Software Requirements Specification

for

NextStep

Version 1.0

Prepared by

Aarushi Goel	**202412002**	**202412002@daiict.ac.in**
Mohsin Pathan	**202412070**	**202412070@daiict.ac.in**
Annirudh Pratap	**202412008**	**202412008@daiict.ac.in**

Instructor: Prof. Jayprakash Lalchandani

Course: Software Engineering (IT 632)

Date: 27th February 2024

INDEX

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1	Mohsin Pathan Annirudh Pratap Aarushi Goel	This SRS document is the first version of SRS in which we have described the initial overview of the system requirements and other specifications	26/02/25

1 Introduction

NextStep is an innovative software designed to guide young individuals towards better opportunities by bridging the gap between their needs and available resources. It serves as a one-stop platform where youngsters can register, log in, and access tailored recommendations based on their personal information, such as age, gender, and location. In this Section you will find a brief introduction of the NextStep application.

1.1 Document Purpose

The purpose of this document is to define the software requirements for the **NextStep Application**. This document serves as a guide for developers, project managers, testers, and other stakeholders, ensuring a clear understanding of the system's capabilities and functionalities.

This SRS covers all the core features and components of the NextStep Application, including user authentication, job and education opportunity recommendations, and a structured user interface for seamless navigation. It details the system's objectives, intended audience, functional and non-functional requirements, and overall design considerations.

1.2 Product Scope

The **NextStep Application** is a web-based platform designed to assist individuals, particularly young professionals and students, in discovering relevant job opportunities, educational programs, and government policies based on their location and profile. The application provides area-wise recommendations, ensuring users receive the best opportunities available in their region.

Benefits:

- **Personalized Job and Education Matching:** Area-wise suggestions based on user preferences and location.
- **Integrated Government Policies & Health Services:** Users can access policies and health facilities available in their region.
- **User-Friendly Web Interface:** Designed for a smooth and engaging user experience.
- **Seamless Registration and Authentication:** Secure login and profile management system

1.3 Intended Audience and Document Overview

This document is intended for:

- **Developers:** To understand functional and non-functional requirements for implementation.
- **Project Managers:** To oversee development progress and ensure compliance with requirements.
- **Testers:** To validate and verify the software's functionality and usability.
- **Clients (End Users & Professors):** To ensure the system aligns with their expectations and needs.

Document Overview

The document is structured as follows:

- **Section 1: Introduction**
  - Defines the purpose and scope of NextStep.
  - Identifies the intended audience and explains document conventions.
  - Lists relevant references and acknowledgments.
- **Section 2: Overall Description**
  - Provides an overview of NextStep and its core functionalities.
  - Highlights design and implementation constraints.
  - Details assumptions and dependencies affecting system development.
- **Section 3: Specific Requirements**
  - Defines external interface requirements for user interaction.
  - Lists functional requirements specifying system behavior.
  - Includes a Use Case Model to illustrate key system interactions.
- **Section 4: Other Non-Functional Requirements**
  - Outlines performance, safety, and security requirements.
  - Defines critical software quality attributes for system reliability and usability.
- **Section 5: Other Requirements**
  - Covers any additional requirements necessary for successful implementation.
- **Appendices**
  - **Appendix A – Data Dictionary:** Defines key data elements used within NextStep.
  - **Appendix B – Group Log:** Documents contributions and responsibilities of project members.

1.4 Definitions, Acronyms and Abbreviations

NextStep: The web-based platform designed to provide job and educational opportunities to young individuals based on their location and profile.

User: Refers to individuals who register on the NextStep platform to access job, education, and government policy recommendations.

SRS (Software Requirements Specification): A document that outlines the functional and nonfunctional requirements of a software system.

UI (User Interface): The visual layout and interactive components through which users interact with the NextStep application.

API (Application Programming Interface): A set of protocols that allow different software components to communicate with each other, enabling backend and frontend integration.

CRUD (Create, Read, Update, Delete): The four basic operations performed on a database to manage stored information.

DBMS (Database Management System): Software that allows users to store, retrieve, and manage structured data efficiently.

MongoDB: A NoSQL database used for storing flexible and dynamic data structures, including user interactions, logs, and unstructured data.

React.js: A JavaScript library used for building the frontend of the NextStep application.

Node.js: A runtime environment that executes JavaScript code server-side, used for the backend of the NextStep application.

Express.js: A backend web application framework for Node.js used to build APIs for NextStep.

Authentication: The process of verifying a user's identity before granting access to the application.

Government Policies: Various schemes and initiatives introduced by the government, which are displayed in NextStep based on the user’s region.

Health Services: Information on hospitals, clinics, and health insurance availability in different regions, accessible through NextStep.

GDPR: General Data Protection Regulation

Use Case: A detailed description of system functionality, outlining user interactions and expected behavior.

IEEE 830-1998: The standard for Software Requirements Specification, which defines best practices for structuring SRS documents.

COMET Methodology: A structured approach used for software design and development in NextStep.

UML (Unified Modeling Language): A standard modeling language used for system diagrams in the design of NextStep.

1.5 Document Conventions

This document follows the **IEEE Standard for Software Requirements Specifications (IEEE 830-1998)**. The formatting guidelines used are:

- **Font Style:** Arial
- **Font Size:** 12 for body text, 14 for section headings
- **Spacing:** Single-spaced with 1” margins
- **Text Formatting:** Italics for comments, bold for key terms
- **Section Numbering:** Follows IEEE format with hierarchical numbering (e.g., 1.1, 1.2, etc.)

1.6 References and Acknowledgments

IEEE Software Requirements Specification Standard (IEEE 830-1998)

React Official Documentation

Node Official Documentation

PostgreSQL Database Management Guidelines

Special thanks to Prof. Jayprakash Lalchandani for guidance and feedback

2 Overall Description

2.1 Product Overview

The **NextStep** Application is a web-based platform designed to bridge the gap between job seekers, students, and available opportunities. It serves as a one-stop solution for individuals looking for jobs, educational programs, government support, and health services. The application integrates external job boards, government databases, and educational institutions to ensure users receive real-time, location-based recommendations to enhance their career and education prospects.

Stages:
User Interface
Authentication System
    Opportunities module
    Education module
    Government policy module
    Health Services module
Database and Recommendations

The system is structured into multiple components to ensure a seamless user experience:

1. User Interface (UI)

- A modern, intuitive web-based UI for easy navigation.
- Designed for desktop and mobile devices to maximize accessibility.
- Minimalistic yet functional design with search filters and category-based navigation.

2. Authentication System

- **User Registration & Login:** Secure login using email/password.
- **Profile Management:** Users can update their personal details, location, and preferences.

3. Opportunities Module (Jobs)

- **Area-Based Job Listings:** Users receive real-time job recommendations based on their location.
- **Job preferences:** Users can see preferences of the job they are interested in.

4. Education Module (Courses and Institutes)

- **Area-Based Education Opportunities:** Lists universities, colleges, and training programs based on location.

- **Accreditation Verification:** Ensures that listed institutions are **government-recognized**.

5. Government Policy Module

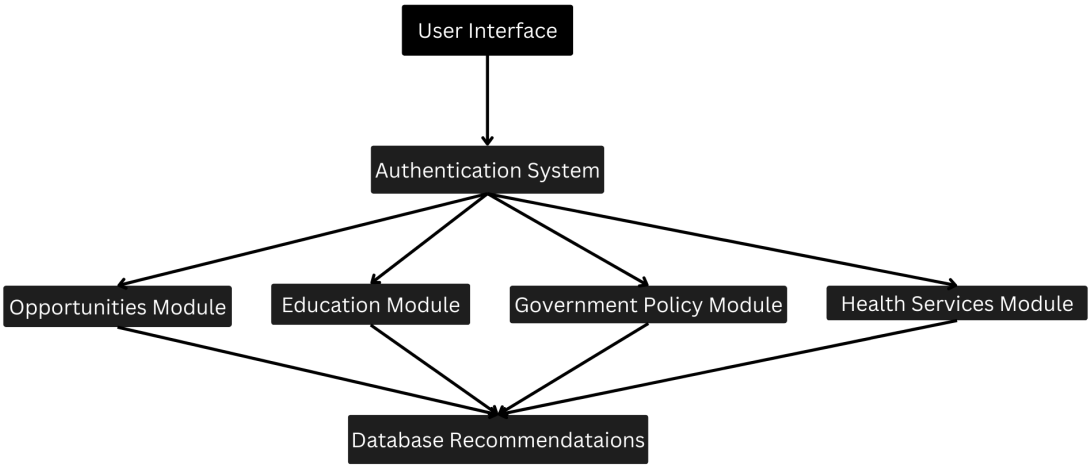
- Displays government policies on employment, education, financial aid, and skill development programs.
- Users can check for grants, scholarships, and subsidies applicable to their region.
- Policy updates are fetched from official government websites.

6. Health Services Module (Hospitals, Clinics, and Pharmacies)

- Lists healthcare facilities, insurance schemes, and wellness programs.
- Integration with government health policies to guide users on free/subsidized healthcare programs.
- List of all the facilities in their region.

7. Database & Recommendation Engine

- **User Data Storage:** Stores profile details, job applications, education history, and preferences.



2.2 Product Functionality

The major functionalities of the NextStep Application include:

- **User Registration & Authentication:** Migrants can securely register for an account and log in to personalize their experience.
- **Area-wise Job & Education Opportunities:** The platform offers tailored recommendations on jobs, schools, health services, and government policies based on where the user is located.
- **Government Policies & Health Services Access:** The users can view the health services and Government policies
- **Interactive Dashboard:** The platform features a dashboard that visually presents available opportunities in different regions, making it easy to track what's available and trending
- **Search and Filter Options for Opportunities:** The user will be able to see the opportunities in the region where he/she is residing.
- **Profile Management & Opportunities:** Managing the user's profile for better recommendations to them in their area where they live.

2.3 Design and Implementation Constraints

2.3.1 Technology Stack:

- **Frontend:** The application will use **React.js** for a responsive and dynamic user interface.
- **Backend: Node.js** with **Express.js** will handle API requests, authentication, and business logic.
- **Database:**
    - **MongoDB** will be used for handling **dynamic data**, ensuring **rapid retrieval and scalability** for high-traffic scenarios.

2.3.2 Software Development Methodology:

- The project follows the **COMET Methodology**, ensuring structured modular development and continuous testing.
- System design and documentation will follow the UML Modeling Language for clear representation of use cases, class diagrams, and process workflows.

2.3.3 Security & Performance Constraints:

- **End-to-end encryption (SSL/TLS)** for secure data transmission.
- The system must be **scalable** to support thousands of users simultaneously.

2.4 Assumptions and Dependencies

2.4.1 ASSUMPTIONS

2.4.1.1 User Behavior & System Usage

- Users will enter valid and accurate personal details during registration (e.g., correct location, education level, job preferences).
- Users will actively engage with the platform, updating their profiles, applying for jobs, and exploring opportunities.
- Employers and educational institutions will post updated and relevant job listings or education programs.
- Users will be able to understand and use basic search and filter functionalities for jobs, education, and policies.

2.4.1.2 Data Accuracy & Availability

- Job postings and educational opportunities are assumed to be up to date, and providers are responsible for removing expired listings.
- Government policies and health services data are assumed to be accurate and regularly updated by relevant agencies.
- The application assumes job providers and educational institutions will verify the authenticity of job listings and admissions.
- Users are assumed to be eligible for the jobs or education programs they apply for based on the system's filtering criteria.

2.4.1.3 Security & Privacy Considerations

- Users are responsible for keeping their credentials secure, and the platform assumes no fraudulent activity.
- Personal data will be stored securely, and users assume responsibility for updating or deleting their profiles if needed.
- The platform assumes no unauthorized access to government policy databases or sensitive employment data.

2.4.1.4 Technical & Infrastructure Assumptions

- The system will be hosted on a reliable cloud infrastructure to ensure high availability and scalability.
- The platform assumes minimal downtime and that system updates will not disrupt user activities significantly.
- The application assumes that React.js (frontend) and Node.js (backend) will continue to be supported frameworks.
- MongoDB databases will be regularly backed up to prevent data loss.
- The system assumes a moderate-to-high traffic load, with performance optimizations in place for scaling.

2.4.1.5 User Device & Browser Compatibility

- The platform assumes users will access the system via modern web browsers (Chrome, Firefox, Edge, Safari).
- The system will be optimized for both desktop and mobile devices, assuming users will use mobile devices for quick job searches and applications.
- Users are assumed to have basic digital literacy to navigate the platform effectively.

2.4.2 DEPENDENCIES

2.4.2.1 System Maintenance & Monitoring

- Logging & Monitoring Tools: The platform will use tools like Prometheus, Grafana, or Datadog for performance tracking.
- Bug Tracking & Issue Management: Taiga or GitHub Issues will be used for reporting and fixing software bugs.
- Automated Backups: Regular database backups to prevent data loss and ensure business continuity.

2.4.2.2 Data & External Services

- Job Listings & Education Data: Collected from trusted job portals and universities.
- Government Policies & Health Services Data: Sourced from official government websites.

2.4.2.3 Software & Technology

- **Frontend:** Built using React.js (for UI) and Material UI (for styling).
- **Backend:** Uses Node.js with Express.js (for handling user requests).
- **Database:**MongoDB (for saving logs and tracking user activity).
- **Authentication:** Uses JWT (for login security)

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The NextStep Application will feature a modern, intuitive web-based interface designed for ease of use and accessibility. The key components of the user interface include:

1. **Registration Screen:**
- Login Name - User has to enter their name
- Email ID- User has to enter their Email-ID
- Password- User has to enter password
- Location- User has to enter their location
- Gender- User has to enter their gender
- Date of Birth- User has to enter their date of birth

2. **Login Screen:**
- Login Name- Enter the name
- Password- Enter the password

3. **Dashboard:**
- Displays job opportunities, education programs, healthcare facilities, and government policies based on the user's location.

4. **Search & Filters:**
- Users can refine searches for jobs, education, healthcare facilities, and government policies based on their location, eligibility, and preferences.

5. **Profile Management:**
- Users can update their details, preferences, and saved opportunities.

6. **Opportunity Details Page:**
- Displays comprehensive information about selected job, education, and healthcare opportunities.
- Provides details on government policies related to employment, education, and health.

7. **Job Interface:**
- Lists available job opportunities for industry, salary range, remote/on-site options, and experience level.
- Displays job descriptions, required qualifications, and employer details.
- .Direct users directly to the opportunity and provide them the details

8. **Education Interface:**
- Lists available education programs with filters for course type, institution, location, and eligibility criteria.
- Provides details about admissions, fees, and course duration..

9. **Healthcare Interface:**
- Displays healthcare facilities in the user's region, including hospitals, clinics, and pharmacy providers.
- Provides information on available health facilities, clinics and pharmacies.
- Allows users to filter based on specialties and affordability based on location.

10. **Government Policies Interface:**
- Lists government policies related to employment, education, and healthcare.
- Provides details on benefits, eligibility criteria, and application processes.
- Offers links to official government resources for applying for schemes and benefits.

11. **Links Providing:**
- Providing users with links to different opportunities and services they may not be aware of, including job portals, university admissions, and healthcare resources.

3.1.2 Hardware Interfaces

- The application requires a minimum screen resolution of **800x600** for proper and complete viewing of screens.

3.1.3 Software Interfaces

- Any windows based operating system.
- Postgresql Server Database or MongoDB Server Database

3.2 Functional Requirements

Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks, or functions the system is required to perform. This section details the different product functions in continuation of the general functional requirements.

3.2.1 F1: User Registration and Authentication

- The system shall allow users to register by providing personal details such as name, email, gender, date of birth, place of origin, etc.
- The system shall verify user email upon registration.
- The system shall allow users to log in using a valid email and password.
- The system shall allow users to reset their password if forgotten.
- The system shall store user credentials securely using encryption.

3.2.2 F2: User Profile Management

- The system shall allow users to update their education, employment, and migration details.
- The system shall allow users to delete their profiles if needed.

3.2.3 F3: Government Policies and Health Facilities

- The system shall display relevant government policies based on the user's location.
- The system shall provide information about nearby health facilities and accepted health insurance policies.

3.2.4 F4: Opportunities & Recommendations

- The system shall display employment and education opportunities based on the user's region.

3.2.5 F5: Database Management & Admin Control

- The system shall allow admin users to manage records in all tables (Youngster, Migration, Employment, Education, etc.).
- The system shall allow admin users to create, update, and delete records.

3.3 Use Case Model

3.3.1 Use Case #1: User Registration (U1)

Author: Mohsin Pathan, Aarushi Goel, Annirudh Pratap
Purpose: Allow users to register in the system with personal details.
Requirements Traceability: F1
Priority: High
Preconditions: User must provide valid details (name, email, password, etc.).
Postconditions: The user is successfully registered and stored in the database.
Actors: Youngster, System
Flow of Events:

- **Basic Flow:**
  - User enters registration details.
  - System validates the input.
  - System registers the user and provides login access.
- **Alternative Flow:**
  - If the email already exists, the system prompts an error.
- **Exceptions:**
  - If the details are incorrect, the registration is incomplete.

