

HASOFT

Job Connect



Group MEMBERS

Hamza Naveed (22i-0961) Saba Riaz (21i-0624) Abdul Momin(22I1326)

Contents

Abstract …………………………………………………………………………………………………………………5

Functional Nonfunctional requirements………………………………………………………………………….8

GOF and GRASP………………………………………………………………………………………………15,16

[Use Case: Manage User Accounts 19](#_Toc183557166)

[ Stakeholders and Interests: 19](#_Toc183557167)

[ Preconditions: 19](#_Toc183557168)

[ Postconditions: 19](#_Toc183557169)

[Main Success Scenario (Table Format) 19](#_Toc183557170)

[Use Case: Login 21](#_Toc183557171)

[ Stakeholders and Interests: 21](#_Toc183557172)

[ Preconditions: 21](#_Toc183557173)

[ Postconditions: 21](#_Toc183557174)

[Main Success Scenario (Table Format) 21](#_Toc183557175)

[Use Case: MCQ Test for Career Identification 23](#_Toc183557176)

[ Stakeholders and Interests: 23](#_Toc183557177)

[ Preconditions: 23](#_Toc183557178)

[ Postconditions: 23](#_Toc183557179)

[Main Success Scenario (Table Format) 23](#_Toc183557180)

[Use Case: View Test Results 25](#_Toc183557181)

[ Stakeholders and Interests: 25](#_Toc183557182)

[ Preconditions: 25](#_Toc183557183)

[ Postconditions: 25](#_Toc183557184)

[Main Success Scenario (Table Format) 25](#_Toc183557185)

[Use Case: Course Registration Based on Test Results 27](#_Toc183557186)

[ Stakeholders and Interests: 27](#_Toc183557187)

[ Preconditions: 27](#_Toc183557188)

[ Postconditions: 27](#_Toc183557189)

[Main Success Scenario (Table Format) 27](#_Toc183557190)

[Level: 29](#_Toc183557191)

[Scope: 29](#_Toc183557192)

[Primary Actor: 29](#_Toc183557193)

[Stakeholders and Interests: 29](#_Toc183557194)

[Preconditions: 29](#_Toc183557195)

[Postconditions: 29](#_Toc183557196)

[Main Success Scenario: 29](#_Toc183557197)

[ 2a. Invalid Location Input: 30](#_Toc183557198)

[ 3b. Network Failure During Search: 30](#_Toc183557199)

[ 4a. Job Listing Expired or Closed: 30](#_Toc183557200)

[ 5a. Job Details Page Error: 30](#_Toc183557201)

[Use Case: Personalized Job Recommendation 31](#_Toc183557202)

[Use Case Name: 31](#_Toc183557203)

[Level: 31](#_Toc183557204)

[Scope: 31](#_Toc183557205)

[Primary Actor: 31](#_Toc183557206)

[Stakeholders and Interests: 31](#_Toc183557207)

[Preconditions: 31](#_Toc183557208)

[Postconditions: 31](#_Toc183557209)

[Main Success Scenario: 31](#_Toc183557210)

[ 2a. Insufficient Profile Data: 32](#_Toc183557211)

[ 3a. Recommendations Not Relevant: 32](#_Toc183557212)

[ 4a. Job Recommendation Expired: 32](#_Toc183557213)

[ 5a. Unable to Save Recommended Job: 32](#_Toc183557214)

[Use Case: Apply for a Job 33](#_Toc183557215)

[Use Case Name: 33](#_Toc183557216)

[Level: 33](#_Toc183557217)

[Scope: 33](#_Toc183557218)

[Primary Actor: 33](#_Toc183557219)

[Stakeholders and Interests: 33](#_Toc183557220)

[Preconditions: 33](#_Toc183557221)

[Postconditions: 33](#_Toc183557222)

[Main Success Scenario: 33](#_Toc183557223)

[ 3a. Resume Upload Fails: 34](#_Toc183557224)

[ 4a. Application Submission Error: 34](#_Toc183557225)

[ 5a. Employer Declines Application: 34](#_Toc183557226)

[ 5b. Job Seeker Withdraws Application: 34](#_Toc183557227)

[Use Case: Create a Job Post 35](#_Toc183557228)

[Use Case Name: 35](#_Toc183557229)

[Level: 35](#_Toc183557230)

[Scope: 35](#_Toc183557231)

[Primary Actor: 35](#_Toc183557232)

[Stakeholders and Interests: 35](#_Toc183557233)

[Preconditions: 35](#_Toc183557234)

[Postconditions: 35](#_Toc183557235)

[Main Success Scenario: 35](#_Toc183557236)

[ 2a. Incomplete Job Post Information: 36](#_Toc183557237)

[ 3a. Duplicate Job Post: 36](#_Toc183557238)

[ 4a. Publishing Delay: 36](#_Toc183557239)

[ 5a. Job Post Rejected Due to Policy Violations: 36](#_Toc183557240)

[Use Case: View Candidate Information 36](#_Toc183557241)

[Use Case Name: 36](#_Toc183557242)

[Level: 37](#_Toc183557243)

[Scope: 37](#_Toc183557244)

[Primary Actor: 37](#_Toc183557245)

[Stakeholders and Interests: 37](#_Toc183557246)

[Preconditions: 37](#_Toc183557247)

[Postconditions: 37](#_Toc183557248)

[Main Success Scenario: 37](#_Toc183557249)

[ 1a. Candidate Withdraws Application: 37](#_Toc183557250)

[ 2a. Candidate Profile Incomplete: 38](#_Toc183557251)

[ 3a. Error in Document Download: 38](#_Toc183557252)

[ 4a. Shortlisting Technical Error: 38](#_Toc183557253)

[Hamza Naveed Fully Dressed Use cases 38](#_Toc183557254)

[Use Case: Send Interview Information 38](#_Toc183557255)

[ Stakeholders and Interests: 38](#_Toc183557256)

[Main Success Scenario: 39](#_Toc183557257)

[ 4a. Invalid interview date: 39](#_Toc183557258)

[ 6a. Candidate email not available: 39](#_Toc183557259)

[Use Case: Course Enrollment 40](#_Toc183557260)

[ Stakeholders and Interests: 40](#_Toc183557261)

[Main Success Scenario: 40](#_Toc183557262)

[ 6a. Prerequisites not met: 40](#_Toc183557263)

[ 8a. Email confirmation failure: 40](#_Toc183557264)

[Use Case: Feedback on Course or Job Experience 41](#_Toc183557265)

[ Stakeholders and Interests: 41](#_Toc183557266)

[Main Success Scenario: 41](#_Toc183557267)

[ 5a. Incomplete form: 41](#_Toc183557268)

[Use Case: Skill Assessment after Course Completion 42](#_Toc183557269)

[ Stakeholders and Interests: 42](#_Toc183557270)

[Main Success Scenario: 42](#_Toc183557271)

[ 6a. Network error during submission: 42](#_Toc183557272)

[Use Case: User Support 43](#_Toc183557273)

[ Stakeholders and Interests: 43](#_Toc183557274)

[Main Success Scenario: 43](#_Toc183557275)

[ 4a. Issue not resolved automatically: 43](#_Toc183557276)

**Abstract:**

The JobConnect application is a comprehensive platform for job search, management, and administration. It bridges the gap between job seekers and recruiters, offering a dynamic interface for both administrators and users. The system provides features such as job postings, user management, advanced filtering options, and notifications. With a clean design and modular architecture, the platform ensures seamless user interaction and efficient data handling.

**Introduction:**

The JobConnect app is a job portal system that allows students, guests, and administrators to interact with job-related functionalities. The platform focuses on connecting job seekers to job opportunities while providing administrative control for managing users and job postings. Its key components include job filtering, application management, and analytics, all designed to ensure ease of use and efficiency. Built using modern technologies, the app prioritizes modularity, maintainability, and scalability.

**Problem Statement and Description:**

In today’s competitive job market, job seekers and employers face challenges in

finding the right fit. Job seekers often struggle to discover opportunities that match

their qualifications and skills, while employers spend considerable time filtering

through unsuitable candidates. Current job portals provide broad listings without

personalized guidance, leading to inefficiencies in the job search and hiring

processes. This results in job seekers being overwhelmed with irrelevant

opportunities and employers unable to find qualified candidates quickly.

Job Connect addresses this problem by offering a platform where users take a

tailored MCQ test to determine their field of expertise. Based on the test results, the

app recommends relevant educational courses to enhance the user's skills and align

them with appropriate job opportunities. This approach simplifies the job search

process by helping users focus on the right career path and upskill effectively. The

project's feasibility is high, relying on structured tests and course

recommendations, ensuring a practical solution.

**System Features:**

1. **User Registration and Login**: Secure authentication for users.
2. **Dynamic Dashboard**: Tailored for different user roles (Admin, Student, Guest).
3. **Job Listings and Filters**: Advanced search and filtering for job postings.
4. **Job Applications**: Allows users to apply for jobs and view their application history.
5. **Notifications**: Real-time updates about jobs, applications, and system changes.
6. **Scene Management**: Smooth navigation between application pages.
7. **Performance Analytics**: Admin-focused data visualizations for system usage.

**Administrator Features:**

1. **Admin Dashboard**: Centralized access to manage jobs, users, and analytics.
2. **Job Posting Management**: Add, edit, and delete job postings.
3. **User Management**: Manage registered users, edit profiles, and delete inactive users.
4. **Analytics and System Logs**: Access detailed analytics and system logs for monitoring performance.
5. **Job Application Monitoring**: Track and manage job applications submitted by users.

**Student Features**

1. **Job Search**: Browse and filter through job postings.
2. **Job Applications**: Apply for jobs and view application statuses.
3. **Profile Management**: Update and manage user profiles.
4. **Career Tests**: Take assessments to determine career interests.
5. **Notifications**: Receive updates on job statuses, deadlines, and system messages.

**Guest Features:**

1. **Job Listings (Limited)**: Browse publicly available job postings without an account.
2. **Registration**: Sign up to access the full features of the platform.
3. **Basic Filters**: Search for jobs with limited filtering options.
4. **General Support**: Access information about the platform and contact support.

**Technologies Used:**

1. **JavaFX**: For building the graphical user interface (GUI).
2. **Java**: Backend logic and application framework.
3. **SceneBuilder**: For designing JavaFX scenes.
4. **CSV**: For data persistence and storage.
5. **Design Patterns**: Utilized Factory, Singleton, Adapter, Observer, and Strategy patterns.
6. **Version Control**: GitHub for collaboration and version management.

**Project Structure:**

1. **application Package**:
   * Dashboard.java: Main user dashboard for managing features.
   * SceneManager.java: Handles scene navigation across the application.
   * JobConnectApp.java: Main entry point for the application.
   * ManageJobApplications.java: Logic for managing job applications.
   * Other components like JobSelectionPage.java and JobManagementPage.java handle specific UI elements.
2. **application. admin Package**:
   * AdminDashboard.java: Centralized dashboard for admin operations.
   * ManageUsersPage.java and ManageJobsPage.java: Specific admin tools for managing users and jobs.
   * AnalyticsPage.java: Visual representation of system analytics for admins.
3. **application.extras Package**:
   * AdvancedFiltersPage.java: Provides dynamic filtering options for job listings.
   * CareerTestPage.java: Enables career assessments for students.
4. **application.jobs Package**:
   * Job.java: Core job model containing job details and data.
   * JobListingsPage.java: Displays job postings with sorting and filtering.
   * PostJobPage.java: Admin interface for posting new jobs.
5. **application.shared Package**:
   * NotificationsPage.java: Manages user notifications.
   * UserSupportPage.java: Provides support options for users.
6. **application.user Package**:
   * LoginPage.java: User login interface.
   * RegisterPage.java: User registration form.
   * ProfilePage.java: Enables users to manage their profiles.
7. **Helper Files**:
   * package-info.java: Documentation for each package.
   * Configurations for consistent styling and file structure.

**Design Patterns:**

**Grasp*:***

Below is the analysis of your **Job Connect App** classes categorized and evaluated according to the requested design principles and patterns.

**1. Classes Acting as Controllers**

**Controllers** are responsible for managing user interactions, delegating tasks, and coordinating between views and models.

* **Controllers in Your App:**
  + **Main.java**: The main entry point of the application. It initializes the application and handles the scene transitions through SceneManager.
  + **SceneManager.java**: Manages scene transitions and acts as a central point for scene-related operations.
  + **Dashboard.java**: Coordinates between different pages and views based on user roles (Job Seeker, Job Poster, or Admin).
  + **ManageJobApplications.java**: Acts as a controller for managing job applications.
  + **Admin Dashboard.java**: Controls the admin-related views and their navigation.

**2. Classes Acting as Creators**

**Creators** are responsible for creating objects of other classes, adhering to the "Creator" GRASP principle.

* **Creators in Your App:**
  + **Main.java**: Creates and initializes core components, including SceneManager and primary views.
  + **SceneManager.java**: Dynamically creates and manages scenes for reuse across the application.
  + **JobConnectApp.java**: (If this is a factory/initializer class, it likely creates application-level resources or configurations.)

**3. Classes Acting as Information Experts**

**Information Experts** are classes that have the most relevant data or knowledge for fulfilling a task.

* **Information Experts in Your App:**
  + **Job.java**: Represents the domain model for a job, containing its title, description, and related metadata.
  + **JobApplicationPage.java**: Likely holds detailed information about specific job applications.
  + **ProfilePage.java**: Contains and displays user profile information.
  + **NotificationsPage.java**: Manages and displays user notifications.
  + **EditUserPage.java**: Manages user-specific data for editing.
  + **AnalyticsPage.java**: Processes and displays analytical data for the admin.
  + **SystemLogsPage.java**: Provides logging and system-related metadata for admins.

**4. Classes with Low Coupling**

**Low Coupling** indicates that a class has minimal dependencies on other classes, increasing reusability and maintainability.

* **Classes with Low Coupling:**
  + **Job.java**: Represents a standalone domain entity with minimal dependencies.
  + **CareerTestPage.java**: Likely operates independently to provide career testing functionality.
  + **PostJobPage.java**: Focused on job creation with limited reliance on other components.
  + **Advanced FiltersPage.java**: Provides filters for job searches with limited interaction with other modules.

**5. Classes with High Cohesion**

**High Cohesion** means a class focuses on a single, well-defined responsibility.

* **Classes with High Cohesion:**
  + **MCQTestPage.java**: Manages the functionality of MCQ tests for job applications.
  + **JobDetailsPage.java**: Focuses on displaying job-specific details.
  + **RegisterPage.java**: Dedicated to user registration tasks.
  + **EditJobPage.java**: Focused on editing job-related information.

**6. Pure Fabricated Classes**

**Pure Fabrication** classes do not belong naturally to the problem domain but are created to achieve other design goals (e.g., reusability, separation of concerns).

* **Pure Fabricated Classes in App:**
  + **SceneManager.java**: Abstracts and centralizes scene transitions for the application.
  + **ManageJobsPage.java**: Likely provides utility functionality for managing jobs that doesn't belong directly to the domain.
  + **SystemLogsPage.java**: Provides fabricated functionality to manage logs and is not part of the core domain.

**Summary :**

| **Class Name** | **Role** | **Low Coupling** | **High Cohesion** | **Pure Fabrication** |
| --- | --- | --- | --- | --- |
| Main.java | Controller, Creator | ❌ | ✅ | ❌ |
| SceneManager.java | Controller, Creator | ✅ | ✅ | ✅ |
| Dashboard.java | Controller | ❌ | ✅ | ❌ |
| ManageJobApplications.java | Controller | ❌ | ✅ | ❌ |
| Job.java | Information Expert | ✅ | ✅ | ❌ |
| JobListingsPage.java | View, Information Expert | ❌ | ✅ | ❌ |
| MCQTestPage.java | High Cohesion | ✅ | ✅ | ❌ |
| PostJobPage.java | Low Coupling | ✅ | ✅ | ❌ |
| SystemLogsPage.java | Fabrication | ✅ | ✅ | ✅ |
| NotificationsPage.java | Information Expert | ✅ | ✅ | ❌ |
| RegisterPage.java | High Cohesion | ✅ | ✅ | ❌ |

**GOF:**Creational Patterns:

1. Factory Pattern

* Relevant Classes:
  + SceneManager.java: Acts as a factory for creating and managing JavaFX scenes dynamically. It provides methods to return instances of different pages based on application navigation, making it a factory.
  + JobConnectApp.java: Could utilize a factory pattern to initialize different parts of the application, such as creating the main stage and various UI elements.

2. Singleton Pattern

* Relevant Classes:
  + SceneManager.java: Likely implements the Singleton pattern to ensure that there is only one instance of the SceneManager managing scenes across the application.
  + JobConnectApp.java: The main application class could follow the Singleton pattern to ensure one entry point into the application lifecycle.

Structural Patterns:

3. Adapter Pattern

* Relevant Classes:
  + JobManagementPage.java: Could act as an adapter if it bridges between the application's job data and the UI components for job management.
  + ManageJobApplications.java: May adapt application-related data into a format that can be displayed or interacted with on the UI.

4. Facade Pattern

* Relevant Classes:
  + Dashboard.java: Serves as a facade by simplifying access to various application components (like job management, user profile, etc.) from one centralized interface.
  + AdminDashboard.java: Likely acts as a facade for admin-specific operations, providing a simplified interface for managing users, jobs, and analytics.

Behavioral Patterns

5. Observer Pattern

* Relevant Classes:
  + NotificationsPage.java: Implements an observer to listen for notifications or updates (e.g., new job posts or application status changes) and updates the UI accordingly.
  + UserSupportPage.java: Could also act as an observer for events related to user feedback or support requests.

6. Strategy Pattern

* Relevant Classes:
  + AdvancedFiltersPage.java: Likely implements the Strategy pattern to allow dynamic application of different job filtering strategies (e.g., filtering by salary, location, or job type).
  + CareerTestPage.java: May use the Strategy pattern to implement different testing or evaluation algorithms for users.
  + MCQTestPage.java: Could also implement the Strategy pattern to switch between different types of questions or scoring mechanisms.

Detailed Class-Pattern Analysis

application Package

* Dashboard.java: Follows the Facade Pattern, simplifying access to other components like job management, user profiles, and admin functionalities.
* JobConnectApp.java: Uses the Singleton Pattern to ensure a single instance of the main application lifecycle.
* SceneManager.java: Implements Factory and Singleton Patterns by managing scenes and ensuring one instance exists globally.

application.admin Package

* AdminDashboard.java: Follows the Facade Pattern by unifying admin features like user management, analytics, and job editing.
* ManageJobsPage.java: Likely an Adapter Pattern to interact with the job data model and provide it to the admin UI.
* SystemLogsPage.java: Could implement Observer Pattern if it tracks and updates logs in real time.

application.extras Package

* AdvancedFiltersPage.java: Implements the Strategy Pattern for dynamic filtering options.
* CareerTestPage.java: Uses the Strategy Pattern to apply different career evaluation algorithms.

application.jobs Package

* JobListingsPage.java: May act as an Adapter Pattern to adapt the job list data for display in the UI.
* Job.java: As a data model, it doesn’t directly implement patterns but may interact with others like Factory or Adapter.

application.shared Package

* NotificationsPage.java: Implements the Observer Pattern to dynamically update notifications.
* UserSupportPage.java: May follow the Observer Pattern to track user requests and update the admin interface.

application.user Package

* LoginPage.java: May not directly follow a pattern but interacts with SceneManager for navigation.
* ProfilePage.java: May interact with the Facade (Dashboard) for accessing user-related data.
* RegisterPage.java: Similar to LoginPage, it interacts with the SceneManager.

