

## 1. Software Requirements Specification (SRS)

**Project Name:** Recruiter Job Portal (Resume Builder)

**Objective:** A portal for recruiters to build and manage resumes, connect with candidates, track job applications, and generate reports in real-time.

**Users:** 1. Recruiters 2. Admin (optional)

**Functional Requirements:** - User Authentication (JWT) - Resume Builder: Create, Edit, Delete resumes - Job Postings: Create, Edit, Delete jobs - Candidate Management: Track resumes - Real-time notifications (Socket.IO) - Search & Filter - Analytics Dashboard (optional)

**Non-Functional Requirements:** - Backend: Python, FastAPI, MSSQL - Frontend: React, Redux/RTK - Socket.IO for notifications - Responsive design - Security: JWT, hashed passwords - PDF export for resumes

---

## 2. Database Design

**Tables:**

**Table 1: Recruiters**

```
CREATE TABLE Recruiters (
    RecruiterID INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(100),
    Email NVARCHAR(100) UNIQUE,
    PasswordHash NVARCHAR(255),
    CreatedAt DATETIME DEFAULT GETDATE()
);
```

**Table 2: Resumes**

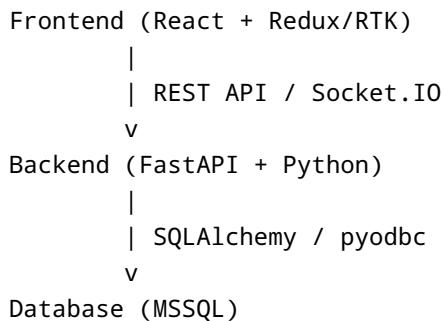
```
CREATE TABLE Resumes (
    ResumeID INT PRIMARY KEY IDENTITY(1,1),
    RecruiterID INT FOREIGN KEY REFERENCES Recruiters(RecruiterID),
    FullName NVARCHAR(100),
    Email NVARCHAR(100),
    Phone NVARCHAR(15),
    Skills NVARCHAR(MAX),
    Experience NVARCHAR(MAX),
    Education NVARCHAR(MAX),
```

```
    CreatedAt DATETIME DEFAULT GETDATE(),
    UpdatedAt DATETIME
);
```

#### Sample View:

```
CREATE VIEW RecruiterResumes AS
SELECT
    r.RecruiterID,
    r.Name AS RecruiterName,
    r.Email AS RecruiterEmail,
    res.ResumeID,
    res.FullName AS CandidateName,
    res.Email AS CandidateEmail,
    res.Skills,
    res.Experience,
    res.Education,
    res.CreatedAt AS ResumeCreated
FROM Recruiters r
LEFT JOIN Resumes res
ON r.RecruiterID = res.RecruiterID;
```

### 3. Architecture Diagram



**Components:** - **Frontend:** Login, Resume Builder, Job Management, Dashboard - **Backend:** REST APIs, WebSocket server (Socket.IO), Auth, Business Logic - **Database:** MSSQL tables + Views

### 4. Folder Structure

#### Frontend (React):

```
frontend/
|
└─ public/
```

```
|- src/
|   |- api/           # API calls
|   |- components/   # Reusable components
|   |- pages/         # Page-level components
|   |- store/         # Redux + RTK slices
|   |- utils/         # Utility functions
|   |- App.jsx
|   |- index.js
|- package.json
```

#### Backend (FastAPI):

```
backend/
|
|- app/
|   |- api/           # Route definitions
|   |- models/         # SQLAlchemy models
|   |- schemas/        # Pydantic schemas
|   |- services/       # Business logic
|   |- core/           # Config, settings, JWT
|   |- main.py          # FastAPI app
|   |- db.py            # Database connection
|- requirements.txt
```

**Socket.IO Integration:** - In `backend/app/main.py` use `fastapi-socketio` or `socketio.AsyncServer`.

---

**End of Document**