3.3.2 Use Case #2: Login (U2)

Author: Mohsin Pathan, Aarushi Goel, Annirudh Pratap
Purpose: Allow users to log in securely.
Requirements Traceability: F1
Priority: High
Preconditions: Users must be registered.
Postconditions: User is logged into the system.
Actors: Youngster, System
Flow of Events:

- **Basic Flow:**
  - User enters an email and password.
  - System validates credentials.
  - If valid, the system logs in the user.
- **Alternative Flow:**
  - If incorrect credentials are entered, an error message is displayed.
- **Exceptions:**
  - If the account is locked due to multiple failed attempts, access is restricted.

3.3.3 Use Case #3: Displaying the Opportunities (U3)

Author: Mohsin Pathan, Aarushi Goel, Annirudh Pratap
Purpose: Allow users to apply for jobs or education opportunities.
Requirements Traceability: F4
Priority: High
Preconditions: User must be logged in.
Postconditions: The user successfully applies for an opportunity.
Actors: Youngster, System, Employer/Educational Institute
Flow of Events:

- **Basic Flow:**
  - Users browse available opportunities.
  - User selects an opportunity.
  - The system directs the user to the opportunity and other details.
- **Alternative Flow:**
  - If the user enters the wrong area or opportunity, an error message is displayed.
- **Exceptions:**
  - If the opportunity is already closed, the user cannot apply.

Use Case #4: Update User Profile (U4)

Author: Mohsin Pathan, Aarushi Goel, Annirudh Pratap
Purpose: Allow users to update their personal details.
Requirements Traceability: F2
Priority: Medium
Preconditions: The user must be logged in.
Postconditions: User profile is updated successfully.
Actors: User, System

Flow of Events:

Basic Flow:

1. User navigates to the profile update section.
2. User modifies personal details.
3. System validates the updated data.
4. System saves changes.
5. Confirmation is displayed.

Alternative Flow:

- If an invalid format is entered, an error message is displayed.

Exceptions:

- If a required field is left blank, the update is not processed.

Use Case #5: View Government Policies (U5)

Author: Mohsin Pathan, Aarushi Goel, Annirudh Pratap
Purpose: Allow users to view policies related to employment, education, and migration.
Requirements Traceability: F3
Priority: Medium
Preconditions: None
Postconditions: Users can view relevant government policies.
Actors: User, System

Flow of Events:

Basic Flow:

1. User selects the "Government Policies" section.
2. System fetches policies based on the user's region.
3. System displays relevant policies.

Alternative Flow:

- If no policies exist for the region, a message is displayed.

Exceptions:

- If the system fails to fetch policies, an error message is displayed.

Use Case #6: View Health Facilities (U6)

Author: Mohsin Pathan, Aarushi Goel, Annirudh Pratap
Purpose: Allow users to view health facilities or medical services.
Requirements Traceability: F3
Priority: Medium
Preconditions: User must be logged in.
Postconditions: Application is successfully submitted.
Actors: Youngster, System, Health Facility

Flow of Events:

Basic Flow:

1. Users browse available health facilities.
2. System fetches facilities based on the user's region.
3. System displays relevant facilities.

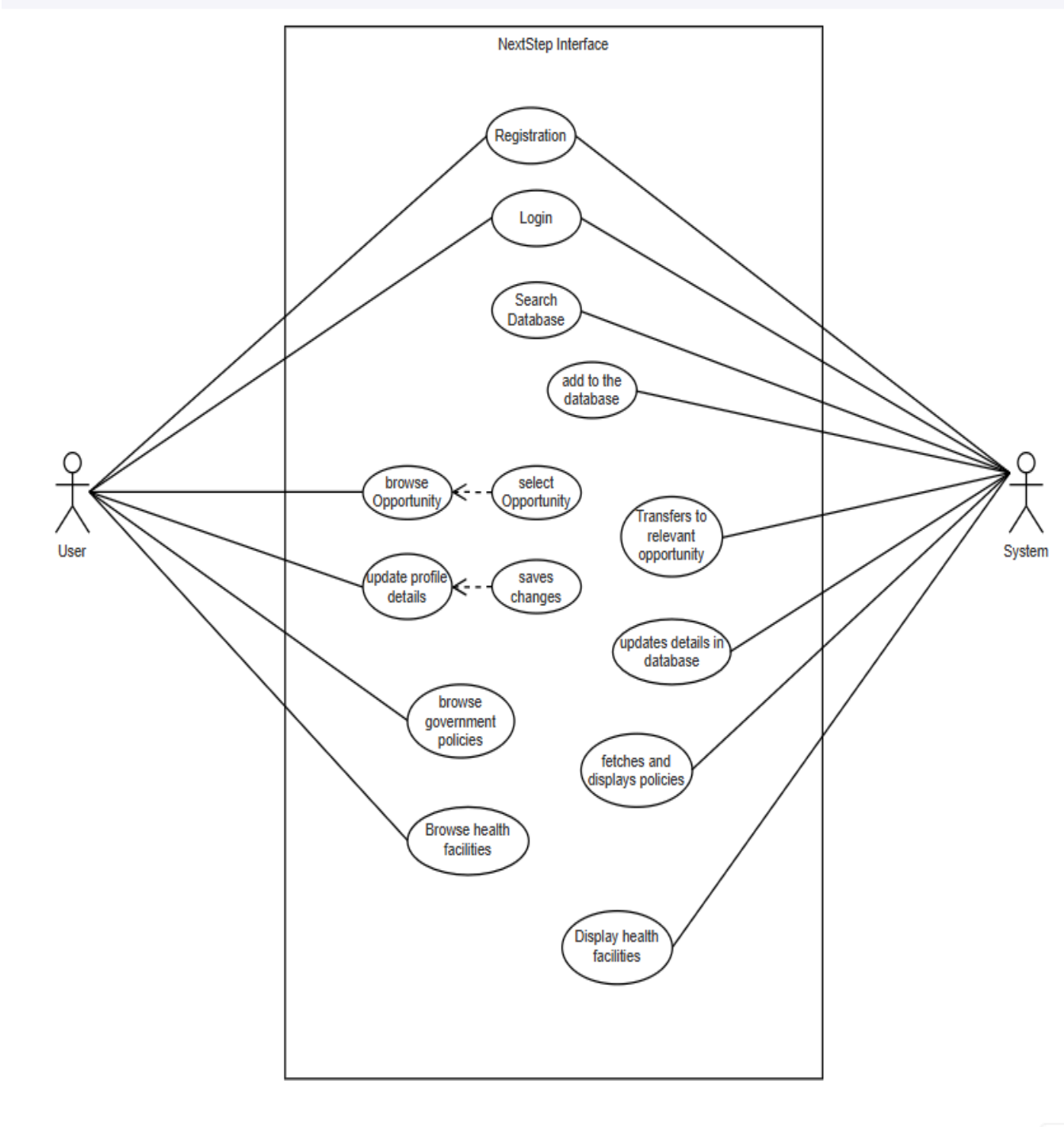
Alternative Flow:

- If the user is ineligible, an error message is displayed.

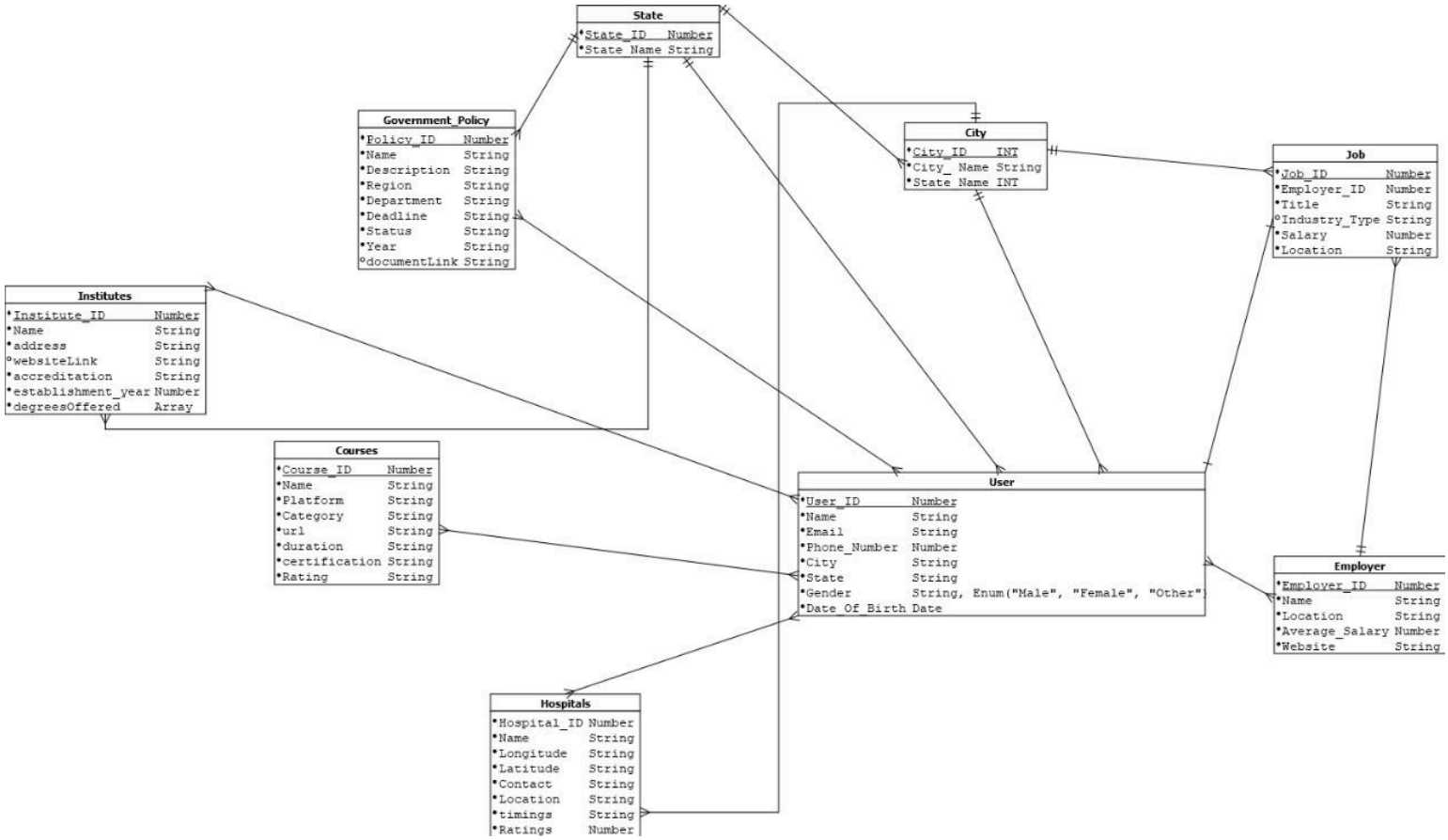
Exceptions:

- If the facility is not accepting applications, the user cannot apply.

USE CASE DIAGRAM



ER DIAGRAM


State
*State_ID Number
*State_Name String
Government_Policy
*Policy_ID Number
*Name String
*Description String
*Region String
*Department String
*Deadline String
*Status String
*Year String
°documentLink String
City
*City_ID INT
*City_ Name String
*State_Name INT
Job
*Job_ID Number
*Employer_ID Number
*Title String
°Industry_Type String
*Salary Number
*Location String
Institutes
*Institute_ID Number
*Name String
*address String
°websiteLink String
*accreditation String
*establishment_year Number
*degreesOffered Array
Courses
*Course_ID Number
*Name String
*Platform String
*Category String
*url String
*duration String
*certification String
*Rating String
User
*User_ID Number
*Name String
*Email String
*Phone_Number Number
*City String
*State String
*Gender String, Enum("Male", "Female", "Other")
*Date_Of_Birth Date
Employer
*Employer_ID Number
*Name String
*Location String
*Average_Salary Number
*Website String
Hospitals
*Hospital_ID Number
*Name String
*Longitude String
*Latitude String
*Contact String
*Location String
*timings String
*Ratings Number

4 Other Non-functional Requirements

4.1 Performance Requirements

P1. The system shall process user registration requests within 2 seconds.

P2. The system shall support up to 1000 concurrent users without performance degradation.

P3. The system shall retrieve user profile data within 1 second.

4.2 Safety and Security Requirements

S1. The system shall encrypt all user passwords using a secure hashing algorithm.

S2. The system shall comply with GDPR and other data protection regulations.

S3. The system shall enforce secure HTTPS communication for all data transmission.

4.3 Software Quality Attributes

4.3.1 Reliability

- The system shall have an uptime of at least 99.9%.
- The system shall automatically recover from minor failures and log all system errors for debugging.

4.3.2 Maintainability

- The system shall use a modular code structure to allow easy updates and debugging.
- The database schema shall be designed to accommodate future expansions without major refactoring.

4.3.3 Usability

- The user interface shall be designed to be intuitive and accessible.
- The system shall provide tooltips and help documentation for new users.

5 Other Requirements

5.1 System Requirements

- The system shall have an internet connection.
- The system shall be able to enable a location setting and GPS.
- The system shall be able to generate an interface.

5.2 User Requirements

- The user should be able to provide their location
- The user shall be able to provide their details for accurate recommendations.
- The details entered by the user should be valid.

Appendix A – Data Dictionary

1. User

Attribute	Data type	Description	Constraints
User_ID	ObjectID (PK)	Unique identifier for the user	Primary Key
Name	String	Name of the employer	Required
email	String	Email of the user	Unique, Required
Phone_no	Number	Phone number of the user	Unique, Required
City	String	City of the user	Required
State	String	State of the user	Required
Gender	String	Gender of the user	Required, Enum ("Male", "Female", "Other")
Date of birth	Date	Date of Birth of the user	Required

2. State

Attribute	Data type	Description	Constraints
State_ID	ObjectID (PK)	Unique identifier for the state	Primary Key
State_Name	String	Name of the state	Unique, Required

3. City

Attribute	Data type	Description	Constraints
City_ID	ObjectID (PK)	Unique identifier for the city	Primary Key
City_Name	String	Name of the city	Required
State_ID	ObjectID (Reference)	Unique identifier for the state	Required, Foreign Key

4. Job

Attribute	Data type	Description	Constraints
Job_ID	ObjectID (PK)	Unique identifier for the job	Primary Key
Employer_ID	ObjectID (Reference)	Identifier for the employer	Foreign Key, Required
Title	String	Job Title	Required
Industry_Type	String	Type of the Industry	Required
Salary	Number	Salary of the job	Required, positive
Location	String	Location where the Job is present	Required

5. Employer

Attribute	Data type	Description	Constraints
Employer_ID	ObjectID (PK)	Unique identifier for the employer	Primary Key
Name	String	Name of the employer	Required, Unique
Location	String	Unique identifier for the state	Required
Average_Salary	Number	Average salary provided by the employer	Required, positive
Website	String	Website of the employer	Unique

6. Government Policy

Attribute	Data type	Description	Constraints
Policy_ID	ObjectID (PK)	Unique identifier for the Government Policy	Primary Key
Name	String	Name of the policy	Required, Unique
Description	String	Description of the policy	Required
Region	String	Where the policy is active	Required
Department	String	Department of the policy	Required
Deadline	String	Deadline of the policy	Required
Status	String	Status of the policy	Required
Year	String	Year of the policy	Required
documentLink	String	Link to the policy	Unique

7. Courses

Attribute	Data type	Description	Constraints
Course_ID	ObjectID (PK)	Unique identifier for the Course	Primary Key
Name	String	Name of the course	Required
Platform	String	Platform of the course	Required
Category	String	Domain of the course	Required
url	String	link	Unique, Required
duration	String	Duration of the policy	Required
certification	Boolean	Is certificate given or not	Required
rating	Number	rating of the court	Required, Between 0-5

8. Institutes

Attribute	Data type	Description	Constraints
Institute_ID	ObjectID (PK)	Unique identifier for the Institute	Primary Key
Name	String	Name of the institute	Required
address	String	Address of the institute	Required
websiteLink	String	website link of the institute	Unique
accreditation	String	accreditation of the institute	Required
establishment_year	Number	year of establishment	Required, positive
degreesOffered	Array	number of degrees	Required

9. Hospitals

Attribute	Data type	Description	Constraints
Hospital_ID	ObjectID (PK)	Unique identifier for the health facility	Primary Key
Name	String	Name of the health facility	Unique, Required
Longitude	String	longitude of the facility	Required
Latitude	String	latitude of the facility	Required
Contact	String	Contact number of the facility	Required, Unique
Location	String	Location of the Facility	Required
Timings	String	Timings of the facility	Required
Rating	Number	The rating of the facility	Required, Between 0-5

Appendix B - Group Log

Epics Reports:
https://api.taiga.io/api/v1/epics/csv?uuid=849ccc3a9cd2425187e27c1fa13e30b6

User Stories Reports:
https://api.taiga.io/api/v1/userstories/csv?uuid=a6542c33500a4fec8af5f130bb68e7af

Tasks Reports:
https://api.taiga.io/api/v1/tasks/csv?uuid=c9ee1dbdd5d24e05a5338dc60a26b5ce

Issues Reports:
https://api.taiga.io/api/v1/issues/csv?uuid=3a0508ba6fa941d4a0179eb3aca0c41b

User Stories

and

Sprints

for

NextStep

Prepared by

Group ID: 21

Aarushi Goel	**202412002**	**202412002@daiict.ac.in**
Mohsin Pathan	**202412070**	**202412070@daiict.ac.in**
Annirudh Pratap	**202412008**	**202412008@daiict.ac.in**

Instructor: Prof. Jayprakash Lalchandani

Course: Software Engineering (IT 632)

Date: 11th March 2025

1 User Stories

User Authentication & Profile Management

User Story 1: User Registration & Login

● As a user, I want to create and manage my account securely so that I can access personalized features.

