



UNIVERSITY OF ENGINEERING  
& TECHNOLOGY PESHAWAR

# JOB PORTAL PROJECT REPORT

---

This project demonstrates the practical application of database management concepts using Laravel, turning academic knowledge into a functional and dynamic system.

2025

## **Prepared By :**

---

Mohsin Sajjad : 22pwcse2149

Muhammad Hassan : 22pwcse2105

Muhammad Afnan khan : 22pwcse2155

## **Prepared For :**

**Engr.Summaya Salahuddin**

# TABLE OF CONENTS

1. PROJECT OVERVIEW	
<a href="#">1.1 PROJECT VISION</a>	3
<a href="#">1.2 CORE OBJECTIVES</a>	3
<a href="#">1.3 PROJECT SCOPE</a>	3
2. TECHNICAL ARCHITECTURE	
<a href="#">2.1 TECHNOLOGY STACK</a>	3
<a href="#">2.2 SYSTEM ARCHITECTURE</a>	4
3. DATABASE DESIGN & ANALYSIS	
<a href="#">3.1 DATABASE DESIGN PHILOSOPHY</a>	4
<a href="#">3.2 ENTITY ANALYSIS</a>	5
<a href="#">3.3 ADVANCED BUSINESS RULES IMPLEMENTATION</a>	5
4. ENTITY-RELATIONSHIP DIAGRAM	
<a href="#">4.1 ER DIAGRAM</a>	8
<a href="#">4.2 ENHANCED ENTITY-RELATIONSHIP DIAGRAM</a>	8
5. IMPLEMENTATION DETAILS	
<a href="#">5.1 LARAVEL MVC IMPLEMENTATION</a>	9
<a href="#">5.1.1 MODEL LAYER</a>	9
<a href="#">5.1.2 CONTROLLER LAYER</a>	9
<a href="#">5.1.3 VIEW LAYER</a>	10
6. DEPLOYMENT ARCHITECTURE	
<a href="#">6.1 DEPLOYMENT WORKFLOW DIAGRAM</a>	10
7. DEPLOYMENT STEPS	
<a href="#">7.1 PUSH THE PROJECT ON GITHUB</a>	11
<a href="#">7.2 CREATE MYSQL DATABASE ON AIVEN.IO</a>	11
<a href="#">7.3 DEPLOY BACKEND TO RAILWAY</a>	12
<a href="#">7.4 CONFIGURING VARIABLE</a>	13
<a href="#">7.5 ACCESS USING MYSQL CLIENT</a>	14
8. SYSTEM FUNCTIONALITY	
<a href="#">8.1 ROLE-BASED ACCESS CONTROL (RBAC)</a>	14
<a href="#">8.1.1 ADMINISTRATOR FUNCTIONS</a>	14
<a href="#">8.1.2 EMPLOYER FUNCTIONS</a>	16
<a href="#">8.1.3 JOB SEEKER FUNCTIONS</a>	17
9. SECURITY IMPLEMENTATION	
<a href="#">9.1 ACCESS CONTROL</a>	18
10. FUTURE ENHANCEMENTS	
<a href="#">10.1 SCALABILITY IMPROVEMENTS</a>	18
<a href="#">10.2 FEATURE EXPANSIONS</a>	18
11. CONCLUSION	
<a href="#">11.1 CONCLUSION</a>	19
<a href="#">11.2 THE PROJECT SUCCESSFULLY DEMONSTRATES</a>	19
12. REFERENCES	
<a href="#">12.1 REFERENCES</a>	20
<a href="#">12.2 DEPLOYMENT URL</a>	20
<a href="#">12.3 GITHUB LINK</a>	20

# PROJECT TITLE: JOB PORTAL

## 1.PROJECT OVERVIEW:

### 1.1 PROJECT VISION:

To develop a scalable, secure, and user-centric job portal that demonstrates advanced database management principles while solving real-world employment challenges through technology.

### 1.2 CORE OBJECTIVES:

- Primary Objective: Design and implement a normalized relational database system supporting complex job market operations
- Secondary Objectives:
  - Implement role-based access control (RBAC) with three distinct user types
  - Develop RESTful API endpoints following Laravel conventions
  - Create responsive, mobile-first user interfaces
  - Deploy on cloud infrastructure with proper CI/CD practices
  - Ensure data integrity through comprehensive validation and constraints

### 1.3 PROJECT SCOPE:

The system encompasses:

- User Management: Registration, authentication, and profile management
- Job Lifecycle: Posting, searching, filtering, and application tracking
- Administrative Controls: Employer verification, application moderation
- Document Management: Resume upload, storage, and retrieval
- Notification System: Real-time status updates and email notifications

---

## 2. TECHNICAL ARCHITECTURE

### 2.1 TECHNOLOGY STACK

- Backend Framework: Laravel 12 (PHP 8.3)
- Frontend Technologies: Blade Templating Engine, Bootstrap 5, HTML5/CSS3
- Database Management: MariaDB/MySQL 8.0
- Development Environment: XAMPP (Apache, MySQL, PHP)
- Version Control: Git/GitHub
- Cloud Deployment: Railway.app (Backend), Aiven.io (Database)
- Additional Tools: Composer, NPM, Artisan CLI

## 2.2 SYSTEM ARCHITECTURE

The application follows the Model-View-Controller (MVC) architectural pattern:

Model Handles:

- Storing user information in the database
- Saving job postings
- Keeping track of job applications
- Managing company profiles
- Handling resume uploads
- Following business rules like "one person can only apply once to same job"

View Handles:

- The web pages users see
- Job listing pages
- Application forms
- User profile pages
- Login and registration screens
- Making everything look nice with colors and layouts

Controller Handles:

- Taking user requests like "show me all jobs"
  - Processing form submissions like job applications
  - Deciding which page to show next
  - Getting data from database and sending it to web pages
  - Handling user actions like clicking buttons
- 

## 3. DATABASE DESIGN & ANALYSIS

### 3.1 DATABASE DESIGN PHILOSOPHY

Our database design adheres to the following principles:

- Normalization: All tables are normalized to 3NF to eliminate redundancy
- Referential Integrity: Foreign key constraints ensure data consistency
- Scalability: Design supports horizontal and vertical scaling
- Security: Prepared statements prevent SQL injection attacks
- Performance: Indexed columns for optimal query performance

## 3.2 ENTITY ANALYSIS

ENTITY	PRIMARY KEY	KEY ATTRIBUTES	DESCRIPTION
USER	UserID	Name, Email, Password, UserType	Stores common account data for both JobSeekers and Employers.
JOBSEEKER	SeekerID	UserID (FK), DateOfBirth, Address, ProfileSummary	Subtype of User; holds seeker-specific profile details.
EMPLOYER	EmployerID	UserID (FK), CompanyName, CompanyDescription, CompanyWebsite, VerifiedStatus	Subtype of User; holds employer-specific company details.
RESUME	ResumeID	SeekerID (FK), FilePath, UploadDate, LastUpdated	Stores uploaded resumes for JobSeekers.
JOB	JobID	EmployerID (FK), Title, Description, Requirements, Location, SalaryRange, PostedDate	Represents job listings posted by Employers.
APPLICATION	ApplicationID	JobID (FK), SeekerID (FK), ResumeID (FK), AppliedDate, Status	Captures a JobSeeker's application to a specific Job with one resume.

