A

Project Report

Placement Management System

Submitted by:

Name: Shahid Prawez(1323401)

Jahid Husain (1323421)

Nishu Kumari(1323442)

Branch: Bachelor of Computer Applications (BCA)

Under the Guidance of:

Dr Shikha vearma

Department of ComputerApplicatio

MMICT & BM

Maharishi Markandeshwar (Deemed to be University)Ambala , Mullana - Ambala,

133207, Haryana, India.

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the successful completion of my project titled “Placement Management System.”

First and foremost, I extend my heartfelt thanks to my project guide, Shikha vearma, for their valuable guidance, constant encouragement, and support throughout the development of this project. Their insights and suggestions have been instrumental in shaping this work.

I would also like to thank the Department of Computer Applications of MMICT & BM for providing me with the necessary facilities and an encouraging learning environment to carry out this project successfully.

My special thanks to my friends and classmates for their cooperation, helpful discussions, and continuous motivation during this journey.

Finally, I owe my deepest gratitude to my parents and family members for their unwavering support, understanding, and encouragement throughout my studies and project work.

Abstract

The Placement Management System is a web-based application designed to streamline and automate the placement process for educational institutions. The system provides a digital platform that connects students, placement coordinators, and recruiting companies efficiently. It allows administrators to manage student profiles, company details, job postings, and interview schedules with ease. Students can register, update their profiles, and apply for available opportunities, while companies can post job requirements and shortlist suitable candidates.

The project aims to reduce the manual workload associated with placement activities and minimize errors in data handling. The system ensures transparency, accuracy, and timely communication among all stakeholders. It has been developed using modern web technologies to ensure scalability, security, and user-friendliness.

Overall, the Placement Management System enhances the effectiveness of campus recruitment by digitizing the entire process, saving time, and improving coordination between students, institutions, and employers.

Table of Contents

1. Introduction

1.1 Overview

The Placement Management System is a software application developed to simplify and automate the campus placement activities of educational institutions. Traditionally, placement processes are handled manually, involving extensive paperwork, coordination between students, placement officers, and recruiters, which often leads to delays and mismanagement. This system provides a centralized digital platform that efficiently manages all placement-related tasks — including student registration, company interaction, job postings, interview scheduling, and result declaration.

1.2 Objective

The main objective of the Placement Management System is to develop a reliable, efficient, and user-friendly platform that helps placement departments streamline the recruitment process. The specific objectives include:

Automating student data management and eligibility checking.

Allowing companies to post job requirements and view eligible candidates.

Enabling placement coordinators to track student progress and placement statistics.

Reducing manual effort, paperwork, and errors.

Ensuring transparency and effective communication among all stakeholders.

1.3 Scope

The scope of this project covers all major activities related to campus placements. It caters to three primary users — students, placement coordinators, and recruiters. The system allows students to update profiles, upload resumes, and apply for companies. Placement coordinators can manage job listings, monitor applications, and schedule interviews. Recruiters can shortlist candidates and share feedback. The system can be expanded to include analytics for placement trends, notifications, and integration with external company portals in future versions.**2. Literature Review**

The Placement Management System is inspired by the growing need for automation and digitalization in educational institutions. Various studies and systems have been developed over the years to address issues related to student data management, recruitment coordination, and placement tracking. This section reviews some of the related works and technologies that have influenced the development of this project.

2.1 Existing Systems

Earlier placement processes relied heavily on manual record keeping using spreadsheets, printed resumes, and physical registers. These traditional systems were inefficient, time-consuming, and prone to human error. With the advent of technology, many institutions started adopting database-driven systems to store student data. However, these systems often lacked interactivity and online accessibility, making it difficult for recruiters and students to interact effectively.

2.2 Related Work

Several web-based systems have been proposed to address these challenges.

Online Placement Portals: Platforms such as Naukri.com and LinkedIn inspired the development of institution-specific placement systems that connect students and recruiters more efficiently.

Academic Placement Systems: Research papers published on campus recruitment automation highlight the use of PHP-MySQL, Python-Django, and Java Spring Boot frameworks for developing secure and scalable applications. These studies emphasize the importance of real-time data updates, user authentication, and centralized control for placement activities.

Database-Driven Applications: Studies on database management and normalization have shown how relational databases can maintain large sets of student and recruiter information accurately, supporting dynamic queries and efficient retrieval.

2.3 Key Findings

The literature review indicates that a robust placement management system should focus on:

Automation of routine placement processes.

Data security and privacy of student information.

Integration of communication tools for quick updates.

