CS673 Software Engineering

Team 4 - {Enter Project Name Here}

Project Proposal and Planning

Your project Logo here if any

Team Member	Role(s)	Signature	Date
Hemant Krishnakumar	Team Lead		09 - 24- 2024
Faizan Ahmad	Design and Implementation Lead		09 - 24- 2024
Tushar	Requirement Lead		09 - 24- 2024
Shubh Gupta	Q/A Lead		09 - 24- 2024
Amruth Reddy	Security Lead		09 - 24- 2024
Jaindra Parvathaneni	Configuration Lead		09 - 24- 2024

Revision history

Version	**Author**	**Date**	**Change**
0	**Hemant Krishnakumar**	**09-24-2024**	

1. Overview

An ATS-like resume reviewer with a recruiter and applicant side. Recruiters get full access to all resumes ranked(scores based on JD). They would also have a personal dashboard which would provide advanced analytics and insights for decision making. On the user front, they can upload a specific number of resumes based on the tier they subscribed to. They too have a personal dashboard with additional analytics which would assist them with improving their resumes to match a JD and increase their chances of being shortlisted.

Motivation

Job applicants and recruiters look forward to better insights and streamlined information. We also see a lack of transparency which leads to applicants not getting a fair understanding of their competencies. Recruiters require a fair platform to judge applicants and shortlist quality resources based on their JD.

Purpose

12.4 M Americans are going to be searching for jobs in the upcoming year. Recruiters spend 6-8 seconds to read a resume and an applicant takes an average of 5 months to find a job. We aim to make this whole lifecycle more efficient on both fronts. Applicants need a streamlined application lifecycle that would increase their chances of being shortlisted for a role that they deserve. Recruiters would like to increase the confidence interval of shortlisting applicants who are qualified for a specific JD.

Potential User
All job seekers and recruiters.

Basic Functionality
Uploading resumes and JDs, reviewing resumes, ranking resumes, providing analytics

Tech Stack
React, Fastify(NodeJS), Docker,, PostgreSQL, GitHub Actions, AWS ECS, AWS Cloudwatch

2. Related Work

LinkedIn, JobScan, and WordIT give users a score and a general understanding of competencies but no advanced information. For recruiters LinkedIn.

3. Proposed High-level Requirements

a. Functional Requirements

(For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)

<u>ESSENTIAL FEATURES:</u>

Signup
As a recruiter or an applicant (user), I want to create an account so that I can access the dashboard

Login
As a user, I want to log in so that I can view the dashboard custom to my profile

Manage account and subscription
As a user, I want to be able to view my profile, manage it, view subscription plans and subscribe/unsubscribe.

Analytics and metrics
As a user, I want to view analytics and metrics (no. of resume uploads, no. of remaining uploads, types of JDs, TBD, etc) for my account

Resume upload
As a user, I want to upload resumes so that I am able to view them and use them for further analysis

JD upload and score
As a user (applicant and recruiter), I want to upload a job description and use my uploaded resumes to see how they score against the job description

Resume analysis - Applicant
As an applicant, I want to see a detailed analysis (requirements, competencies etc) of the uploaded resume against a particular JD so that I am able to improve the resume for a JD based on the feedback/analysis

Resume analysis and ranking - Recruiter
As a recruiter, I want to see detailed analysis (requirements, competencies etc) of the provided resumes against a particular JD and rank the resumes based on their scores so that I am able to choose the best resumes to move forward in the recruiting process.

DESIRABLE FEATURES:

Keyword search
As a recruiter, I want to be able to search all resumes with specific keywords/tags

Get best resumes
As a recruiter, I want to be able to filter the best resumes in the entire system for a specific job description

OPTIONAL FEATURES:

Upload Job Listing
As a recruiter, I want to upload job listings so that applicants can apply to them

Manage Job Listing
As a recruiter, I want to manage my job listings so that I can update them

View Job Listing
As an applicant, I want to view job listings so that I can apply to those roles

Apply to Job Listing

As an applicant, I want to apply to a job listing so that I can be considered for the role

Edit resume with AI
As an applicant, I want to tweak my current resume using AI to fit the JD better

b. Nonfunctional Requirements
i. User security: We want to make sure that user data is secure from unauthorized access and data breaches
ii. Platform security: We want to ensure that our platform is secure from cybersecurity threats such as DDoS attacks, MITM attacks, etc.
iii. Reliability: We want to ensure we minimize system failures and downtimes
iv. Availability: We want to ensure that we maximize our operational time
v. Maintainability: We want to ensure that our system is easy to maintain, debug, understand, and enhance
vi. Performance: We want to ensure that our system is performant, scales with increased usage, and does not degrade over time

4. Management Plan

a. Objectives and Priorities

Complete all proposed (essential) features: The primary objective is to ensure that all MVP (Minimum Viable Product) features are developed and functional. These include the applicant and recruiter dashboards, document upload, analytics, and resume matching system.