## 3.3 ADVANCED BUSINESS RULES IMPLEMENTATION

### I. USER REGISTRATION RULE

- Rule: Every account must register as either JobSeeker or Employer
- Simple Explanation: You can't create an account without choosing your role
- Real Example: During signup, you must click either "I'm looking for jobs" or "I'm hiring"

### II. UNIQUE EMAIL RULE

- Rule: The Email in User table must be unique and validated
- Simple Explanation: One email = one account only
- Real Example: If [john@gmail.com](mailto:john@gmail.com) is taken, you must use different email

### III. EMAIL VERIFICATION RULE

- Rule: Only Employers with VerifiedStatus = TRUE may post Job listings
- Simple Explanation: Companies must be verified before they can post jobs
- Real Example: New companies can't post jobs until admin approves them

### IV. JOB POSTING RULE

- Rule: An Employer may post zero or more Jobs; each Job must reference exactly one Employer
- Simple Explanation: Jobs must belong to a company, and companies can post multiple jobs
- Real Example: Google can post 50 jobs, but each job belongs only to Google

### V. RESUME UPLOAD RULE

- Rule: A JobSeeker may upload multiple Resumes, but only one active Resume per Application
- Simple Explanation: You can have many resumes saved, but pick one for each job application
- Real Example: You have "Tech Resume" and "Marketing Resume" but apply with only one

### VI. PROFILE COMPLETENESS RULE

- Rule: JobSeekers must complete mandatory fields (DateOfBirth, Address, Profile Summary) before applying
- Simple Explanation: Must fill required profile info before applying to jobs
- Real Example: Can't apply until you enter your birth date, address, and summary

### VII. APPLICATION CONSTRAINT RULE

- Rule: A JobSeeker may apply to the same Job at most once
- Simple Explanation: You can only apply once to each job
- Real Example: Applied to "Software Engineer at Google"? Can't apply again to same posting

### VIII. APPLICATION STATUS RULE

- Rule: Each Application's Status must be: Pending, Reviewed, Interview, Offered, or Rejected
- Simple Explanation: Applications follow specific status progression
- Real Example: Pending → Reviewed → Interview → Offered (or Rejected at any step)

### IX. POSTING REQUIREMENTS RULE

- Rule: Each Job must include: Title, Description, Requirements, Location, and Posted Date
- Simple Explanation: All job postings must have these 5 essential pieces of information
- Real Example: Can't post job without job title, description, requirements, location, and date

### X. AUTOMATIC TIMESTAMPS RULE

- Rule: Applied Date and Posted Date default to current system date
- Simple Explanation: System automatically records when jobs are posted and applications are submitted
- Real Example: Apply today = Applied Date becomes today's date automatically

## XI. RESUME UPDATE RULE

- Rule: When JobSeeker updates Resume, old version may be retained but only latest links to new Applications
- Simple Explanation: Keep history of old resumes but new applications use newest version
- Real Example: Upload Resume v2.0, but Resume v1.0 stays for old applications

## XII. DELETION RULES

- Rule: Deleting User cascades to delete JobSeeker/Employer records; deleting Employer prevented if open Job listings exist
- Simple Explanation: Delete account = delete profile, but can't delete company with active job postings
- Real Example: Delete your account = your profile disappears; Google can't delete account if they have 20 open jobs

## WHY THESE RULES MATTER:

### DATA QUALITY:

- Ensures all information is correct and complete
- Prevents garbage data from entering system
- Maintains professional standards

### USER EXPERIENCE:

- Users know exactly what to expect
- Clear process flow from registration to job application
- Prevents confusion and errors

### SYSTEM INTEGRITY:

- Database stays organized and reliable
- Prevents data corruption
- Ensures smooth operation

### BUSINESS LOGIC:

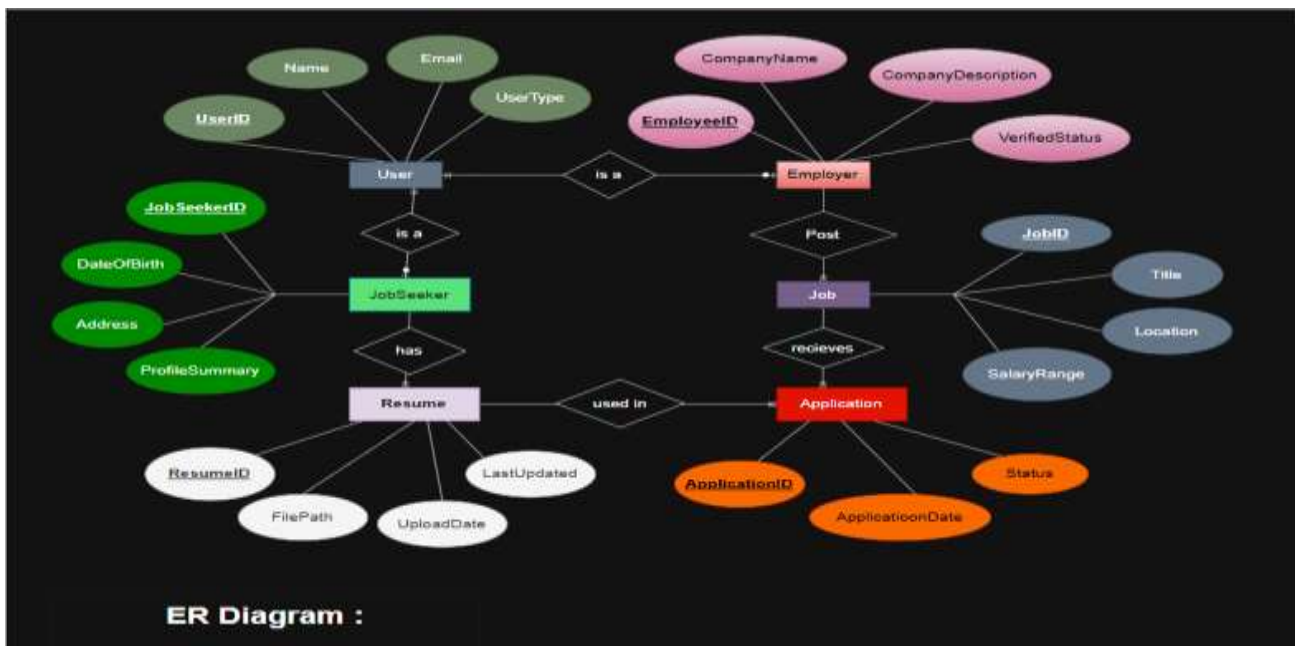
- Reflects real-world hiring processes
- Protects both job seekers and employers
- Maintains platform credibility

### SECURITY:

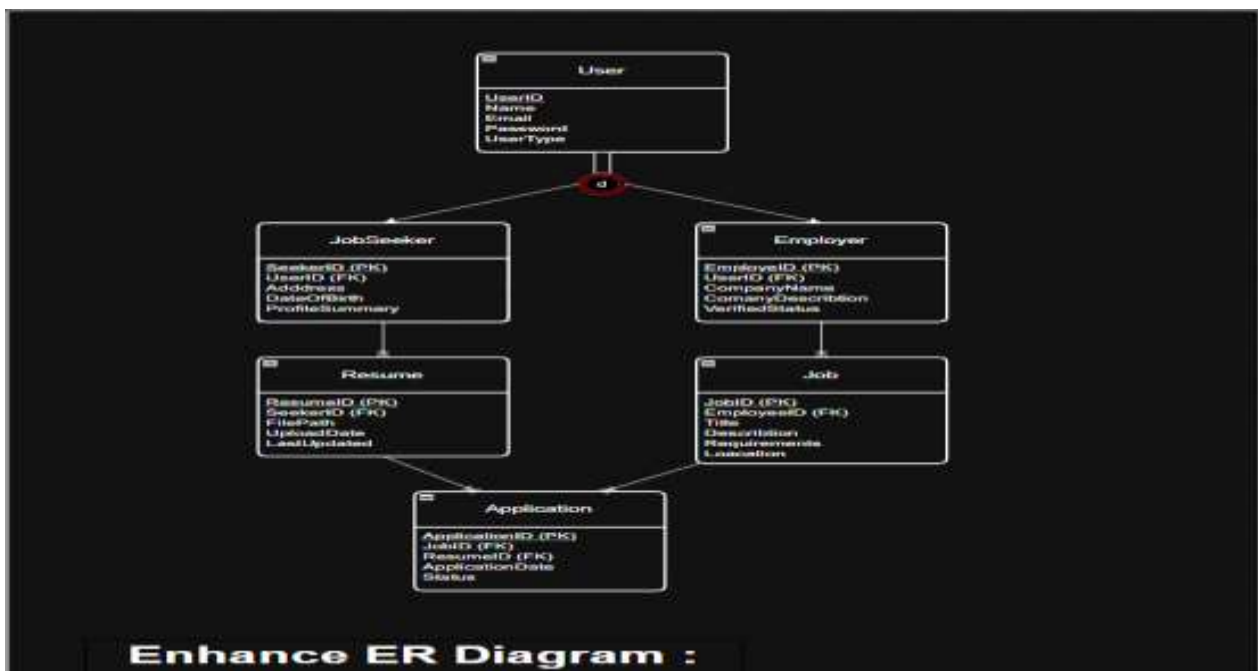
- Prevents unauthorized actions
- Ensures proper verification
- Protects user data

