Dinesh Kumar Baalajee Jothi

Bjdineshkumar08@gmail.com

Abstract

This document provides you the details of the project and how it works

JOB tracker application

Documentation of the whole application

# **Job Tracker Application**

**Phase 1 - Planning and Setup**

**Introduction**

This document provides detailed instructions for Phase 1 of the Job Tracker Application development. The goal of this phase is to plan the project structure, set up the development environment, and establish the core components for the frontend, backend, and database.

**Deliverables**

* Frameworks of the web application.
* Project planning of the features expected.
* Definition of the architecture and the usage.
* Initial setup needed for the frontend, backend, and database.

**Technology used:**

|  |  |  |
| --- | --- | --- |
| **Category** | **Technology/Tool** | **Purpose** |
| Frontend Framework | React | Build the user interface and handle frontend logic. |
| UI Library | Material-UI (MUI) | Create visually appealing and consistent UI components. |
| Design | Figma | |  | | --- | | Prototype and design the application's user interface. |  |  | | --- | |  | |
| State Management | Redux or Context API | |  | | --- | | Manage application state efficiently. |  |  | | --- | |  | |
| Backend Framework | Spring Boot | |  | | --- | | Build the RESTful API and handle backend operations. |  |  | | --- | |  | |
| Programming Languages | JavaScript/TypeScript (Frontend) | |  | | --- | | Write scalable and maintainable frontend code. |  |  | | --- | |  | |
|  | Java (Backend) | |  | | --- | | Write backend services and business logic. |  |  | | --- | |  | |
| Database | MongoDB | Store job application data in a flexible NoSQL database. |
| Containerization | Docker | |  | | --- | | Package the app and its dependencies into containers. |  |  | | --- | |  | |
| Version Control | |  | | --- | |  |  |  | | --- | | GitHub | | |  | | --- | | Manage source code and collaborate on the project. |  |  | | --- | |  | |
| Hosting (Frontend) | Netlify or Vercel | |  | | --- | | Deploy and host the React frontend application. |  |  | | --- | |  | |
| Hosting (Backend) | Render or AWS Lambda | |  | | --- | | Deploy and host the backend API services. |  |  | | --- | |  | |
| Development Tools | Node.js and npm | |  | | --- | | Manage dependencies and run development servers. |  |  | | --- | |  | |
|  | MongoDB Compass | |  | | --- | | Visualize and manage MongoDB data (optional). |  |  | | --- | |  | |
| Visualization Library | Chart.js or Recharts | |  | | --- | | Display analytics and graphs in the dashboard. |  |  | | --- | |  | |
| Email Notifications | Amazon SES or NodeMailer | Send follow-up reminders to users. |

**Features:** The below are the features which will be developed for our web application.

 **Job Tracking**:

* CRUD operations for job applications.
* Status updates (e.g., Applied, Interviewing, Offered).

 **Analytics Dashboard**:

* Visual insights using charts and graphs.

 **Integrations**:

* LinkedIn, GitHub, Gmail, and HackerRank for automation.

 **Notifications:**

* Reminders for interviews and follow-ups.