Sub-Stories:

✔ As a user, I can register using my email and password and other details.
✔ As a user, I can login and log out securely.
✔ As a user, I get an error message if I enter incorrect login credentials.

User Story 2: User Profile Management

● As a user, I want to manage my profile so that I can personalize my experience.

Sub-Stories:

✔ As a user, I can update my name and other details.
✔ As a user, I can change my password securely.
✔ As a user, I can delete my account if I no longer want to use the platform.

User Story 3: Location Storage

● As a user, I want to store my location so that I can receive relevant career and education opportunities.

Sub-Stories:

✔ As a user, I can provide my state and city during registration.
✔ As a user, I can update my location in my profile settings.

Education Opportunities

User Story 4: Viewing Educational Institutions

● As a user, I want to see educational institutions near me so that I can explore learning options.

Sub-Stories:

✔ As a user, I can view a list of institutes based on my state/city.
✔ As a user, I can filter institutes by location and other filters. ✔
As a user, I can visit the institute's website for more details.

User Story 5: Course Details

● As a user, I want to see details about courses offered by an institution so that I can find relevant programs.

Sub-Stories:

✔ As a user, I can view course names, duration, and eligibility criteria.
✔ As a user, I can see admission deadlines and tuition fees.
✔ As a user, I can visit the institution's website for application details.

Job Listings

User Story 6: Viewing Job Listings

● As a user, I want to see job opportunities in my location so that I can apply for relevant positions.

Sub-Stories:

✔ As a user, I can view a list of jobs based on my city/state.
✔ As a user, I can filter jobs by my location and other filters.
✔ As a user, I can visit the employer's website to apply for a job.

User Story 7: Job Details Page

● As a user, I want to see job details so that I can determine if the job is suitable for me.

Sub-Stories:

✔ As a user, I can view job descriptions, required skills, and salary range.
✔ As a user, I can check whether the job is remote, hybrid, or on-site.
✔ As a user, I can see employer details and company reviews.

Health Facilities

User Story 8: Viewing Health Facilities

● As a user, I want to find hospitals and clinics near me so that I can access healthcare services.

Sub-Stories:

✔ As a user, I can view a list of hospitals and clinics in my city/state.
✔ As a user, I can filter health facilities by type (hospital, clinic, pharmacy).
✔ As a user, I can view contact details of healthcare providers.

User Story 9: Searching for Health Services

● As a user, I want to search for specific health services so that I can find the right facility.

Sub-Stories:

✔ As a user, I can search for timings, ratings.
✔ As a user, I can filter health facilities in my region.
✔ As a user, I can check the hospital location easily.

Government Policies

User Story 10: Viewing Government Policies

● As a user, I want to see relevant government policies so that I can benefit from available schemes.

Sub-Stories:

✔ As a user, I can view policies related to education, employment, and healthcare.
✔ As a user, I can filter policies by location and other attributes.
✔ As a user, I can visit government websites for detailed information.

User Story 11: Policy Details

● As a user, I want to see the details of a government policy so that I can understand its benefits.

Sub-Stories:

✔ As a user, I can see eligibility criteria, benefits, and deadlines.
✔ As a user, I can find links to the official government website.
✔ As a user, I can check if the policy applies to my location.

User Story 12: Search & Filters

● As a user, I want to search and filter results efficiently so that I can find relevant

information quickly.

Sub-Stories:

✔ As a user, I can search for institutes, jobs, and health facilities using my location.
✔ As a user, I can apply filters to narrow down search results based on my preferences.

User Story 13: Favorites & Saved Listings

● As a user, I want to save my favorite jobs, institutions, and health facilities based on my location so that I can view them later.

Sub-Stories:

✔ As a user, I can add jobs, institutes, and hospitals to my saved list.
✔ As a user, I can view my saved items from a dedicated section.
✔ As a user, I can remove items from my saved list.

Testing & Deployment

User Story 14: Application Testing

● As a developer, I want to test my application so that I can ensure it works smoothly.

Sub-Stories:

✔ As a developer, I can run unit tests for backend APIs.
✔ As a developer, I can perform UI testing for the frontend.
✔ As a developer, I can check for bugs and performance issues.

2 Taiga.io

Taiga.io is an Agile project management tool designed to streamline software development workflows. It allows teams to create projects, set a name, description, logo, and add team members, just like we did.

Once the project is set up, we can create multiple sprints based on the project's needs and timelines. Each sprint can contain user stories, which represent key features or functionalities. Within each user story, we can add different tasks, breaking down the work into smaller, manageable pieces.

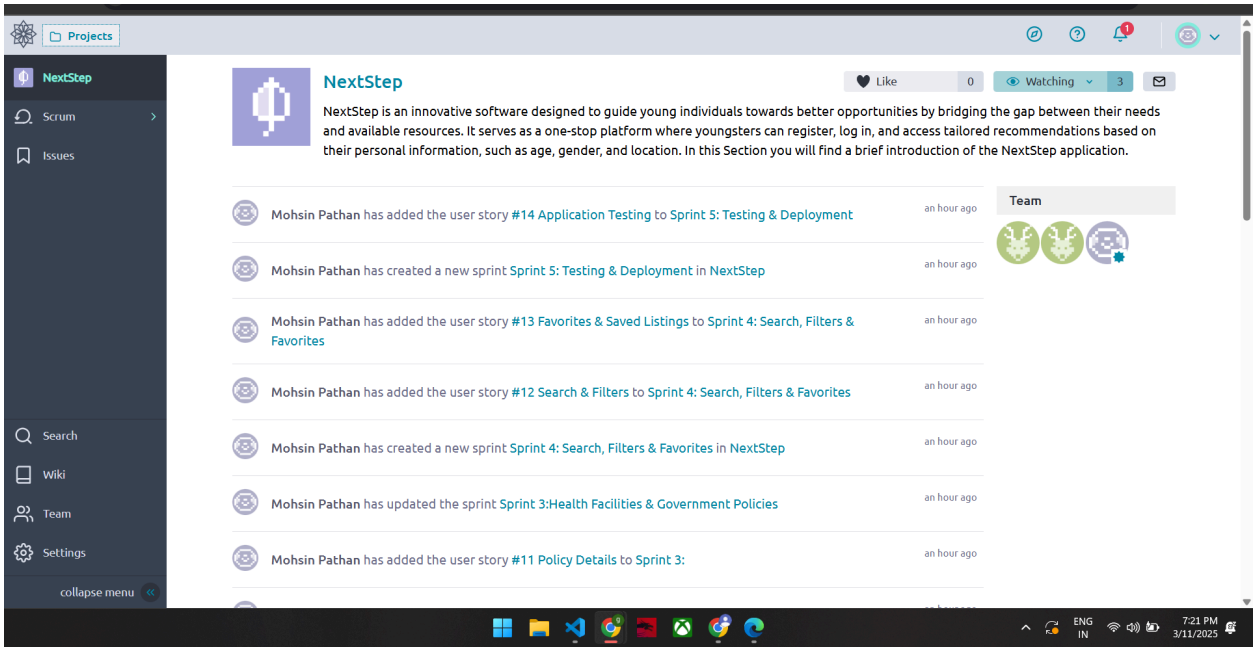
Taiga provides a status tracking system where tasks and user stories can be moved through various stages:

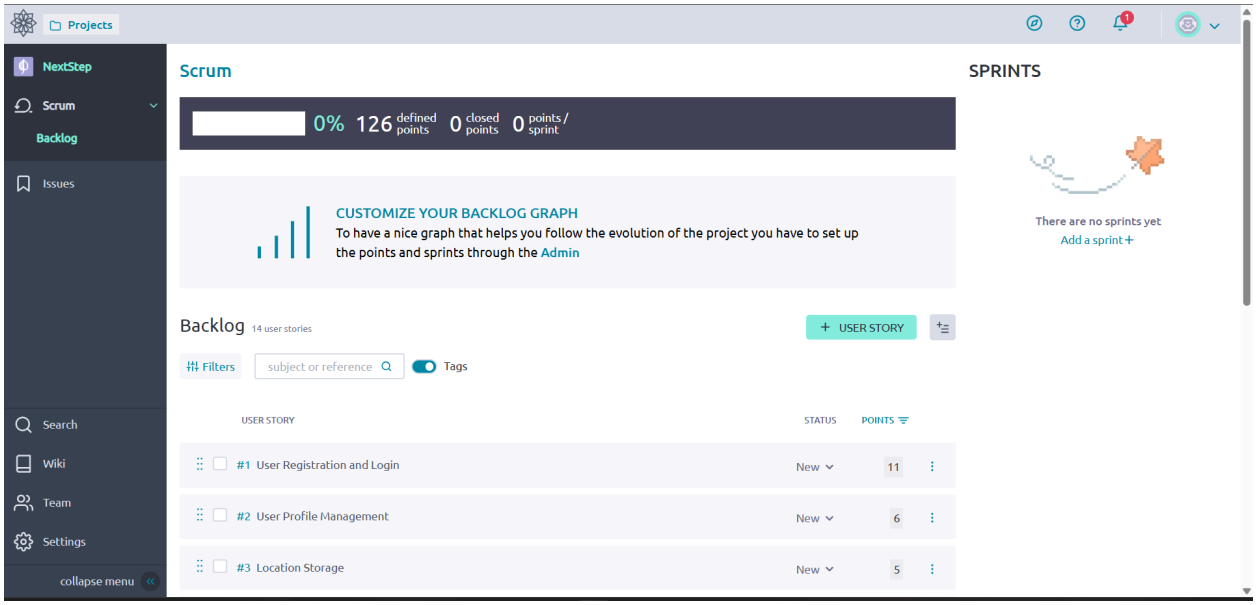
New
Ready
In Progress
Ready for Testing
Done
Archived

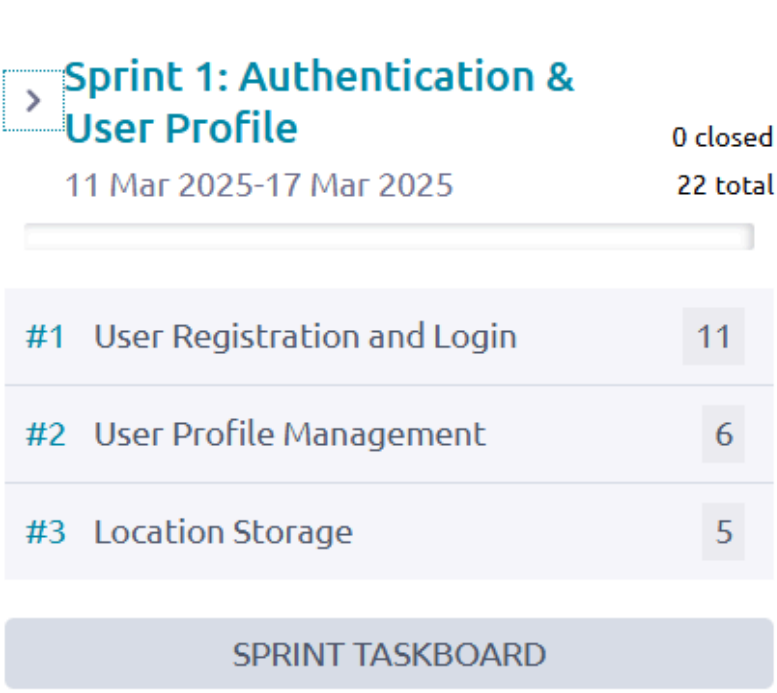
Additionally, we can assign story points to each user story based on the effort required, whether it's design, frontend, backend, UX, or other aspects. This helps in tracking progress and understanding how many points are completed versus how many are remaining.

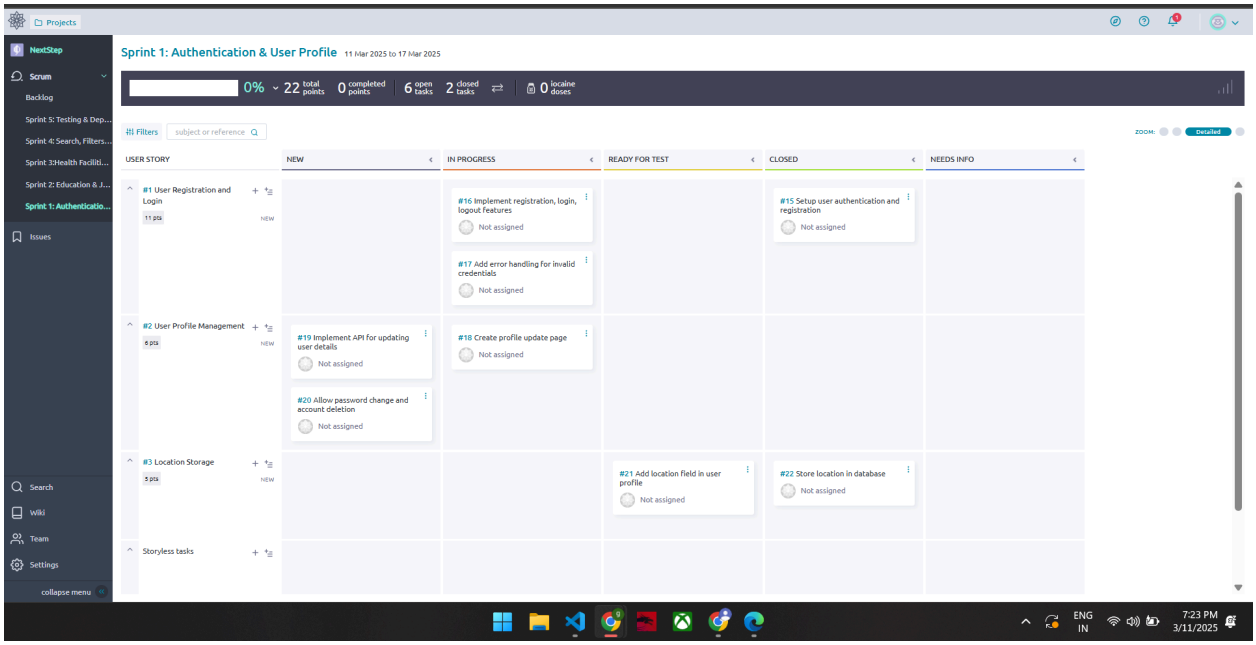
Taiga ensures efficient project tracking, team collaboration, and Agile development transparency, making it easier to manage our NextStep project effectively.