## 4.ENTITY-RELATIONSHIP DIAGRAM

### 4.1 ER DIAGRAM:



### 4.2 ENHANCED ENTITY-RELATIONSHIP DIAGRAM:

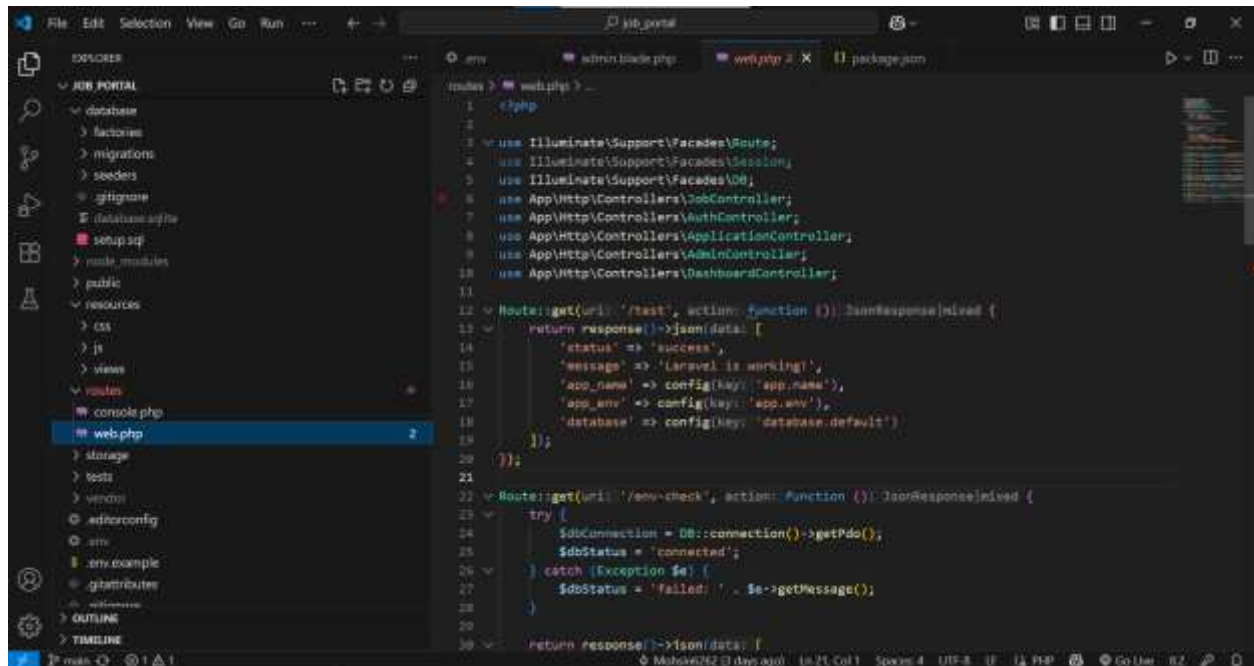




## 5. IMPLEMENTATION DETAILS

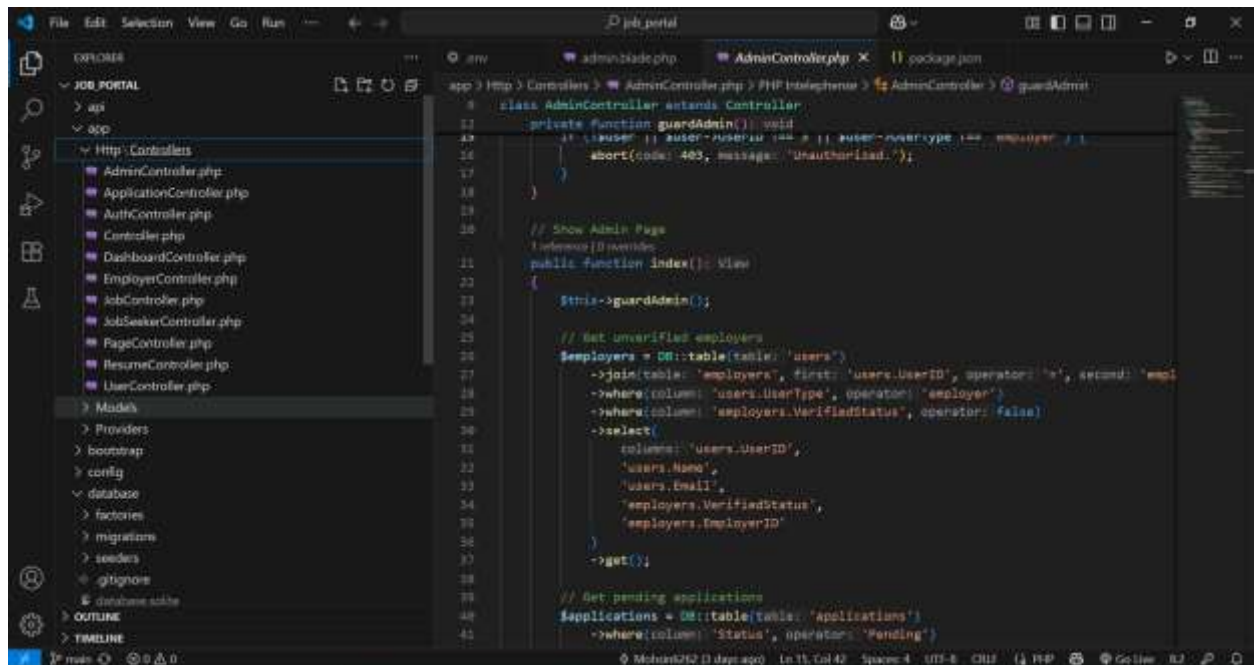
### 5.1 LARAVEL MVC IMPLEMENTATION

#### 5.1.1 MODEL LAYER



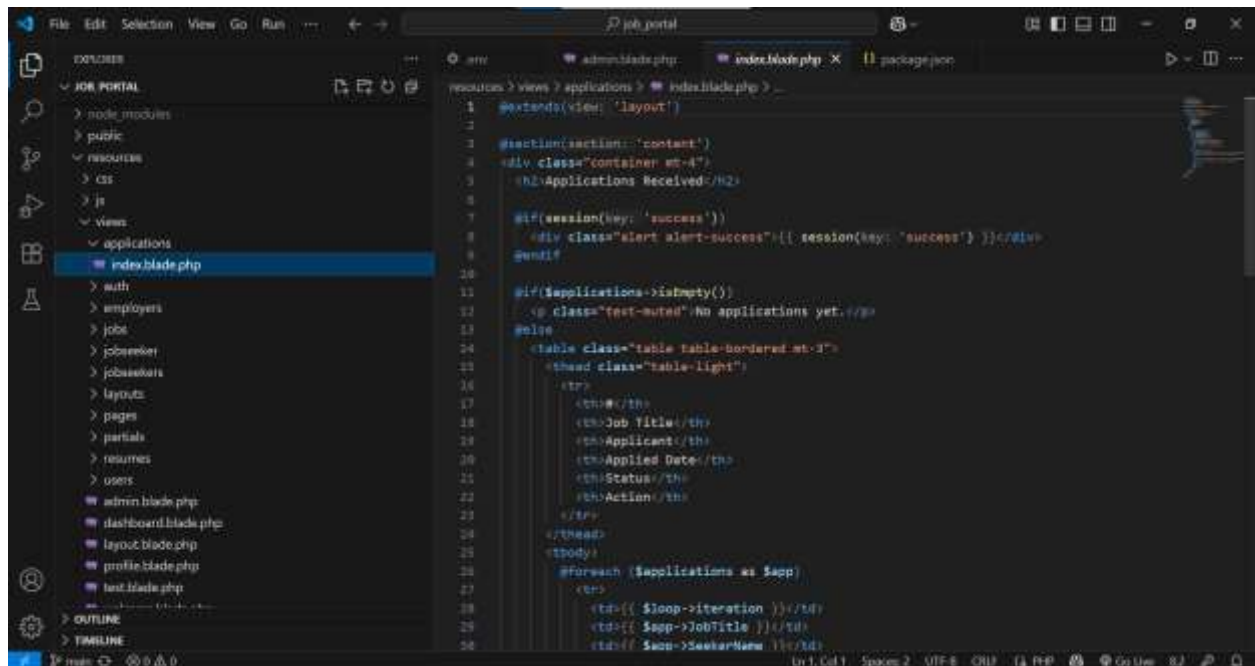
```
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use Illuminate\Support\Facades\Session;
5 use Illuminate\Support\Facades\DB;
6 use App\Http\Controllers\JobController;
7 use App\Http\Controllers\AuthController;
8 use App\Http\Controllers\ApplicationController;
9 use App\Http\Controllers\AdminController;
10 use App\Http\Controllers\DashboardController;
11
12 Route::get(url: '/test', action: function () { $jsonResponse = json([
13     'status' => 'success',
14     'message' => 'Laravel is working!',
15     'app_name' => config(key: 'app.name'),
16     'app_env' => config(key: 'app.env'),
17     'database' => config(key: 'database.default')
18 ]);
19 });
20
21 Route::get(url: '/env-check', action: function () { $jsonResponse = json([
22     try {
23         $dbConnection = DB::connection()->getPdo();