Summary Table

| Class | Design Pattern |
| --- | --- |
| SceneManager | Factory, Singleton |
| Dashboard | Facade |
| AdminDashboard | Facade |
| NotificationsPage | Observer |
| AdvancedFiltersPage | Strategy |
| CareerTestPage | Strategy |
| JobListingsPage | Adapter |
| ManageJobsPage | Adapter |
| SystemLogsPage | Observer |
| JobConnectApp | Singleton |

**Functional Requirements for Job Connect App**

**User Registration and Login:**

Users (students, guests, administrators) should be able to register and login to the application.

User credentials should be stored securely using encryption (e.g., SHA-256 for password hashing).

Users should be able to retrieve forgotten passwords via email.

**User Profile Management:**

Users should be able to view and edit their profile, including their personal details, resume, skills, etc.

Users should be able to upload their resumes in PDF or Word formats.

**Job Management (Admin Features):**

Administrators should be able to manage job listings (create, edit, delete).

Administrators should be able to manage user accounts (approve, delete, edit roles).

Administrators should be able to view and analyze job application statistics.

Admin should have access to system logs and reports.

**Job Listings:**

Users (students and guests) should be able to view the list of job openings.

Users should be able to filter job listings based on criteria such as location, job type, salary range, etc.

Users should be able to search for jobs by keywords.

**Job Applications:**

Registered users should be able to apply for jobs through the app.

Job applications should allow users to submit a cover letter, resume, and other relevant documents.

Users should be able to track the status of their applications (pending, shortlisted, rejected).

**Notifications:**

Users should receive notifications for important events such as new job postings, application status updates, etc.

Notifications should be displayed in the app and sent to users via email if required.

**Job Test Integration:**

Users should be able to take career-related tests (e.g., MCQs) through the app.

Results of the tests should be recorded and made available to the users and administrators.

**Job Filters:**

Users should be able to apply filters to job listings, such as job type, location, salary, and company.

Advanced filters should be available to refine job search results.

**Admin Dashboard:**

The admin should have access to a dashboard to view overall system statistics, such as the number of users, job applications, and active job listings.

**User Support:**

Users should have access to a user support page where they can ask questions or contact support.

**System Logs:**

The system should maintain logs of user activities, job applications, and administrative actions.

**Non-Functional Requirements for Job Connect App**

**Performance:**

The application should be responsive, with loading times not exceeding 3 seconds for most pages.

Job search and application submission should be optimized for speed and efficiency, with results returned within a few seconds.

**Security:**

User data should be encrypted using secure algorithms such as AES-128.

The app should ensure secure user authentication using technologies like OAuth or JWT for session management.

Passwords should be stored securely (using a one-way hashing algorithm like SHA-256 or bcrypt).

Secure communication (SSL/TLS) should be enforced for data transmission over the network.

Regular security testing and vulnerability assessments should be conducted.

**Scalability:**

The app should be scalable to handle an increasing number of users and job listings without performance degradation.

The app should support a growing database of users, jobs, applications, and logs.

The system should be able to handle traffic spikes, especially during job application deadlines.

**Availability and Reliability:**

The app should have a high availability rate (99.9% uptime) and should recover from failures without data loss.

Critical systems like user authentication and job applications should have failover mechanisms in place.

**Scheduled maintenance and downtime should be communicated to users in advance.**

**Usability:**

The user interface should be intuitive, with easy navigation between pages.

Users should be able to complete their primary tasks (e.g., applying for jobs, viewing job listings) without requiring assistance.

The application should provide an easy-to-use search interface and filtering system.

**Compatibility:**

The app should be compatible across a wide range of devices (desktop, tablet, mobile) and operating systems (Windows, macOS, Linux, iOS, Android).

The app should support modern web browsers like Chrome, Firefox, Safari, and Edge.

**Maintainability:**

The codebase should be modular and well-documented to facilitate easy updates and maintenance.

Developers should be able to fix bugs and deploy updates without disrupting the service for users.

**Backup and Recovery:**

The system should regularly back up critical data, such as user accounts, job listings, and applications.

The app should have a disaster recovery plan to restore services in the event of a system failure or data loss.

**Legal and Compliance:**

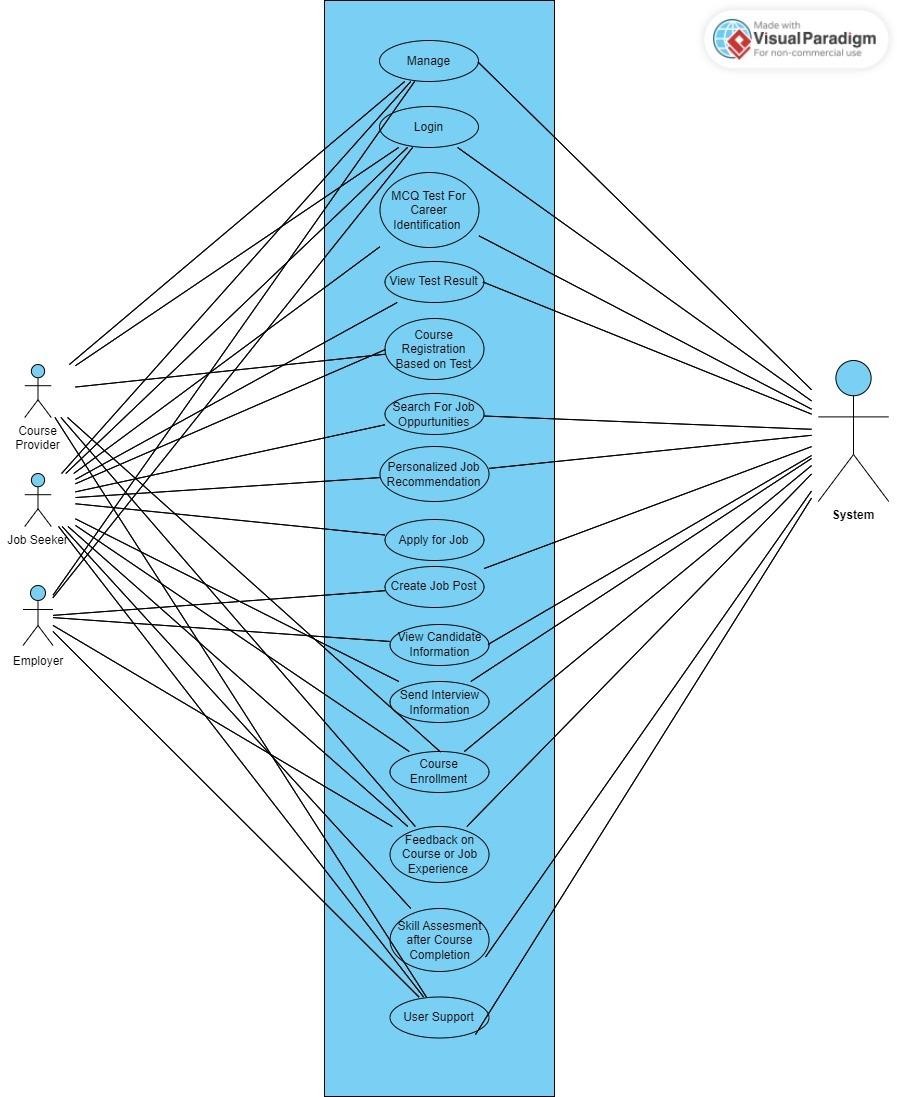
The app should comply with data protection laws such as GDPR (General Data Protection Regulation) if operating in the EU or similar data privacy regulations in other regions.

The app should have clear terms of service and privacy policies in place.

**Interoperability:**

The app should be able to integrate with third-party services for functionalities like payment gateways, email notifications, etc.

The system should support easy integration with other job portals or career platforms, if necessary.

**Use case diagram:**

# Abdul Momin Fully Dressed Use cases

## Use Case: Manage User Accounts

* **Level**: User-goal
* **Scope**: Job Connect Platform
* **Use Case Name**: Manage User Accounts
* **Primary Actor**: System

### Stakeholders and Interests:

* + **Job Seeker/Employer**: Wants their account details to be managed automatically and accurately.
  + **System**: Automatically manages user account creation, updates, and deletion based on specific triggers.

### Preconditions:

* + User accounts exist in the system.
  + The system has predefined rules for managing user accounts (e.g., account inactivity, and updates based on profile changes).

### Postconditions:

* + User account details are automatically updated, created, or deleted based on system logic.
  + System logs all changes for audit purposes.

### Main Success Scenario (Table Format)

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker or employer registers or updates details. | 2. The system automatically creates, updates, or deletes accounts based on predefined rules. |
| 3. User changes or updates their profile information. | 4. The system validates and applies changes to the user account. |

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
|  | 1. System removes or merges inactive/duplicate accounts according to predefined criteria. 2. Inactive or duplicate accounts are identified. 3. User account status is updated automatically. |
|  | 1. The system logs all account changes and ensures data consistency. 2. System sends confirmation notifications (if required). |
|  | 10. Send email notifications to users about major account changes. |

**Extensions**

* **3a.** User Attempts to Update Invalid Data:
  + **3a1.** The system rejects invalid changes and prompts the user to correct the information.
* **4a.** Duplicate Accounts Identified:
  + **4a1.** The system merges duplicate accounts and notifies the user about the consolidation.
* **5a.** System Detects Security Issues:
  + **5a1.** The system locks the account and notifies the user for security purposes.

## Use Case: Login

* **Level**: User-goal
* **Scope**: Job Connect Platform
* **Use Case Name**: Login
* **Primary Actor**: Job Seeker/Employer

### Stakeholders and Interests:

* + **Job Seeker/Employer**: Wants to access their account quickly, whether logging in or registering for the first time.
  + **System**: Ensures secure authentication and manages user registrations.

### Preconditions:

* + User has an account (for login) or is new and needs to register.

### Postconditions:

* + User is logged into their account or successfully registered and logged in.
  + System records the login or registration activity.

### Main Success Scenario (Table Format)

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. The user navigates to the login/registration page. | 2. Displays options for logging in or creating a new account. |
| 3. The user chooses to log in or register. | 4. If logging in, prompt for username and password; if registering, prompt for user details. |
| 5. The user enters login credentials or registration information. | 6. Validate the provided information against stored data. |
| 7. User submits the form. | 8. If logging in, grants access and redirects to the dashboard; if registering, create a new account and log in the user. |
|  | 9. Send a confirmation message and record the session activity. |

**Extensions**

* **2a.** User Enters Invalid Login Credentials:
  + **2a1.** The system notifies users of the error and prompts them to try again.
* **2b.** User Registration Information is Incomplete or Invalid:
  + **2b1.** The system prompts the user to complete or correct the information before proceeding.
* **4a.** Account Already Exists During Registration:
  + **4a1.** The system informs the user that the account already exists and prompts for login instead.
* **4b.** User Forgets Password:
  + **4b1.** The system offers a password recovery option.

## Use Case: MCQ Test for Career Identification

* **Level**: User-goal
* **Scope**: Job Connect Platform
* **Use Case Name**: MCQ Test for Career Identification
* **Primary Actor**: Job Seeker

### Stakeholders and Interests:

* + **Job Seeker**: Wants to take a test that identifies suitable career paths based on their skills and interests.
  + **System**: Provides a reliable test and evaluates the results to recommend courses and career paths.

### Preconditions:

* + Job seeker is logged into their account.
  + Test is available and has been designed with relevant questions.

### Postconditions:

* + Job seeker completes the test and receives immediate results.
  + System stores the results and suggests suitable career options.

### Main Success Scenario (Table Format)

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker navigates to the career identification test section. | 2. Displays information about the test and a "Start Test" button. |
| 3. Job seeker starts the MCQ test. | 4. Presents the first question along with answer options. |
| 5. Job seeker selects an answer for each question. | 6. Records the selected answer and presents the next question until all questions are answered. |
| 7. Job seeker submits the test after answering all questions. | 8. Evaluates the answers and calculates the results based on predefined scoring criteria. |
| 9. System displays the results along with recommended career paths. | 10. Suggests relevant courses and career options based on the test results. |

**Extensions**

* **2a.** Job Seeker Abandons the Test:
  + **2a1.** The system saves progress if the user exits before completion, allowing them to resume later.
* **3a.** Job Seeker Changes Mind About Answer:
  + **3a1.** The system allows the job seeker to change their answer before submission.
* **4a.** Job Seeker Submits the Test with Unanswered Questions:
  + **4a1.** The system prompts the job seeker to complete all questions before submitting.
* **5a.** No Suitable Career Options Found:
  + **5a1.** The system provides a message indicating that more information is needed or suggests retaking the test with updated interests.

## Use Case: View Test Results

* **Level**: User-goal
* **Scope**: Job Connect Platform
* **Use Case Name**: View Test Results
* **Primary Actor**: Job Seeker

### Stakeholders and Interests:

* + **Job Seeker**: Wants to review their test results to understand their strengths and potential career paths.
  + **Admin**: Ensures that the results are accurately stored and presented to users.

### Preconditions:

* + Job seeker has completed the career identification test.
  + Job seeker is logged into their account.

### Postconditions:

* + The test results are displayed to the job seeker, including suggested career paths and related courses.

### Main Success Scenario (Table Format)

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker navigates to the “Test Results” section. | 2. Displays the option to view completed test results. |
| 3. Job seeker selects to view their test results. | 4. Retrieves the stored test results from the database. |
| 5. Job seeker views their test results. | 6. Displays the results in a clear and organized format, including strengths and weaknesses. |
| 7. Job seeker reviews suggested career paths and courses. | 8. Provides detailed information about suggested careers and courses based on the results. |
| 9. Job seeker may choose to save or download the results. | 10. Offers options to save or download the test results as a file. |

**Extensions**

* **2a.** Test Results Not Available:
  + **2a1.** The system informs the job seeker that results are not yet available and suggests checking back later.
* **3a.** System Fails to Retrieve Results:
  + **3a1.** The system displays an error message and prompts the job seeker to try again later.
* **4a.** Job Seeker Has Questions About Results:
  + **4a1.** The system provides options for the job seeker to contact support for further clarification.

## Use Case: Course Registration Based on Test Results

* **Level**: User-goal
* **Scope**: Job Connect Platform
* **Use Case Name**: Course Registration Based on Test Results
* **Primary Actor**: Job Seeker

### Stakeholders and Interests:

* + **Job Seeker**: Wants to enroll in courses that will enhance their skills based on their test results.
  + **System**: Provides recommendations and manages the registration process seamlessly.
  + **Course Providers**: Interested in attracting qualified candidates to their courses.

### Preconditions:

* + Job seeker has completed the career identification test.
  + Job seeker has logged into their account.
  + The system has analyzed the test results and generated course recommendations.

### Postconditions:

* + Job seeker is successfully registered for the selected course.
  + The system updates the job seeker’s profile with the course registration details.

### Main Success Scenario (Table Format)

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker navigates to the recommended courses section. | 2. Displays a list of courses recommended based on test results. |
| 3. Job seeker selects a course to view details. | 4. Displays detailed information about the selected course, including duration, content, and fees. |
| 5. Job seeker confirms their choice to register for the course. | 6. Validates course availability and checks the job seeker’s eligibility for registration. |

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 7. Job seeker submits registration information and payment (if applicable). | 8. Processes the registration and payment, confirming the job seeker’s enrollment. |
| 9. System confirms registration and updates user profile. | 10. Displays a confirmation message and updates the job seeker’s profile with course registration details. |

**Extensions**

* **2a.** Selected Course is No Longer Available:
  + **2a1.** The system informs the job seeker that the course is no longer available and suggests alternative courses.
* **3a.** Job Seeker is Ineligible for the Course:
  + **3a1.** The system informs the job seeker of the eligibility requirements and suggests courses that they can register for.
* **4a.** Payment Failure During Registration:
  + **4a1.** The system displays a payment error message and prompts the job seeker to retry or use a different payment method.
* **5a.** Registration Limit Reached for the Course:
  + **5a1.** The system notifies the job seeker that the course registration limit has been reached and recommends other available courses.

**Saba Riaz Fully Dressed Use cases Use Case: Search for Job Opportunities Use Case Name:**

Search for Job Opportunities

### Level:

User-goal

### Scope:

Job Connect Platform

### Primary Actor:

Job Seeker

### Stakeholders and Interests:

* **Job Seeker:** Wants to quickly find job opportunities that match their skills and location.
* **Employer:** Wants their job postings to be easily discoverable by qualified candidates.
* **System:** Facilitates accurate and efficient job searches based on criteria provided by job seekers.

### Preconditions:

* Job seeker is logged into their account.
* The system has a database of active job postings.

### Postconditions:

* The system provides job opportunities that match the search criteria.
* The system logs the job seeker's search activity for future personalization.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker navigates to the job search section. | 1. The system displays search fields (e.g., job title, location, industry). |
| 2. Job seeker enters search criteria (e.g., job title, location). | 2. The system validates the input and performs the search across the job database. |

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 3. Job seeker submits the search request. | 3. The system retrieves matching job postings based on the provided criteria. |
| 4. System displays matching job opportunities. | 4. The system sorts and filters job results and presents them to the job seeker. |
| 5. Job seeker clicks on a job posting for details. | 5. The system loads and displays detailed job information. |

**Extensions:**

### 2a. Invalid Location Input:

* + The job seeker enters a location that is not recognized.
  + The system prompts for corrections or suggests valid locations.

### 3b. Network Failure During Search:

* + The system encounters a network error during the search process.
  + The system displays an error message and suggests retrying later.

### 4a. Job Listing Expired or Closed:

* + Job seeker clicks on a job, but it has expired.
  + The system informs the job seeker and removes the expired listing.

### 5a. Job Details Page Error:

* + A technical issue prevents the job details page from loading.
  + The system notifies the job seeker and prompts them to try again.

## Use Case: Personalized Job Recommendation

### Use Case Name:

Personalized Job Recommendation

### Level:

User-goal

### Scope:

Job Connect Platform

### Primary Actor:

System

### Stakeholders and Interests:

* **Job Seeker:** Wants to receive relevant job recommendations based on their profile.
* **Employer:** Wants their job postings to be recommended to the right candidates.
* **System:** Delivers accurate job recommendations by analyzing the job seeker’s profile and search behavior.

### Preconditions:

* Job seeker has a completed profile with relevant skills and preferences.
* The system has access to job seeker activity and job listings.

### Postconditions:

* The system provides personalized job recommendations to the job seeker.
* The system logs recommendation activity to improve future suggestions.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker accesses the "Recommended Jobs" section. | 1. The system analyzes the job seeker’s profile, skills, and previous searches. |
| 2. System generates personalized job recommendations. | 2. The system retrieves relevant job postings based on analysis. |
| 3. Job seeker reviews recommended jobs. | 3. The system presents the recommended job opportunities. |

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 4. Job seeker clicks on a recommended job for more details. | 4. The system displays the detailed job posting. |
| 5. Job seeker decides whether to apply or save the job. | 5. The system offers options to apply for or save the job. |

**Extensions:**

### 2a. Insufficient Profile Data:

* + The job seeker’s profile lacks information.
  + The system prompts the user to update their profile for more accurate recommendations.

### 3a. Recommendations Not Relevant:

* + The job seeker finds the recommendations irrelevant.
  + The system allows the user to provide feedback to improve future recommendations.

### 4a. Job Recommendation Expired:

* + The job posting is no longer available.
  + The system notifies the job seeker and suggests alternative jobs.

### 5a. Unable to Save Recommended Job:

* + A technical error prevents saving the recommended job.
  + The system prompts the job seeker to retry or report the issue.

