Web-Based Job Portal System

Backend Architecture & API Specification

Technology Stack - Backend Framework: NestJS - ORM: Prisma - Database: MySQL - Authentication: JWT (Role-Based Access Control)

---

1. Introduction

This document presents the professional backend design and API specification for the Web-Based Job Portal System. The purpose of this document is to translate the system analysis and design artifacts into a real-world, production-ready backend architecture using NestJS, Prisma ORM, and MySQL.

The system supports three primary user roles: - **Admin** – System supervisor and moderator - **Employer** – Job provider and recruiter - **Job Seeker** – Applicant searching and applying for jobs

---

2. System Architecture Overview

The backend follows a **modular, layered architecture**:

- **Controller Layer** – Handles HTTP requests and responses
- **Service Layer** – Implements business logic
- **Data Access Layer** – Prisma ORM for database interaction
- **Security Layer** – JWT authentication and role-based authorization

Each business function is implemented as an independent NestJS module to ensure scalability, maintainability, and clear separation of concerns.

---

3. Module Decomposition

3.1 Authentication Module

Purpose - User registration and login - JWT token generation - Role-based authentication n**Database Tables and Fields**

User Table - id (INT, PK) - email (VARCHAR, UNIQUE) - password (VARCHAR) - role (ENUM: ADMIN, EMPLOYER, JOB_SEEKER) - isActive (BOOLEAN) - createdAt (DATETIME) - updatedAt (DATETIME)

API Endpoints and Descriptions

- **POST /auth/register**
  Registers a new user by saving credentials into the User table after password hashing.

- **POST /auth/login**
  Validates user credentials against stored records and issues JWT on success.

- **GET /auth/me**
  Retrieves authenticated user data from the User table.

- **POST /auth/logout**
  Ends the user session on the client side.

3.2 User Management Module

Purpose - Manage shared user data across all roles

Database Tables and Fields

User Table - id (INT, PK) - fullName (VARCHAR) - email (VARCHAR) - phone (VARCHAR) - role (ENUM) - isActive (BOOLEAN) - createdAt (DATETIME) - updatedAt (DATETIME)

API Endpoints and Descriptions

- **GET /users**
  Fetches all user records from the User table (Admin only).

- **GET /users/{id}**
  Retrieves a single user record by user ID.

- **PUT /users/{id}**
  Updates user information fields such as name or phone.

- **DELETE /users/{id}**
  Removes a user record permanently from the database.

3.3 Job Seeker Module

Purpose - Manage job seeker profiles and job applications

Database Tables and Fields

JobSeekerProfile Table - id (INT, PK) - userId (INT, FK → User) - education (VARCHAR) - skills (TEXT) - experience (TEXT) - address (VARCHAR) - createdAt (DATETIME) - updatedAt (DATETIME)

API Endpoints and Descriptions

- **POST /job-seekers/profile**
  Creates a new JobSeekerProfile record linked to the User table.

- **GET /job-seekers/profile**
  Retrieves job seeker profile data from JobSeekerProfile table.

- **PUT /job-seekers/profile**
  Updates education, skills, and experience fields.

- **GET /job-seekers/jobs**
  Retrieves approved job records from Job table.

- **POST /job-seekers/apply/{jobId}**
  Inserts a new record into Application table.

- **GET /job-seekers/applications**
  Retrieves application records linked to the job seeker.

3.4 Employer Module

Purpose - Enable employers to post and manage job vacancies

Database Tables and Fields

EmployerProfile Table - id (INT, PK) - userId (INT, FK → User) - companyName (VARCHAR) - companyAddress (VARCHAR) - contactEmail (VARCHAR) - approvalStatus (ENUM: PENDING, APPROVED, REJECTED) - createdAt (DATETIME) - updatedAt (DATETIME)

API Endpoints and Descriptions

- **POST /employers/profile**
  Creates employer profile and sets approval status to PENDING.

- **GET /employers/profile**
  Retrieves employer profile data.

- **PUT /employers/profile**
  Updates employer information.

- **POST /employers/jobs**
  Creates a job record linked to EmployerProfile.

- **PUT /employers/jobs/{id}**
  Updates job record fields.

- **DELETE /employers/jobs/{id}**
  Deletes job record from Job table.

- **GET /employers/applications/{jobId}**
  Retrieves application records linked to the job.