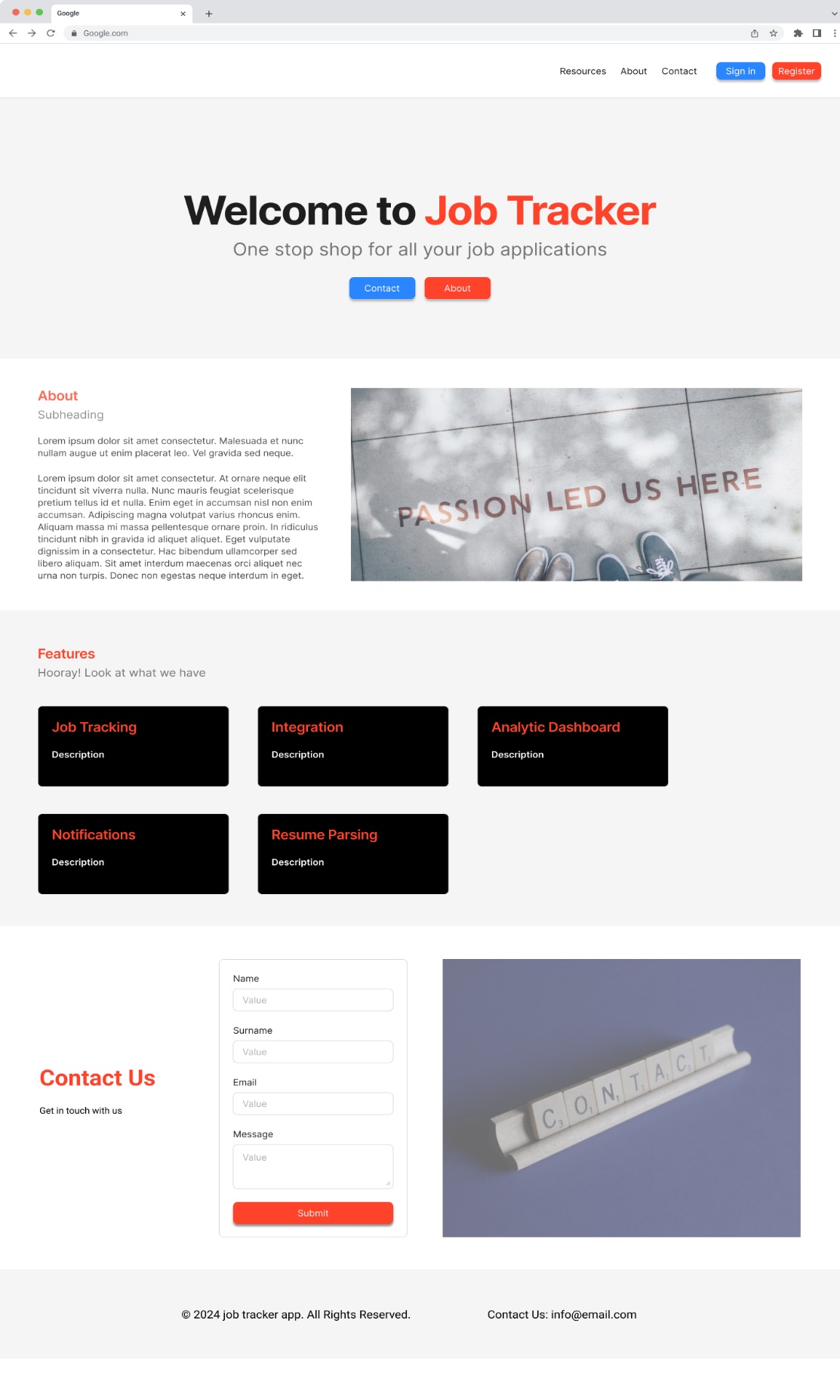
 **Resume Parsing**:

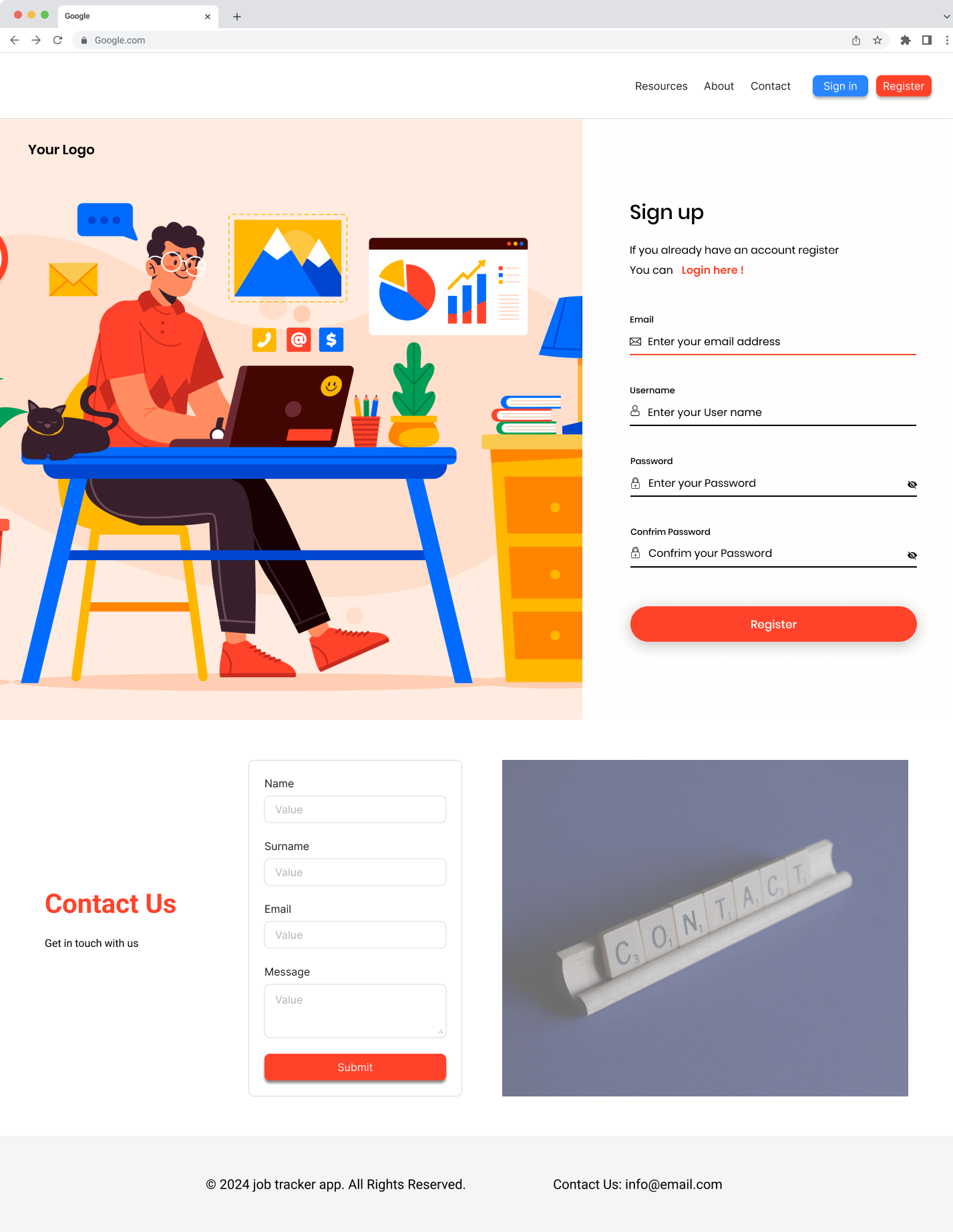
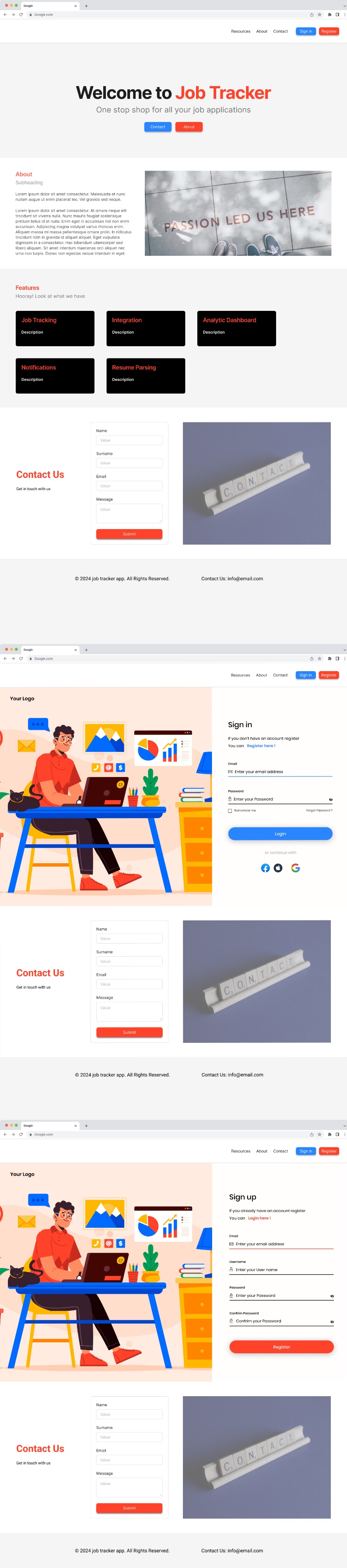