## Use Case: Apply for a Job

### Use Case Name:

Apply for Job

### Level:

User-goal

### Scope:

Job Connect Platform

### Primary Actor:

Job Seeker

### Stakeholders and Interests:

* **Job Seeker:** Wants to easily apply for jobs that match their qualifications.
* **Employer:** Wants to receive applications from qualified candidates.
* **System:** Facilitates the application process by ensuring the submission is complete and accurate.

### Preconditions:

* The job seeker is logged in and has identified a job to apply for.
* The job posting is active and accepting applications.

### Postconditions:

* The job seeker successfully applies for the job.
* The system logs the application and notifies the employer.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job seeker clicks "Apply" on a job posting. | 1. The system presents the application form or prompts to upload a resume. |
| 2. Job seeker fills in application details or uploads a resume. | 2. The system validates the form data or resume format. |
| 3. Job seeker submits the application. | 3. The system submits the application and records the job seeker's details. |

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 4. System confirms the application submission. | 4. The system sends confirmation to the job seeker and notifies the employer. |
| 5. Job seeker receives confirmation of successful application. | 5. The system logs the submission and updates the job seeker's application history. |

**Extensions:**

### 3a. Resume Upload Fails:

* + The resume upload fails due to format issues.
  + The system prompts the job seeker to retry with a valid format.

### 4a. Application Submission Error:

* + A system error occurs during submission.
  + The system provides options to retry or save the application for later.

### 5a. Employer Declines Application:

* + The employer declines the application.
  + The system notifies the job seeker and provides feedback if available.

### 5b. Job Seeker Withdraws Application:

* + The job seeker decides to withdraw the application.
  + The system updates the application status and notifies the employer.

## Use Case: Create a Job Post

### Use Case Name:

Create Job Post

### Level:

User-goal

### Scope:

Job Connect Platform

### Primary Actor:

Employer

### Stakeholders and Interests:

* **Employer:** Wants to create job posts easily to attract qualified candidates.
* **Job Seeker:** Wants access to detailed job postings.
* **System:** Ensures job posts are accurately created and available to job seekers.

### Preconditions:

* The employer is logged in and has permission to create job posts.

### Postconditions:

* The job post is successfully created and listed for job seekers.
* The system logs the post creation for tracking purposes.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Employer navigates to the job posting section. | 1. The system displays the job post creation form. |
| 2. Employer fills in the job post details (e.g., title, description, location). | 2. The system validates the input data. |
| 3. Employer submits the job post. | 3. The system creates the job post and adds it to the job listing database. |
| 4. System confirms job post creation. | 4. The system displays a confirmation message and provides options to edit or delete the post. |

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 5. Job seekers can now view the job post. | 5. The system makes the job post available to job seekers. |

**Extensions:**

### 2a. Incomplete Job Post Information:

* + The employer submits incomplete details.
  + The system highlights the missing fields and requests completion.

### 3a. Duplicate Job Post:

* + The system detects a similar job post.
  + The system alerts the employer and suggests editing the existing post.

### 4a. Publishing Delay:

* + A delay occurs in making the job post live.
  + The system informs the employer and provides an estimated time for the post to be visible.

### 5a. Job Post Rejected Due to Policy Violations:

* + The system identifies a violation in the job post content.
  + The system rejects the post and provides feedback to the employer.

## Use Case: View Candidate Information

### Use Case Name:

View Candidate Information

### Level:

User-goal

### Scope:

Job Connect Platform

### Primary Actor:

Employer

### Stakeholders and Interests:

* **Employer:** Wants to view candidate information to evaluate job applications.
* **Candidate:** Wants their profile to be presented accurately to potential employers.
* **System:** Ensures candidate information is displayed correctly and securely.

### Preconditions:

* Employer is logged in and has access to candidate applications.
* Candidate profiles are complete and available.

### Postconditions:

* The employer successfully views the candidate’s information.
* The system logs the employer's activity.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Employer navigates to the list of applicants. | 1. The system displays a list of candidates who applied for the job. |
| 2. Employer clicks on a candidate’s profile. | 2. The system retrieves and displays the candidate’s details (e.g., resume, contact information). |
| 3. Employer reviews the candidate’s profile. | 3. The system provides options to download the resume, contact the candidate, or shortlist the candidate. |
| 4. Employer shortlists the candidate for further evaluation. | 4. The system marks the candidate as shortlisted and updates the application status. |
| 5. Employer contacts the candidate if needed. | 5. The system logs the employer's actions and updates the candidate’s status. |

**Extensions:**

### 1a. Candidate Withdraws Application:

* + The candidate withdraws their application before the employer views their profile.
  + The system informs the employer that the candidate is no longer available.

### 2a. Candidate Profile Incomplete:

* + The candidate’s profile lacks essential information.
  + The system alerts the employer and requests additional details from the candidate.

### 3a. Error in Document Download:

* + The system fails to download the candidate’s resume.
  + The system notifies the employer and offers to retry or contact the candidate.

### 4a. Shortlisting Technical Error:

* + The system encounters an issue while marking the candidate as shortlisted.
  + The system displays an error and prompts the employer to retry.

# Hamza Naveed Fully Dressed Use cases

## Use Case: Send Interview Information

* **Level:** User goal
* **Scope:** Job Management System
* **Use Case Name:** Send Interview Information
* **Primary Actor:** Employer

### Stakeholders and Interests:

* + **Employer:** Wants to schedule interviews with selected candidates efficiently.
  + **Candidate:** Expects to receive timely interview information.
* **Precondition:** The employer has selected a candidate for an interview, and the candidate contact information is available.
* **Postcondition:** The interview is scheduled, and the candidate receives the interview information.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Employer selects a candidate for an interview. | 2. The system displays the candidate’s details for review. |
| 3. Employer chooses the interview date, time, and location. | 4. System validates the details and schedules the interview. |
| 5. Employer sends the interview invitation. | 6. System sends the interview notification to the candidate. |
| 7. Employer confirms interview scheduling. | 8. System updates the candidate's status and records the interview. |

**Extensions:**

### 4a. Invalid interview date:

* + The system prompts the employer to select a valid interview date and reschedule.

### 6a. Candidate email not available:

* + The system requests additional contact information for the candidate.

## Use Case: Course Enrollment

* **Level:** User goal
* **Scope:** Course Management System
* **Use Case Name:** Course Enrollment
* **Primary Actor:** Job Seeker / Student

### Stakeholders and Interests:

* + **Job Seeker / Student:** Wants to enroll in a course related to their career path.
  + **Course Provider:** Wants to ensure students are properly enrolled and meet prerequisites.
* **Precondition:** The student is logged in and has completed any necessary course prerequisites.
* **Postcondition:** The student is enrolled in the course and receives confirmation.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job Seeker / Student browses available courses. | 2. System displays the list of available courses. |
| 3. Job Seeker / Student selects a course for enrollment. | 4. System provides course details and enrollment form. |
| 5. Job Seeker / Student submits the enrollment request. | 6. System validates enrollment, checks prerequisites. |
| 7. Job Seeker / Student receives confirmation of enrollment. | 8. System updates course status and sends enrollment confirmation. |

**Extensions:**

### 6a. Prerequisites not met:

* + The system informs the student and suggests other courses or guides them to complete the prerequisites.

### 8a. Email confirmation failure:

* + The system retries sending the email or notifies the student to manually check the status.

## Use Case: Feedback on Course or Job Experience

* **Level:** Subfunction
* **Scope:** Course and Job Management System
* **Use Case Name:** Feedback on Course or Job Experience
* **Primary Actor:** Job Seeker / Student

### Stakeholders and Interests:

* + **Job Seeker / Student:** Wants to provide feedback to help improve future experiences.
  + **Course Provider / Employer:** Wants to gather feedback to improve services.
* **Precondition:** The course/job experience has been completed, and the feedback feature is accessible.
* **Postcondition:** The feedback is successfully submitted and recorded.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job Seeker / Student navigates to the feedback section. | 2. System displays the feedback form for the relevant course/job. |
| 3. Job Seeker / Student fills in feedback (rating, comments, etc.). | 4. System validates that all fields are complete. |
| 5. Job Seeker / Student submits the feedback. | 6. System stores feedback and updates the relevant record. |
| 7. Job Seeker / Student receives a confirmation of submission. | 8. System sends confirmation and stores the feedback. |

**Extensions:**

### 5a. Incomplete form:

* + The system prompts the user to fill in all required fields before submission.

## Use Case: Skill Assessment after Course Completion

* **Level:** Subfunction
* **Scope:** Course Management System
* **Use Case Name:** Skill Assessment after Course Completion
* **Primary Actor:** Job Seeker / Student

### Stakeholders and Interests:

* + **Job Seeker / Student:** Wants to verify their skills after completing a course.
  + **Course Provider:** Wants to assess the student's learning outcomes.
* **Precondition:** The student has completed the course, and the skill assessment is available.
* **Postcondition:** The student completes the assessment and receives their results.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job Seeker / Student selects the completed course. | 2. System displays available skill assessments for the course. |
| 3. Job Seeker / Student starts the assessment. | 4. System provides instructions and initiates the assessment. |
| 5. Job Seeker / Student answers assessment questions. | 6. System validates responses and gives instant feedback. |
| 7. Job Seeker / Student submits the assessment. | 8. System grades the assessment and provides the result. |

**Extensions:**

### 6a. Network error during submission:

* + The system attempts to reconnect and submit the assessment again.

## Use Case: User Support

* **Level:** Subfunction
* **Scope:** User Support System
* **Use Case Name:** User Support
* **Primary Actor:** Job Seeker / Employer

### Stakeholders and Interests:

* + **Job Seeker / Employer:** Wants quick and effective resolution of issues or queries.
  + **System Support Team:** Wants to handle and resolve user queries efficiently.
* **Precondition:** The user has an account and can access the support system.
* **Postcondition:** The issue is resolved, and the support ticket is closed.

### Main Success Scenario:

|  |  |
| --- | --- |
| **Actor Action (or Intention)** | **System Responsibility** |
| 1. Job Seeker or Employer encounters an issue or has a query. | 2. System provides access to a helpdesk, chatbot, or support ticketing system. |
| 3. Job Seeker or Employer submits a support request. | 4. System processes the request and either resolves it automatically or forwards it to a support team. |
| 5. Job Seeker or Employer receives resolution. | 6. System updates the user’s support history and closes the ticket. |

**Extensions:**

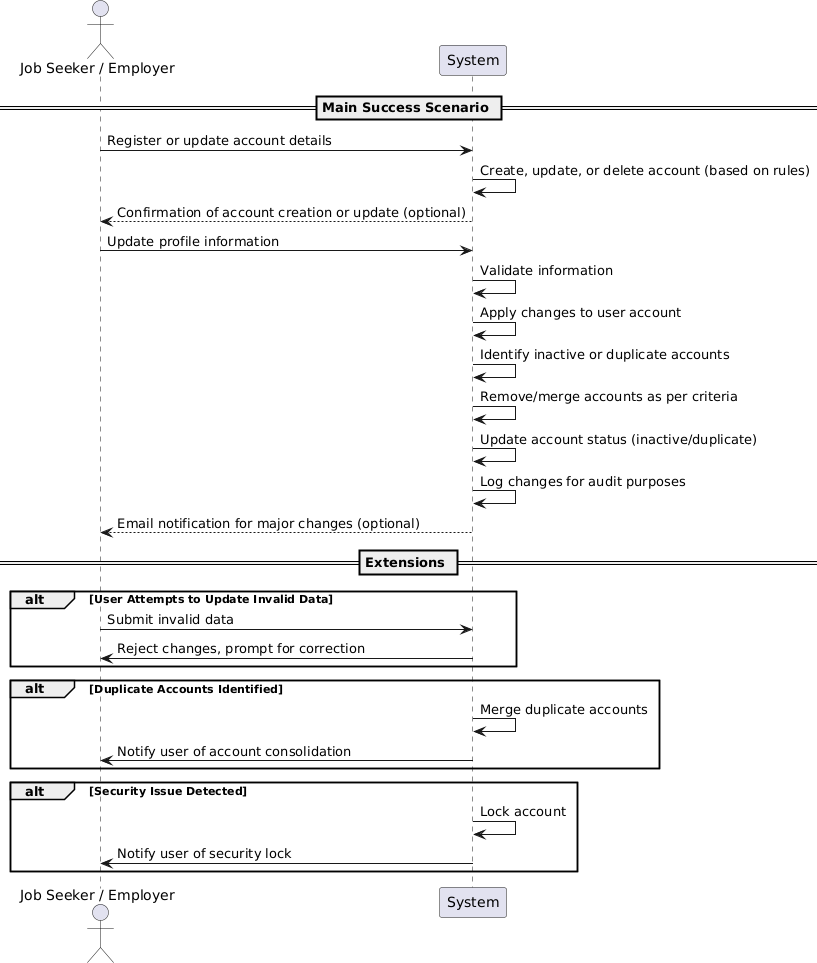
### 4a. Issue not resolved automatically:

* + The system escalates the issue to a live support team, and the user is notified of the delay.