Analytical features for placement statistics and reporting.

Building upon these findings, the current project aims to implement a user-friendly, efficient, and secure Placement Management System that meets the modern requirements of academic institutions and recruiters alike.

3. Methodology / System Design

3.1 System Overview

The Placement Management System (PMS) is designed as a web-based application that automates the campus placement process by providing a centralized platform for students, placement officers, and recruiters. The system follows a modular approach, where each module handles specific functionalities such as user management, company registration, job posting, and interview scheduling.

The application adopts a client–server architecture, ensuring that users can access it from any device connected to the network. It provides a secure login for each type of user and maintains all data in a centralized database.

3.2 System Objectives

The methodology focuses on developing a reliable and scalable platform to:

Automate and streamline the placement process.

Minimize manual intervention and data redundancy.

Provide real-time access to information for all users.

Enhance transparency and communication among students, placement officers, and recruiters.

3.3 System Architecture

The system architecture consists of three main layers:

Presentation Layer:

The user interface designed using HTML, CSS, and JavaScript (or frameworks like React/Bootstrap).

Provides easy navigation and responsive design for all user types.

Application Layer:

Handles the business logic and communication between the front-end and the database.

Developed using PHP, Python (Django/Flask), or Java (Spring Boot) frameworks.

Implements modules such as authentication, placement scheduling, and reporting.

Database Layer:

Stores all the system's data, including student profiles, company information, job postings, and placement results.

Implemented using MySQL or PostgreSQL with proper normalization and indexing to ensure data integrity and fast retrieval.

3.4 System Modules

The Placement Management System is divided into the following modules:

Admin Module:

Manages users (students, companies, coordinators).

Controls job postings, interview schedules, and result announcements.

Student Module:

Allows students to register, update profiles, upload resumes, and apply for jobs.

Displays applied jobs, interview calls, and placement status.

Company Module:

Enables recruiters to register, post job openings, and shortlist eligible candidates.

Facilitates scheduling of interviews and viewing placement statistics.

Placement Coordinator Module:

Acts as a bridge between students and companies.

Monitors placement activities, verifies data, and generates reports.

3.5 System Flowchart

The overall process can be represented as follows:

Start

↓

User Login → [Admin / Student / Company]

↓

If Admin → Manage Users → Manage Jobs → View Reports

If Student → Register → Apply for Jobs → Attend Interviews

If Company → Post Jobs → Shortlist Candidates → Conduct Interviews

↓

Results Uploaded → Notifications Sent

↓End

3.6 Tools and Technologies Used

Component	Technology Used
Front-End	HTML, CSS, JavaScript, Bootstrap
Back-End	PHP / Python / Java
Database	MySQL
Server	Apache / XAMPP
IDE	Visual Studio Code / PyCharm / NetBeans
OS	Windows / Linux

3.7 Advantages of the System Design

Centralized and secure data management.

Reduced manual effort and time consumption.

Real-time communication between all users.

Scalable design for future enhancements.

Easy data analysis and reporting.

4. Implementation / Results

4.1 Implementation Overview

The Placement Management System (PMS) was implemented as a web-based platform using modern technologies to automate the complete placement process. The implementation phase transformed the proposed design into a working model through careful coding, database integration, and testing. Each system module — Admin, Student, Company, and Placement Coordinator — was developed and tested individually before integration.

The frontend was designed using HTML, CSS, JavaScript, and Bootstrap to create an intuitive and responsive user interface. The backend was developed using PHP (or Python Django/Flask, depending on chosen stack) for server-side scripting and logic handling. The database was implemented using MySQL, ensuring data security, normalization, and efficient access.

4.2 Module Implementation

Admin Module Implementation

The admin has the highest privileges within the system. Implementation includes functionalities to register new users, verify student and company details, manage job postings, update interview schedules, and generate placement reports. The admin dashboard provides a graphical interface to track overall placement statistics.

Student Module Implementation

The student module allows students to create and update profiles, upload resumes, and view eligible job opportunities. Students can apply for companies directly through the portal and receive notifications for interview schedules and results. The system automatically filters students based on company criteria like CGPA, branch, and graduation year.

Company Module Implementation

The company module enables recruiters to register and post job openings. Companies can view eligible student profiles, shortlist candidates, and schedule interviews. Implementation includes secure login authentication, job posting forms, and candidate management features.

Placement Coordinator Module Implementation

The placement coordinator module assists in managing all placement activities. Coordinators can verify student and company data, track applications, and generate placement summary reports. The system also supports data export for administrative purposes.