Deploy the software successfully: The second priority is the successful deployment of the web application, ensuring it is accessible and operational in the designated environment.

High-quality code with minimal bugs: Maintain a high standard of code quality, following best practices, and aim to have no known critical bugs by the end of each iteration.

Optimize the backend system: Focus on building an efficient backend using Fastify, Prisma, and Redis, ensuring scalability and security.

User-friendly UI/UX: Ensure the user interface is simple, intuitive, and effective in supporting the backend functionalities.

b. Risk Management (need to be updated constantly)

The main risks identified for the project are:

1. **Technology risk**: The team is new to some technologies (Fastify, Prisma, Redis). To mitigate this risk, team members will continue researching and experimenting with these tools, collaborating to solve challenges.
2. **Integration issues**: Problems might arise when integrating the frontend and backend, especially with complex data structures. Regular integration testing will be done to identify and fix issues early.
3. **Timeline delays**: Due to unfamiliarity with some technologies, there is a risk of delayed progress. To manage this, the team will meet weekly to review the timeline, adjust tasks, and ensure the project remains on track.
4. **Team collaboration risk**: Miscommunication or lack of adherence to GitHub protocols could lead to version control issues. Clear protocols and regular communication will be maintained.

c. Timeline

d.

Iteration	Functional Requirements(Essential/ Disable/Option)	Tasks (Cross requirements tasks)	Estimated/real person hours
1	Essential: Set up Git, MVP feature definitions	Git setup, high-level requirements, team collaboration setup, backend tech research	20
2	Essential: Backend setup, MVP feature development	Implement CRUD using Fastify & Prisma, initial UI/UX design, Redis integration	30
3	Essential: Integration of backend and frontend, testing	Frontend-backend integration, testing, debugging, UI/UX refinement	35

5. Configuration Management Plan

a. Tools

We are using the following tools in the project - VScode as the IDE, Prisma as ORM, Postgres as database, React as Frontend framework with material UI as UI package, Fastify for backend,

Keycloak for sso with oauth2.0. We Are also using Docker containers which will be deployed on a K8 cluster and hosted using ECS => EC2

b. Code Commit Guideline and Git Branching Strategy

We are working on different branches before committing the changes to the Main branch, we have created an initial_setup branch and pushed our commits there. Once we review the changes we will create a pull request, review it and then merge into the main branch. We planned on creating more branches if we need it for everyone in the team to work separately. We have also set up a git restriction which requires 2 people to review the changes before the merge request is fulfilled.

c. CI/CD Plan if applicable

We are going to deploy our react app through AWS elastic beanstalk and we will deploy our fastify app on AWS ec2 through AWS EKS and deploy postgres on AWS RDS..

1. Quality Assurance Plan

a. Metrics

Metric Name	Description
LOC	I will monitor total lines to assess project size and complexity
Cyclomatic Complexity	I will measure the complexity to identify high-risk modules needing targeted testing.
Defect Density	I will track defects per KLOC to maintain quality control over the software development lifecycle.
Test Case Pass Rate	I will record test pass rates to evaluate and improve the effectiveness of our test cases.
Story Completion Rate	I will ensure user stories are tracked per sprint to monitor team output and adjust workloads.

b. Coding Standard

Automated Tools: Use Prettier and ESLint for automated code checking and formatting, incorporating plugins for Node.js, React, and TypeScript recommended practices.

Commenting and Documentation: Maintain thorough comments and documentation to enhance code understanding and maintenance.

Naming Conventions: Adhere to naming conventions such as camelCase, PascalCase, and snake_case to ensure consistency across the codebase.

Readability: Focus on writing clear and readable code to facilitate easier debugging and future enhancements.

Cyclomatic Complexity: Implement a cyclomatic complexity limit of five paths to keep functions simple and prevent overly complex code.

Code Conciseness: Strive to write concise code, reducing the number of lines without compromising functionality or clarity.

c. Code Review Process

Mandatory Reviews: Every pull request must undergo a code review process.

Reviewer Requirement: A minimum of two team members must review and approve each PR.

Merge Condition: PRs can only be merged following approval from both reviewers.

d. Testing

Unit Testing: Team members are required to write unit tests for their code, with adherence and effectiveness assessed via code coverage metrics.

Integration Testing: Utilizing GitFlow, PRs are merged into a staging branch for soft deployment, where I personally conduct integration tests to ensure new features work seamlessly together.

React Specific: Post-integration testing, I compare the implemented features against the original Figma designs to ensure visual and functional fidelity.

e. Defect Management

Release Blockers: Critical defects that prevent deployment, such as issues that impair key functionalities (e.g., login failures). These must be addressed immediately.

Feature Design Mismatch: Functional features that do not align with specified designs. Requires adjustment to meet design specifications.

Small Bugs: Minor issues such as alignment errors or pixel mismatches. These are non-critical and can be resolved in subsequent iterations.