24         $dbStatus = 'connected';
25     } catch (Exception $e) {
26         $dbStatus = 'failed: ' . $e->getMessage();
27     }
28     return response()->json([data: [
29         'status' => 'success',
30         'message' => 'Laravel is working!',
31         'app_name' => config(key: 'app.name'),
32         'app_env' => config(key: 'app.env'),
33         'database' => config(key: 'database.default')
34     ]);
35 }]);
```

#### 5.1.2 CONTROLLER LAYER



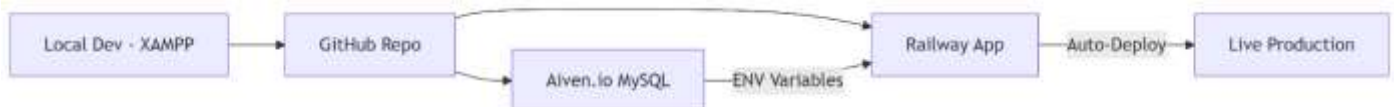
```
1 class AdminController extends Controller
2 {
3     private function guardAdmin() {
4         if (! $user || $user->userType != 'employer') {
5             abort(403, 'Unauthorized.');
```

### 5.1.3 VIEW LAYER



## 6. DEPLOYMENT ARCHITECTURE

### 6.1 DEPLOYMENT WORKFLOW DIAGRAM



## 7. DEPLOYMENT STEPS:

### 7.1 PUSH THE PROJECT ON GITHUB:

```
cd C:\xampp\htdocs\job_portal
```

```
git init
```

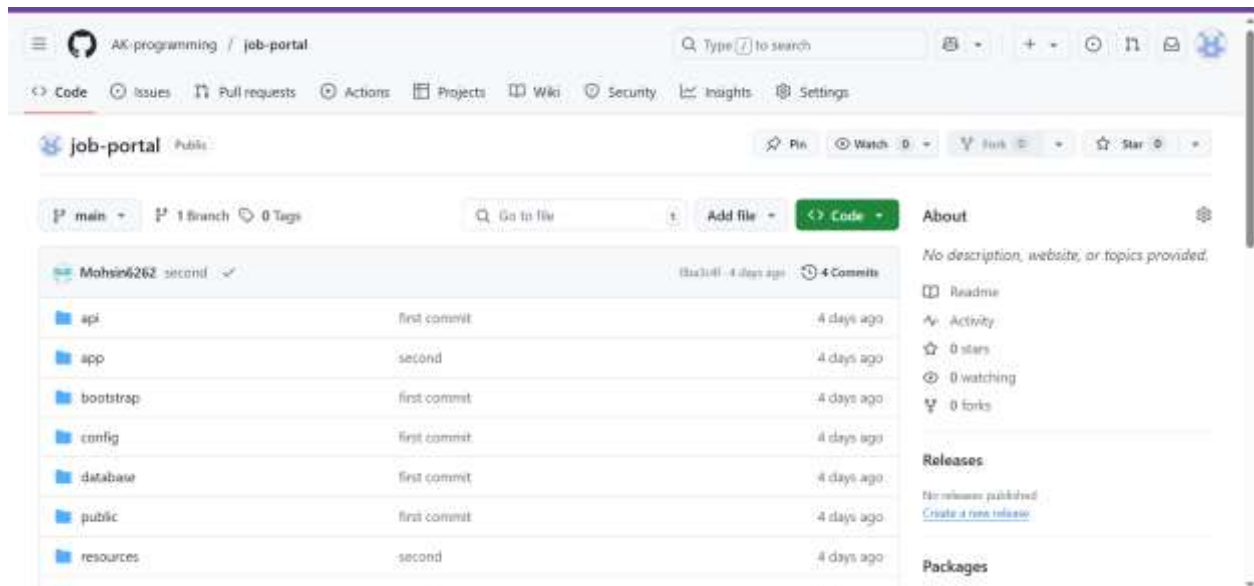
```
git remote add origin https://github.com/Ak-programming/job_portal.git
```

```
git add .
```

```
git commit -m "Pushing project"
```

```
git branch -M main
```

```
git push -u origin main
```