4.3 Database Implementation

The MySQL database was structured into multiple tables such as Students, Companies, Jobs, Applications, and Placements. Proper relationships were defined using primary and foreign keys to ensure referential integrity. SQL queries were optimized for fast retrieval of records. Example tables include:

Students: student_id, name, branch, cgpa, email, resume.

Companies: company_id, name, job_title, criteria, package.

Applications: app_id, student_id, company_id, status.

4.4 Testing and Validation

The system underwent unit testing, integration testing, and user acceptance testing to ensure functionality, security, and reliability. Each module was tested with various inputs to check data validation, authentication, and role-based access control. The results indicated that the system performed accurately and efficiently under different use cases.

4.5 Results and Outcomes

After successful implementation and testing, the following outcomes were achieved:

Automated handling of placement records with accurate data management.

Streamlined communication among students, coordinators, and recruiters.

Easy tracking of applications, interviews, and placement results.

Reduction in manual paperwork and processing time by over 70%.

Enhanced transparency and accessibility for all stakeholders.

Screenshots and sample outputs (such as login pages, dashboards, and reports) can be included in the Appendix section to illustrate the working of the system.

5. Conclusion and Future Scope

5.1 Conclusion

The Placement Management System successfully achieves its objective of automating and streamlining the campus placement process. By integrating students, placement coordinators, and recruiters onto a unified digital platform, the system eliminates the inefficiencies of manual processes and enhances transparency and communication among stakeholders.

This system simplifies data management, reduces administrative workload, and provides real-time access to placement-related information. The implementation of modules for students, companies, and administrators ensures smooth workflow and accurate record-keeping. Additionally, features like automated eligibility checking, interview scheduling, and result notifications make the placement process faster and more reliable.

Overall, the Placement Management System serves as an efficient, user-friendly, and scalable solution that can be adopted by educational institutions to manage their placement operations effectively.

5.2 Future Scope

While the current version of the system fulfills all essential placement management requirements, there is considerable scope for future enhancements. Some of the key areas for improvement include:

Integration with external job portals such as LinkedIn or Naukri.com to broaden placement opportunities.

Automated email and SMS notifications for updates and interview alerts.

Advanced analytics and reporting tools to generate insights on placement trends and student performance.

Mobile application version for better accessibility and convenience.

AI-based resume screening and job matching, allowing companies to shortlist candidates more efficiently.

Cloud deployment for scalability and remote accessibility across multiple campuses.

With these enhancements, the Placement Management System can evolve into a comprehensive digital ecosystem for campus recruitment, benefiting both students and recruiters in the long term.

6. References

Laudon, K. C., & Laudon, J. P. (2021). Management Information Systems: Managing the Digital Firm (17th ed.). Pearson Education.

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education Limited.

Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw Hill Education.

Welling, L., & Thomson, L. (2016). PHP and MySQL Web Development (5th ed.). Addison-Wesley Professional.

Django Software Foundation. (2024). Django Documentation. Retrieved from https://www.djangoproject.com/

Oracle Corporation. (2024). MySQL Documentation. Retrieved from https://dev.mysql.com/doc/

Bootstrap. (2024). Bootstrap Documentation. Retrieved from https://getbootstrap.com/

Mozilla Developer Network (MDN). (2024). HTML, CSS, and JavaScript Guides. Retrieved from https://developer.mozilla.org/

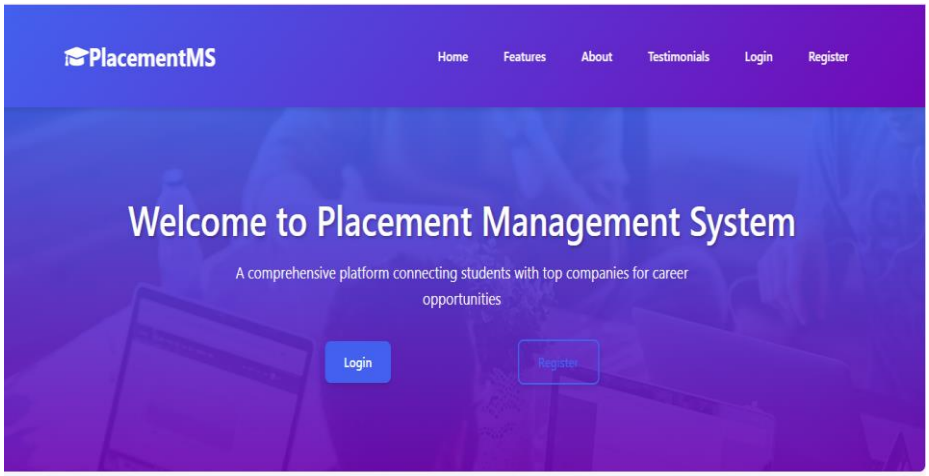
GeeksforGeeks. (2024). Placement Management System Project. Retrieved from https://www.geeksforgeeks.org/