Screenshots:

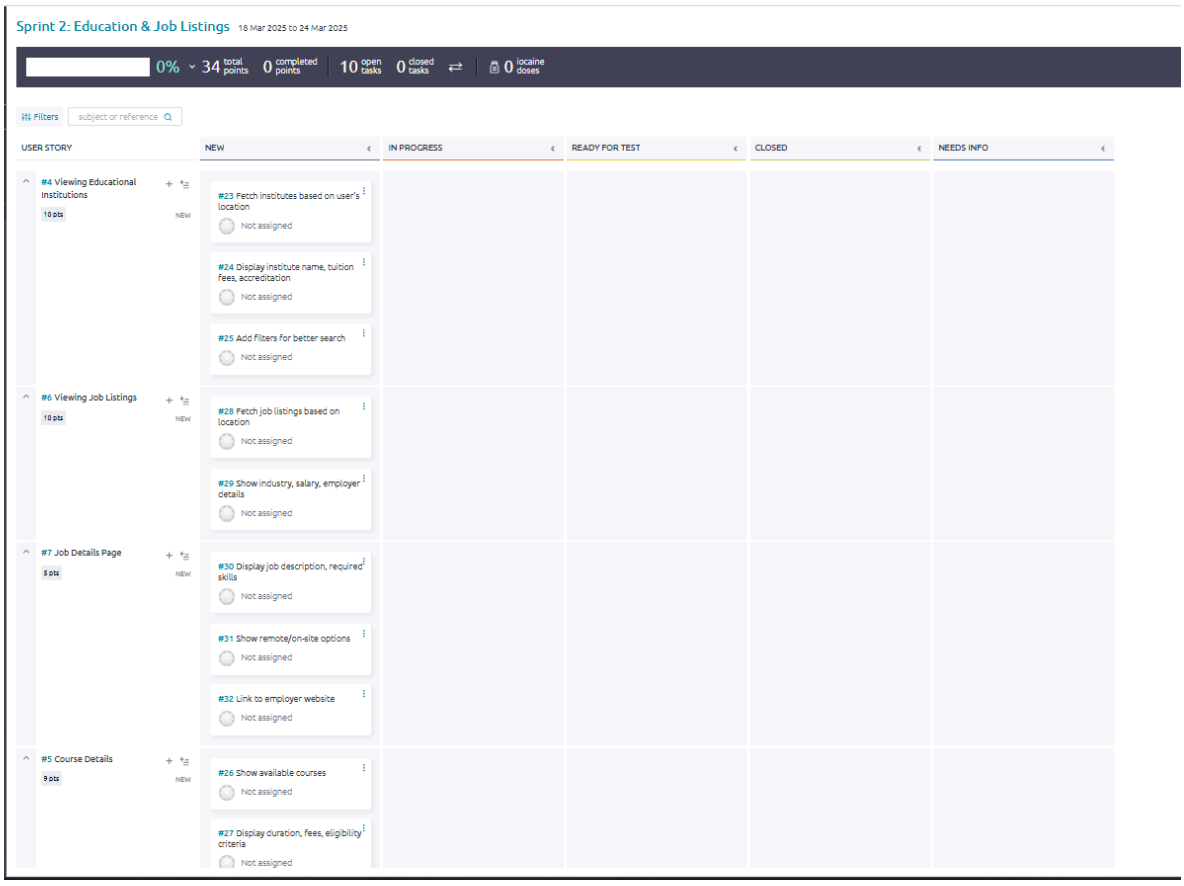




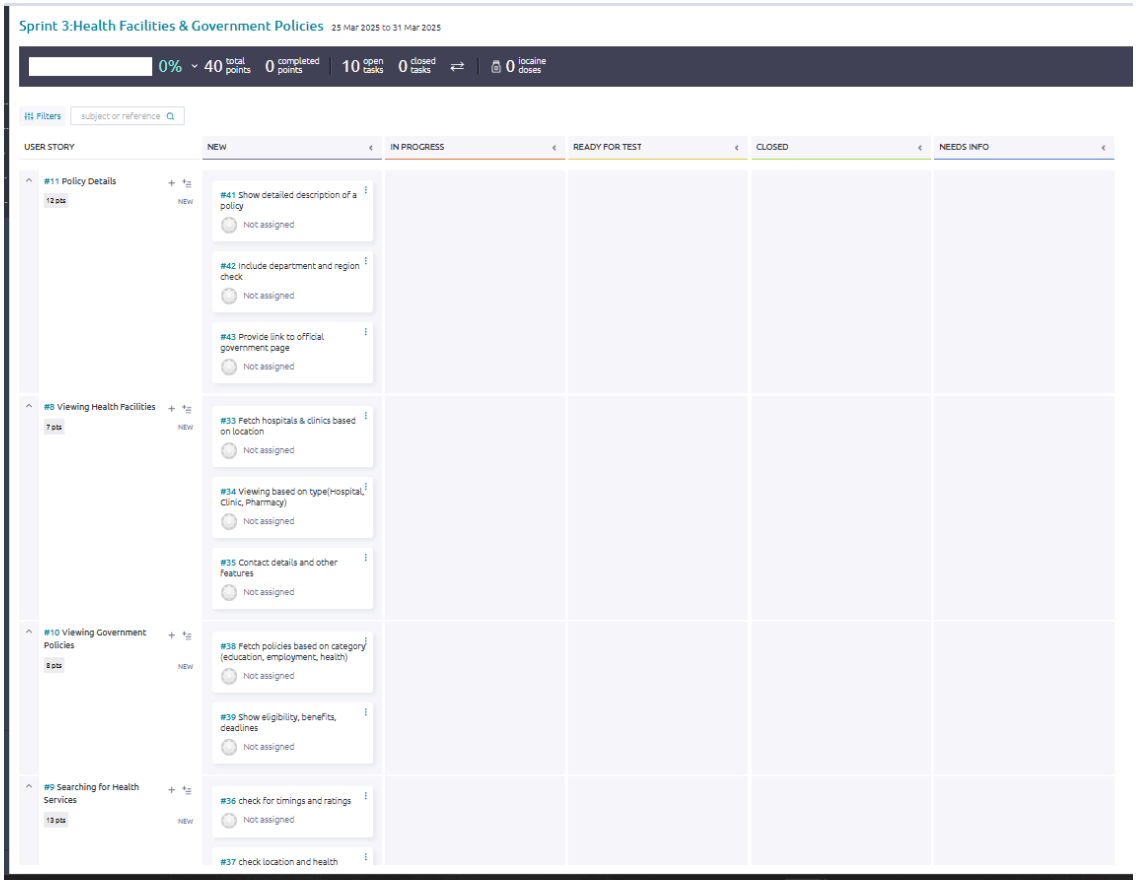

Sprint 1: Authentication & User Profile
0 closed
11 Mar 2025-17 Mar 2025
22 total
#1 User Registration and Login 11
#2 User Profile Management 6
#3 Location Storage 5
SPRINT TASKBOARD

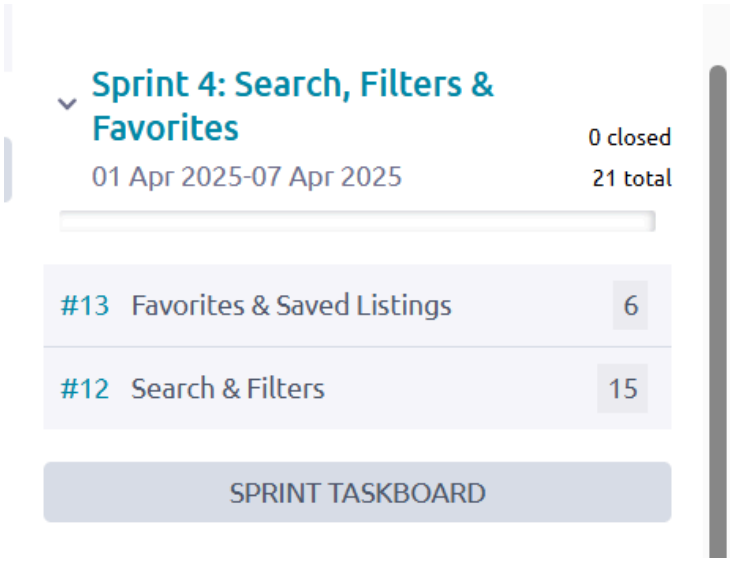

Projects
NextStep
Scrum
Backlog
Issues
Search
Wiki
Team
Settings
collapse menu
Sprint 1: Authentication & User Profile
11 Mar 2025 to 17 Mar 2025
0%
22 total points
0 completed points
6 open tasks
2 closed tasks
0 locaine doses
Filters
subject or reference
ZOOM:
Detailed
USER STORY
NEW
IN PROGRESS
READY FOR TEST
CLOSED
NEEDS INFO
#1 User Registration and Login
11 pts
NEW
#16 Implement registration, login, logout features
Not assigned
#17 Add error handling for invalid credentials
Not assigned
#15 Setup user authentication and registration
Not assigned
#2 User Profile Management
6 pts
NEW
#19 Implement API for updating user details
Not assigned
#20 Allow password change and account deletion
Not assigned
#18 Create profile update page
Not assigned
#3 Location Storage
5 pts
NEW
#21 Add location field in user profile
Not assigned
#22 Store location in database
Not assigned
Storyless tasks
ENG
IN
7:23 PM
3/11/2025

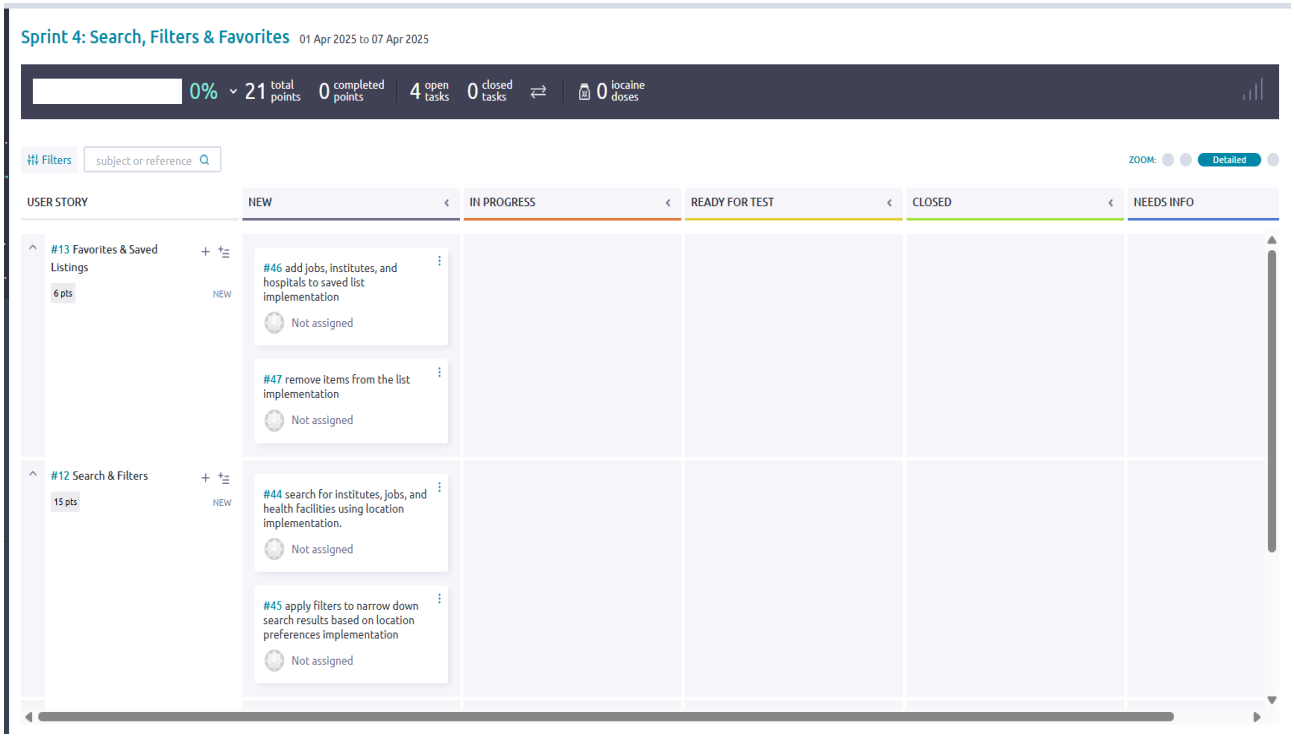

Sprint 2: Education & Job Listings
18 Mar 2025-24 Mar 2025
0 closed
34 total
#4 Viewing Educational Institutions 10
#6 Viewing Job Listings 10
#7 Job Details Page 5
#5 Course Details 9
SPRINT TASKBOARD


Sprint 2: Education & Job Listings 18 Mar 2025 to 24 Mar 2025
0% 34 total points 0 completed points 10 open tasks 0 closed tasks 0 iocaine doses
Filters
subject or reference
USER STORY
NEW
IN PROGRESS
READY FOR TEST
CLOSED
NEEDS INFO
#4 Viewing Educational Institutions
10 pts
NEW
#23 Fetch institutes based on user's location
Not assigned
#24 Display institute name, tuition fees, accreditation
Not assigned
#25 Add filters for better search
Not assigned
#6 Viewing Job Listings
10 pts
NEW
#28 Fetch job listings based on location
Not assigned
#29 Show industry, salary, employer details
Not assigned
#7 Job Details Page
5 pts
NEW
#30 Display job description, required skills
Not assigned
#31 Show remote/on-site options
Not assigned
#32 Link to employer website
Not assigned
#5 Course Details
9 pts
NEW
#26 Show available courses
Not assigned
#27 Display duration, fees, eligibility criteria
Not assigned

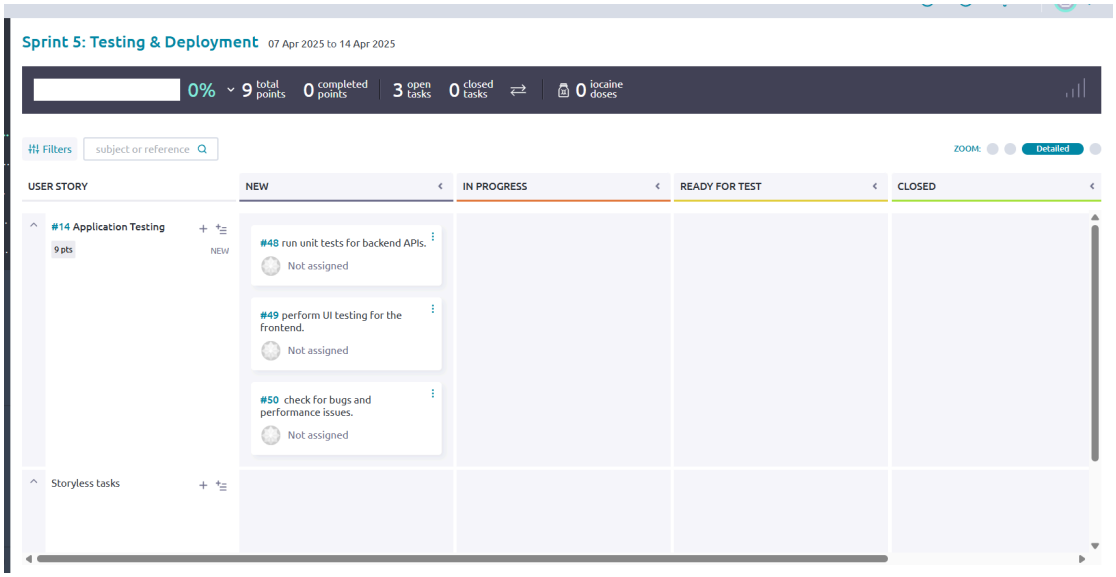

Sprint 3:Health Facilities & Government Policies
25 Mar 2025-31 Mar 2025
0 closed
40 total
#11 Policy Details 12
#8 Viewing Health Facilities 7
#10 Viewing Government Policies 8
#9 Searching for Health Services 13
SPRINT TASKBOARD


Sprint 3:Health Facilities & Government Policies 25 Mar 2025 to 31 Mar 2025
0% 40 total points 0 completed points 10 open tasks 0 closed tasks 0 iocaine doses
Filters subject or reference
USER STORY
NEW
IN PROGRESS
READY FOR TEST
CLOSED
NEEDS INFO
#11 Policy Details
12 pts
NEW
#41 Show detailed description of a policy
Not assigned
#42 Include department and region check
Not assigned
#43 Provide link to official government page
Not assigned
#8 Viewing Health Facilities
7 pts
NEW
#33 Fetch hospitals & clinics based on location
Not assigned
#34 Viewing based on type(Hospital, Clinic, Pharmacy)
Not assigned
#35 Contact details and other features
Not assigned
#10 Viewing Government Policies
8 pts
NEW
#38 Fetch policies based on category (education, employment, health)
Not assigned
#39 Show eligibility, benefits, deadlines
Not assigned
#9 Searching for Health Services
13 pts
NEW
#36 check for timings and ratings
Not assigned
#37 check location and health


Sprint 4: Search, Filters & Favorites
0 closed
01 Apr 2025-07 Apr 2025
21 total
#13 Favorites & Saved Listings 6
#12 Search & Filters 15
SPRINT TASKBOARD


Sprint 4: Search, Filters & Favorites 01 Apr 2025 to 07 Apr 2025
0% 21 total points 0 completed points 4 open tasks 0 closed tasks 0 iocaine doses
Filters
subject or reference
ZOOM: Detailed
USER STORY
NEW
IN PROGRESS
READY FOR TEST
CLOSED
NEEDS INFO
#13 Favorites & Saved Listings
6 pts
NEW
#46 add jobs, institutes, and hospitals to saved list implementation
Not assigned
#47 remove items from the list implementation
Not assigned
#12 Search & Filters
15 pts
NEW
#44 search for institutes, jobs, and health facilities using location implementation.
Not assigned
#45 apply filters to narrow down search results based on location preferences implementation
Not assigned


Sprint 5: Testing & Deployment
07 Apr 2025-14 Apr 2025
0 closed
0 total
#14 Application Testing
9
SPRINT TASKBOARD


Sprint 5: Testing & Deployment
07 Apr 2025 to 14 Apr 2025
0%
9 total points
0 completed points
3 open tasks
0 closed tasks
0 iocaine doses
Filters
subject or reference
ZOOM:
Detailed
USER STORY
NEW
IN PROGRESS
READY FOR TEST
CLOSED
#14 Application Testing
9 pts
NEW
#48 run unit tests for backend APIs.
Not assigned
#49 perform UI testing for the frontend.
Not assigned
#50 check for bugs and performance issues.
Not assigned
Storyless tasks

Data

Flow

Diagrams

for

NextStep

Prepared by

Group ID: 21

Aarushi Goel	**202412002**	**202412002@daiict.ac.in**
Mohsin Pathan	**202412070**	**202412070@daiict.ac.in**
Annirudh Pratap	**202412008**	**202412008@daiict.ac.in**