**System Sequence Diagrams (SSDs):**

**Manage User**

**Account:**

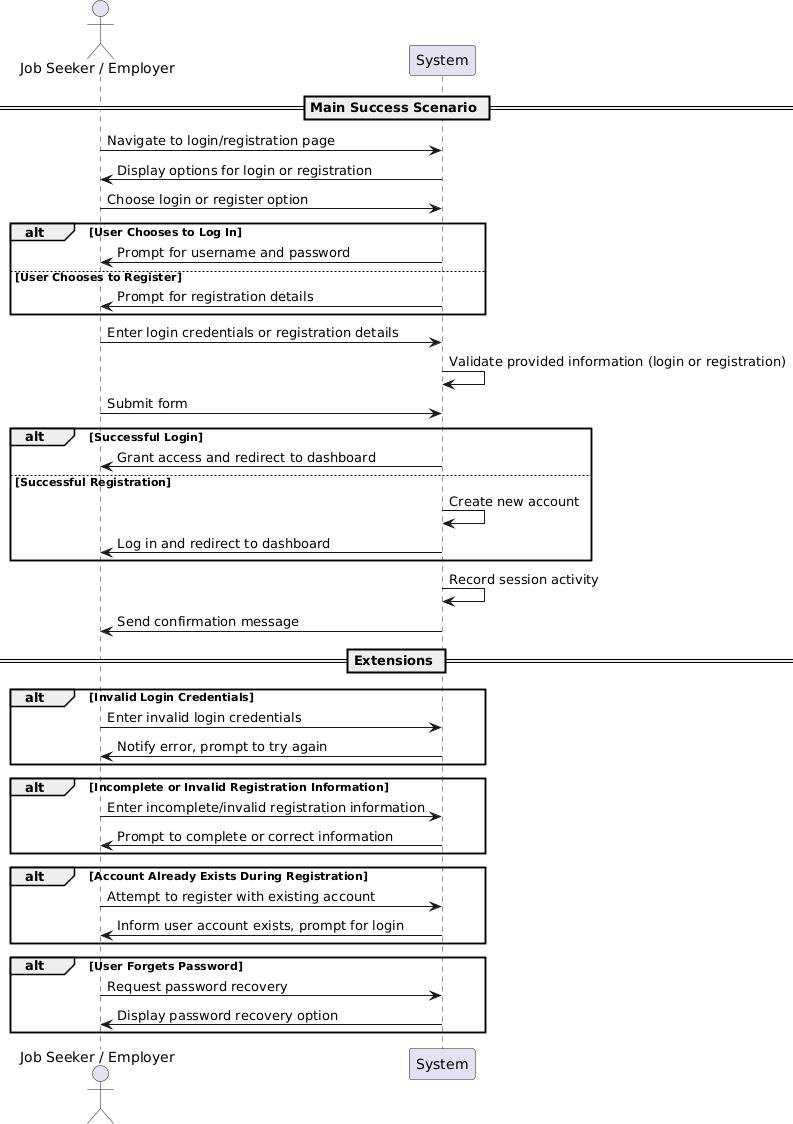
****

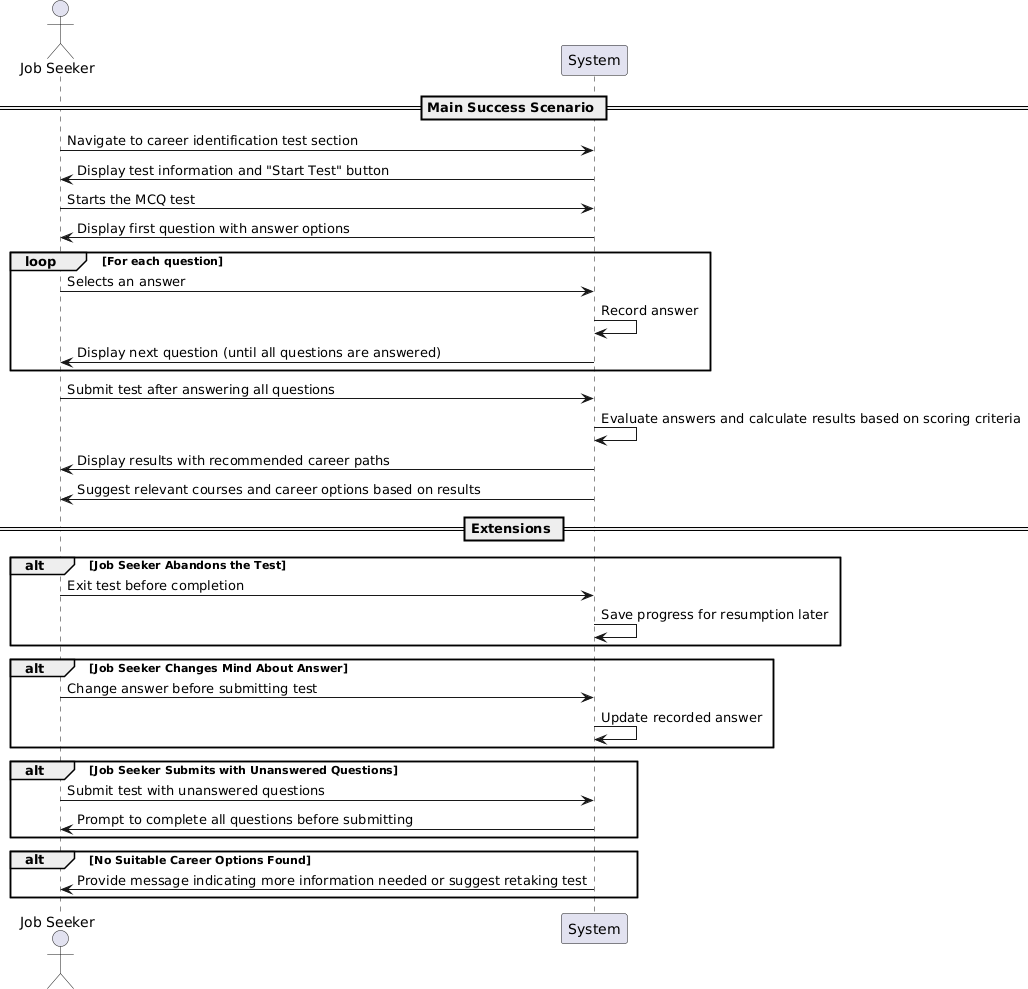
**System Sequence Diagrams (SSDs):**

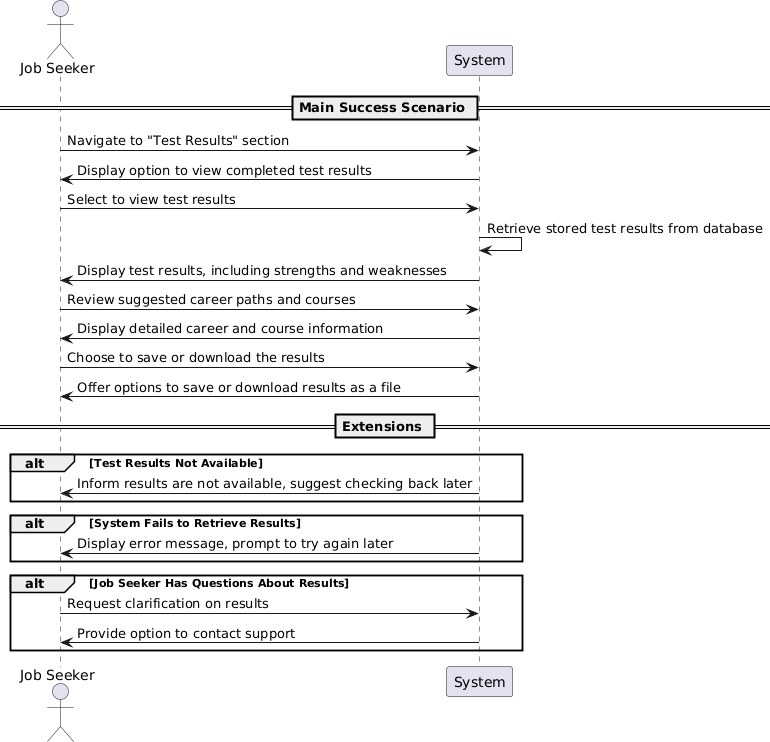
**Manage User**

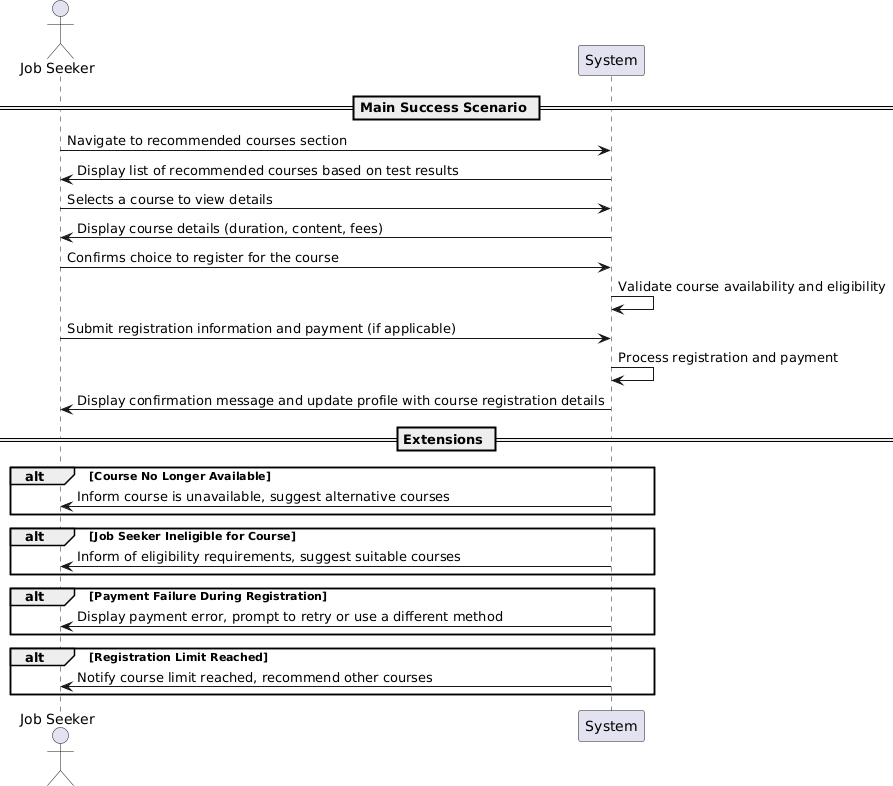
**Account:**

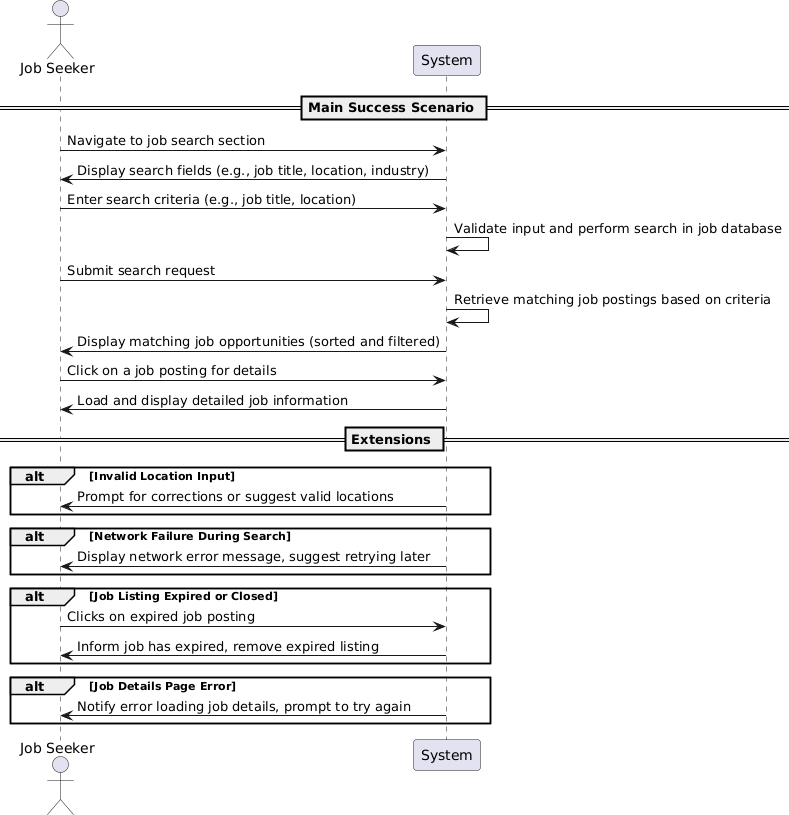
**Login:**

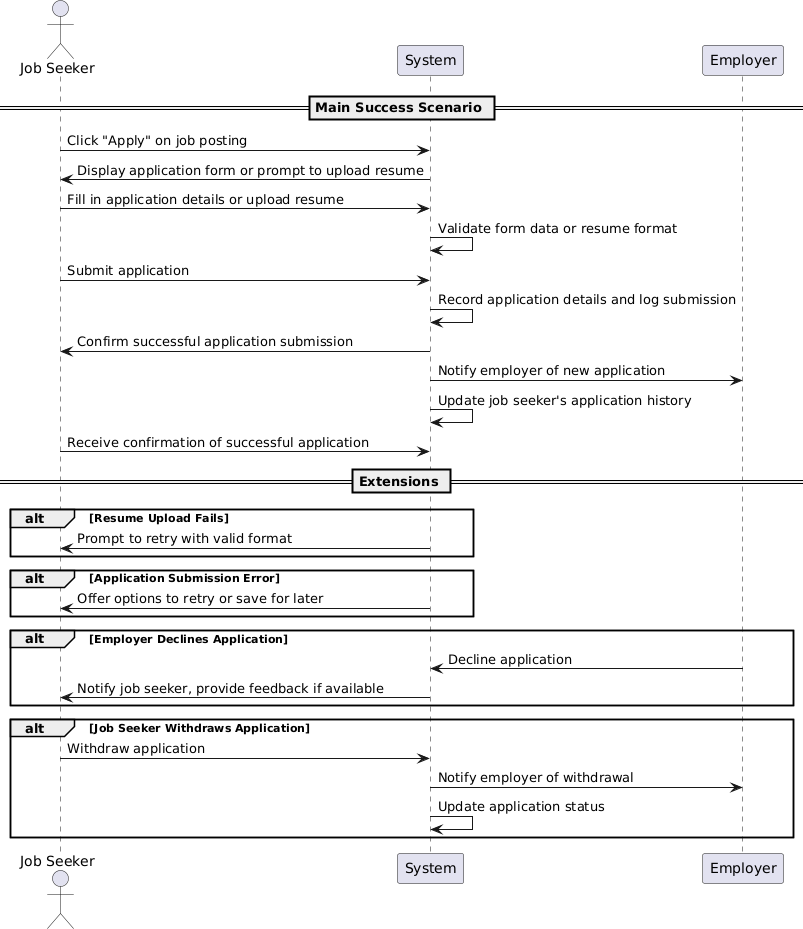
****

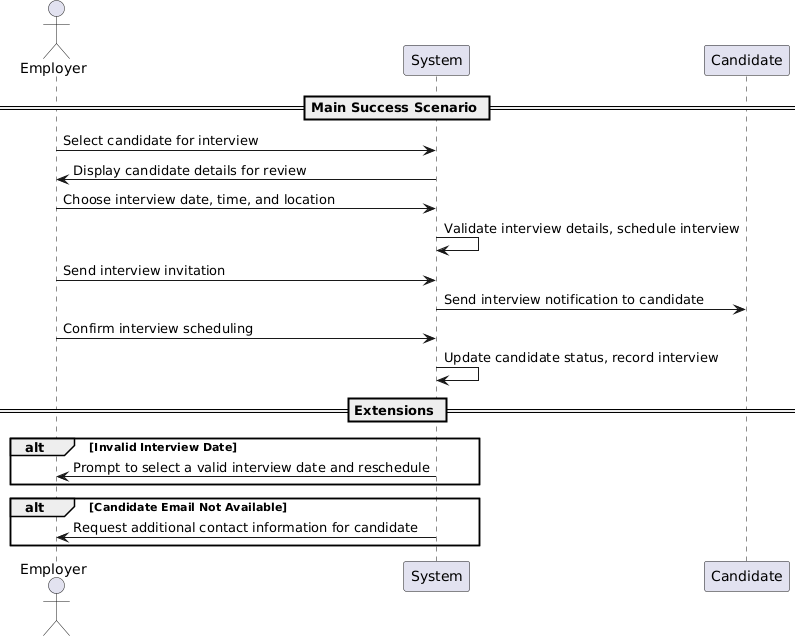


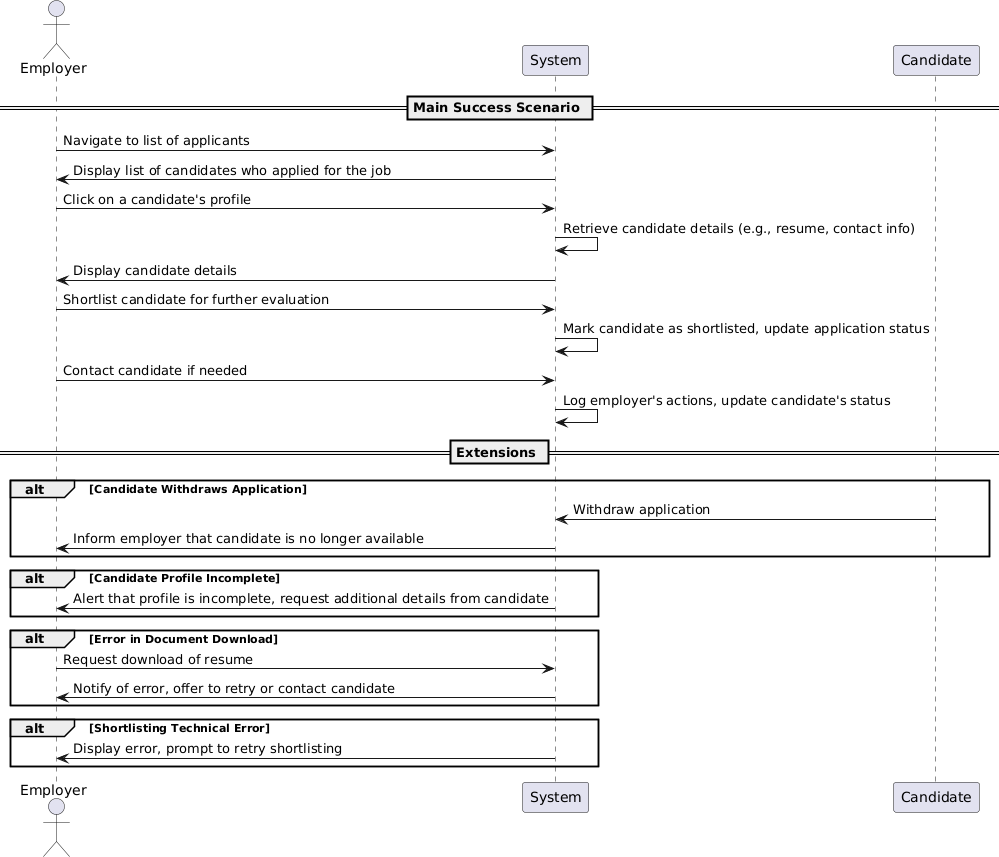