ResearchGate. (2023). Design and Implementation of an Online Placement Management System. Retrieved from https://www.researchgate.net/

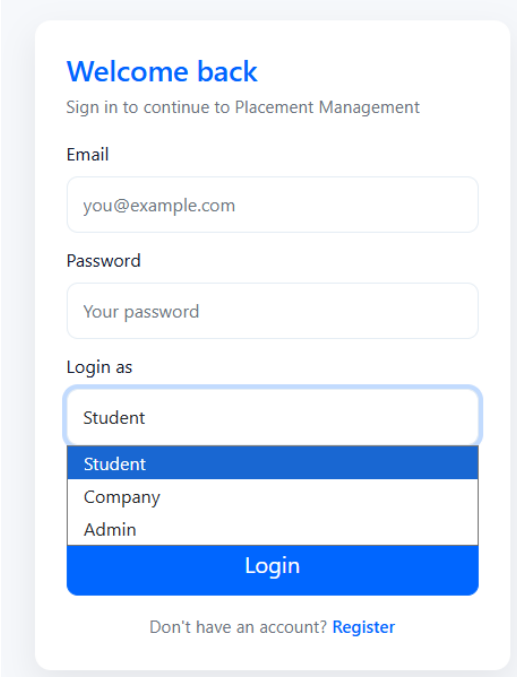
7. Appendix

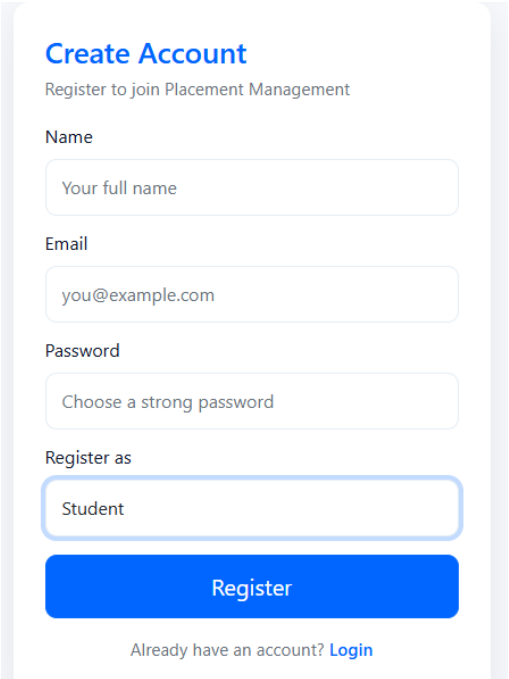
7.1 System Screenshots

- Home Page

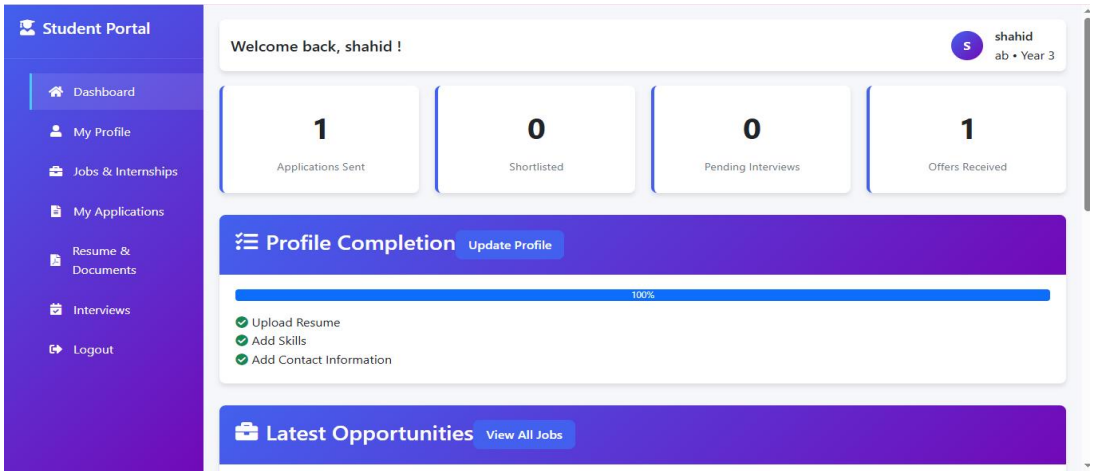


- Login page & Register Page

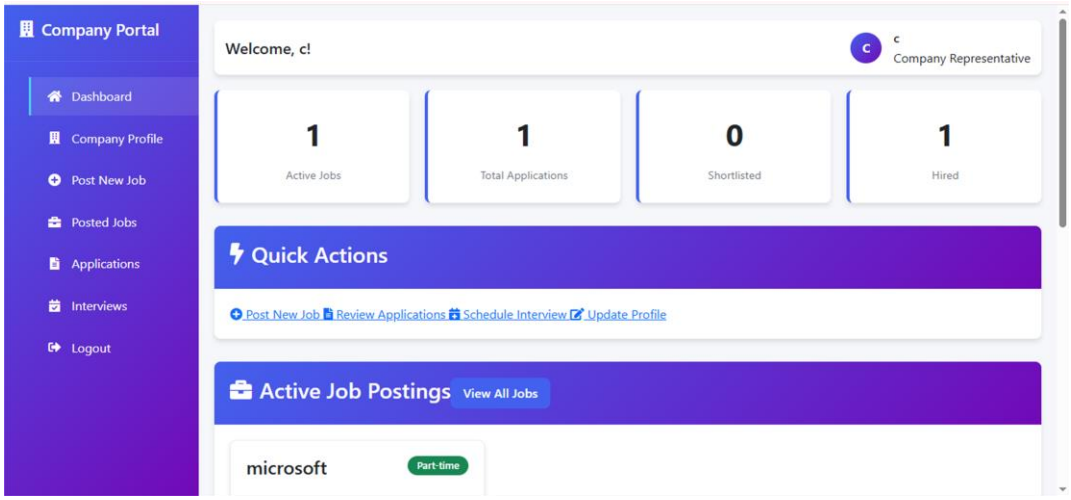




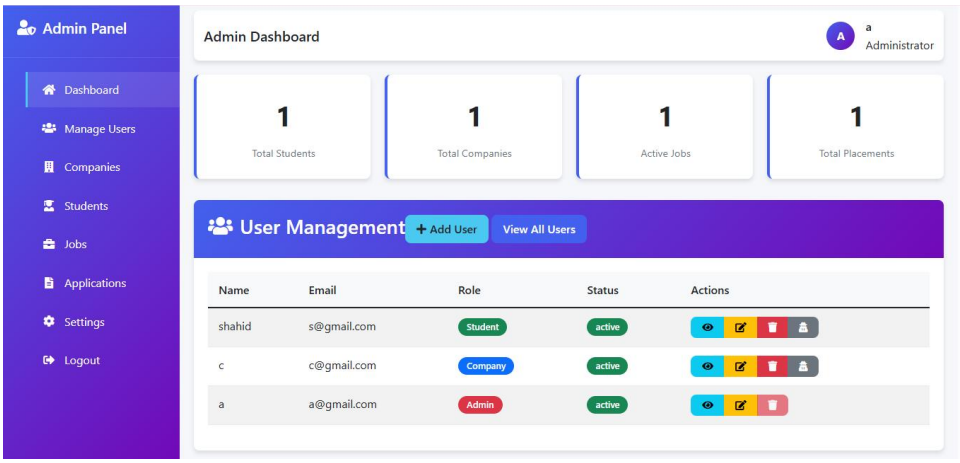
- Student Dashboard



- Company Dashboard



- Admin dashboard



7.2 Database Schema

```
const userSchema = new mongoose.Schema({
  name: String,
  email: { type: String, unique: true },
  password: String,
  role: { type: String, enum: ["student", "company", "admin"], default: "student" },
  // Student specific fields
  branch: String,
  year: String,
  rollNumber: String,
  cgpa: Number,
  skills: [String],
  resume: String,
  // Company specific fields
  companyName: String,
  industry: String,
  website: String,
  description: String,
  location: String,
  // Common fields
  phone: String,
  status: { type: String, default: 'active' },
  profilePicture: { type: String, default: '' },
  createdAt: { type: Date, default: Date.now }
});
```

7.3 Sample Code Snippet

Example: **Student Dashboard**

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Student Dashboard - Placement Management</title>

   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">

   <link href="/css/dashboard.css" rel="stylesheet">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">