Instructor: Prof. Jayprakash Lalchandani

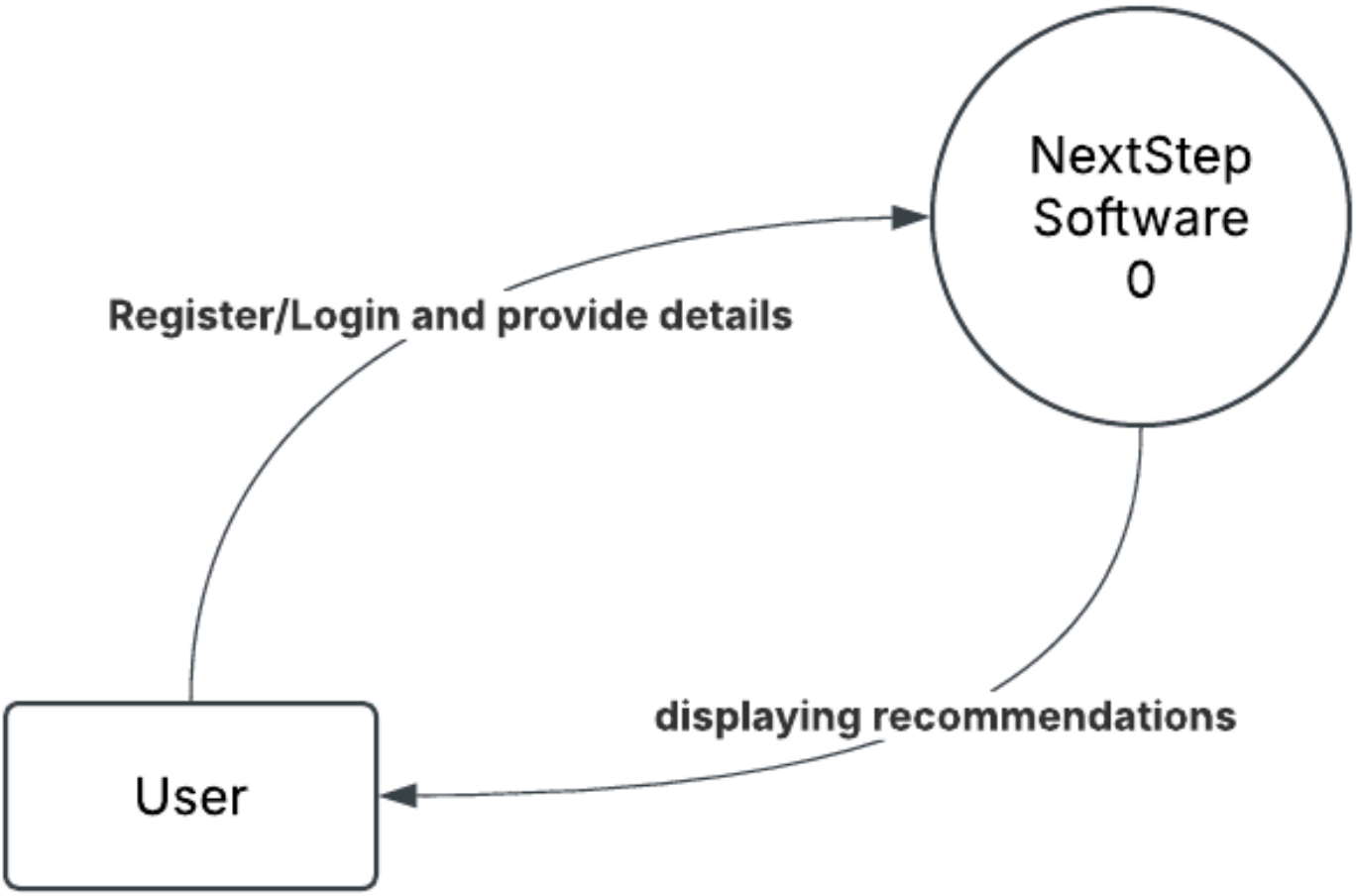
Course: Software Engineering (IT 632)

Date: 13th March 2025

1 Level 0 Diagram

External entity: User

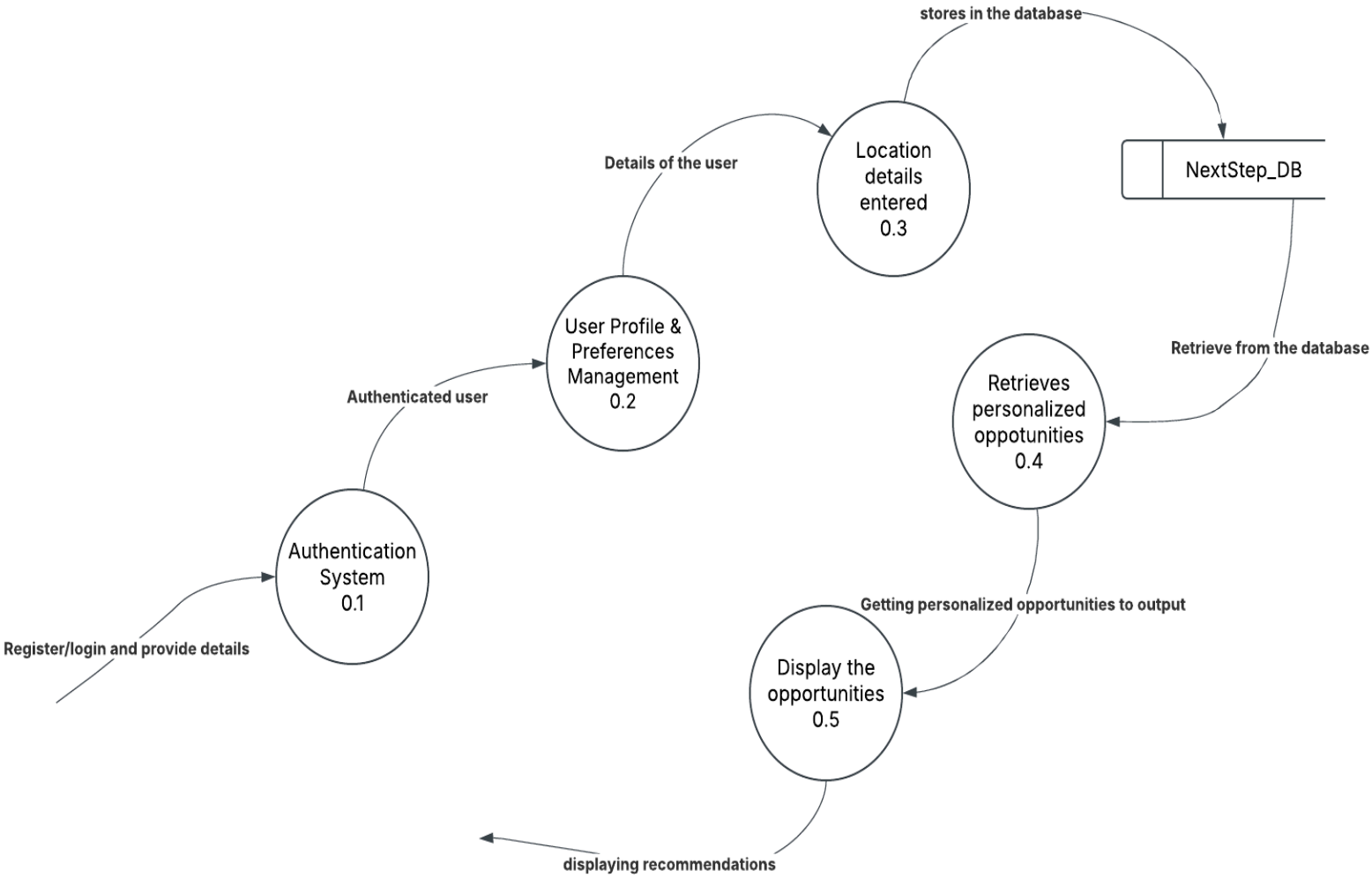
Process: NextStep Software



2 Level 1 Diagram

Process: Authentication system, User profile preferences and management, location details entered, retrieving personalized opportunities, displaying opportunities

Data Store: NextStep_DB



3 Level 2 Diagram

Process:

1. **Authentication system:**
   1.1 Registration
   1.2 Login
   1.3 MongoDB query to check
   1.4 Grants access

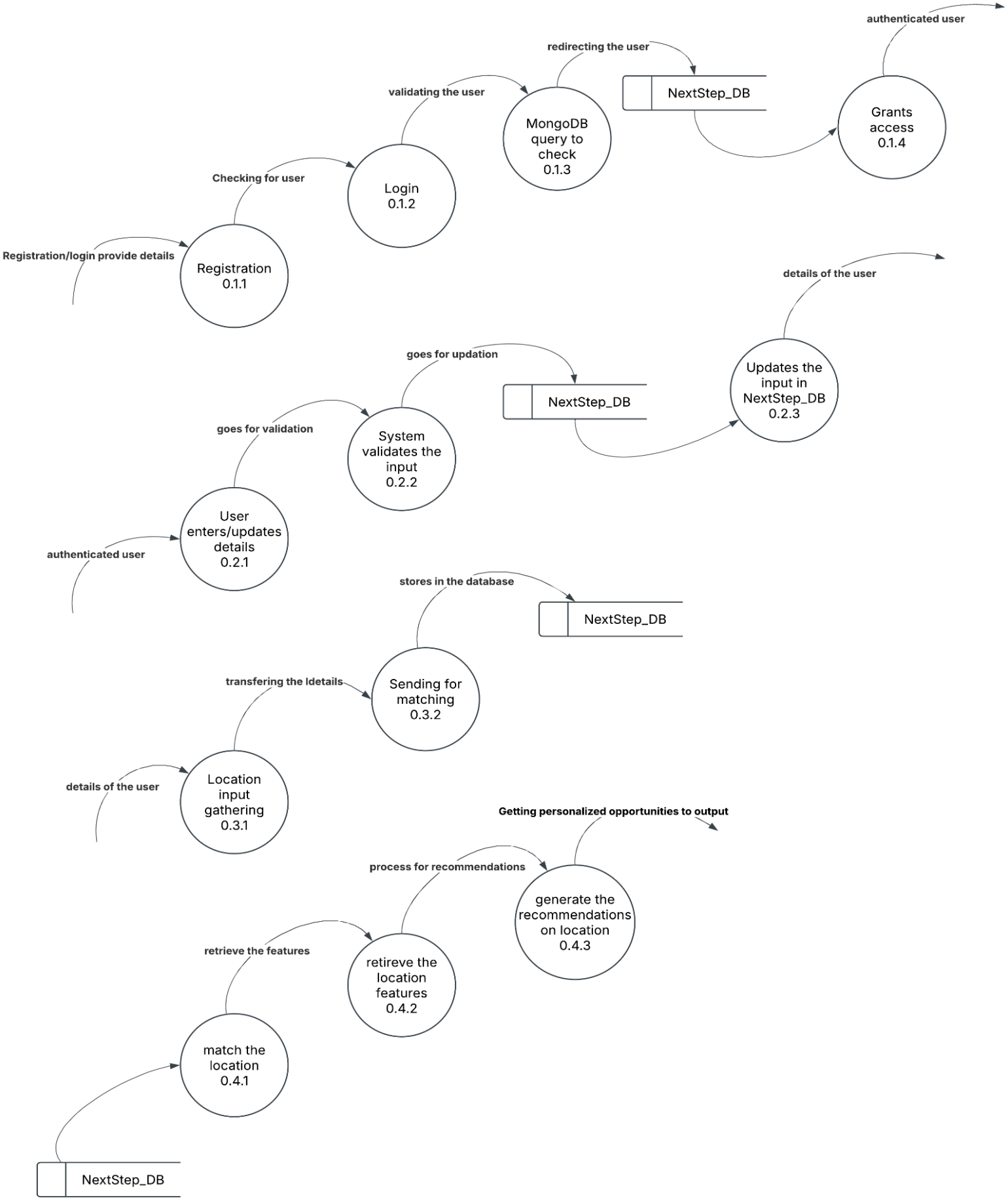
2. **User profile preferences and management:**
   2.1 User enters and updates details
   2.2 System validates the input
   2.3 Updates the details directly in MongoDB

3. **Location details entered:**
   3.1 Location input gathering
   3.2 Sending for matching

4. **Retrieving personalized opportunities:**
   4.1 match the location
   4.2 retrieve location features
   4.3 generate recommendations based on the locations

5. **Displaying opportunities**

Data Store: NextStep_DB


authenticated user
redirecting the user
validating the user
NextStep_DB
MongoDB query to check 0.1.3
Grants access 0.1.4
Checking for user
Login 0.1.2
Registration/login provide details
Registration 0.1.1
details of the user
goes for updation
Updates the input in NextStep_DB 0.2.3
NextStep_DB
goes for validation
System validates the input 0.2.2
User enters/updates details 0.2.1
authenticated user
stores in the database
NextStep_DB
transfering the ldetails
Sending for matching 0.3.2
details of the user
Location input gathering 0.3.1
Getting personalized opportunities to output
process for recommendations
generate the recommendations on location 0.4.3
retrieve the features
retireve the location features 0.4.2
match the location 0.4.1
NextStep_DB

UML Diagrams

for

NextStep

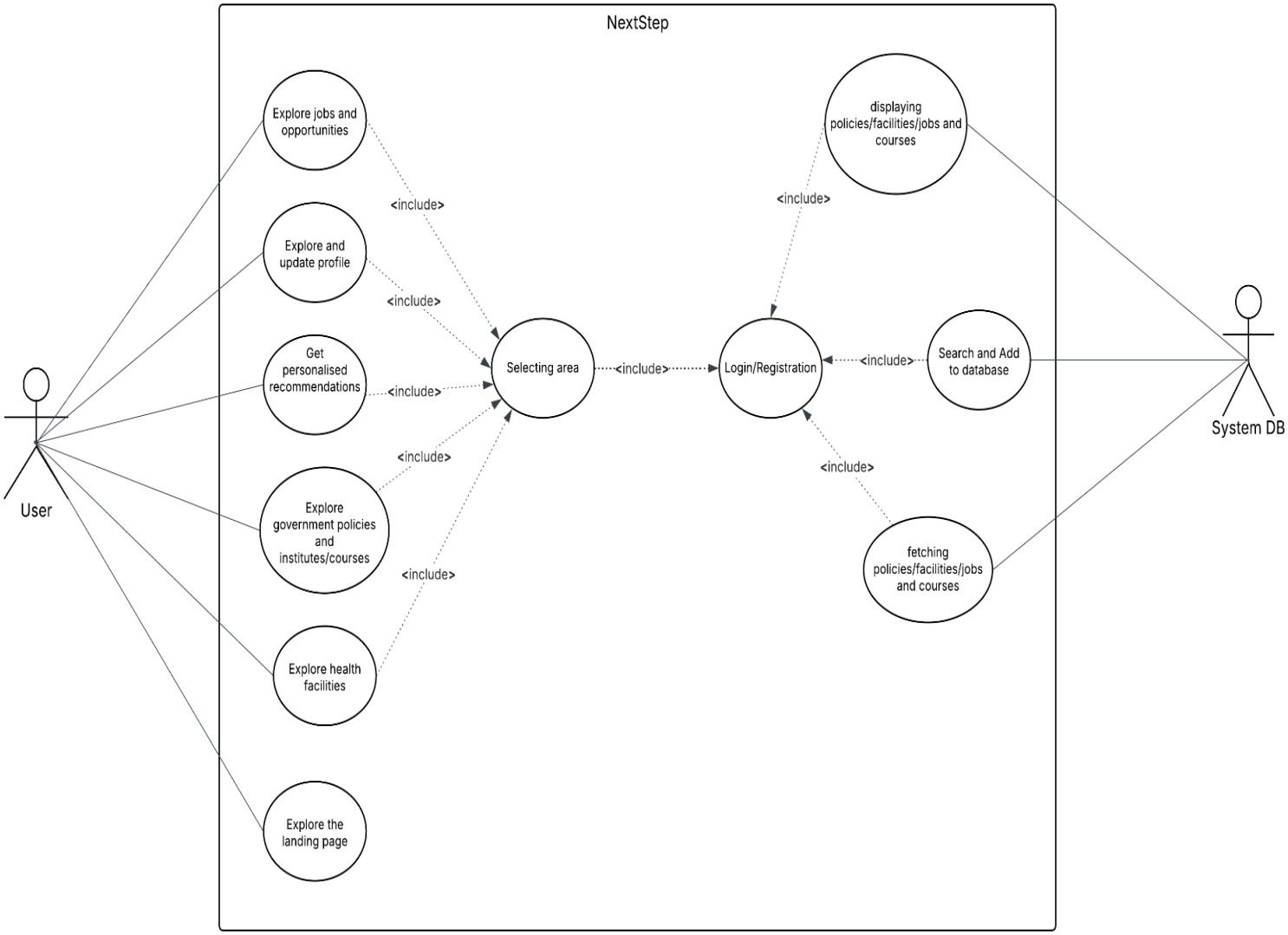
Prepared by

Aarushi Goel	**202412002**	**202412002@daiict.ac.in**
Mohsin Pathan	**202412070**	**202412070@daiict.ac.in**
Annirudh Pratap	**202412008**	**202412008@daiict.ac.in**

Instructor: Jayprakash Lalchandani

Date: 4th April 2025

Use Case Diagram



Textual Use Cases

Use Case 1: Explore the Landing Page

Description: The user views and navigates the homepage of the NextStep application.
Actors: User
Precondition: The user has accessed the application.