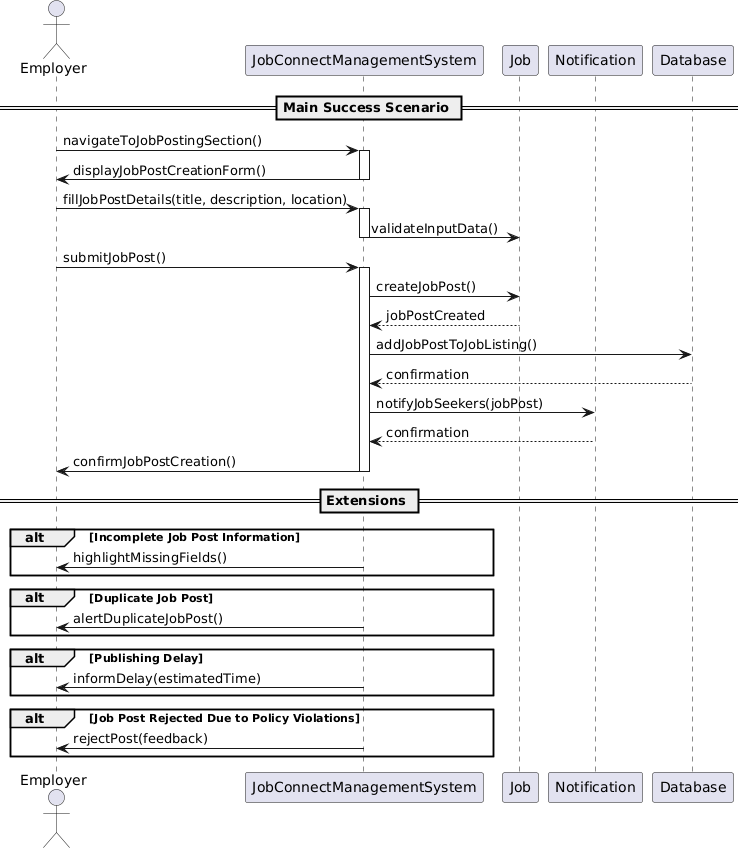


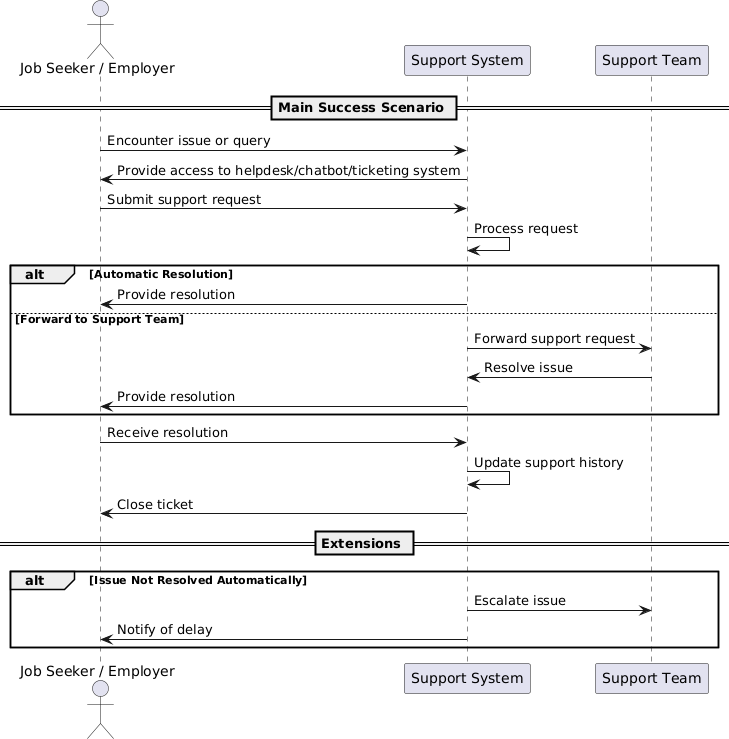


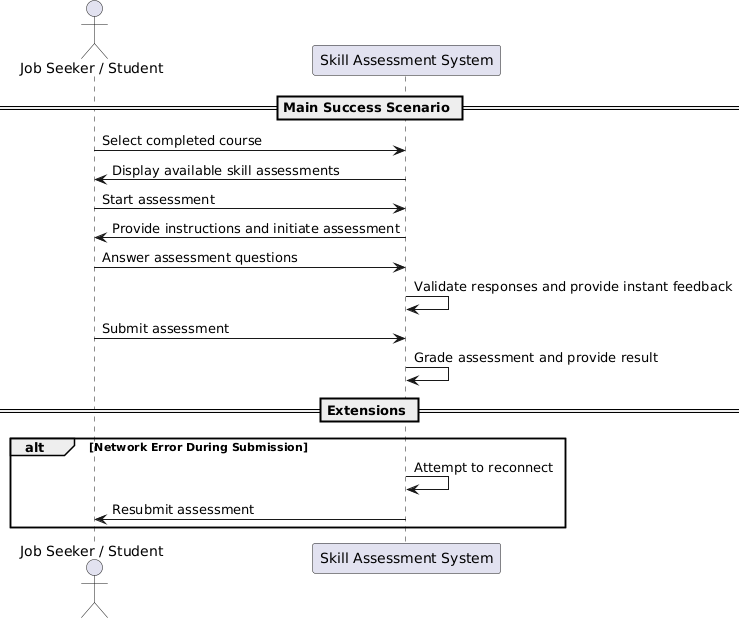


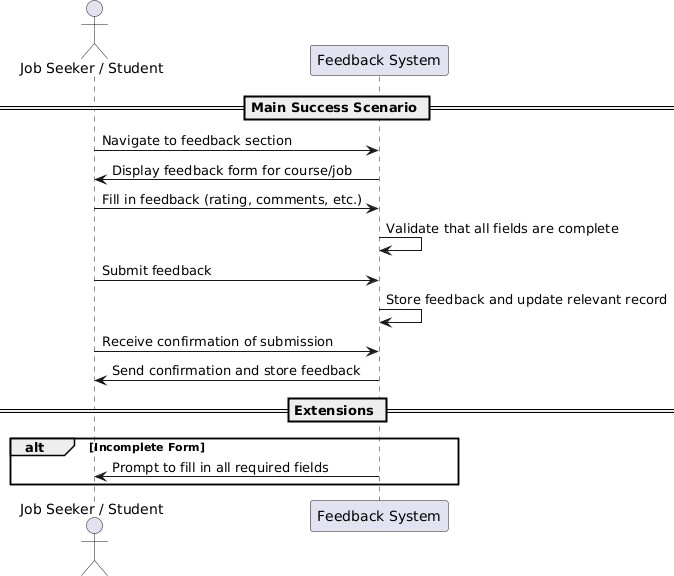


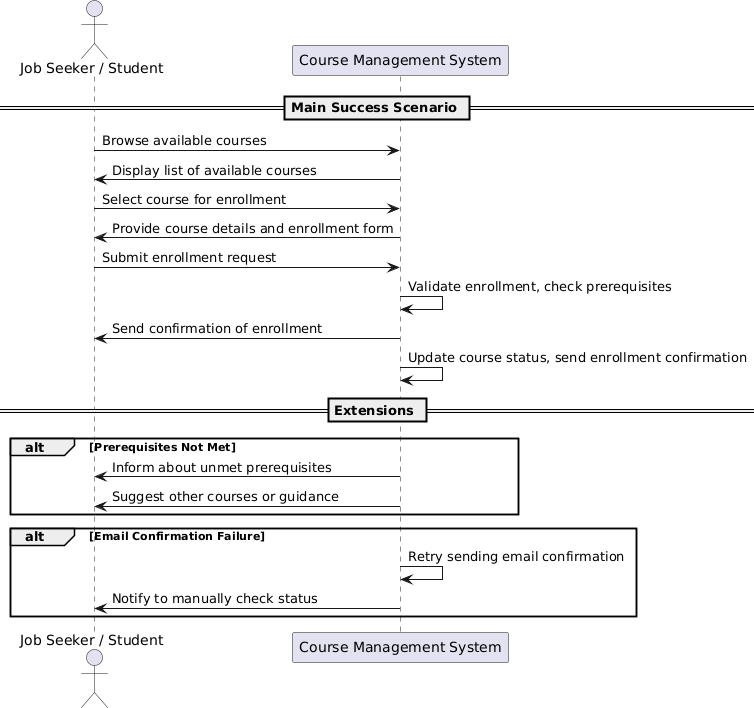




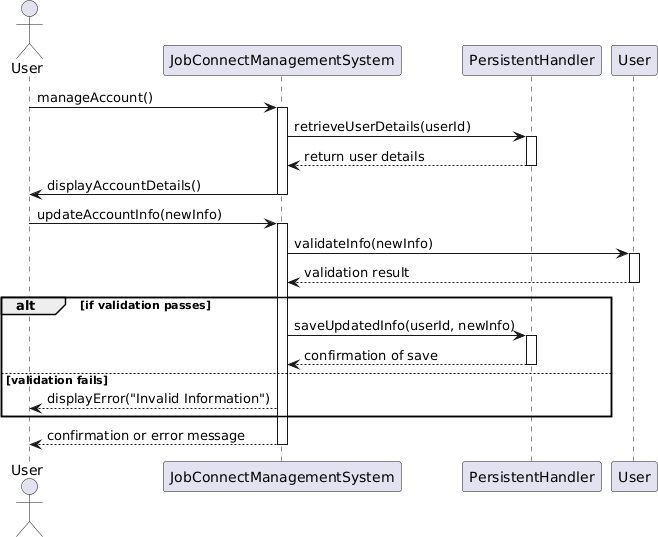




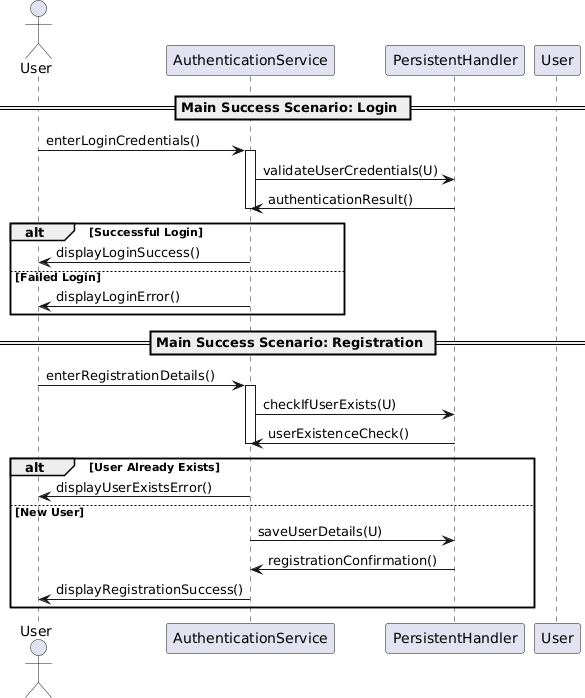




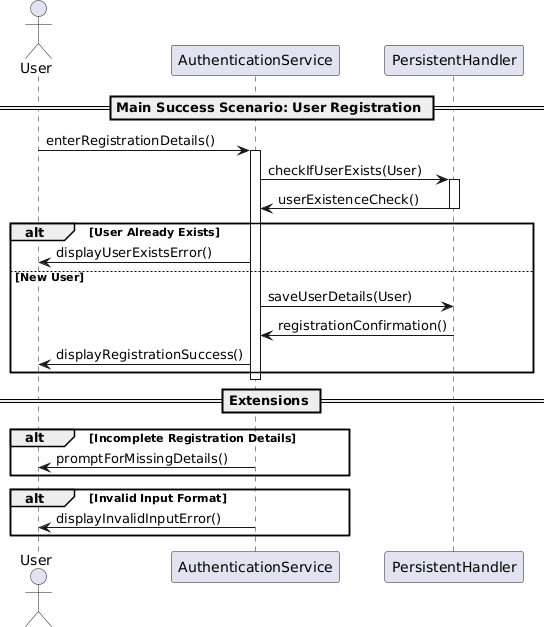
**Manage User Account:**

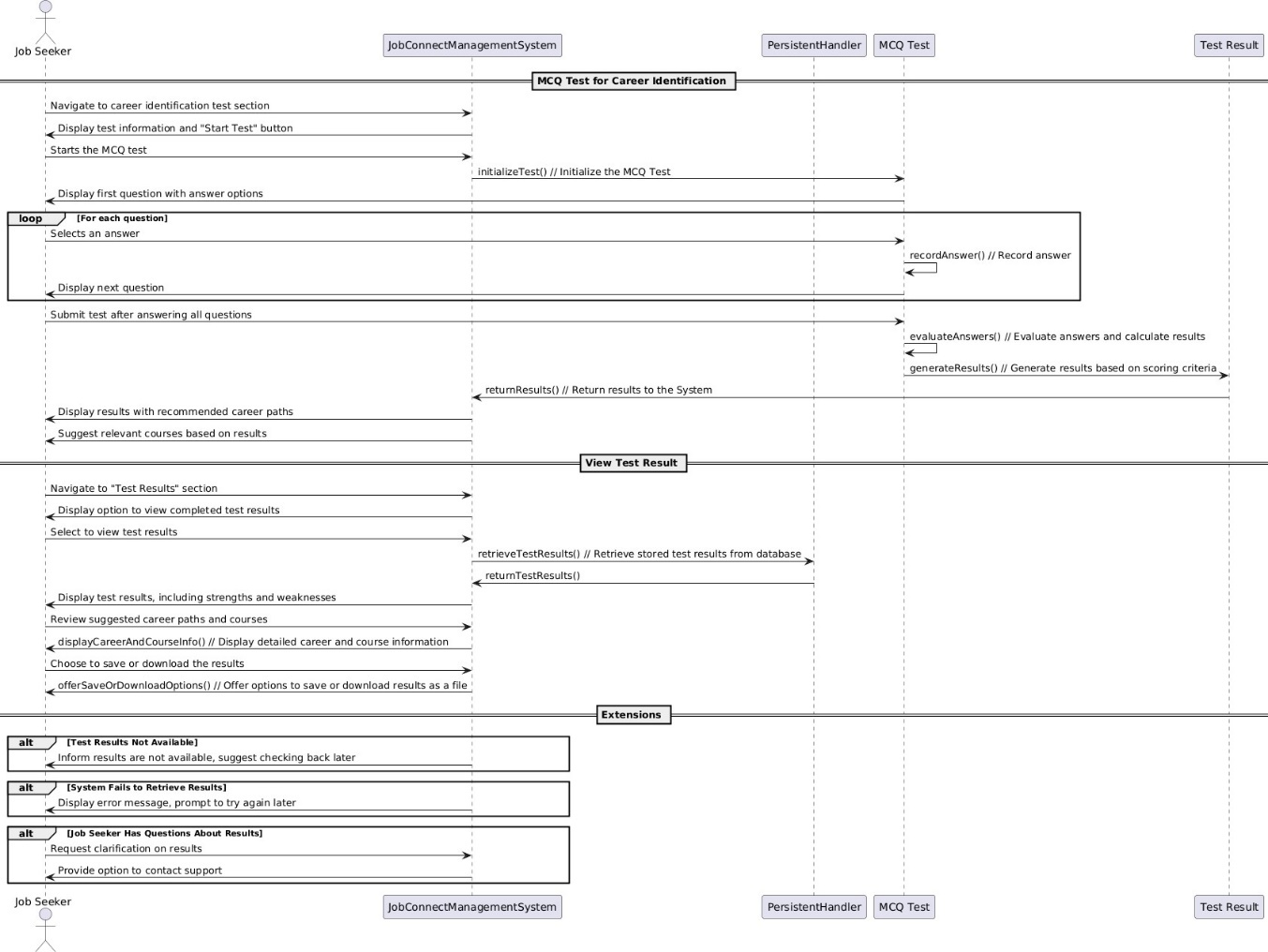
****

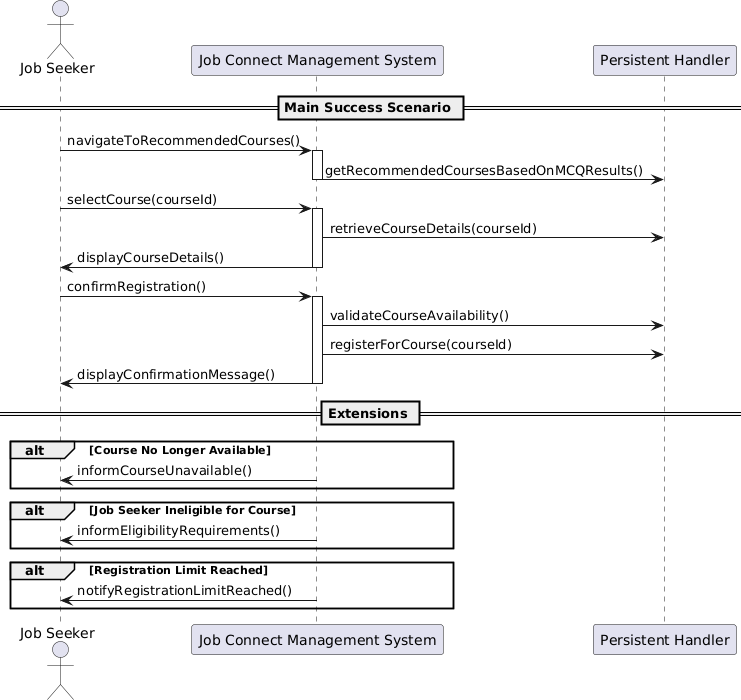
