**📄 JobConnect – Project Documentation**

**🔍 Project Overview**

**JobConnect** is a full-stack job portal application that bridges the gap between job seekers and employers. The platform enables job seekers to browse and apply for jobs, while employers can post and manage job listings. Each user registers with a role and gains access to features accordingly.

**🎯 Project Objectives**

* Enable **Job Seekers** to:
  + Register/login
  + Browse and search job listings
  + View job details
  + Apply for jobs
  + Receive SMS updates
* Enable **Employers** to:
  + Register/login
  + Post, edit, and delete job listings
  + View applications
  + Receive SMS notifications

**🛠️ Tech Stack**

| **Component** | **Technology** |
| --- | --- |
| Backend | Spring Boot, Spring Security, JPA, REST APIs |
| Frontend | Thymeleaf **or** ReactJS |
| Database | MySQL / MongoDB |
| Authentication | JWT (for React) / Session (for Thymeleaf) |
| Notification | Twilio (SMS Integration) |
| Testing | JUnit 5, Mockito |
| Documentation | Swagger / OpenAPI |
| Deployment | Backend: Render/Railway/AWS EC2 |
|  | Frontend: Netlify/Vercel |

**🧩 Functional Modules**

**1. User Authentication & Role Management**

* Registration form allows role selection: JOB\_SEEKER or EMPLOYER
* Authentication:
  + Spring Security with JWT (for React)
  + Session-based for Thymeleaf
* Role-based access control:
  + Job Seeker: View/search/apply jobs
  + Employer: Post/edit/delete jobs, view applications

**2. Job Posting (Employer)**

* Fields: title, description, location, salary, deadline
* CRUD APIs for managing job listings
* Accessible only by users with role EMPLOYER

**3. Job Search (Job Seeker)**

* View all jobs
* Filter/search by keyword and location
* View job details
* Apply to a job (store application and notify employer via SMS)

**4. SMS Notification**

* Twilio API Integration
* Notifications:
  + On user registration
  + On job application submission
  + On job post creation

**📁 Project Structure**

JobConnect/

├── backend/

│ ├── src/main/java/com/jobconnect/

│ │ ├── controller/

│ │ ├── entity/

│ │ ├── repository/

│ │ ├── service/

│ │ └── config/

│ ├── resources/

│ │ └── application.yml

│ └── tests/

├── frontend/ (only if using React)

│ ├── src/

│ └── public/

└── README.md

**✅ Development Steps**

**Step 1: Set Up Backend**

* Create Spring Boot app with dependencies:
  + Spring Web, Spring Data JPA, MySQL, Spring Security, JWT, Lombok, Twilio
* Configure database and JWT properties in application.yml

**Step 2: Build Core Features**

* User Module: Register/login with role
* Job Module: Employer CRUD operations
* Application Module: Job Seeker apply to jobs
* Notification Module: Twilio integration for SMS

**Step 3: Frontend Development**

**Option A: Thymeleaf**

* Add HTML templates for registration, login, dashboard, job listing, etc.
* Use Spring MVC to pass model data to templates

**Option B: React**

* Build pages/components for:
  + Registration/Login
  + Employer Dashboard (Job Management)
  + Seeker Dashboard (Job Search & Apply)
* Use axios to call backend APIs
* Store JWT in localStorage

**Step 4: Documentation**

* Add Swagger/OpenAPI annotations
* Access API docs at /swagger-ui.html

**Step 5: Testing**

* Write **JUnit 5** + **Mockito** tests for:
  + Controllers
  + Services
  + Repositories

**Step 6: Deployment**

* **Backend**: Deploy to Render/Railway/AWS EC2
* **Frontend** (React or Thymeleaf):
  + React → Deploy to Vercel/Netlify
  + Thymeleaf → Part of backend deployment

**🧪 Testing Plan**

| **Test Type** | **Tools** | **Scope** |
| --- | --- | --- |
| Unit Testing | JUnit 5, Mockito | Services, Controllers, Repositories |
| Integration Testing | JUnit | Full request-response flow |
| End-to-End User Testing | Postman or Manual | Login, Apply, Post Jobs, Role control |