Main Scenario:

User	System
1. The user opens the application.	2. The system loads the landing page with available features.
3. The user navigates through the available sections.	4. The system displays relevant content dynamically.

Alternative: None
Exceptional: If the system fails to load the page, it displays an error message.

Use Case 2: Explore and Update Profile

Description: The user can update personal details such as location and interests.
Actors: User
Precondition: The user must be logged in.

Main Scenario:

User	System
1. The user selects the profile section.	2. The system fetches the user's existing profile details.
3. The user updates their information and saves changes.	4. The system validates and updates the profile.

Alternative: The user cancels the update, and no changes are saved.
Exceptional: If validation fails, an error message is displayed.

Use Case 3: Explore Jobs and Opportunities

Description: The user searches and browses job opportunities.
Actors: User
Precondition: The user has an account and is logged in.

Main Scenario:

User	System
1. The user navigates to the jobs section.	2. The system fetches available job listings.
3. The user searches and filters job listings.	4. The system applies filters and displays relevant results.

Alternative: The user resets filters to view all job listings.
Exceptional: If no jobs are found, the system displays a message suggesting profile updates for better recommendations.

Use Case 4: Apply for Jobs

Description: The user applies for a selected job.
Actors: User
Precondition: The user has an updated profile and uploaded a resume.

Main Scenario:

User	System
1. The user selects a job listing.	2. The system displays job details and application options.
3. The user clicks ‘Apply’ and confirms submission.	4. The system processes the application and notifies the employer.

Alternative: The user cancels the application before submission.
Exceptional: If the application fails, the system prompts the user to retry.

Use Case 5: Explore Health Facilities

Description: The user searches for available health facilities in their region.
Actors: User
Precondition: The system has a list of health facilities stored in the database.

Main Scenario:

User	System
1. The user navigates to the health facilities section.	2. The system fetches available health facilities based on the user's location.
3. The user selects a facility for more details.	4. The system displays detailed information about the selected facility.

Alternative: The user searches for facilities in another location.
Exceptional: If no facilities are found, the system suggests nearby regions.

Use Case 6: Explore Government Policies and Courses

Description: The user explores available policies and educational courses.
Actors: User
Precondition: The system contains updated policy and course data.

Main Scenario:

User	System
1. The user navigates to the policies/courses section.	2. The system fetches relevant government policies and educational courses.
3. The user selects a policy/course for more details.	4. The system displays full details.

Alternative: The user filters policies based on eligibility.
Exceptional: If no policies/courses match, the system suggests related options.

Use Case 7: Get Personalized Recommendations

Description: The system suggests jobs and courses based on the user's profile.
Actors: User, System DB
Precondition: The user has provided sufficient profile information.

Main Scenario:

User	System
1. The user accesses the recommendation section.	2. The system fetches user profile data.
3. The system processes data and generates relevant recommendations.	4. The user views and explores suggested jobs/courses.

Alternative: The user updates their profile for better recommendations.
Exceptional: If no recommendations are found, the system suggests adding skills or interests.

Use Case 8: Login/Registration

Description: A user registers or logs into their account.
Actors: User
Precondition: The user must have an internet connection.

Main Scenario:

User	System
1. The user selects login/register.	2. The system prompts for credentials.
3. The user enters credentials.	4. The system validates and grants access.

Alternative: If the user selects 'Forgot Password,' the system sends a reset link.
Exceptional: If credentials are invalid, the system shows an error message.

Use Case 9: Selecting Area

Description: The user selects a specific area to view related data.
Actors: User
Precondition: The system has area-based data.

Main Scenario:

User	System
1. The user chooses a city or region.	2. The system filters and displays area-specific jobs, policies, and opportunities.

Alternative: The user changes the selected area to view different results.
Exceptional: If no data exists for the area, the system suggests the nearest relevant location.

Use Case 10: Displaying Policies, Facilities, Jobs, and Courses

Description: The system fetches and displays relevant data for users.
Actors: System DB
Precondition: The database contains valid records.

Main Scenario:

System
1. The system retrieves policies, jobs, courses, and health facilities.
2. The system displays data on the UI dynamically.

Alternative: The system allows filtering and sorting of displayed data.
Exceptional: If an error occurs, the system logs the issue and prompts the user to retry.

Use Case 11: Search and Add to Database

Description: The system searches for and stores job/policy/course data in the database.
Actors: System DB
Precondition: The database connection is active.

Main Scenario:

System
1. The system scrapes or fetches new jobs, policies, and courses.
2. The system adds relevant data to the database.

Alternative: Admin users manually add data if needed.
Exceptional: If data insertion fails, the system logs an error and retries.

CRC CARDS

Recommendation Engine	
Responsibilities	**Collaborators**
• Suggest jobs, courses, and policies based on user profile.	• User • Profile Manager • Job Model • Policy Model

Profile Manager	
Responsibilities	**Collaborators**
• Store and update user information. • Fetch user details when required. • Update preferences (location, skills).	• User • Recommendation Engine (Uses profile data) • Database

Policy Model	
Responsibilities	**Collaborators**
• Fetch and display government policies.	• User (Searches policies) • Database

User Model	
Responsibilities	**Collaborators**
• Register and log in. • Update profile (location, skills, interests). • Search and apply for jobs. • View recommended opportunities.	• Authentication System (Login/Register) • Profile Manager (Handles user details) • Job Model (Fetches job listings) • City Model • Hospital Model • Course Model • Recommendation Engine

Job Model	
Responsibilities	**Collaborators**
• Fetch and display job listings. • Filter/search jobs based on user input. • Allow users to apply for jobs.	• User (Searches and applies) • Database (Stores job listings) • Recommendation Engine

Course Model	
Responsibilities	**Collaborators**
• Fetch and display educational courses.	• User (Explores courses) • Recommendation Engine • Database

Database	
Responsibilities	**Collaborators**
• Store all data (users, jobs, policies, courses, health facilities). • Retrieve and update records.	• Authentication System • Profile Manager • Job Model • Policy Model • Course Model • Hospital Model • City Model • State Model • Institute Model • Employer Model

Hospital Model	
Responsibilities	**Collaborators**
• Fetch and display health facilities. • Allow users to search by location.	• User (Searches for facilities) • Database

Authentication System	
Responsibilities	**Collaborators**
• Handle user registration and login. • Authenticate users via email/password. • Manage password reset.	• User (Provides credentials) • Database (Stores user details)

Institute Model	
Responsibilities	**Collaborators**
◦ Represent institute data	• Database

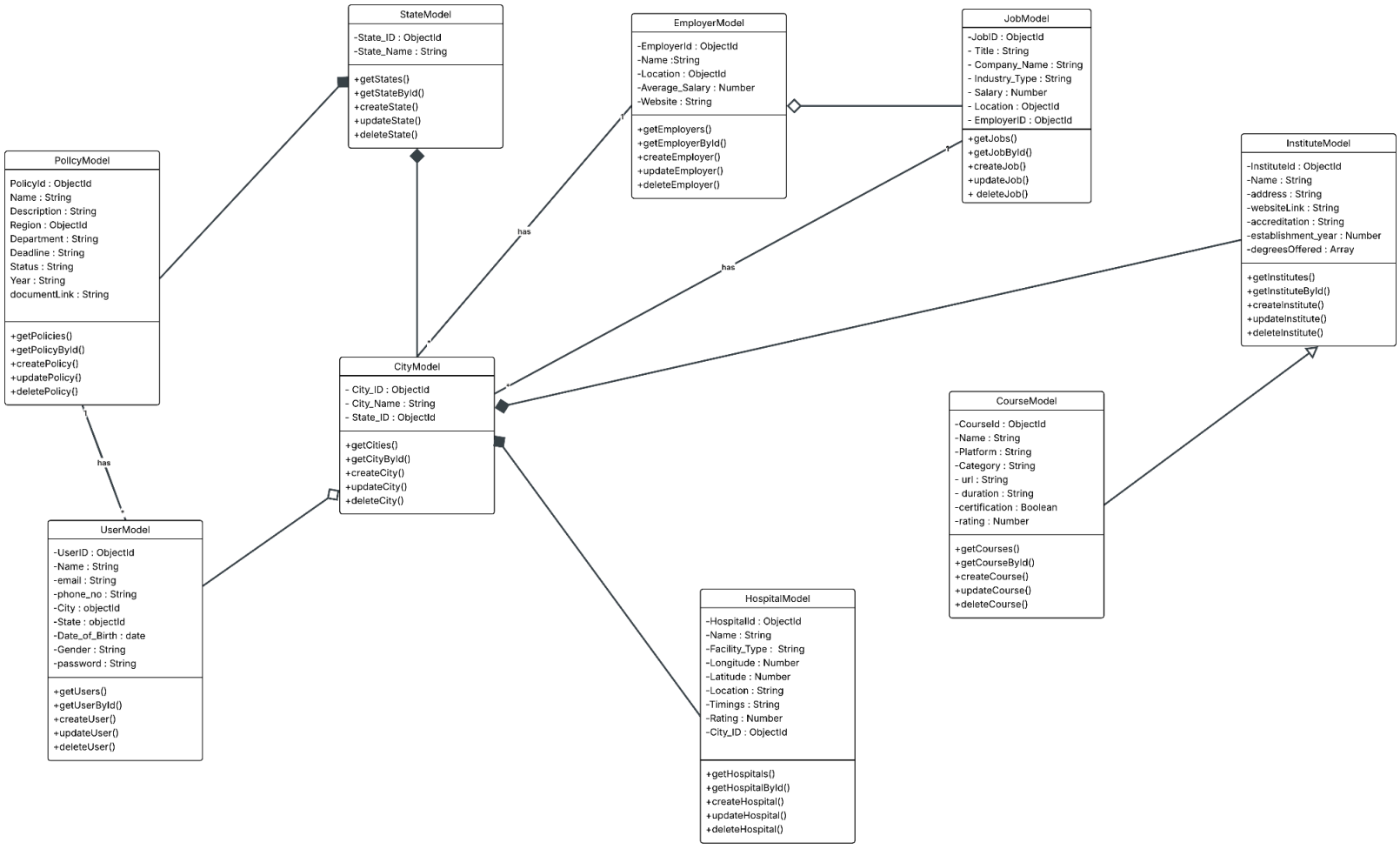
Employer Model	
Responsibilities	**Collaborators**
• Represent employer data	• JobModel • Database

State Model	
Responsibilities	**Collaborators**
• **Responsibilities:** Represent state data, database operations	• CityModel • Database

City Model	
Responsibilities	**Collaborators**
◦ Represent city data	• State Model • User Model

Frontend Components (React)	
Responsibilities	**Collaborators**
• Render UI, call APIs	• API Services • React Router

Class Diagram



1. UserModel

- Attributes: UserID, Name, email, phone_no, City, State, Date_Of_Birth, Gender, password
- Functions: getUsers(), getUserById(), createUser(), updateUser(), deleteUser()
- Relationships:
  has a reference to CityModel using ObjectIds (Aggregation) and Association with PolicyModel.
- Role: Represents registered users in the system who access job, course, health, and policy data.

2. CityModel

- Attributes: City_ID, City_Name, State_ID
- Functions: CRUD operations on cities.
- Relationships:
  Composed of a StateModel (City cannot exist without a State) → Composition.
  Aggregates multiple UserModel, EmployerModel,HospitalModel, → Aggregation.
  InstituteModel, JobModel → Association
- Role: Represents the geographical location context.

3. StateModel

- Attributes: State_ID, State_Name
- Functions: CRUD operations on states.
- Relationships:
  Parent to many CityModels, PolicyModel → Composition.
- Role: Higher-level location that groups multiple cities.

4. EmployerModel

- Attributes: EmployerId, Name, Location, Average_Salary, Website
- Functions: CRUD for employers.
- Relationships:
  Has Aggregation with CityModel (Location).
- Role: Represents companies offering jobs.

5. JobModel

- Attributes: JobID, Title, Company_Name, Industry_Type, Salary, Location, EmployerID
- Functions: CRUD for jobs.
- Relationships:
  Association with EmployerModel (linked by EmployerID).
  Aggregation with CityModel (Location).
- Role: Lists job opportunities available on the platform.

6. InstituteModel

- Attributes: InstituteId, Name, address, websiteLink, accreditation, establishment_year, degreesOffered
- Functions: CRUD for institutes.
- Relationships:
  Association with CityModel for Location.
- Role: Represents educational institutes offering offline courses.

7. CourseModel

- Attributes: CourseId, Name, Platform, Category, url, duration, certification, rating
- Functions: CRUD for online courses.
- Relationships:
  No direct link with other entities. Can have relationship with institutes
- Role: Online learning content shown as opportunities.

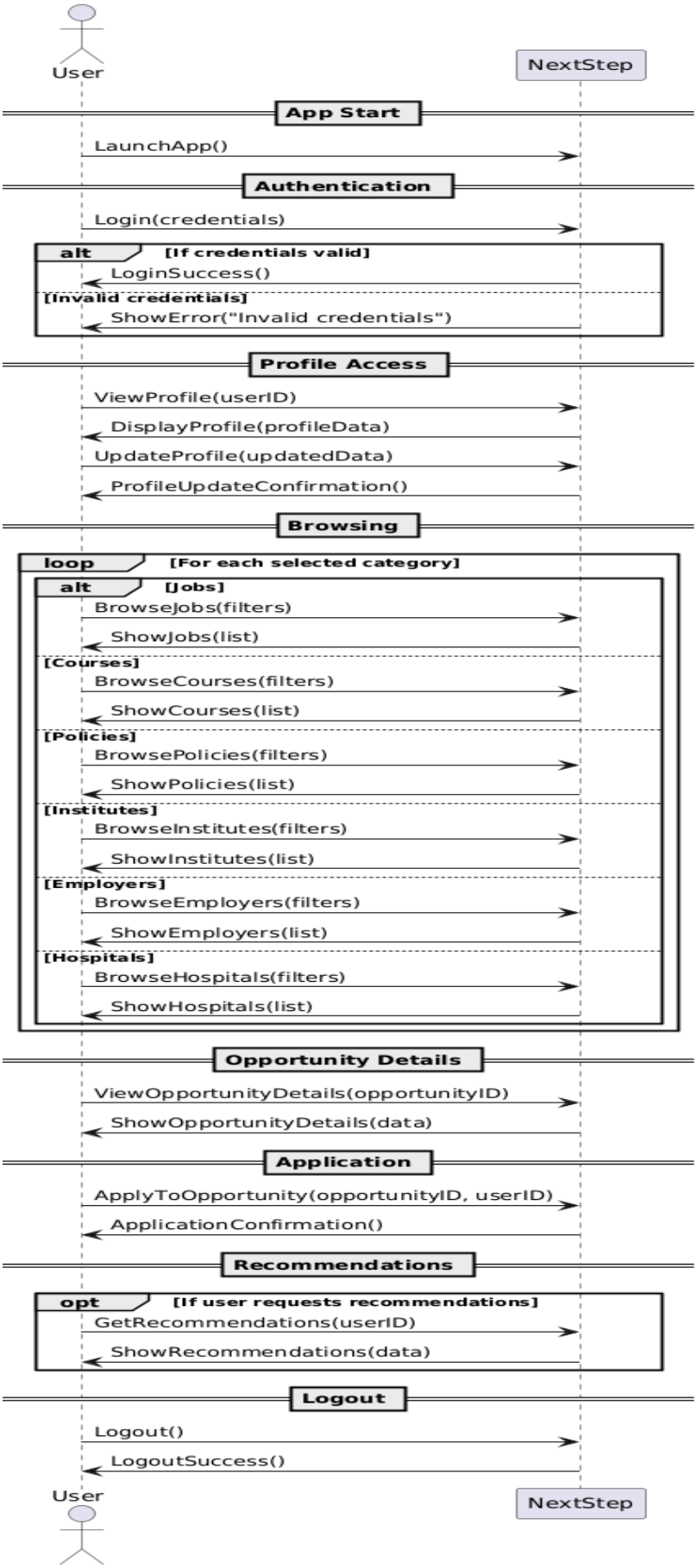
8. HospitalModel

- Attributes: HospitalId, Name, Facility_Type, Longitude, Latitude, Location, Timings, Rating, City_Id
- Functions: CRUD for hospitals.
- Relationships:
  Association with CityModel.
- Role: Healthcare centers relevant to the user's city.