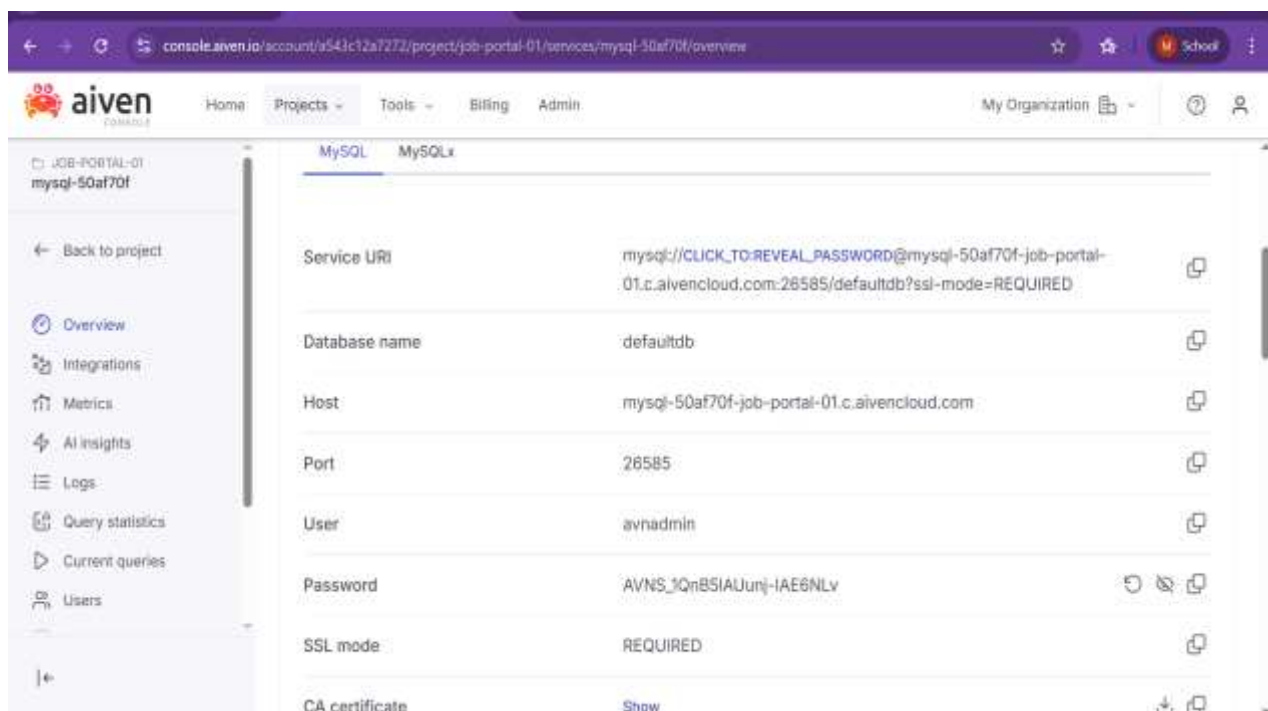
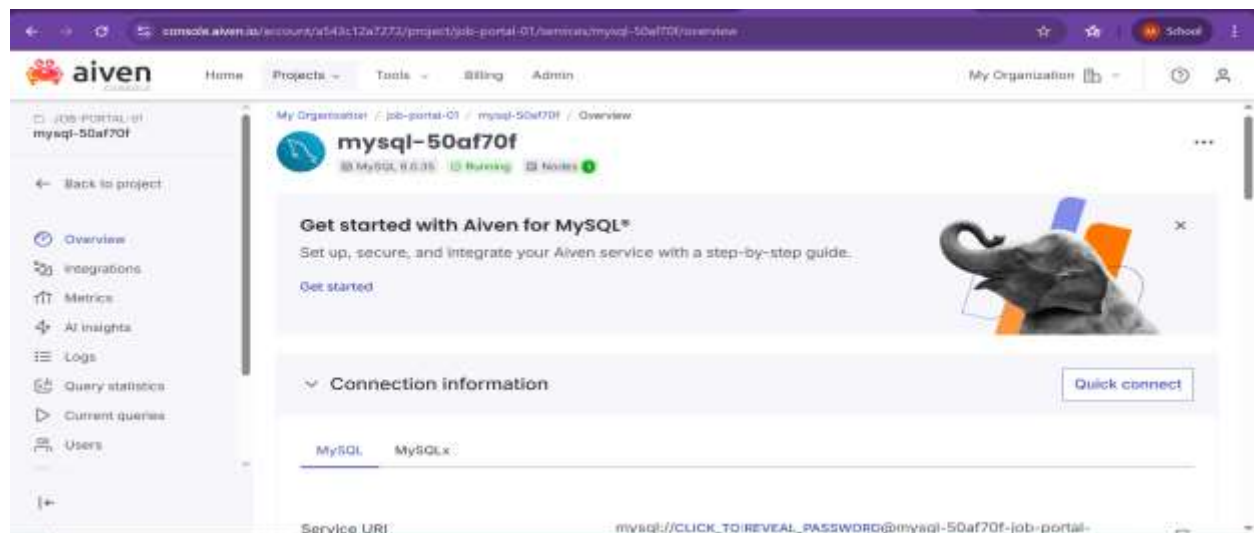


### 7.2 CREATE MYSQL DATABASE ON AIVEN.IO

- Sign in to Aiven.io and create a new MySQL service.
- After creation, note the following credentials:
  - DB\_HOST: mysql-50af70f-job-portal-01.c.aivencloud.com
  - DB\_PORT: 26585
  - DB\_DATABASE: job\_portal
  - DB\_USERNAME: avnadmin
  - DB\_PASSWORD: AVNS\_1QnB5IAUunj-IAE6NLv

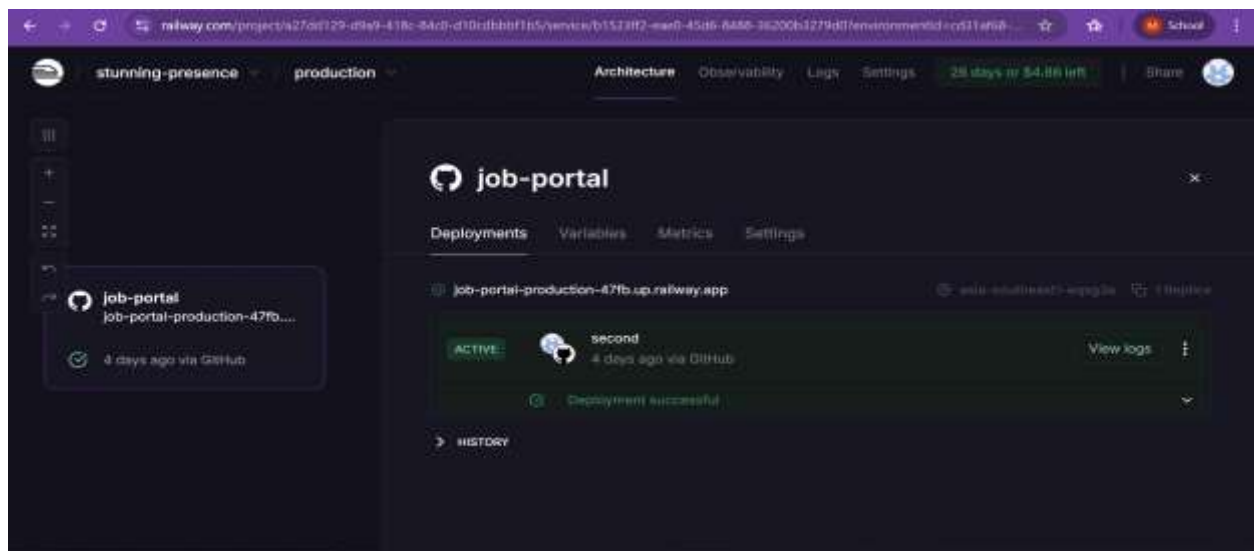
**The database can be accessed using the following information. Paste it directly into the command line:**

```
mysqlsh --sql mysql://avnadmin:AVNS_1QnB5IAUunj-IAE6NLv@mysql-50af70f-job-portal-01.c.aivencloud.com:26585/job_portal?ssl-mode=REQUIRED
```