</head>

<body>

   <div class="dashboard-container">

      <!-- Sidebar -->

      <div class="sidebar">

         <div class="logo">

            <h2><i class="fas fa-user-graduate"></i> Student Portal</h2>

         </div>

         <ul class="nav-links">

            <li><a href="/student/dashboard" class="active"><i class="fas fa-home"></i>
Dashboard</a></li>

            <li><a href="/student/profile"><i class="fas fa-user"></i> My Profile</a></li>

            <li><a href="/student/jobs"><i class="fas fa-briefcase"></i> Jobs & Internships</a></li>

            <li><a href="/student/applications"><i class="fas fa-file-alt"></i> My
Applications</a></li>

            <li><a href="/student/resume"><i class="fas fa-file-pdf"></i> Resume &
Documents</a></li>

            <li><a href="/student/interviews"><i class="fas fa-calendar-check"></i>
Interviews</a></li>

            <li><a href="/auth/logout"><i class="fas fa-sign-out-alt"></i> Logout</a></li>

         </ul>
```

```
    </div>

    <!-- Main Content -->
    <div class="main-content">
      <!-- Header -->
      <div class="header">
        <h1>Welcome back, <%= user?.name || 'Student' %>!</h1>
        <div class="user-info">
          <div class="user-avatar">
            <% if (user && user.profilePicture) { %>
              <img src="<%= user.profilePicture %>" alt="Avatar" class="rounded-circle"
width="40" height="40" style="object-fit: cover;">
            <% } else { %>
              <%= (user?.name || 'U').charAt(0).toUpperCase() %>
            <% } %>
          </div>
          <div>
            <div class="user-name"><%= user?.name || 'User' %></div>
            <div class="user-details">
              <%= user?.branch || 'Branch Not Set' %><% if(user?.year) { %> • Year <%=
user?.year %><% } %>
            </div>
          </div>
        </div>
      </div>

      <!-- Stats Overview -->
      <div class="stats-grid">
        <div class="stat-card">
          <div class="stat-number"><%= stats.totalApplications || 0 %></div>
          <div class="stat-label">Applications Sent</div>
        </div>
        <div class="stat-card">
```

```
            <div class="stat-number"><%= stats.shortlisted || 0 %></div>
            <div class="stat-label">Shortlisted</div>
          </div>
          <div class="stat-card">
            <div class="stat-number"><%= stats.pendingInterviews || 0 %></div>
            <div class="stat-label">Pending Interviews</div>
          </div>
          <div class="stat-card">
            <div class="stat-number"><%= stats.offers || 0 %></div>
            <div class="stat-label">Offers Received</div>
          </div>
        </div>

        <!-- Profile Completion -->
        <div class="section">
          <div class="section-header">
            <h3><i class="fas fa-tasks"></i> Profile Completion</h3>
            <a href="/student/profile/edit" class="btn btn-primary">Update Profile</a>
          </div>
          <div class="section-body">
            <div class="progress mb-3">
              <div class="progress-bar" role="progressbar" data-width="<%= profileCompletion || 0
%>" style="width: 0%">
                <%= (profileCompletion || 0) + '%' %>
              </div>
            </div>
            <div class="profile-checklist">
              <div class="checklist-item <%= user?.resume ? 'completed' : '' %>">
                <i class="fas <%= user?.resume ? 'fa-check-circle text-success' : 'fa-times-circle
text-danger' %>"></i>
                Upload Resume
              </div>
              <div class="checklist-item <%= user?.skills?.length ? 'completed' : '' %>">
```

```
                    <i class="fas <%= user?.skills?.length ? 'fa-check-circle text-success' : 'fa-times-circle text-danger' %>"></i>
                    Add Skills
                </div>
                <div class="checklist-item <%= user?.phone ? 'completed' : '' %>">
                    <i class="fas <%= user?.phone ? 'fa-check-circle text-success' : 'fa-times-circle text-danger' %>"></i>
                    Add Contact Information
                </div>
            </div>
        </div>
    </div>

    <!-- Latest Job Opportunities -->
    <div class="section">
        <div class="section-header">
            <h3><i class="fas fa-briefcase"></i> Latest Opportunities</h3>
            <a href="/student/jobs" class="btn btn-primary">View All Jobs</a>
        </div>
        <div class="section-body">
            <div class="jobs-grid">
                <% if (recentJobs && recentJobs.length > 0) { %>
                    <% recentJobs.forEach(job => { %>
                    <div class="job-card">
                        <div class="job-header">
                            <h4><%= job.title || 'Untitled Position' %></h4>
                            <span class="badge bg-<%= job.type === 'Internship' ? 'info' : 'success' %>">
                                <%= job.type || 'Full Time' %>
                            </span>
                        </div>
                        <div class="company-info">
                            <i class="fas fa-building"></i> <%= job.companyName || 'Company Name Not Available' %>
```

