than waiting someone else may have the same one.

Task Overview

Task 1

- Mock data + UI for Screens 1 and 2 (5 ppl)
- Search and filtering by preference (6 ppl)

Task 2

- Data scraping (3 ppl)
- Data processing (4 ppl)
- Auto-apply functionality (4 ppl)

Task 3

- UI for settings (5 ppl)
- User authentication (6 ppl)

Competitor Insights Summary

- Monster: Strong LinkedIn integration, job alerts, and career tools.
- swipejobs: Match-based, lacks rejection feedback, often overfilters.
- Snagajob: Filter-heavy, no swipe interface, repetitive listings and poor location matching.

Key Takeaways for Our MVP

- Focus on a swipe-to-apply experience with low friction.
- Prioritize effective filtering and user-friendly UI.
- Deprioritize post-application tracking (due to technical constraints).
- Use feedback from competitor shortcomings to guide feature prioritization.

Final Notes

Madhan