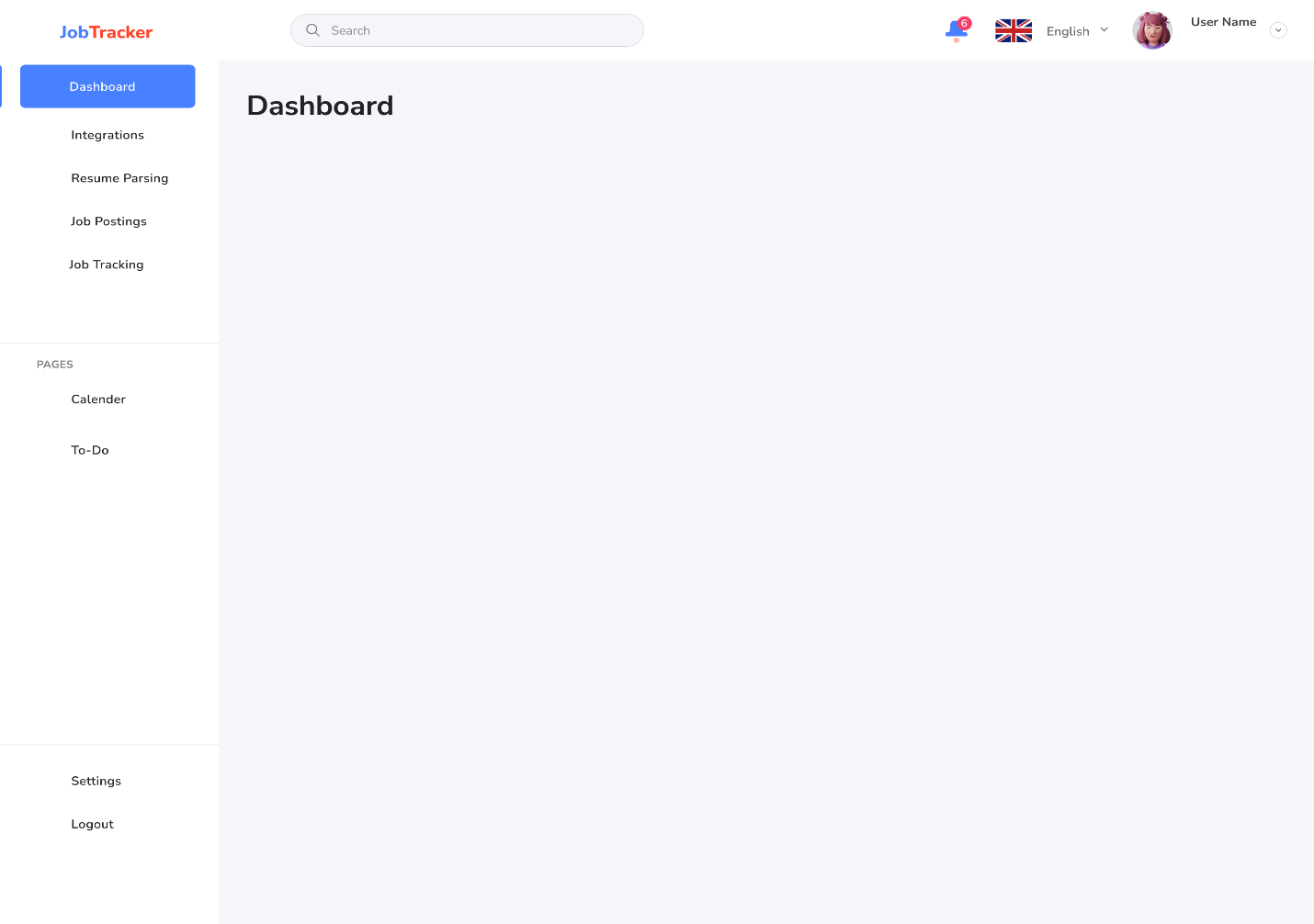
* Automatic extraction of job history and skills.