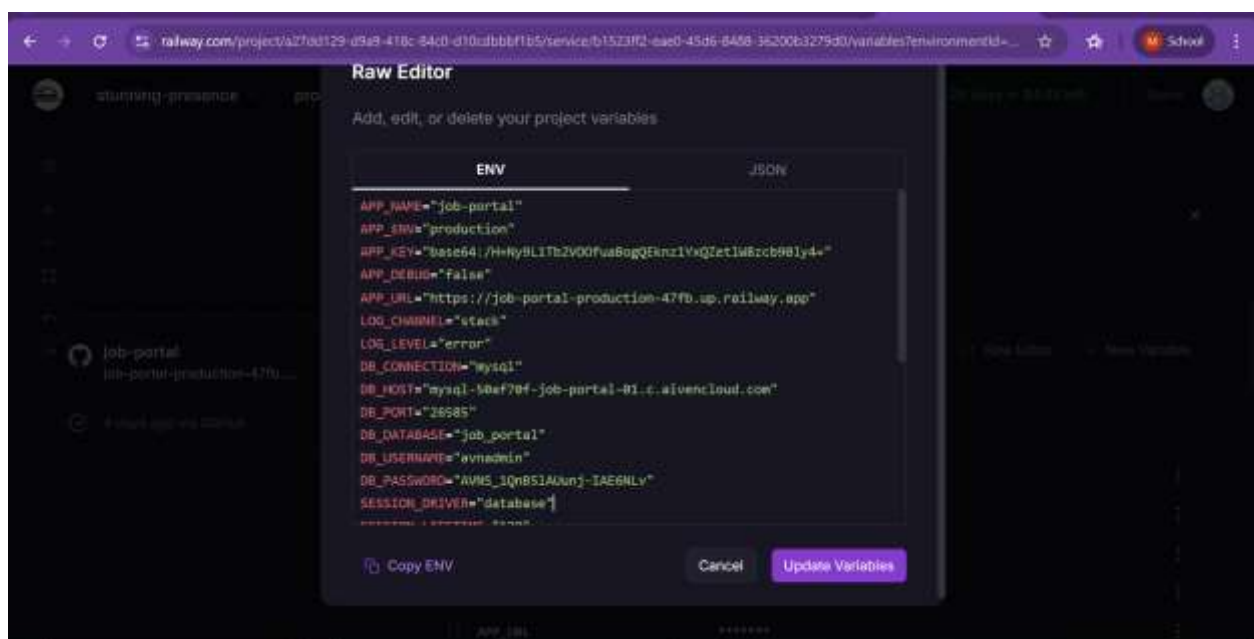
## 7.3 DEPLOY BACKEND TO RAILWAY

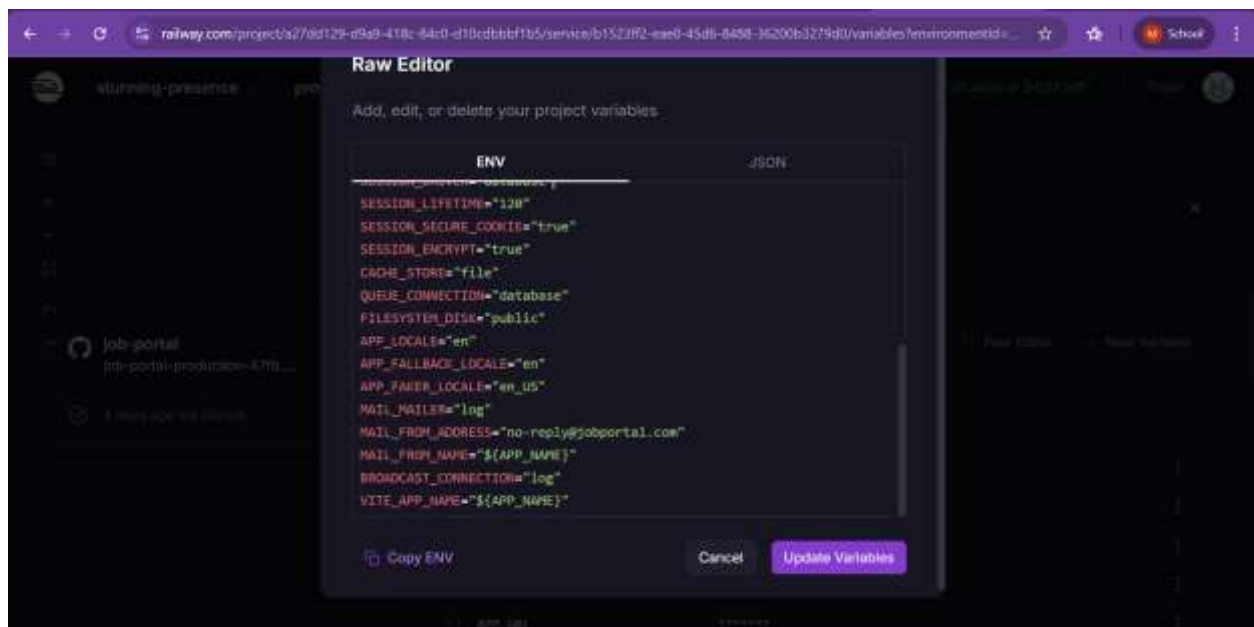
- Go to Railway.app.
- Click on "New Project" → "Deploy from GitHub".
- Select your Laravel project repository



## 7.4 CONFIGURING VARIABLE:

- After deployment, open the “Variables” tab in Railway.
- Add the following environment variables (from your .env file):





## 7.5 ACCESS USING MYSQL CLIENT:

```
Microsoft Windows [Version 10.0.19045.5965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\T470s>mysqlsh --sql mysql://avnadmin:AVNS_1QnB5IAUunj-IAE6NLv@mysql-50af70f-job-portal-01.c.aivencloud.com:26585/job_portal?ssl-mode=REQUIRED
MySQL Shell 8.0.28

Copyright (c) 2016, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
WARNING: Using a password on the command line interface can be insecure.
Creating a Classic session to 'avnadmin@mysql-50af70f-job-portal-01.c.aivencloud.com:26585/job_portal?ssl-mode=REQUIRED'
Fetching schema names for autocompletion... Press ^C to stop.
Fetching table and column names from 'job_portal' for auto-completion... Press ^C to stop.
Your MySQL connection id is 77402
Server version: 8.0.35 Source distribution
Default schema set to 'job_portal'.
MySQL mysql-50af70f-job-portal-01.c.aivencloud.com:26585 ssl job_portal SQL > show tables;

+-----+
| Tables_in_job_portal |
+-----+
| applications         |
| employers            |
| jobs                 |
| jobseekers           |
| migrations           |
| resumes              |
| sessions             |
+-----+
```