3.5 Job Management Module

Purpose - Handle job vacancy data and public job listings

Database Tables and Fields

Job Table - id (INT, PK) - title (VARCHAR) - description (TEXT) - category (VARCHAR) - location (VARCHAR) - salary (DECIMAL) - employerId (INT, FK → EmployerProfile) - status (ENUM: PENDING, APPROVED, CLOSED) - createdAt (DATETIME)

API Endpoints and Descriptions

- **GET /jobs**
  Retrieves all approved job records.

- **GET /jobs/{id}**
  Retrieves a specific job record.

- **GET /jobs/search**
  Filters job records by keyword or category.

3.6 Application Module

Purpose - Manage job application lifecycle

Database Tables and Fields

Application Table - id (INT, PK) - jobId (INT, FK → Job) - seekerId (INT, FK → JobSeekerProfile) - status (ENUM: SUBMITTED, UNDER_REVIEW, SHORTLISTED, HIRED, REJECTED) - appliedAt (DATETIME)

API Endpoints and Descriptions

- **POST /applications/{jobId}**
  Creates a new application record with SUBMITTED status.

- **GET /applications/my**
  Retrieves applications linked to the job seeker.

- **GET /applications/job/{jobId}**
  Retrieves applications linked to a job.

- **PUT /applications/{id}/shortlist**
  Updates application status to SHORTLISTED.

- **PUT /applications/{id}/reject**
  Updates application status to REJECTED.

- **PUT /applications/{id}/hire**
  Updates application status to HIRED.

3.7 CV Management Module

Purpose - Secure upload and retrieval of applicant CVs

Main Responsibilities - File storage management - Access control for CV downloads

API Endpoints - POST /cvs/upload - GET /cvs/{id}/download - DELETE /cvs/{id}

3.8 Admin Module

Purpose - Centralized system administration and moderation

Main Responsibilities - Approve employers and job postings - Manage users - Monitor system activity

API Endpoints - GET /admin/dashboard - PUT /admin/employers/{id}/approve - PUT /admin/employers/{id}/reject - PUT /admin/jobs/{id}/approve - PUT /admin/jobs/{id}/reject - GET /admin/users - DELETE /admin/users/{id}

3.9 Feedback Module

Purpose - Collect user feedback and system reports

API Endpoints - POST /feedback - GET /feedback

4. Database Design Overview (Prisma ORM)

Core Entities - User - JobSeekerProfile - EmployerProfile - Job - Application - CV - Feedback

Key Design Principles - Relational integrity using foreign keys - Enum-based role and status management - Secure storage of sensitive data

---

5. Security Design

- JWT-based authentication
- Role-Based Access Control (RBAC)
- Password hashing using bcrypt
- Route guards and authorization decorators

---

6. Data Transfer Objects (DTO) Design

Data Transfer Objects (DTOs) are used to define the structure of data exchanged between the client and the backend API. DTOs ensure validation, consistency, and security by controlling which fields are accepted or returned by each endpoint.

6.1 Authentication Module DTOs

RegisterUserDto - email: string - password: string - role: enum (EMPLOYER | JOB_SEEKER)

LoginDto - email: string - password: string

6.2 User Management Module DTOs

UpdateUserDto - fullName?: string - phone?: string - isActive?: boolean

---

6.3 Job Seeker Module DTOs

CreateJobSeekerProfileDto - education: string - skills: string - experience: string - address: string

UpdateJobSeekerProfileDto - education?: string - skills?: string - experience?: string - address?: string

---

6.4 Employer Module DTOs

CreateEmployerProfileDto - companyName: string - companyAddress: string - contactEmail: string

UpdateEmployerProfileDto - companyName?: string - companyAddress?: string - contactEmail?: string

CreateJobDto - title: string - description: string - category: string - location: string - salary?: number

UpdateJobDto - title?: string - description?: string - category?: string - location?: string - salary?: number

---

6.5 Job Management Module DTOs

SearchJobDto - keyword?: string - category?: string - location?: string

---

6.6 Application Module DTOs

CreateApplicationDto - jobId: number

UpdateApplicationStatusDto - status: enum (SHORTLISTED | HIRED | REJECTED)

6.7 CV Management Module DTOs

UploadCvDto - file: File (PDF/DOC)

6.8 Feedback Module DTOs

CreateFeedbackDto - message: string

---

7. Conclusion

This document now provides a complete professional backend specification including system architecture, API endpoints, database schema, and DTO definitions. The inclusion of DTOs ensures strong validation, secure data transfer, and clean separation between external requests and internal data models. The system is fully prepared for implementation using NestJS, Prisma ORM, and MySQL.