9. PolicyModel

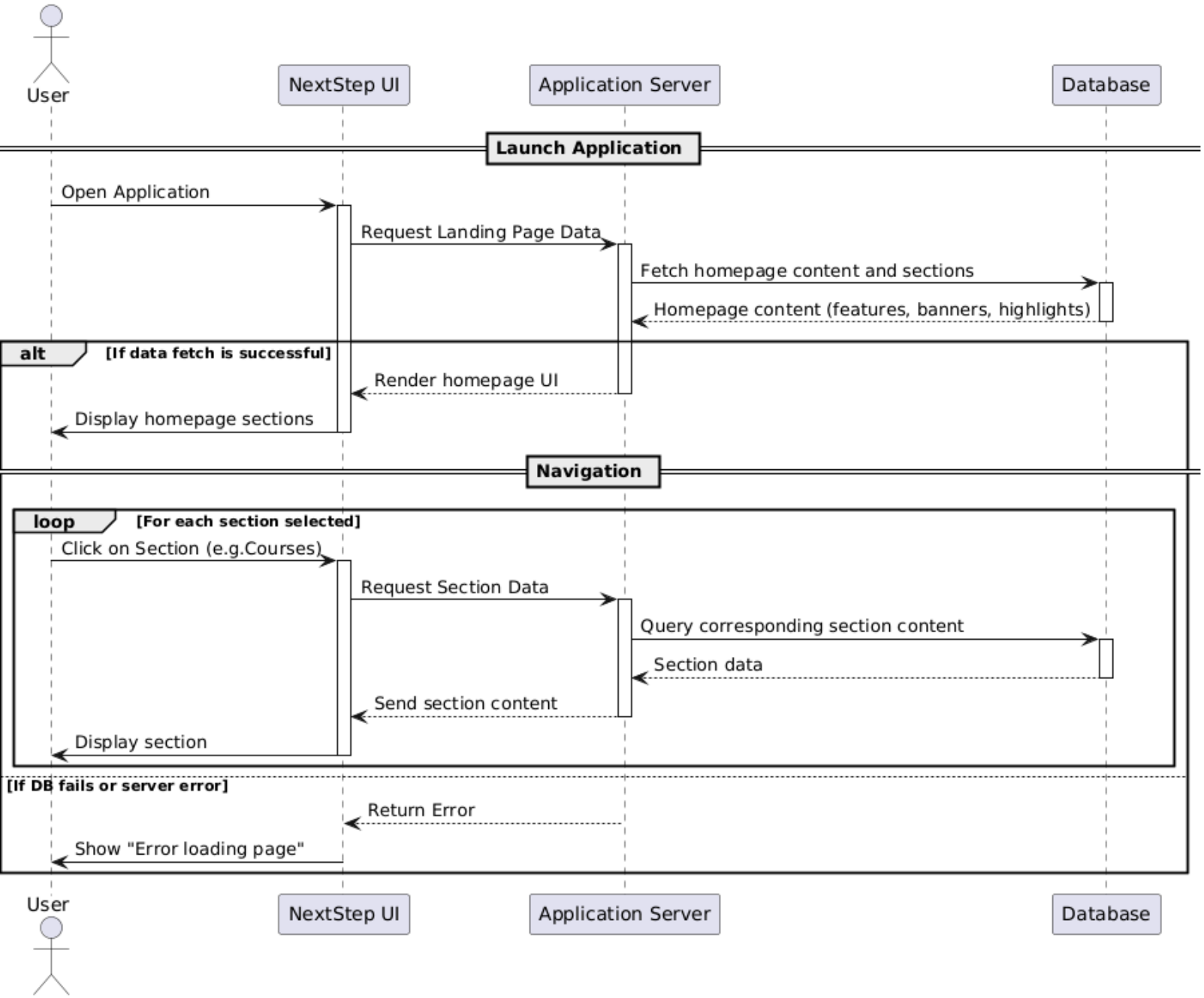
- Attributes: PolicyId, Name, Description, Region, Department, Deadline, Status, Year, documentLink
- Functions: CRUD for policies.
- Relationships:
  Association with StateModel (Region).
- Role: Government policies relevant to youth, jobs, health, etc.

System Sequence Diagram

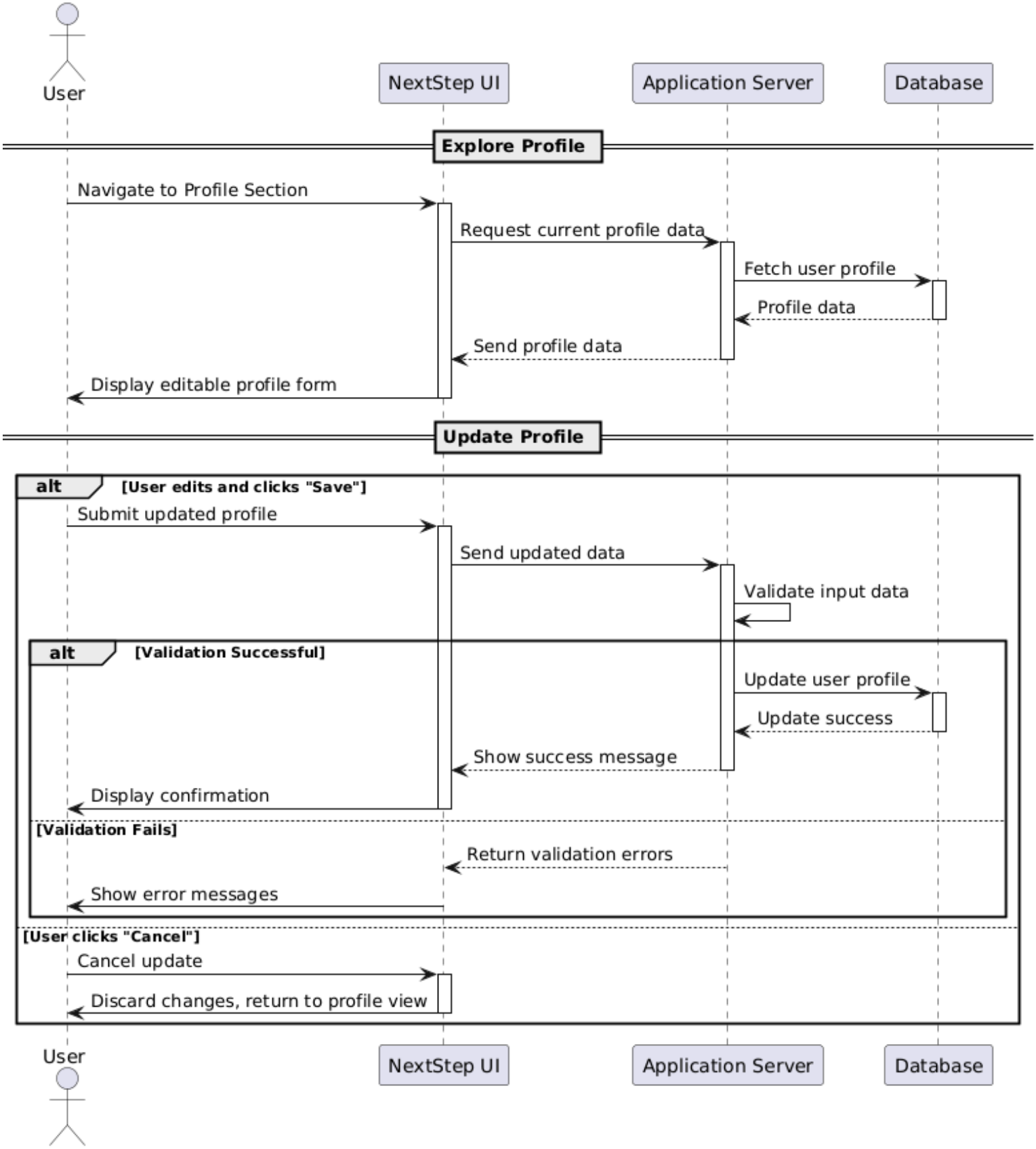


Sequence Diagram

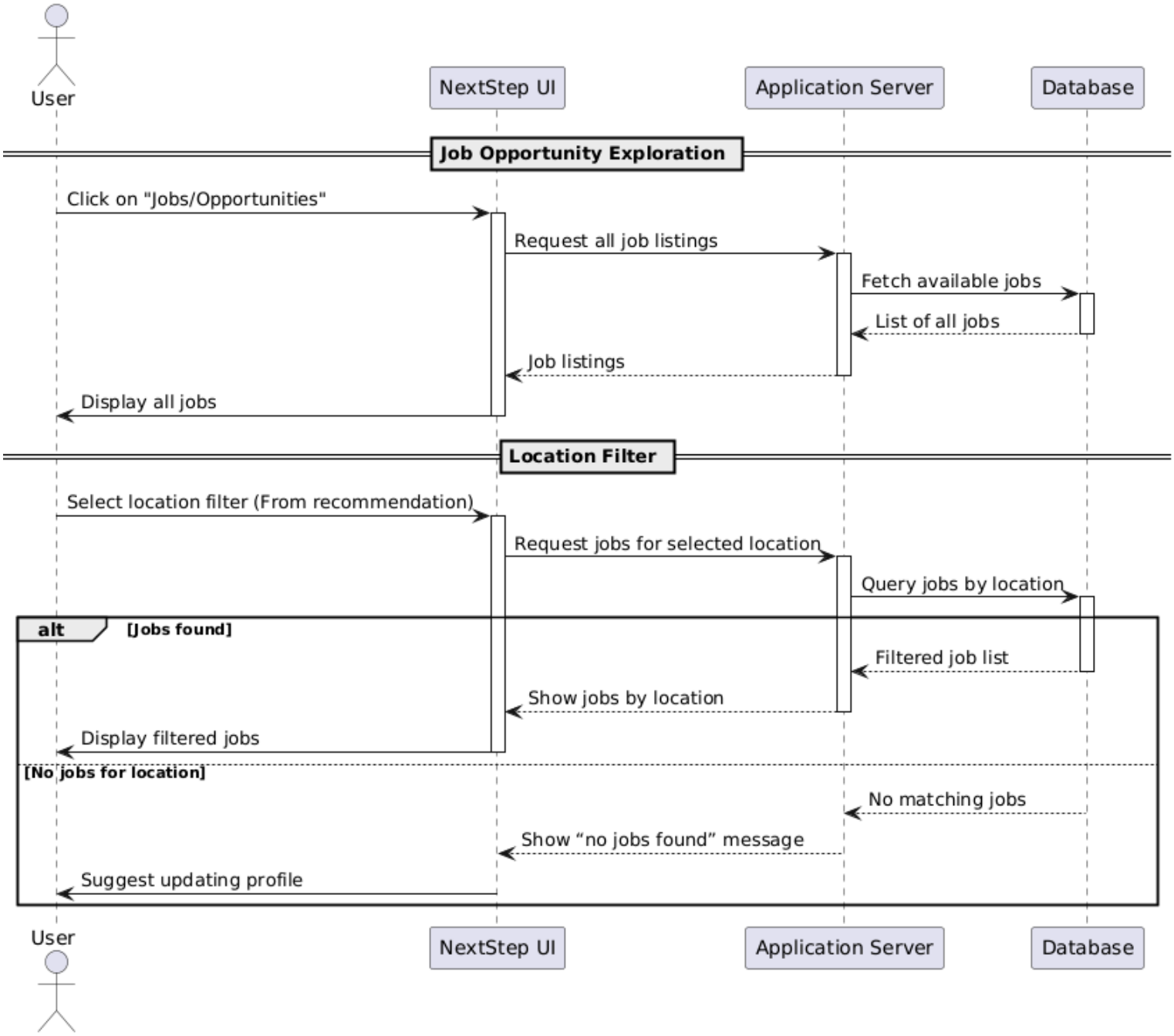
Use Case 1: Explore the Landing Page



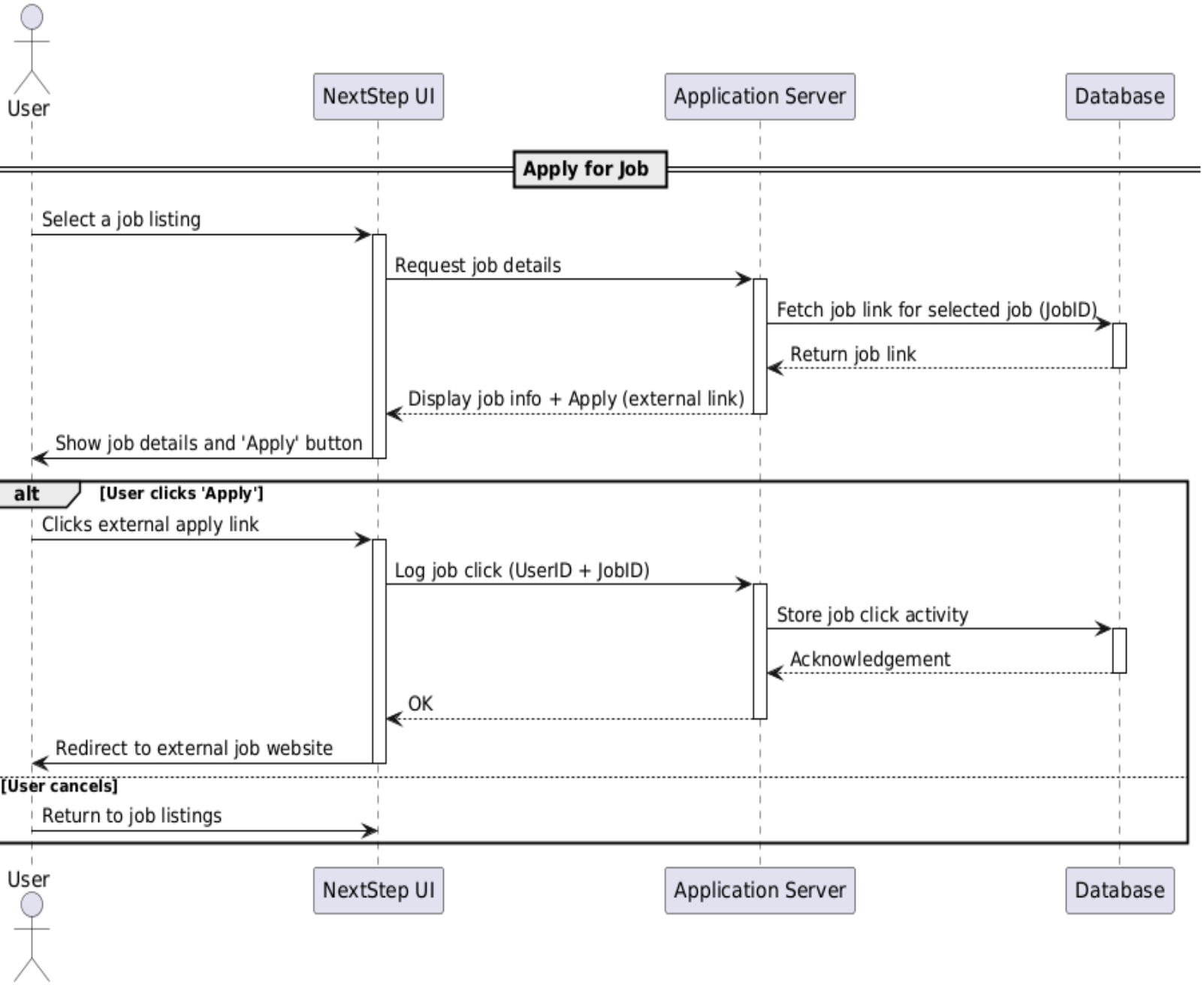
Use Case 2: Explore and Update Profile



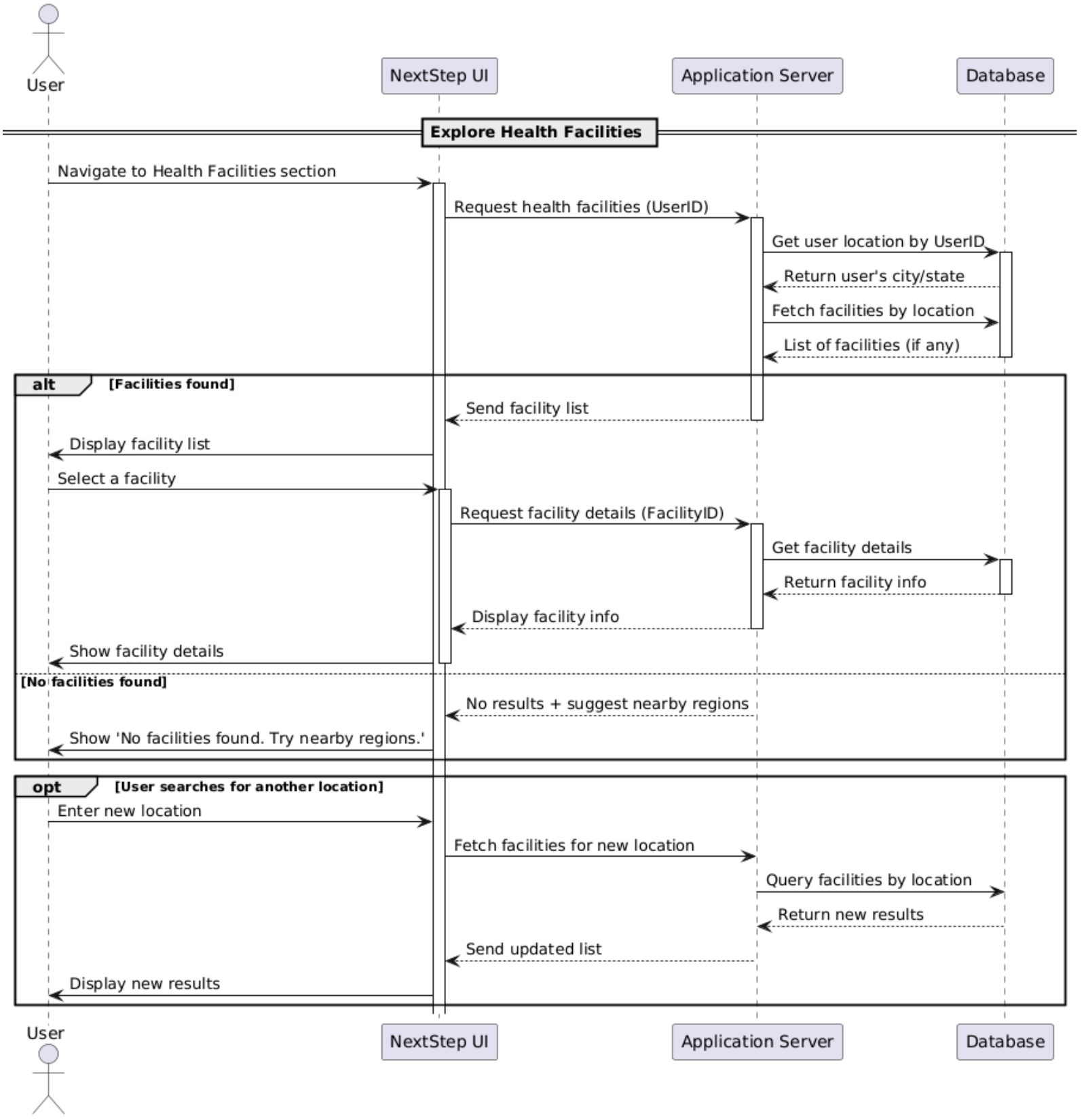
Use Case 3: Explore Jobs and Opportunities