**Login:**

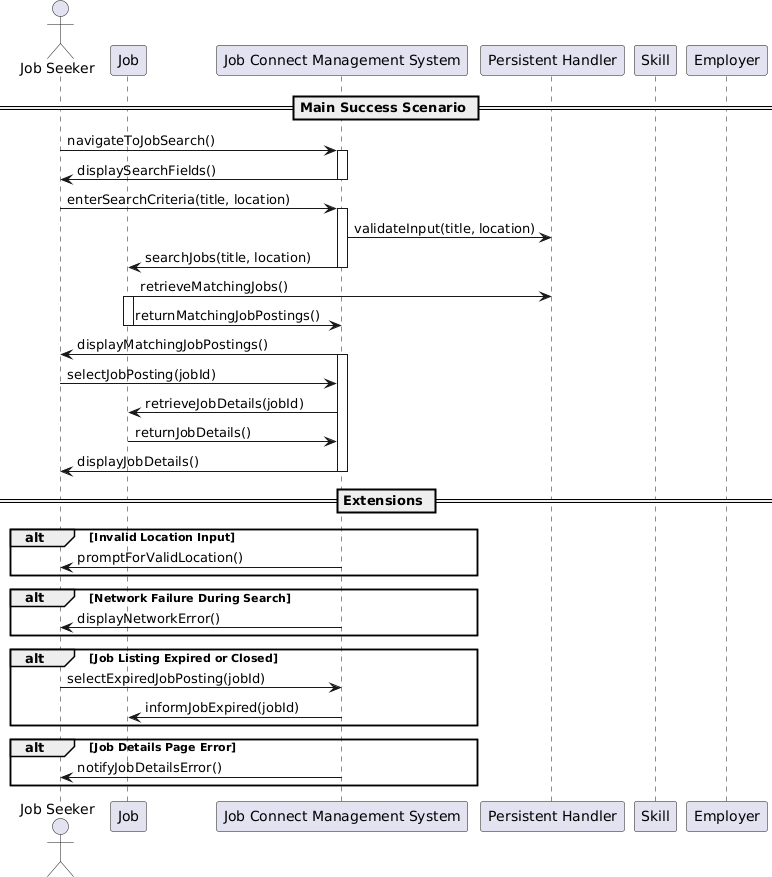


**Registraion**

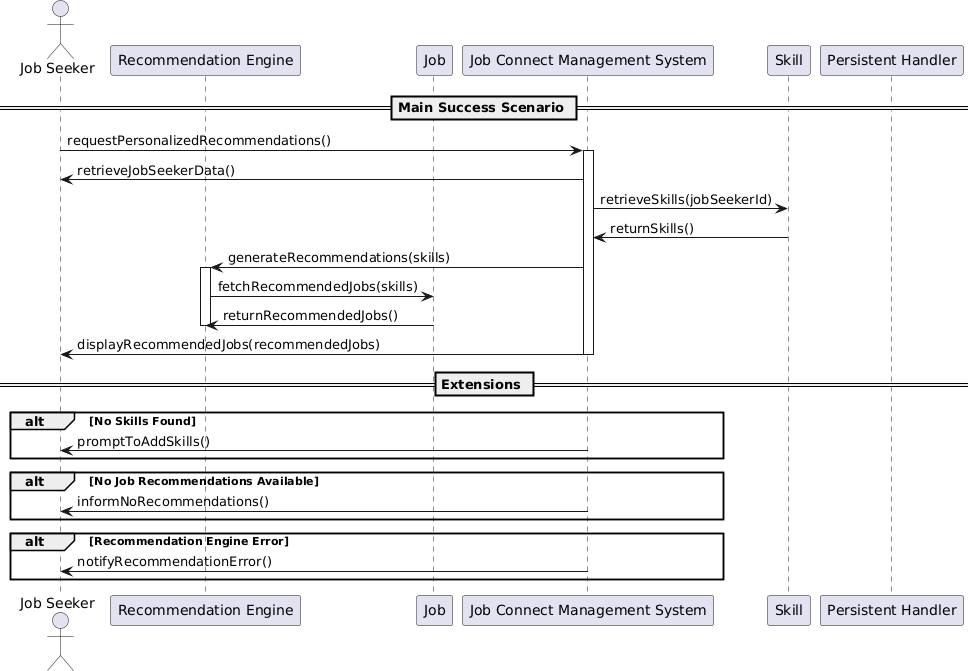




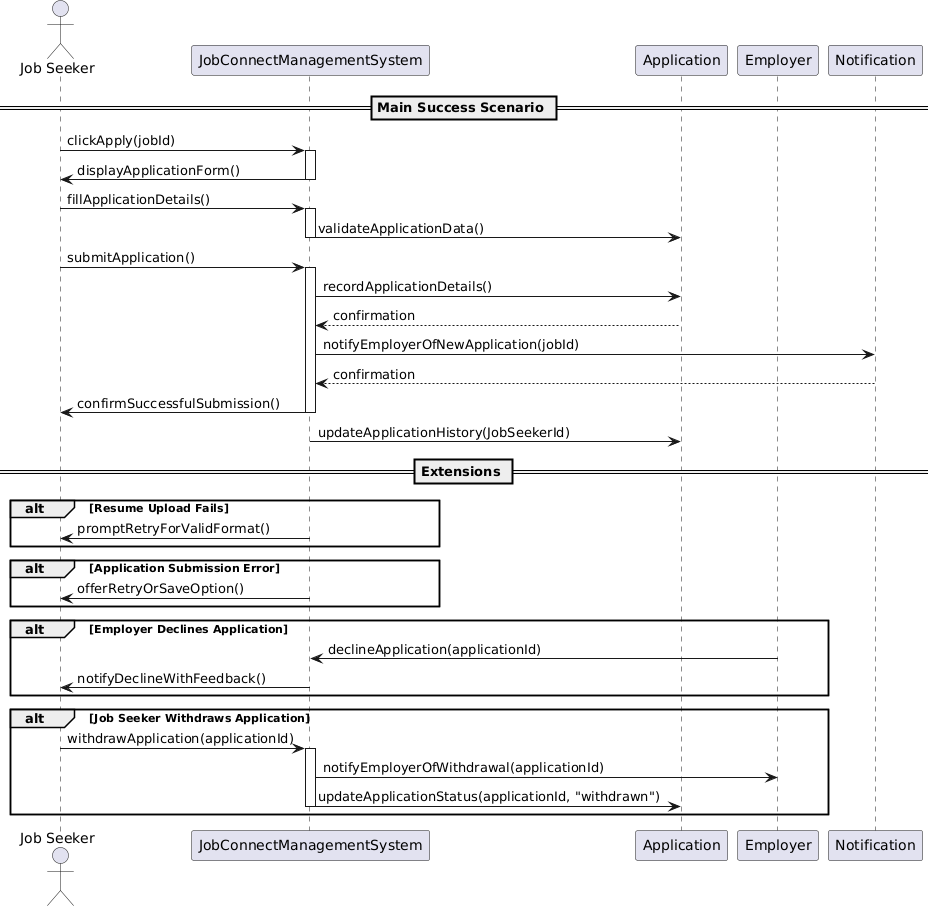




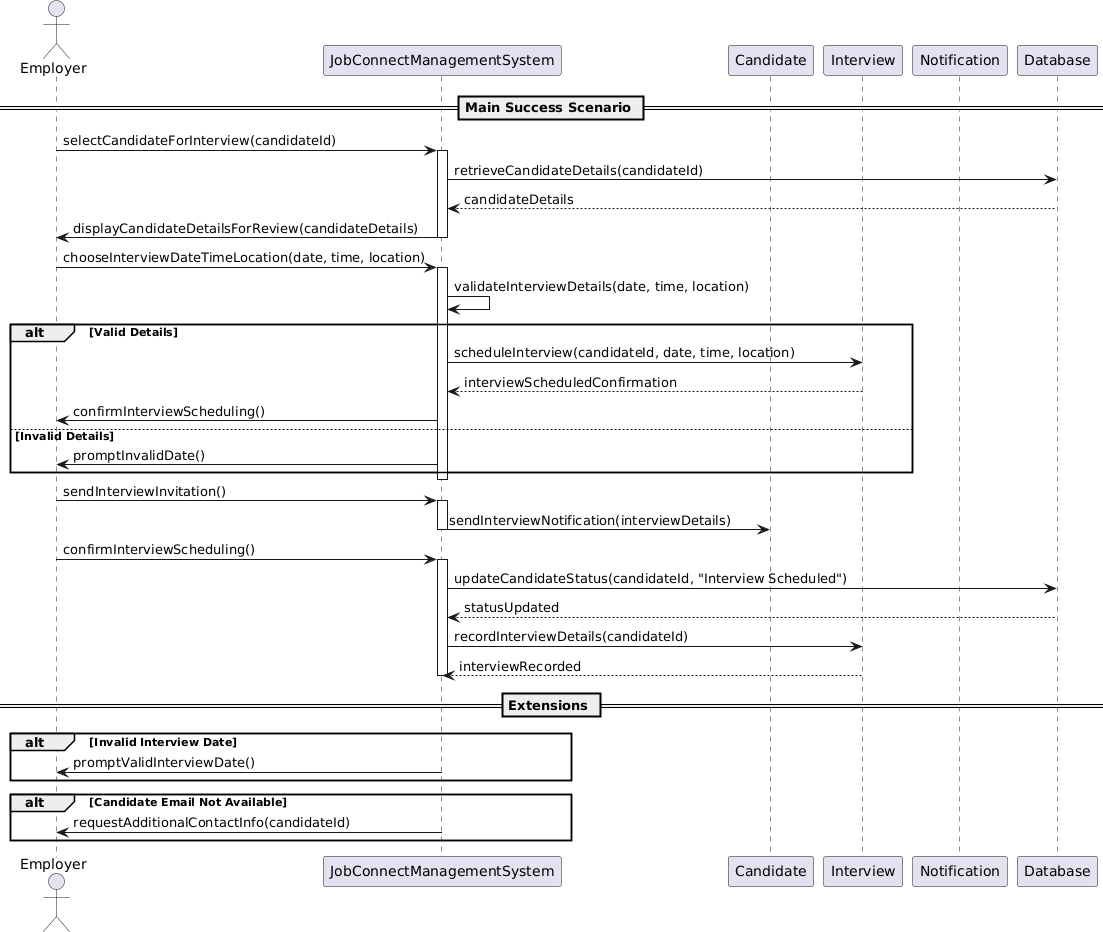
**Personalised Job Recommendations**

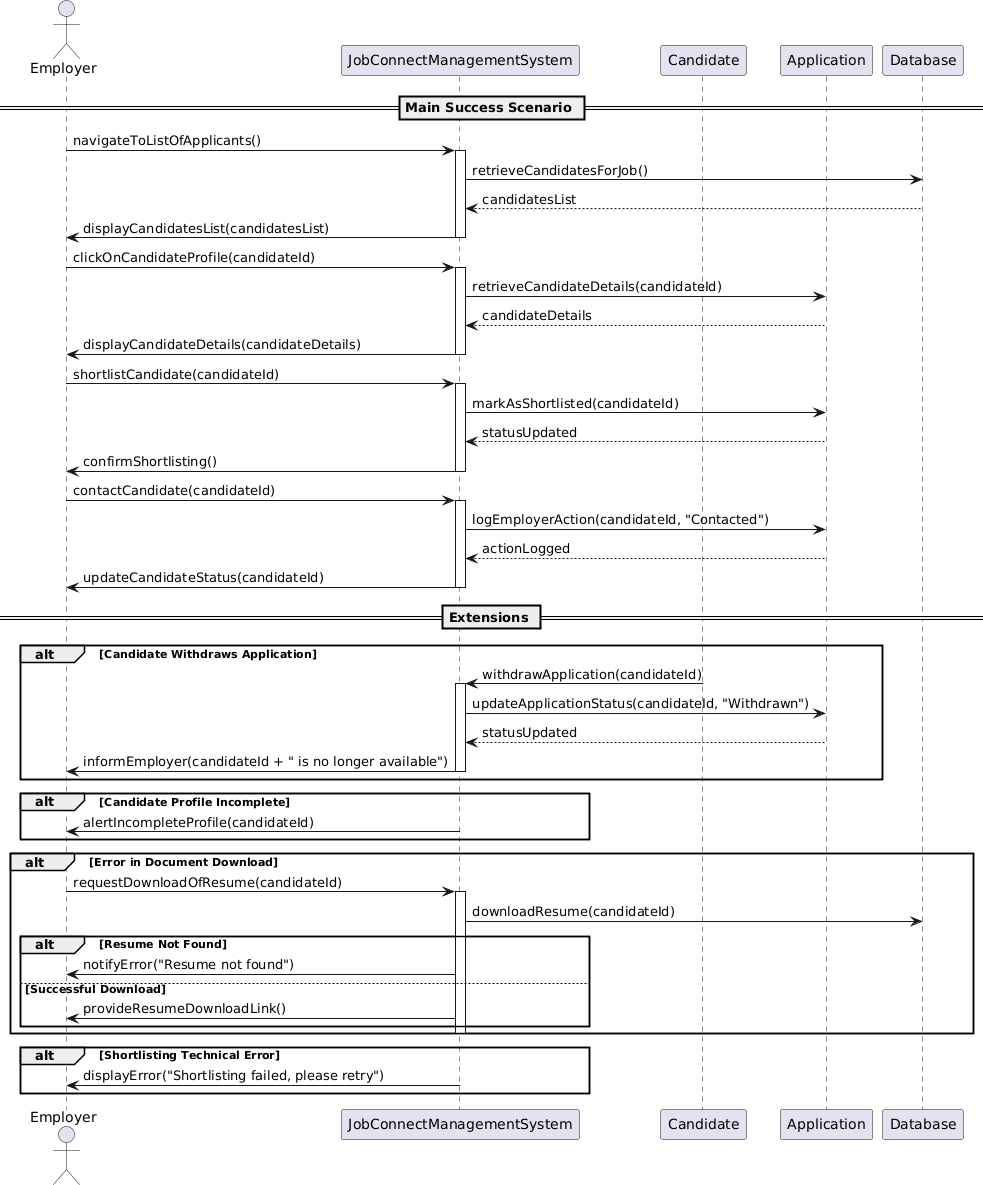


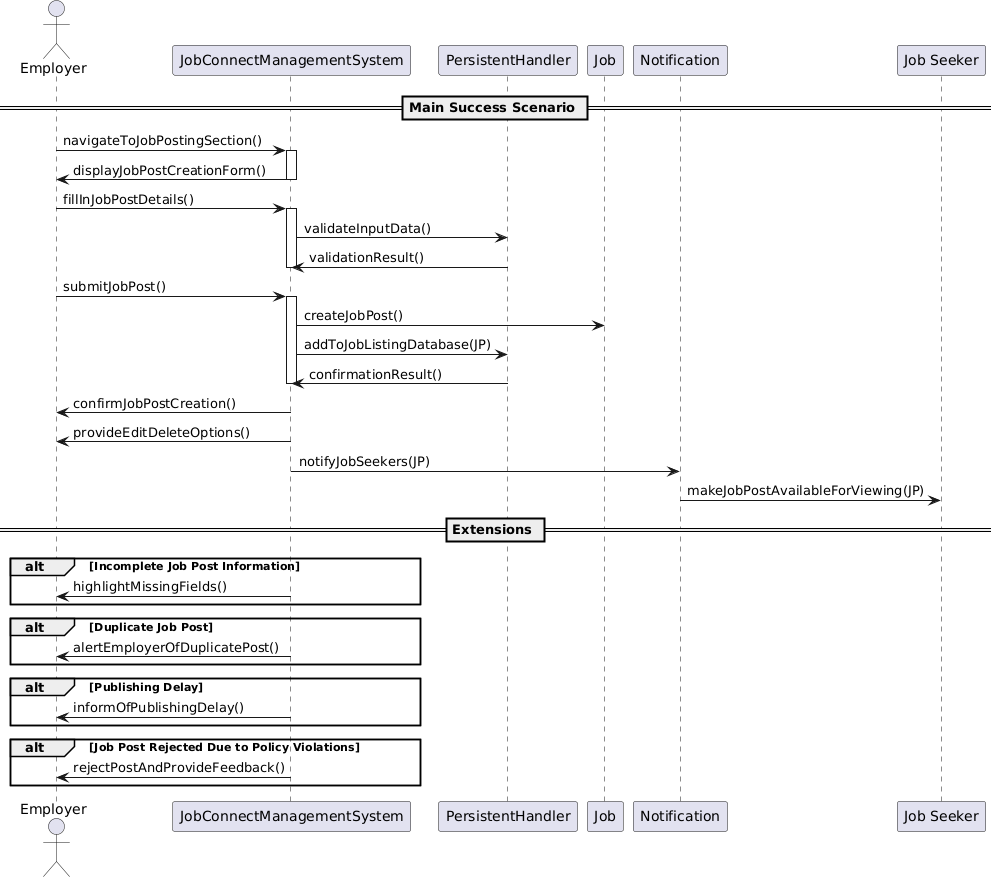
**Apply for Job:**

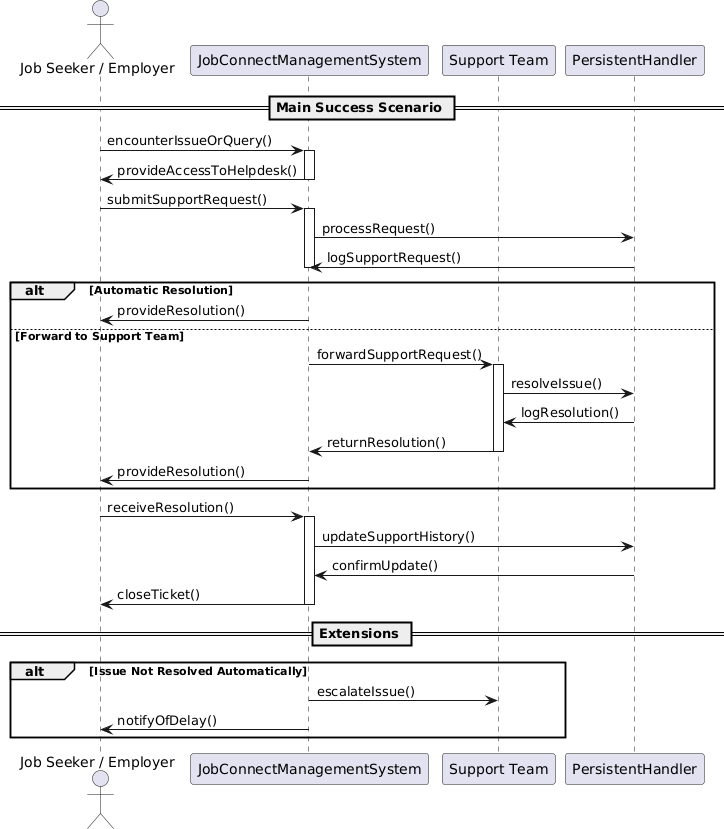
****

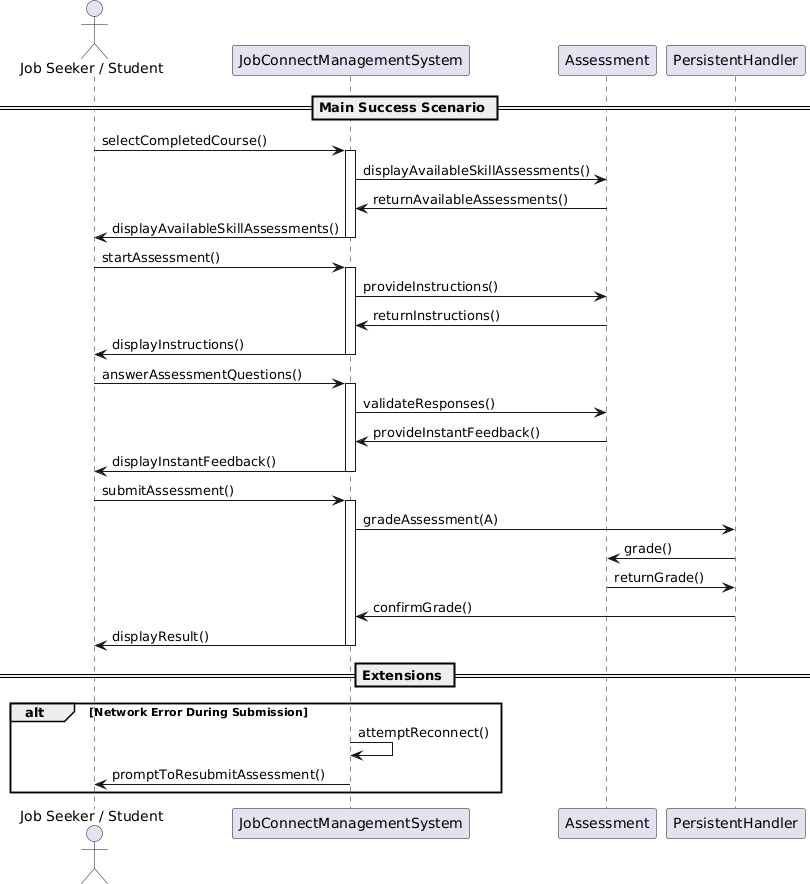