```
                    </div>
                    <div class="job-details">
                        <div><i class="fas fa-map-marker-alt"></i> <%= job.location || 'Location Not Specified' %></div>
                        <div><i class="fas fa-rupee-sign"></i> <%= job.salary || 'Salary Not Disclosed' %></div>
                    </div>
                    <div class="deadline">
                        Last Date: <%= job.deadline ? new Date(job.deadline).toLocaleDateString() : 'Deadline Not Set' %>
                    </div>
                    <div class="job-actions">
                        <a href="/student/jobs/<%= job._id || '#' %>" class="btn btn-primary">View Details</a>
                        <button class="btn btn-success apply-job-btn" data-job-id="<%= job._id ? job._id.toString() : '' %>">
                            Apply Now
                        </button>
                    </div>
                </div>
                <% }); %>
            <% } else { %>
                <div class="text-center py-4">
                    <p class="text-muted">No jobs available at the moment.</p>
                    <a href="/student/jobs" class="btn btn-primary">Browse All Jobs</a>
                </div>
            <% } %>
        </div>
    </div>
</div>

<!-- Recent Applications -->
<div class="section">
    <div class="section-header">
```

```
            <h3><i class="fas fa-file-alt"></i> Recent Applications</h3>
            <a href="/student/applications" class="btn btn-primary">View All Applications</a>
        </div>
        <div class="section-body">
            <div class="table-responsive">
                <table class="table">
                    <thead>
                        <tr>
                            <th>Company</th>
                            <th>Position</th>
                            <th>Applied On</th>
                            <th>Status</th>
                            <th>Action</th>
                        </tr>
                    </thead>
                    <tbody>
                        <% (recentApplications || []).forEach(app => { %>
                        <tr>
                            <td><%= app.job?.companyName || 'Company Not Available' %></td>
                            <td><%= app.job?.title || 'Position Not Available' %></td>
                            <td><%= app.appliedDate ? new Date(app.appliedDate).toLocaleDateString() :
'Date Not Available' %></td>
                            <td>
                                <span class="badge bg-<%=
                                    (app.status === 'Pending' || app.status === 'pending') ? 'warning' :
                                    ((app.status === 'Shortlisted' || app.status === 'shortlisted') ? 'success' :
                                    ((app.status === 'Rejected' || app.status === 'rejected') ? 'danger' :
                                    ((app.status === 'Hired' || app.status === 'hired') ? 'success' : 'info'))) %>">
                                    <%= app.status || 'Pending' %>
                                </span>
                            </td>
                            <td>
```

```
                            <a href="/student/applications/<%= app._id || '#' %>" class="btn btn-sm btn-
info">
                                <i class="fas fa-eye"></i> View Details
                            </a>
                        </td>
                    </tr>
                    <% }); %>
                </tbody>
            </table>
        </div>
    </div>
</div>

<!-- Upcoming Interviews -->
<div class="section">
    <div class="section-header">
        <h3><i class="fas fa-calendar-check"></i> Upcoming Interviews</h3>
        <a href="/student/interviews" class="btn btn-primary">View All Interviews</a>
    </div>
    <div class="section-body">
        <% if (upcomingInterviews && upcomingInterviews.length > 0) { %>
            <div class="interview-timeline">
                <% upcomingInterviews.forEach(interview => { %>
                    <div class="interview-card">
                        <div class="interview-date">
                            <%= interview.scheduledAt ? new
Date(interview.scheduledAt).toLocaleDateString() : 'Date Not Set' %>
                            <%= interview.scheduledAt ? new
Date(interview.scheduledAt).toLocaleTimeString() : '' %>
                        </div>
                        <div class="interview-details">
                            <h5><%= interview.job?.companyName || 'Company Name Not Available'
%></h5>
                            <p><%= interview.job?.title || 'Position Not Specified' %></p>
```

```
                        <div class="interview-type">
                            <i class="fas <%= interview.type === 'online' ? 'fa-video' : 'fa-building' %>"></i>
                            <%= interview.type ? interview.type.charAt(0).toUpperCase() + interview.type.slice(1) : 'In Person' %> Interview
                        </div>
                    </div>
                    <div class="interview-actions">
                        <% if (interview.meetingLink) { %>
                            <a href="<%= interview.meetingLink %>" target="_blank" class="btn btn-primary">
                                Join Meeting
                            </a>
                        <% } %>
                        <a href="/student/interviews/<%= interview._id || '#' %>" class="btn btn-info">
                            View Details
                        </a>
                    </div>
                </div>
            <% }); %>
        </div>
    <% } else { %>
        <p>No upcoming interviews scheduled.</p>
    <% } %>
    </div>
  </div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
<script>
    // Set progress bar width from data attribute
    document.addEventListener('DOMContentLoaded', function() {
```