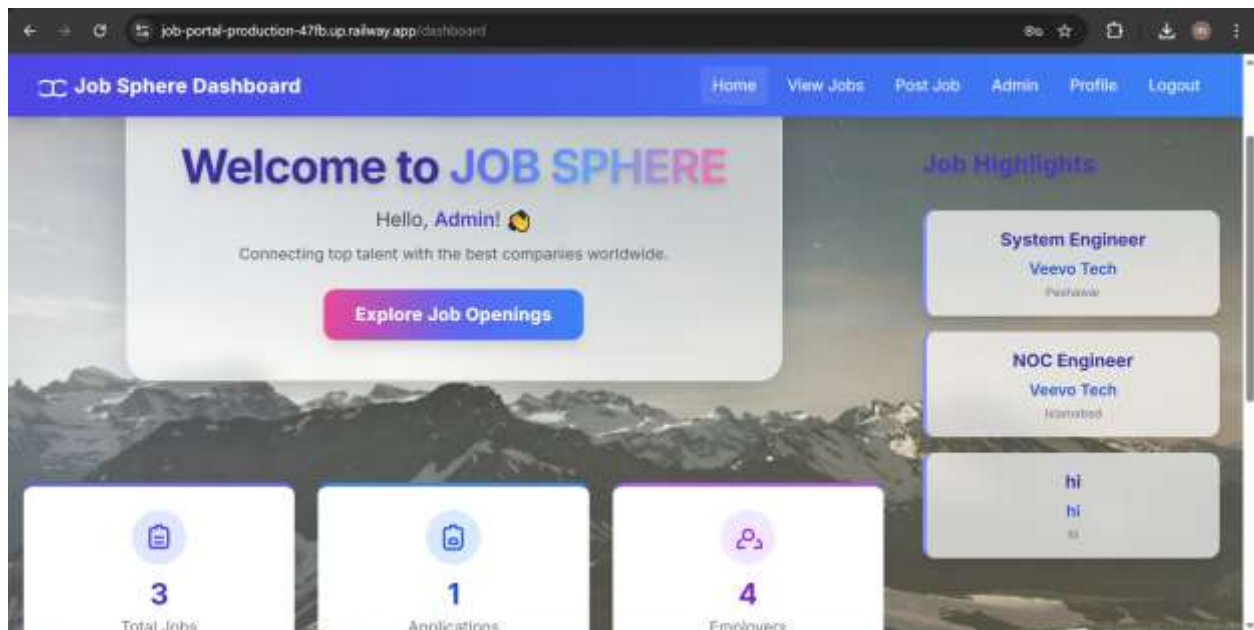
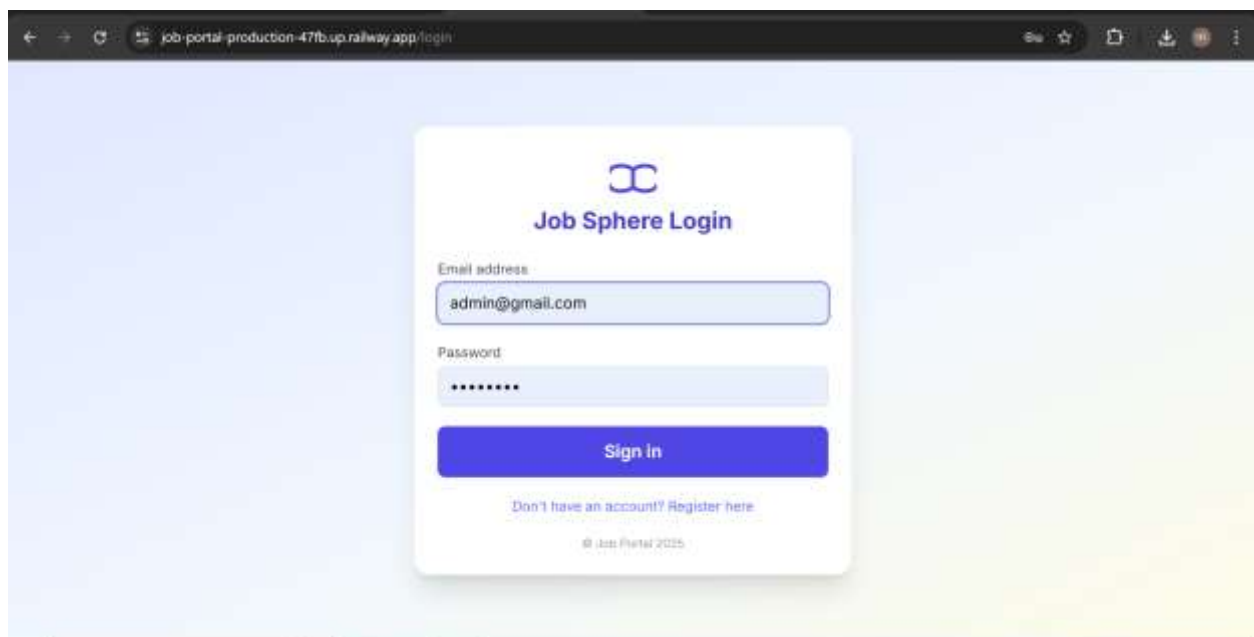


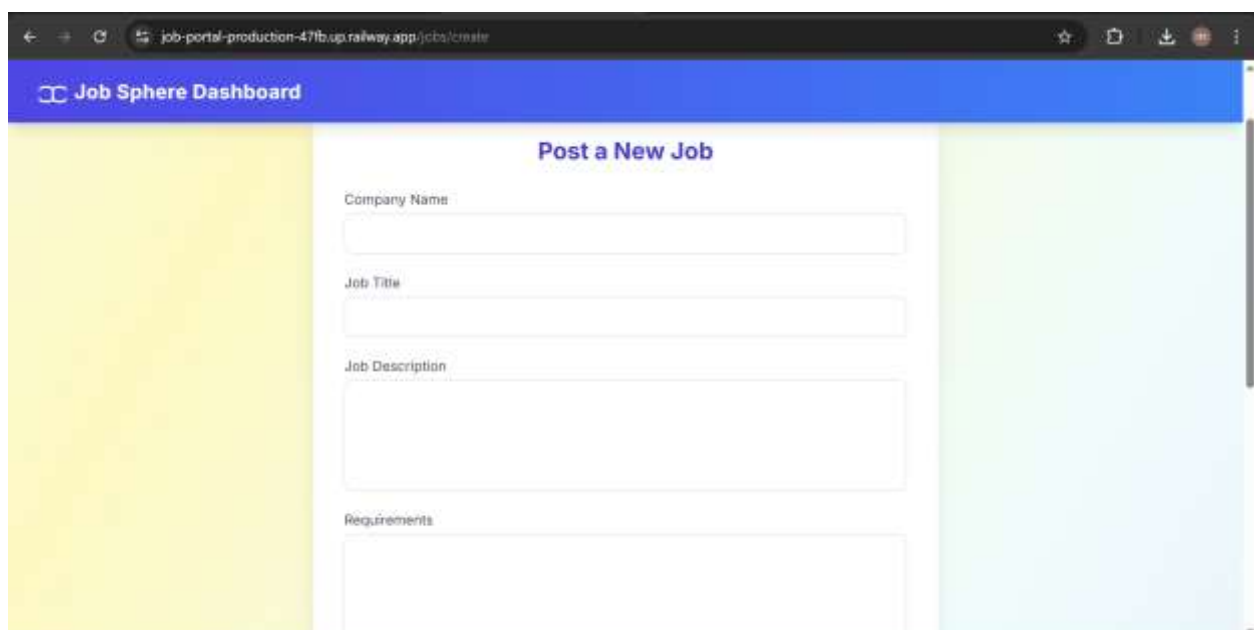
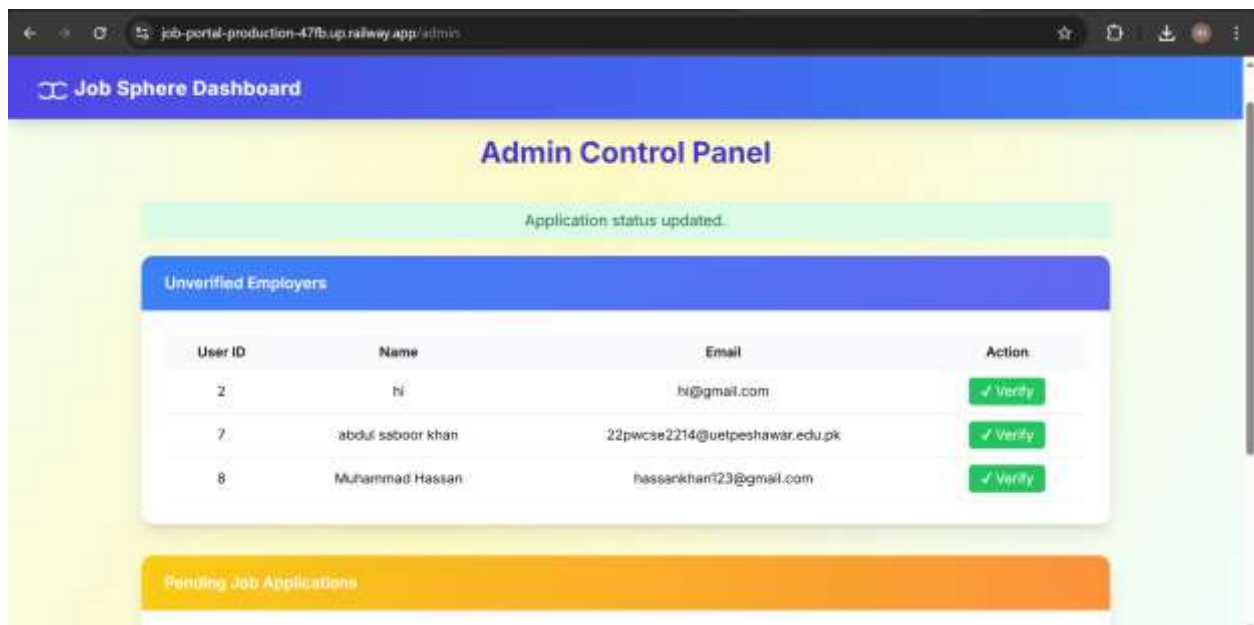
## 8. SYSTEM FUNCTIONALITY