**Send Interview Information:**



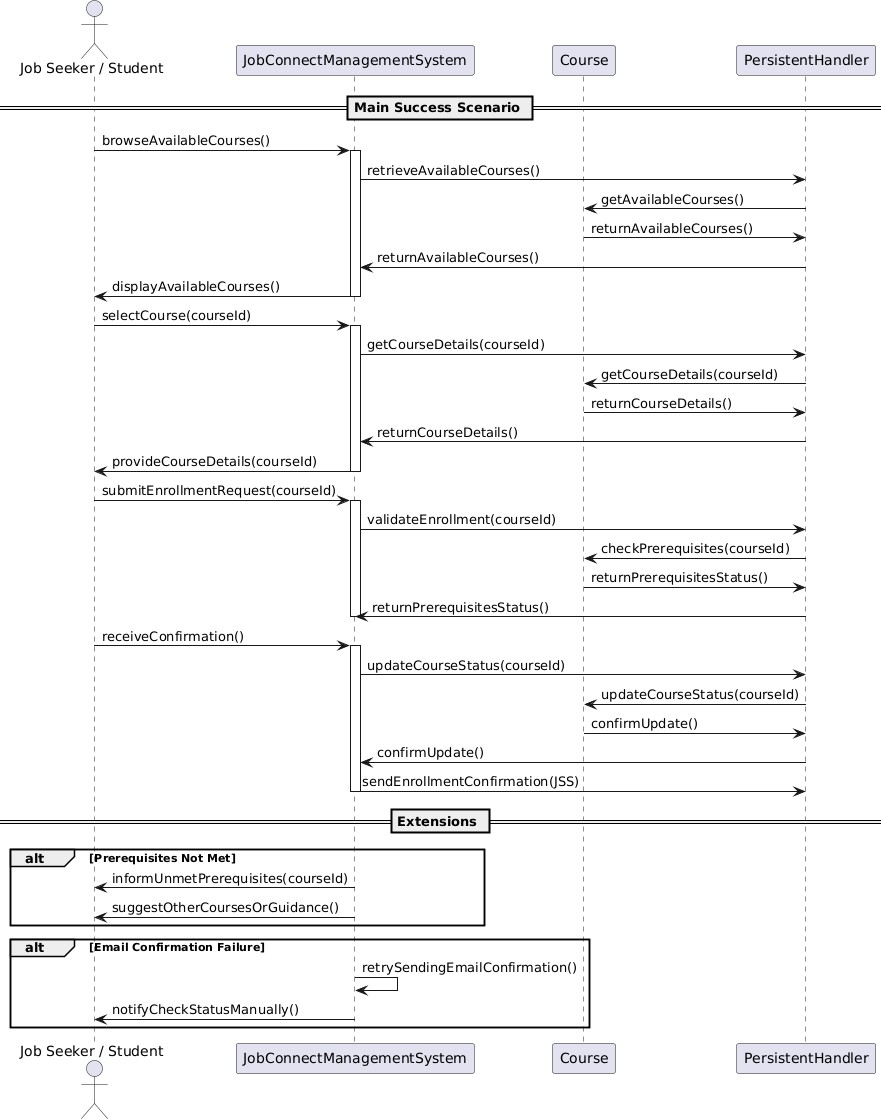




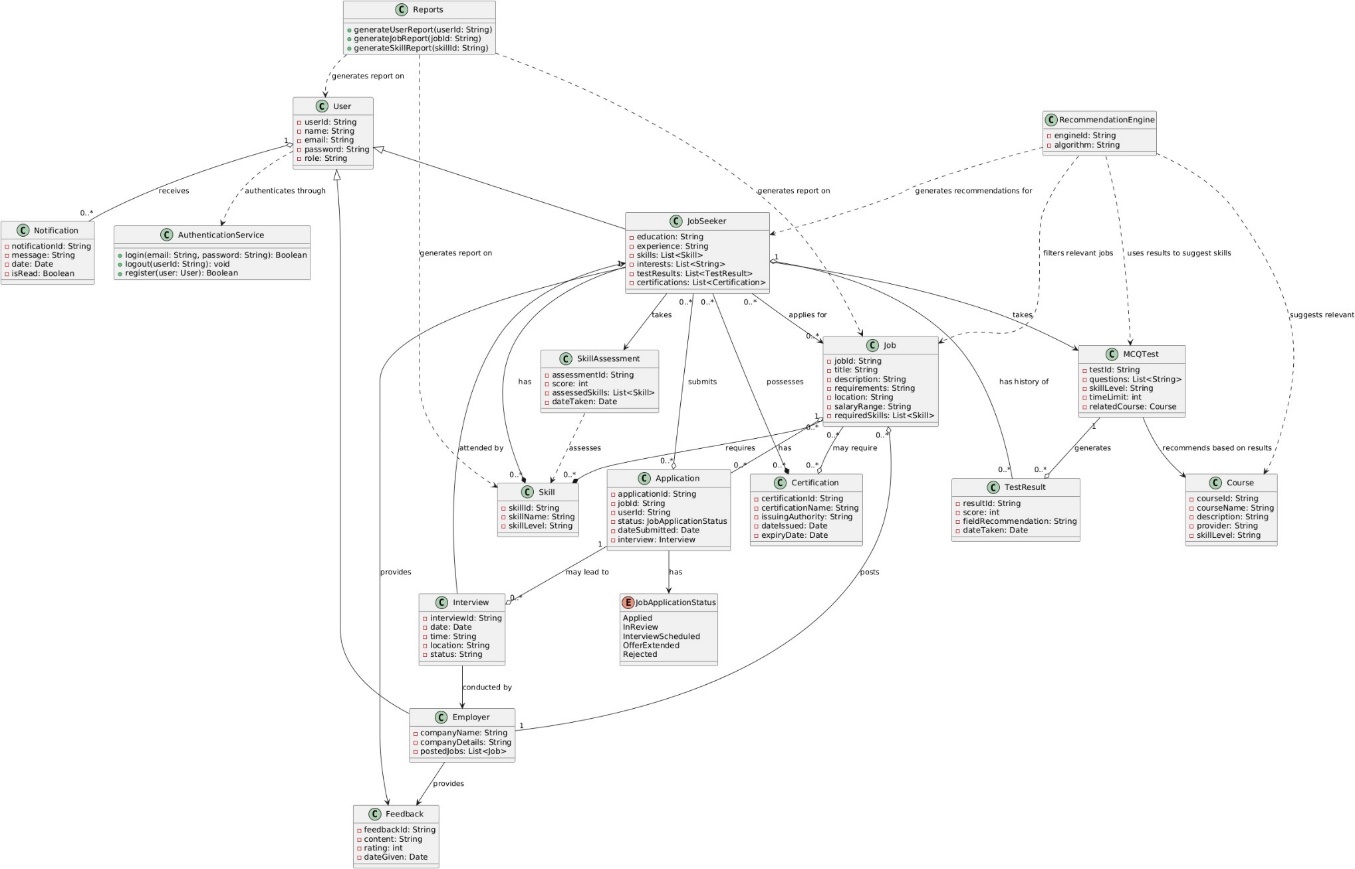




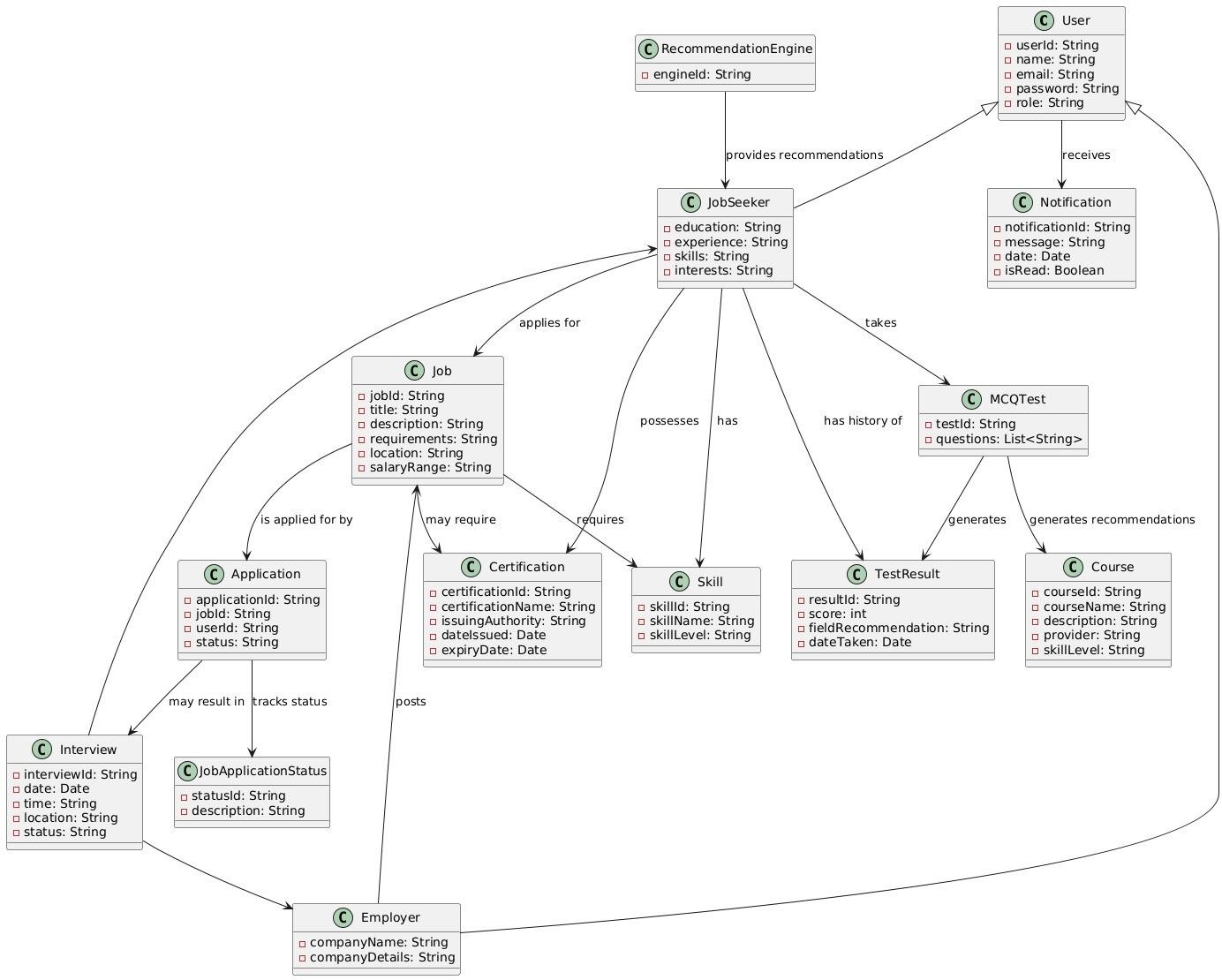




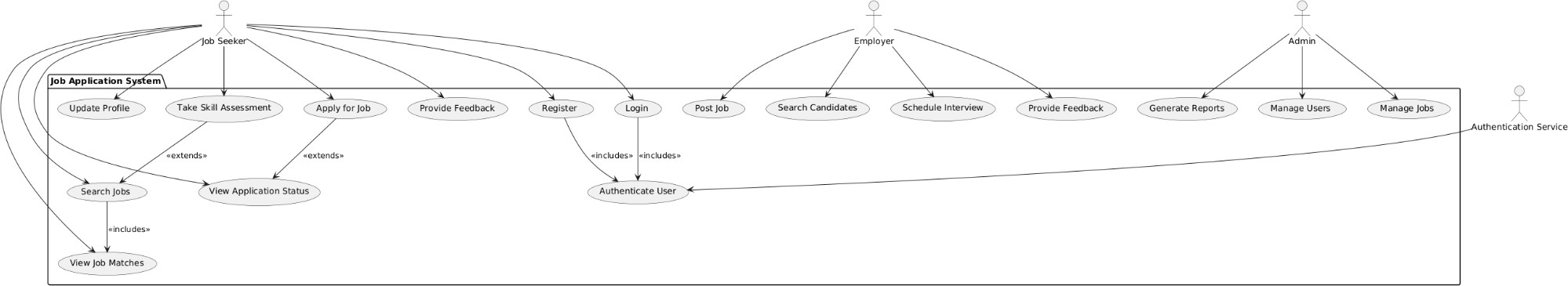
**Class diagram:**



**Diagram model:**



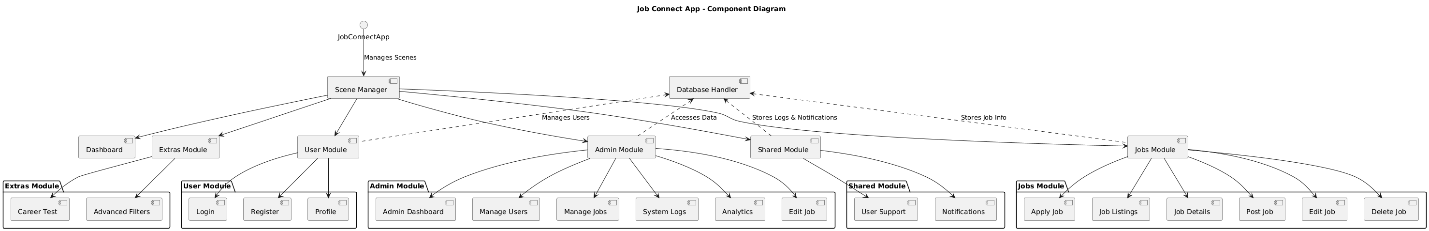
**Update Previous Artifacts:**

****

**Deployment Diagram:**

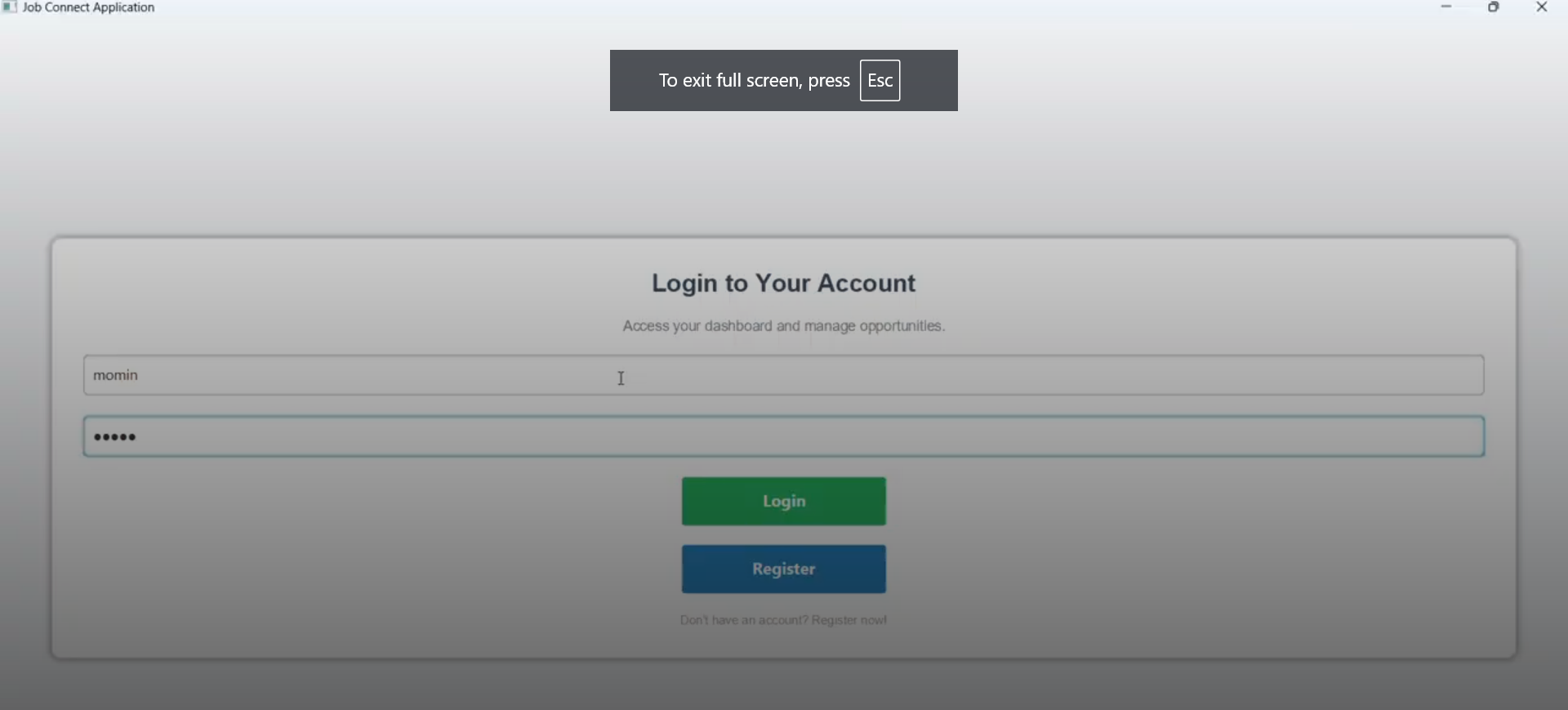
****

**Component Diagram:**

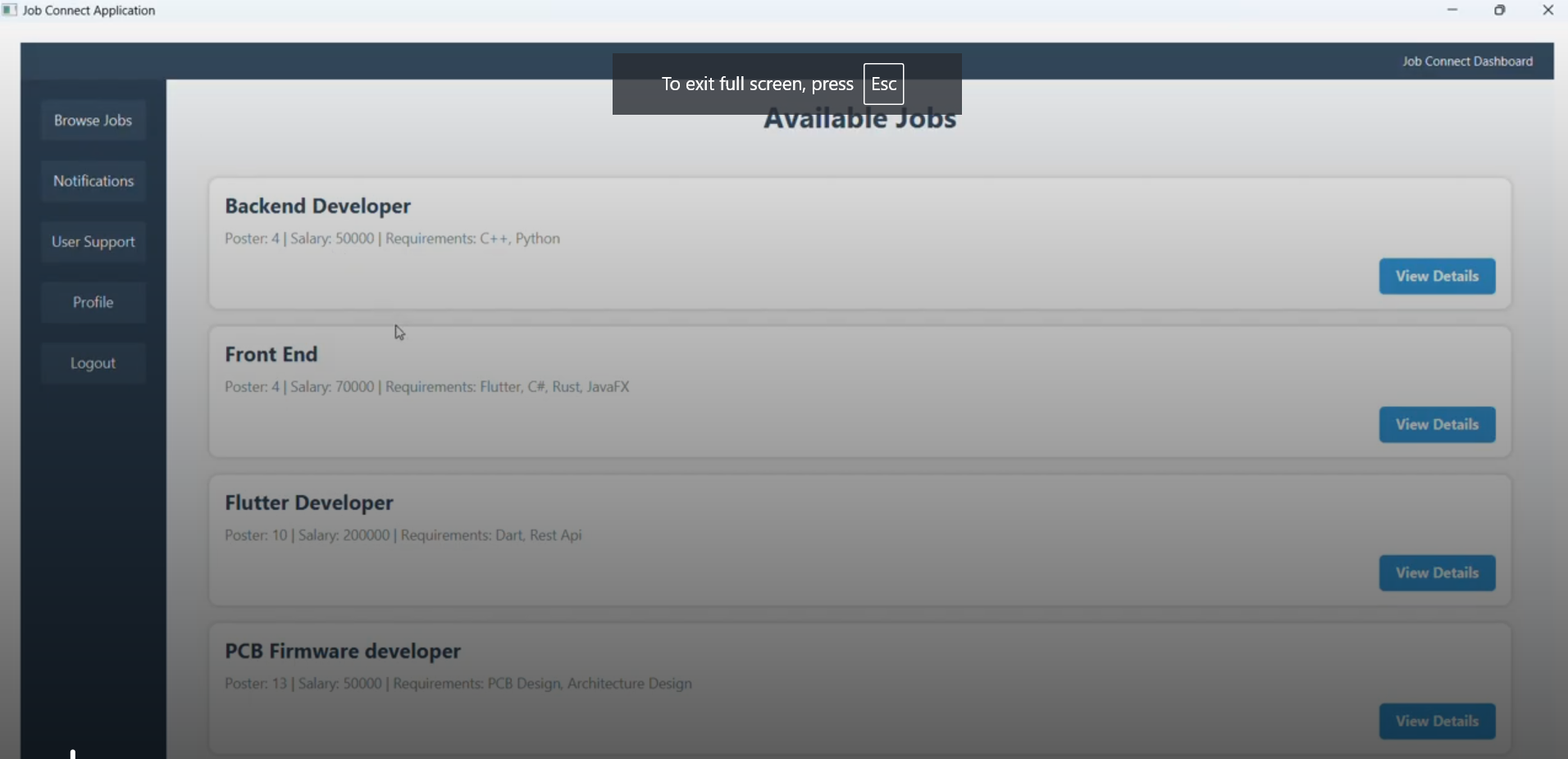
****

**GUI:**

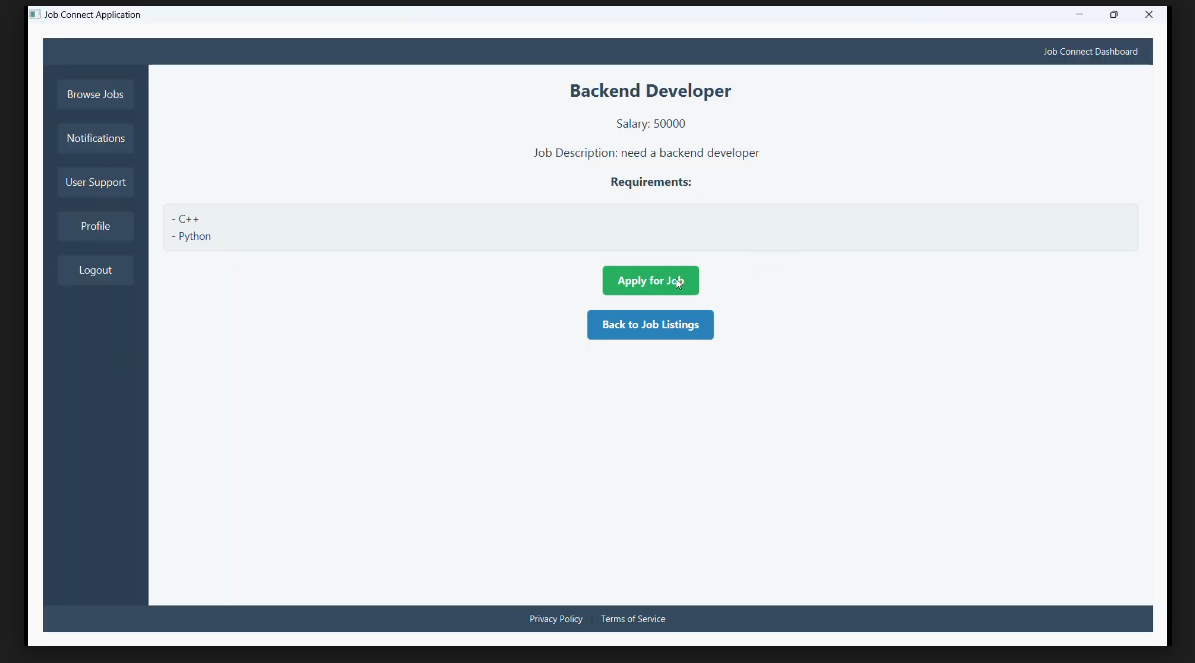
