**PROJECT SYNOPSIS REPORT**

**ON**

**InternSpot – Job Portal**

**SUBMITTED**

**TO**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**FOR**

**Full Stack Engineering (22CS037)**



Submitted To:- Submitted By:-

Mr. Rahul Singh Tanish Gupta(2310991063).

Sulabh Saluja(2310991060).

Tarun Garg(2310991149).

Anmol Garg (2310990285).

# Index

Sr. no Topic Page No

1. Problem Statement 3
2. Title of project 3
3. Objective & Key Learning’s 3
4. Options available to Execute the project 3
5. Tech Stack 3
6. Advantages/ Disadvantages 4
7. Implementation Strategy 4
8. Conclusion 5
9. References 5

### **Problem Statement:-**

In the competitive job market, students and freshers often struggle to find suitable internships and entry-level opportunities, while employers face challenges in reaching the right candidates efficiently. Existing job portals are either too complex, costly, or lack features tailored for academic and fresher recruitment.

There is a need for a simple, user-friendly, and scalable web-based platform that connects job seekers with employers, enabling effective posting, searching, and management of opportunities.

**Title of Project:-** InternSpot – Job Portal

**Objective & Key Learnings:-**

• To develop a role-based system for job seekers and employers with secure authentication.  
• To implement modules for job posting, searching, filtering, and application tracking.  
• To learn integration of MongoDB, Express.js, Node.js, and EJS in a full-stack application.  
• To gain hands-on experience with file uploads (resumes/logos), middleware, and session management.  
• To understand project lifecycle from design, backend APIs, frontend templating, to deployment.

**Options Available to Execute the Project:-**

• **Monolithic Architecture (Chosen):** Easy to develop and manage for academic project scale.  
• **Microservices Architecture:** More scalable, modular, but complex for current scope.  
• **Cloud Deployment:** Can be deployed on services like AWS, Render, or Vercel for real-world use.  
**• Frontend Framework Option:** Could be enhanced with React.js/Angular for richer UI, but EJS was used for simplicity.  
**• Database Options:** MongoDB (chosen for flexibility and JSON document support), or SQL-based databases like MySQL.

### **Tech Stack:**

• **Backend:** Node.js, Express.js  
**• Database:** MongoDB with Mongoose  
• **Frontend:** EJS templates, HTML, CSS, JavaScript  
**• Authentication:** express-session, connect-mongo, bcryptjs  
**• Middleware:** multer (file uploads), express-validator (validation), method-override  
• **Deployment Options:** Local server, cloud hosting (e.g., AWS, Heroku)

### **Advantages and Disadvantages:**

### **Advantages:**

• Simple and user-friendly interface for both employers and seekers.  
• Secure authentication and session handling with MongoDB persistence.  
• Supports file uploads (resume, company logo).  
• Role-based dashboards for better separation of functionality.  
• Scalable due to use of Node.js and MongoDB.

#### Disadvantages:

• UI built with EJS is functional but not as modern as React/Angular.  
• Limited advanced features (no real-time chat, payment integration, analytics).  
• Lacks automated testing and DevOps pipeline.

**Implementation Strategy:**

**1. Backend Development**  
- Set up Express server, configure MongoDB connection.  
- Define Mongoose models: User, Job, Application.  
- Implement routes for auth, job posting, applications, dashboards.

**2. Authentication & Security**  
- Implement role-based authentication (employer/seeker).  
- Secure passwords with bcryptjs.  
- Store sessions in MongoDB using connect-mongo.

**3. Job Portal Features**  
- Employer dashboard for posting and managing jobs.  
- Seeker dashboard for profile management and applying to jobs.  
- Application tracking for both roles.

**4. File Uploads & Validation**  
- Use multer middleware for uploading resumes and logos.  
- Apply validations for input forms using express-validator.

**5. Frontend Integration**  
- Create responsive EJS views for login, registration, dashboards, job listings.  
- Integrate flash messages for feedback.

**6. Testing & Deployment**  
- Manual testing of flows (registration, login, posting, applying).  
- Future deployment on cloud (e.g., AWS/Heroku) with environment variables.

### **Conclusion:**

The InternSpot Job Portal project demonstrates a full-stack web application connecting employers and job seekers on a single platform. With secure authentication, role-based dashboards, job posting, and application tracking, it provides the essential features of a recruitment portal. Built using Node.js, Express, and MongoDB, the project showcases scalability, simplicity, and practical application of full stack engineering concepts. It serves as a strong foundation for further enhancements.

**References:-**

• https://www.mongodb.com/docs/  
• https://expressjs.com/  
• https://nodejs.org/en/docs  
• https://www.geeksforgeeks.org/  
• https://developer.mozilla.org/