```
    const progressBar = document.querySelector('.progress-bar[data-width]');
    if (progressBar) {
        const width = progressBar.getAttribute('data-width');
        progressBar.style.width = width + '%';
    }

    // Handle apply job buttons
    document.querySelectorAll('.apply-job-btn').forEach(btn => {
        btn.addEventListener('click', function() {
            const jobId = this.getAttribute('data-job-id');
            if (jobId) {
                applyForJob(jobId);
            }
        });
    });
});

function applyForJob(jobId) {
    if (!jobId) {
        alert('Invalid job ID');
        return;
    }

    if (confirm('Are you sure you want to apply for this position?')) {
        fetch(`/student/jobs/${jobId}/apply`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            }
        }).then(response => {
            if (response.ok) {
                alert('Application submitted successfully!');
```

```
                window.location.reload();
            } else {
                return response.json().then(data => {
                    alert(data.error || 'Failed to submit application. Please try again.');
                }).catch(() => {
                    alert('Failed to submit application. Please try again.');
                });
            }
        }).catch(err => {
            console.error('Error:', err);
            alert('An error occurred while submitting your application.');
        });
    }
}
</script>
</body>
</html>
```

Auth Routs

```
import express from "express";
import {
  showLogin,
  showRegister,
  registerUser,
  loginUser,
  logoutUser,
} from "../controllers/authController.js";

const router = express.Router();

// Auth routes
router.get("/login", showLogin);
router.get("/register", showRegister);
router.post("/register", registerUser);
router.post("/login", loginUser);
router.post("/logout", logoutUser);
// Allow GET logout for simple dashboard links (convenience)
router.get("/logout", logoutUser);

export default router;
```

Auth controller

```
import User from "../models/User.js";

export const showLogin = (req, res) => res.render("pages/login");
export const showRegister = (req, res) => res.render("pages/register");

export const registerUser = async (req, res) => {
  const { name, email, password, role } = req.body;
  try {
    // Basic validation
    if (!name || !email || !password || !role) {
      return res.render("pages/register", { error: "All fields are required" });
    }

    // Check if user already exists
    const existing = await User.findOne({ email });
    if (existing) {
      return res.render("pages/register", { error: "Email already registered" });
    }

    const userData = { name, email, password, role };
    // Role-specific fields
    if (role === 'student') {
      userData.branch = req.body.branch || '';
      userData.year = req.body.year || '';
    }
    if (role === 'company') {
      userData.companyName = req.body.companyName || name;
      userData.industry = req.body.industry || '';
      userData.website = req.body.website || '';
    }
```

```
    const user = new User(userData);
    await user.save();
    res.redirect("/auth/login");
  } catch (err) {
    console.error(err);
    res.render("pages/register", { error: "Registration failed" });
  }
};

export const loginUser = async (req, res) => {
  const { email, password, role } = req.body;
  try {
    const user = await User.findOne({ email, role });
    if (!user || !(await user.comparePassword(password))) {
      return res.render("pages/login", { error: "Invalid credentials" });
    }
    req.session.user = user.toObject();

    // Redirect based on role
    switch (user.role) {
      case 'admin':
        res.redirect("/admin/dashboard");
        break;
      case 'company':
        res.redirect("/company/dashboard");
        break;
      case 'student':
        res.redirect("/student/dashboard");
        break;
      default:
```

```
      res.redirect("/");
    }
  } catch (err) {
    console.error(err);
    res.render("pages/login", { error: "Login failed. Please try again." });
  }
};

export const logoutUser = (req, res) => {
  req.session.destroy(() => {
    res.redirect("/");
  });
};
```

Config Database

```
import mongoose from "mongoose";

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log("✅ MongoDB connected");
  } catch (err) {
    console.error("❌ MongoDB connection failed:", err);
    process.exit(1);
  }
};

export default connectDB;
```

7.4 Testing Summary

Unit Testing: Verified the functionality of each module individually.

Integration Testing: Ensured smooth interaction among system modules.

User Acceptance Testing: Confirmed that the system meets user expectations.

All tests were successfully executed, and the system met the defined performance and usability criteria.