**Login**

****

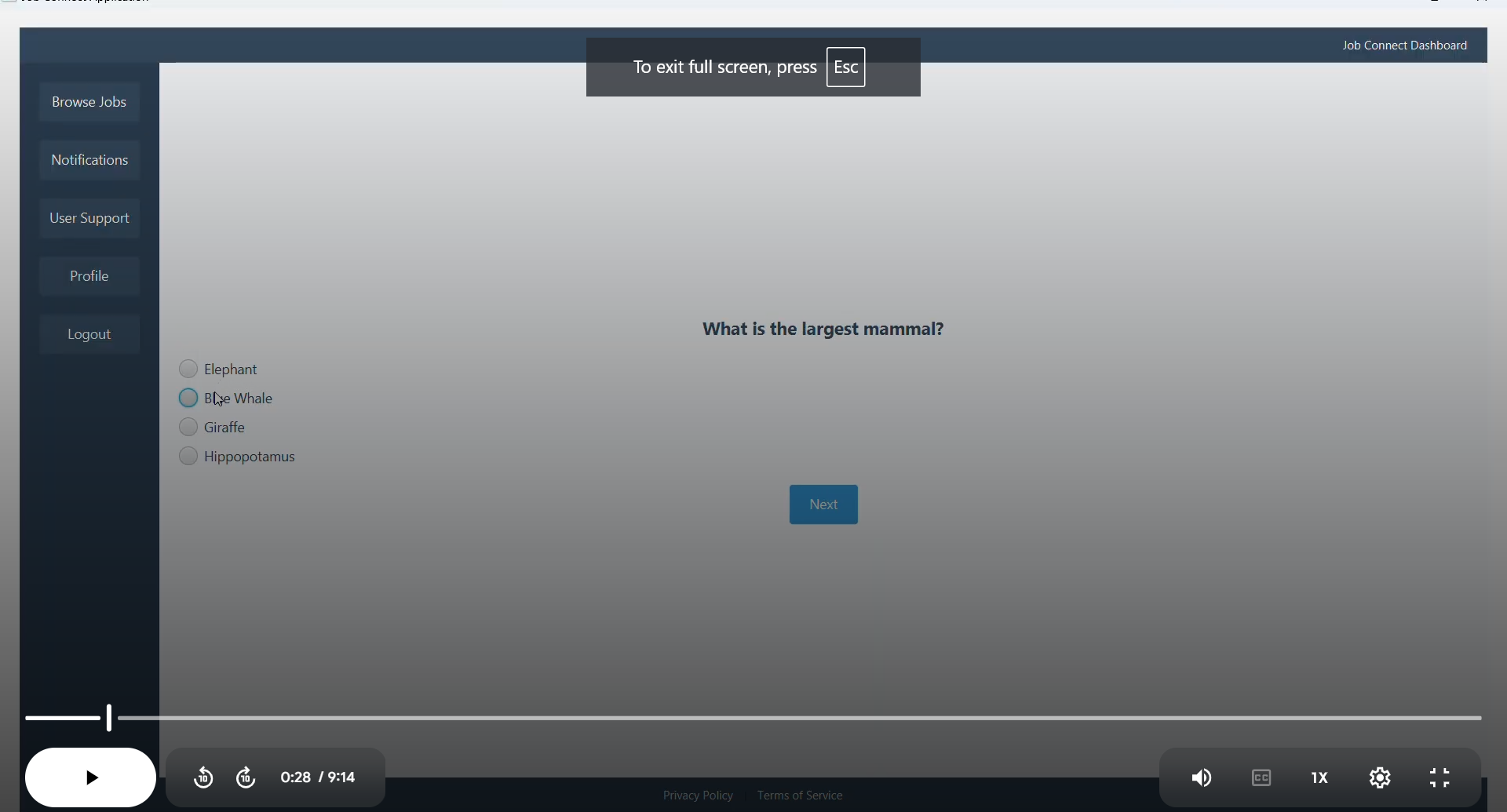
**Available Jobs**

****

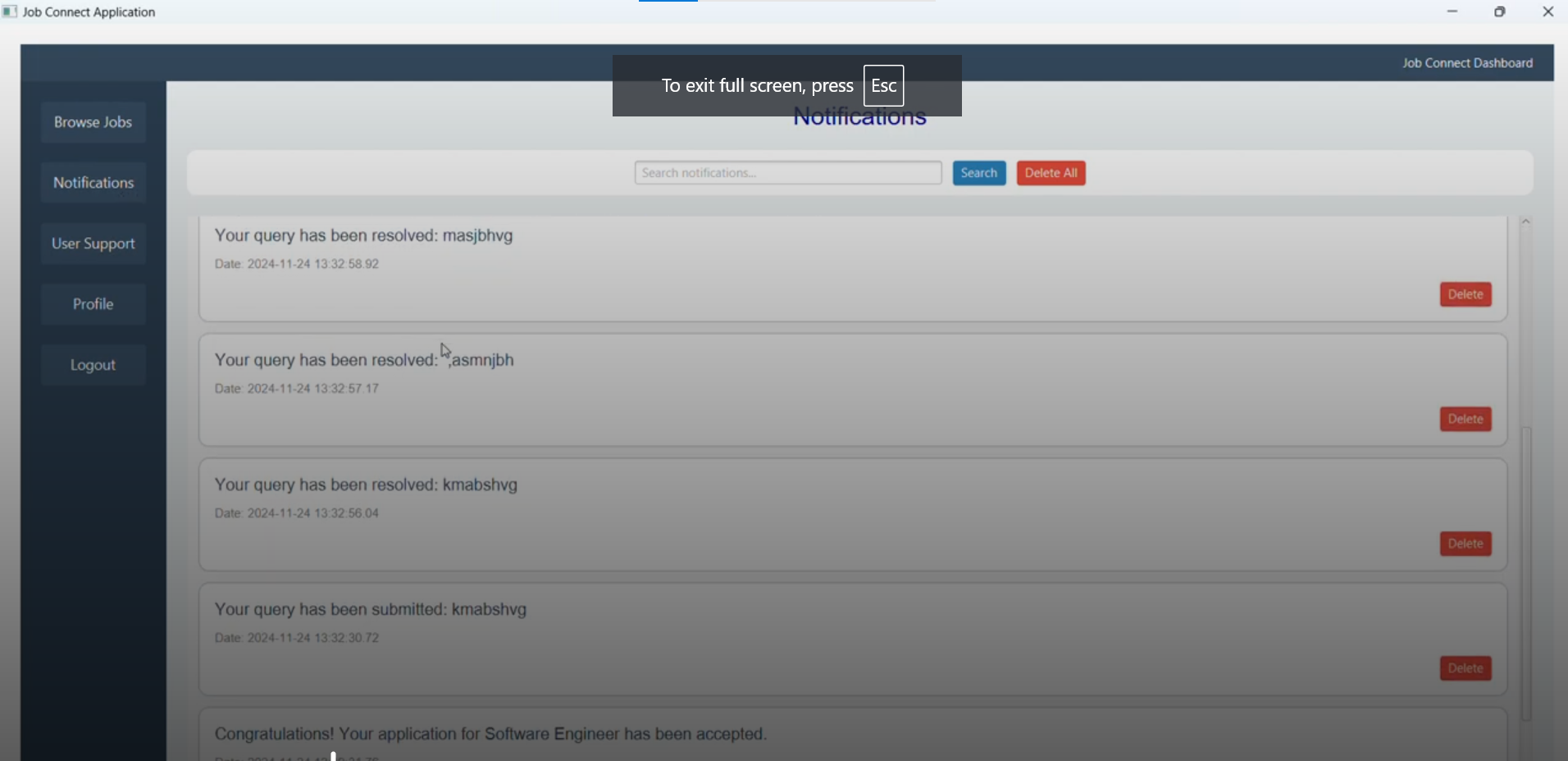
Apply for Job

****

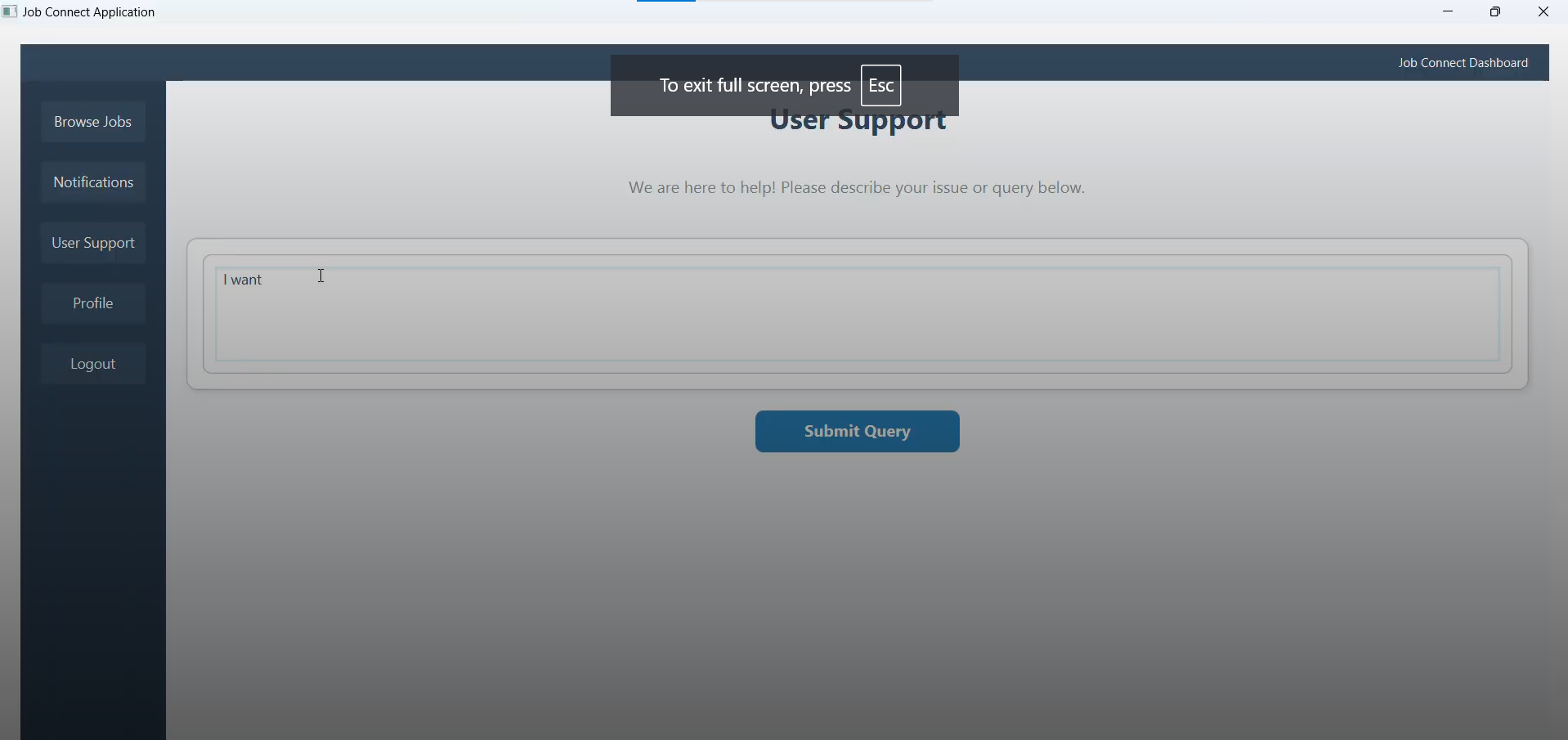
**Job Test**

****

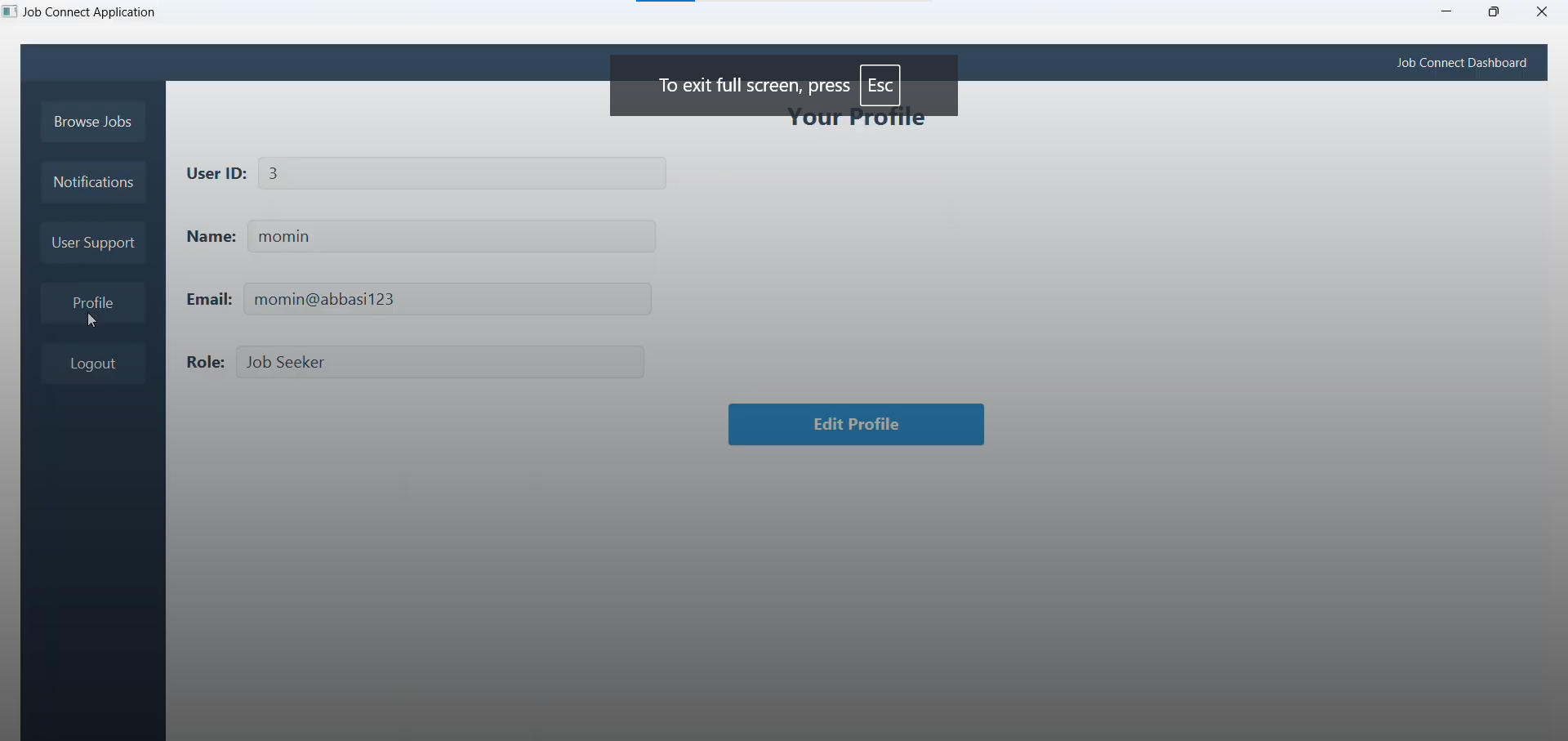
**Notifications**

****

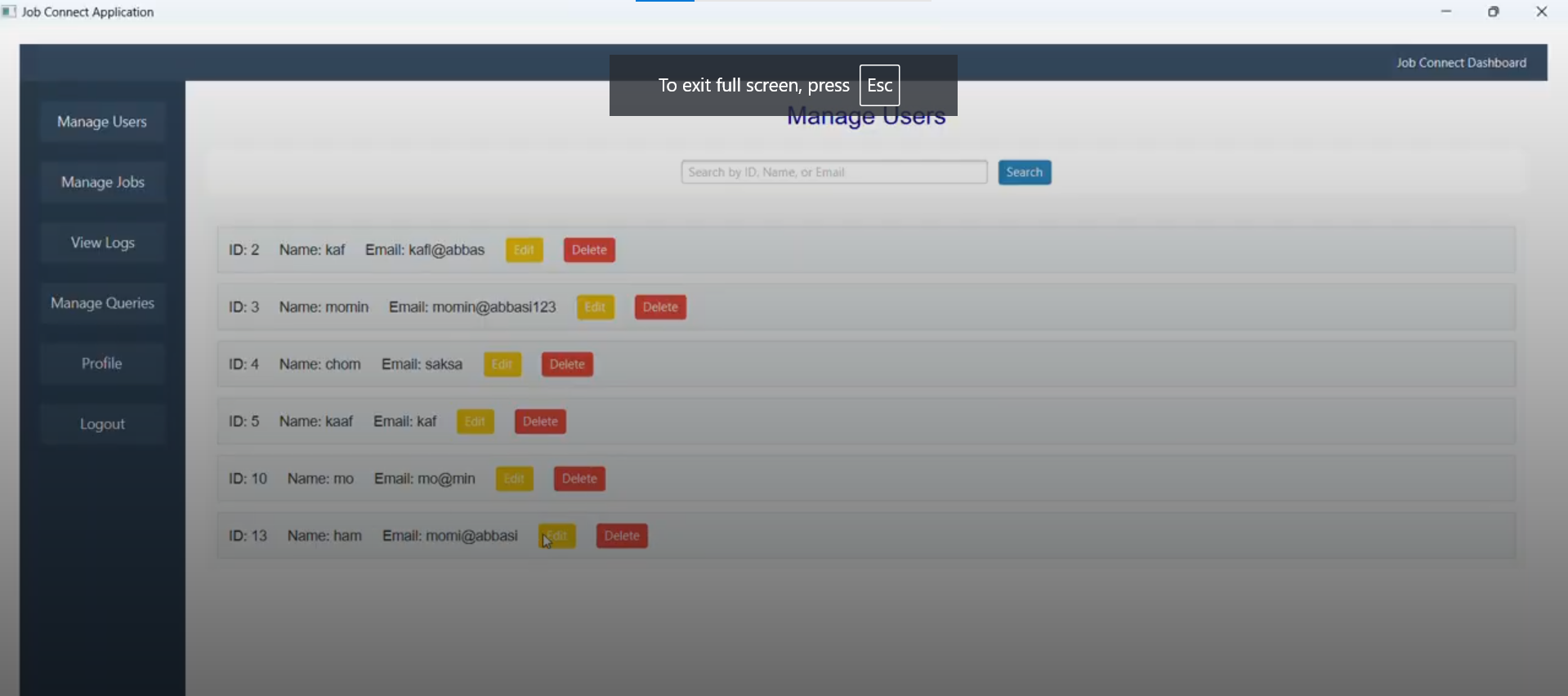
**User Support**

****

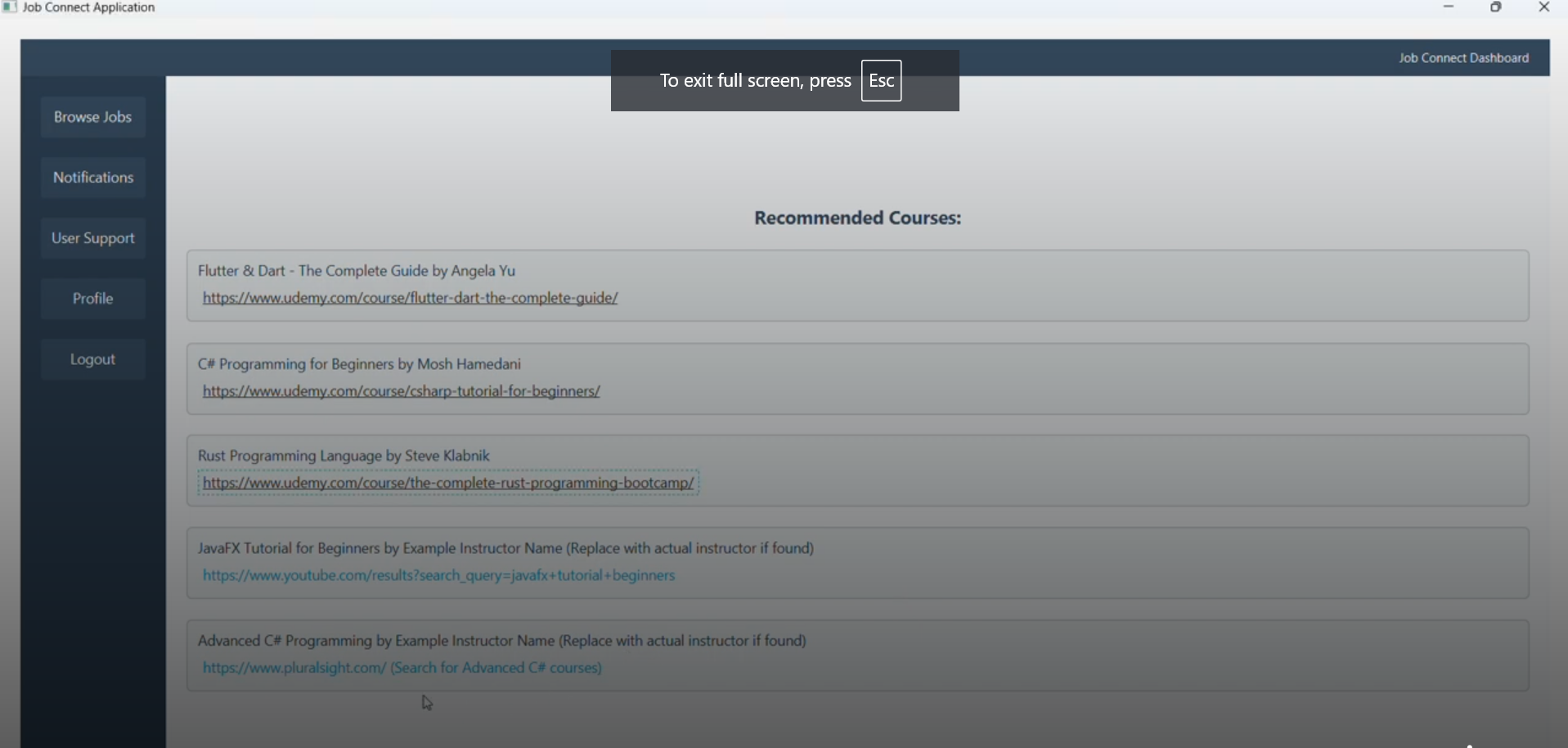
**View user profile**

****

**Manage Users**

****

**Course Recommendations**

****