### 8.1 ROLE-BASED ACCESS CONTROL (RBAC)

#### 8.1.1 ADMINISTRATOR FUNCTIONS

- **User Management:** View, verify, and manage user accounts
- **Content Moderation:** Review and approve job postings
- **Application Oversight:** Monitor application status and statistics
- **System Analytics:** Generate reports on platform usage
- **Employer Verification:** Approve or reject employer registrations





## 8.1.2 EMPLOYER FUNCTIONS

- **Job Management:** Create, edit, and delete job postings
- **Application Review:** View and manage received applications
- **Company Profile:** Update company information and branding
- **Candidate Communication:** Send messages to applicants
- **Analytics Dashboard:** Track job posting performance



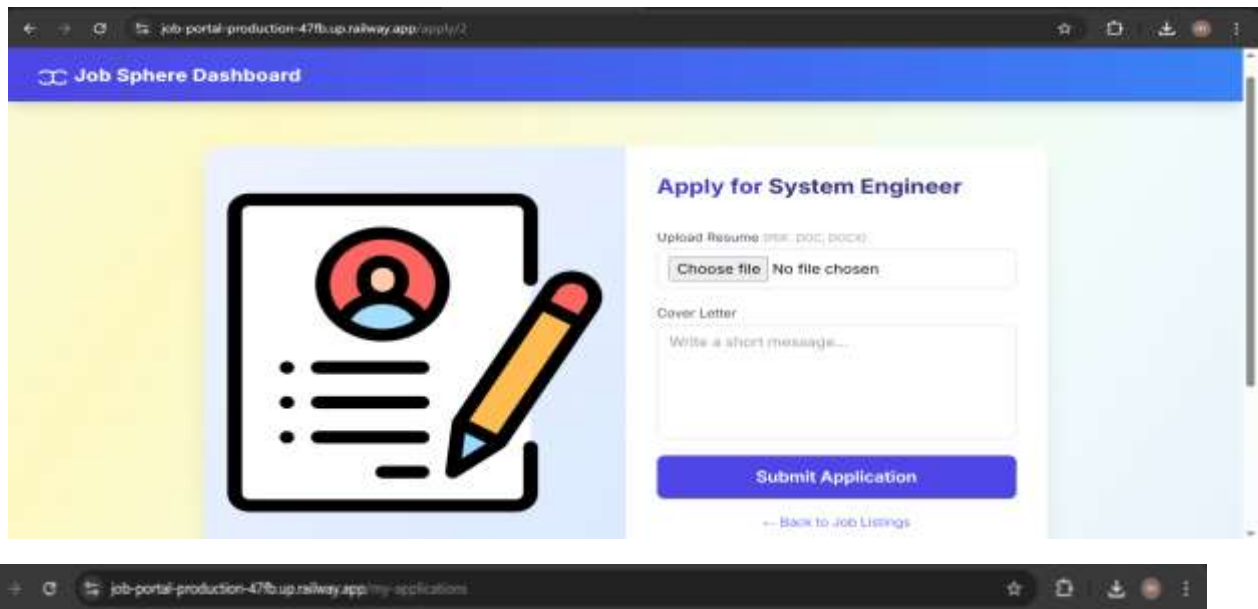
The screenshot shows a web browser window with the URL `job-portal-production-47fb.up.railway.app/jobs/create`. The page has a blue header with the 'Job Sphere Dashboard' logo. The main content area is titled 'Post a New Job' and contains a form with four input fields: 'Company Name', 'Job Title', 'Job Description', and 'Requirements'. The form is set against a background with a yellow-to-blue gradient.

### 8.1.3 JOB SEEKER FUNCTIONS

- **Profile Management:** Update personal information and skills
- **Resume Upload:** Store and manage multiple resume versions
- **Job Search:** Filter and search available positions
- **Application Tracking:** Monitor application status and history
- **Notification Center:** Receive updates on application progress

The screenshot shows a web browser window with the URL `job-portal-production-47fb.up.railway.app/jobs`. The page has a blue header with the 'Job Sphere Dashboard' logo. The main content area is titled 'Find Your Next Opportunity' with the subtitle 'Search thousands of jobs from top companies'. Below this is a search bar with the placeholder text 'Job title, keywords, or company' and a 'Search' button. The page displays three job listings in a grid:

System Engineer	NOC Engineer	hi
<p>% Peshawar</p> <p>Veevo Tech is a leading tech company specializing in innovative software development, A2P SMS/WhatsApp (CPaaS), IoT solu...</p>	<p>% Islamabad</p> <p>Veevo Tech is a leading tech company specializing in innovative software development, A2P SMS/WhatsApp (CPaaS), IoT solu...</p>	<p>% hi</p> <p>hi</p>
<p>Show Details</p> <p>Apply Now</p>	<p>Show Details</p> <p>Apply Now</p>	<p>Show Details</p> <p>Apply Now</p>



## My Applications

Job Title	Applied Date	Status
System Engineer	03 Jul, 2025	Reviewed

## 9. SECURITY IMPLEMENTATION

### 9.1 ACCESS CONTROL

- Authentication: Multi-factor authentication support
- Authorization: Role-based permissions system
- Data Validation: Server-side input validation
- File Upload Security: MIME type validation and virus scanning
- Rate Limiting: API request throttling

## 10. FUTURE ENHANCEMENTS

### 10.1 SCALABILITY IMPROVEMENTS

- Microservices Architecture: Service decomposition for better scalability
- API Development: RESTful API for mobile app integration
- Real-time Features: WebSocket integration for instant notifications
- Machine Learning: Job recommendation algorithms
- Analytics Platform: Advanced reporting and business intelligence

## 10.2 FEATURE EXPANSIONS

- Video Interviews: Integrated video conferencing
- Skills Assessment: Online testing and certification
- Social Features: Professional networking capabilities
- Mobile Application: Native iOS and Android apps
- Multi-language Support: Internationalization features

## 11. CONCLUSION

### 11.1 CONCLUSION:

Job Sphere represents a comprehensive demonstration of database management system principles applied to real-world problem-solving. Through careful design, implementation, and deployment, we have created a scalable, secure, and user-friendly platform that addresses genuine market needs.

### 11.2 THE PROJECT SUCCESSFULLY DEMONSTRATES:

- Database Design Excellence: Normalized schema with proper relationships
- Full-stack Development: Complete web application with modern technologies
- Cloud Deployment: Production-ready hosting with proper CI/CD
- Security Implementation: Industry-standard security practices
- Performance Optimization: Efficient queries and caching strategies

This experience has provided invaluable insights into enterprise-level application development, from conception through deployment, preparing us for professional software development careers.

## 12. REFERENCES

### 12.1 REFERENCES

1. Laravel Documentation. (2024). *Laravel 12 Framework Documentation*. Retrieved from <https://laravel.com/docs/12.x>
2. MySQL Documentation. (2024). *MySQL 8.0 Reference Manual*. Retrieved from <https://dev.mysql.com/doc/>
3. OpenAI. (2024). *ChatGPT conversations for code assistance and debugging*. Retrieved from <https://chat.openai.com/>
4. Railway Documentation. (2024). *Railway hosting documentation*. Retrieved from <https://docs.railway.app/>
5. Aiven Documentation. (2024). *Database service documentation*. Retrieved from <https://aiven.io/docs/>
6. Bootstrap Documentation. (2024). *Bootstrap 5 CSS Framework*. Retrieved from <https://getbootstrap.com/docs/5.0/>
7. Diagrams.net. (2024). *Free diagram software*. Retrieved from <https://www.diagrams.net/>
8. PHP Documentation. (2024). *PHP 8.3 Manual*. Retrieved from <https://www.php.net/manual/en/>

### 12.2 DEPLOYMENT URL:

Live Application URL: <https://job-portal-production-47fb.up.railway.app>

### 12.3 GITHUB LINK:

GitHub Repository: <https://github.com/AK-programming/job-portal.git>