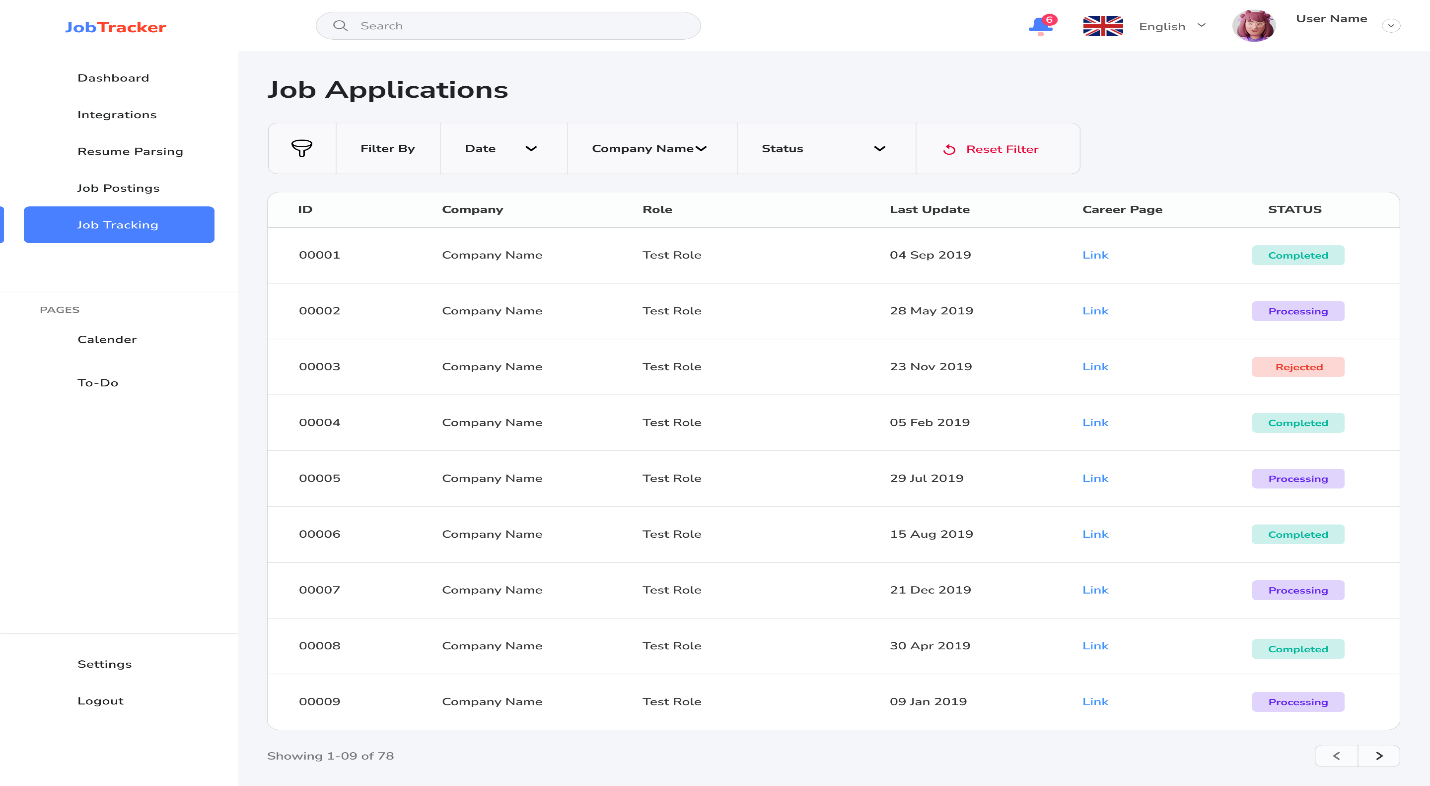
**Framework:**

The below framework’s were designed using Figma, it’s a mock up of the expected design which might change during the production release.

The frameworks will be updated iteratively based on the changing requirements. The below might not be the final design.

**Default Home page:**

**Sign-in page:  
  
  
Sign-up page:  
  
  
Home Dashboard:  
**

**Job Tracking:  
**

**Architecture:**

The below is the planned system design of the web application.

# Version 1: Initial Test Run

This document describes the Version 1 of the Job Tracker Application, which serves as an initial test run. The features and designs outlined here represent the foundational work for the application, including planning, prototyping, and development. This version has been planned from the base, with design prototypes created in Figma.

# Architecture

The system architecture for the Job Tracker Application is designed to ensure scalability, maintainability, and efficiency. Below are the main components and their roles:

**1. Frontend:** Built using React.js with Material-UI for styling and responsiveness. The frontend communicates with the backend through RESTful APIs to manage job data and user interactions. **2. Backend:** Implemented using Spring Boot, which provides the RESTful API layer for business logic, data processing, and communication with the database. **3. Database:** MongoDB is used to store job application data with a flexible schema, ensuring that different types of job-related data can be accommodated. **4. Machine Learning:** A Flask server hosts the machine learning model, which parses emails and resumes, extracting key details like job positions, skills, and keywords. **5. Integration:** Google OAuth and Gmail API are integrated for secure authentication and automated email parsing. This provides seamless synchronization of job-related data. **6. Notifications:** Implemented using tools like NodeMailer or Amazon SES to send reminders for interviews and follow-ups.

# Design Elements

The design for the Job Tracker Application was prototyped in Figma, allowing for rapid iterations and collaboration during the planning phase. The following pages were included in the design:

**1. Default Home Page:** An overview page that welcomes users and provides quick access to main features.  
**2. Sign-In and Sign-Up Pages:** Secure authentication flow for new and returning users.  
**3. Home Dashboard:** Displays a summary of job applications, status updates, and key insights. Includes visualizations for analytics.  
**4. Job Tracking Page:** Allows users to add, update, and view detailed information about their job applications. **5. Resume Analysis Page:** Provides a comparison of uploaded resumes with job descriptions, highlighting strengths and improvement areas.

# Planned Iterations

This document and the initial designs are subject to updates based on feedback and changing requirements. Future iterations will include enhanced analytics, collaborative features, and expanded machine learning capabilities.