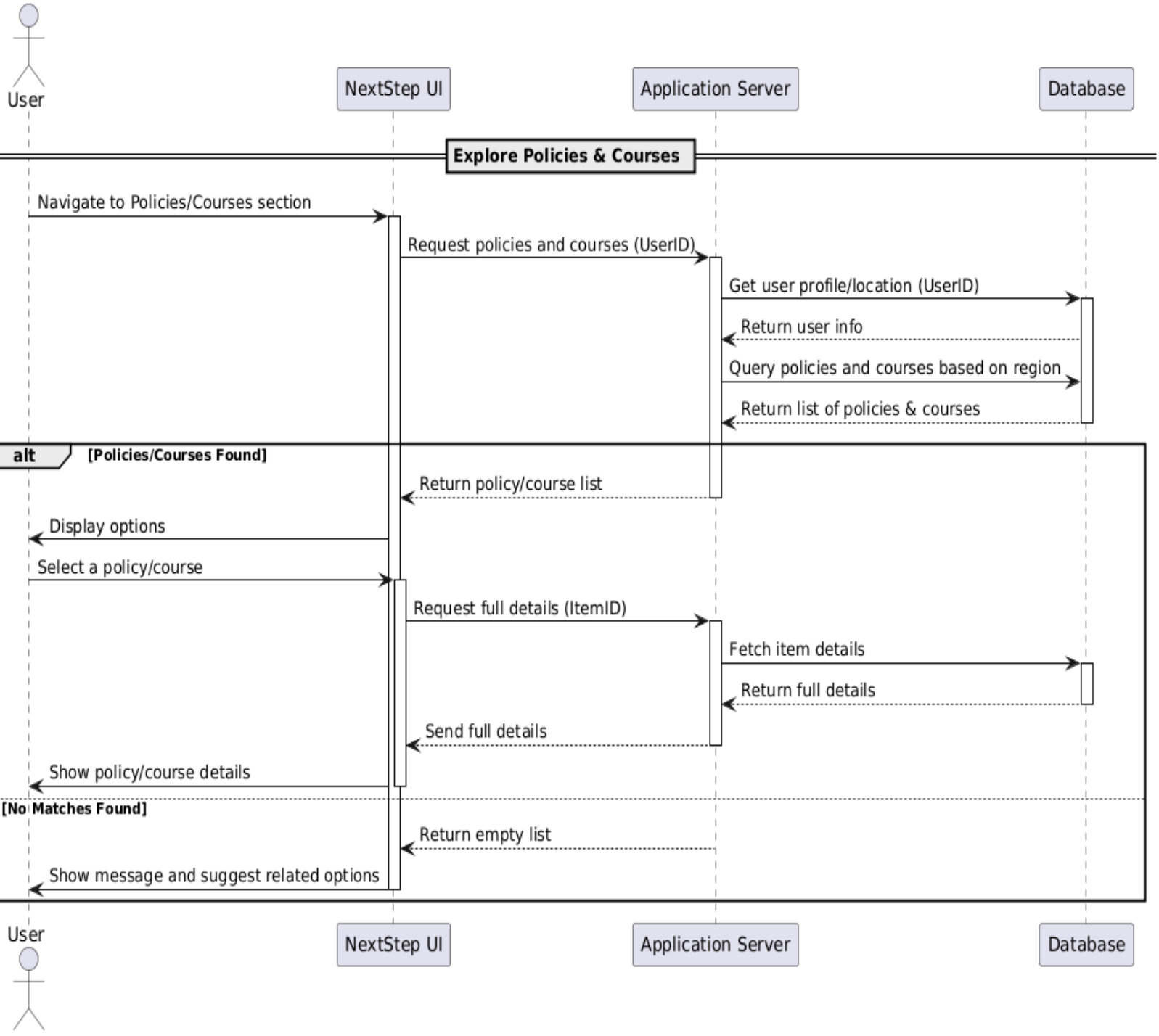
Use Case 4: Apply for Jobs



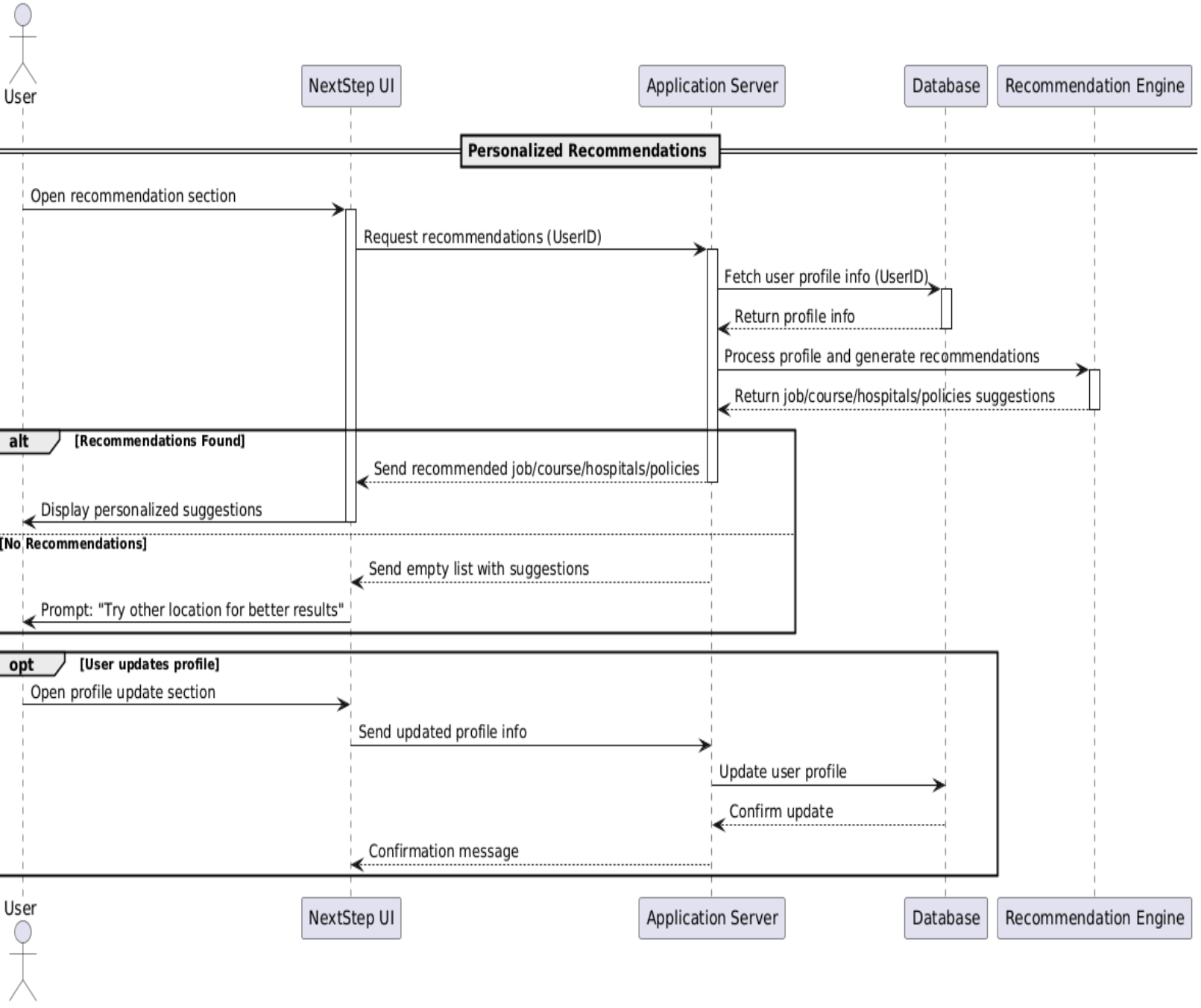
Use Case 5: Explore Health Facilities



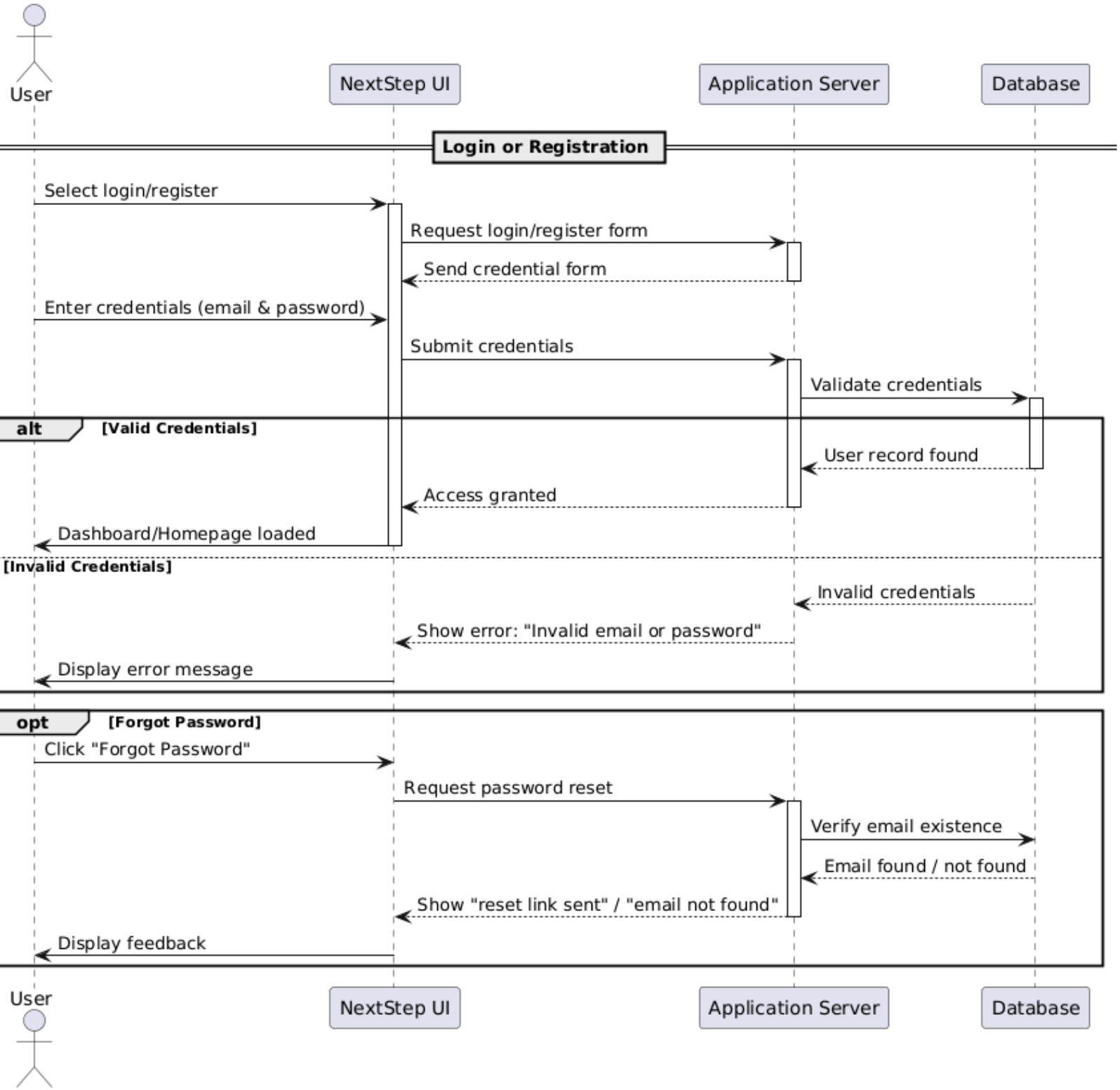
Use Case 6: Explore Government Policies and Courses



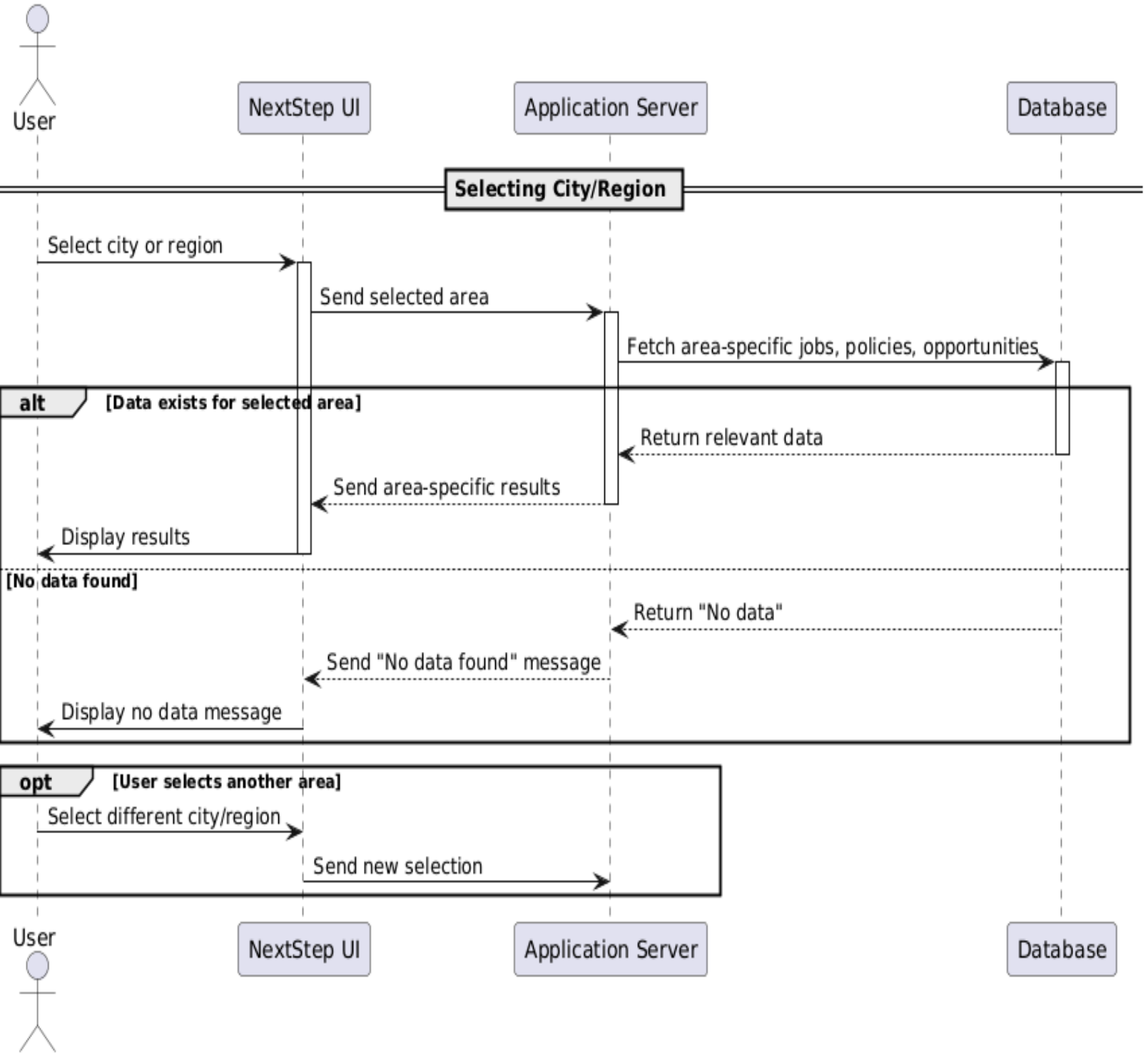
Use Case 7: Get Personalized Recommendations



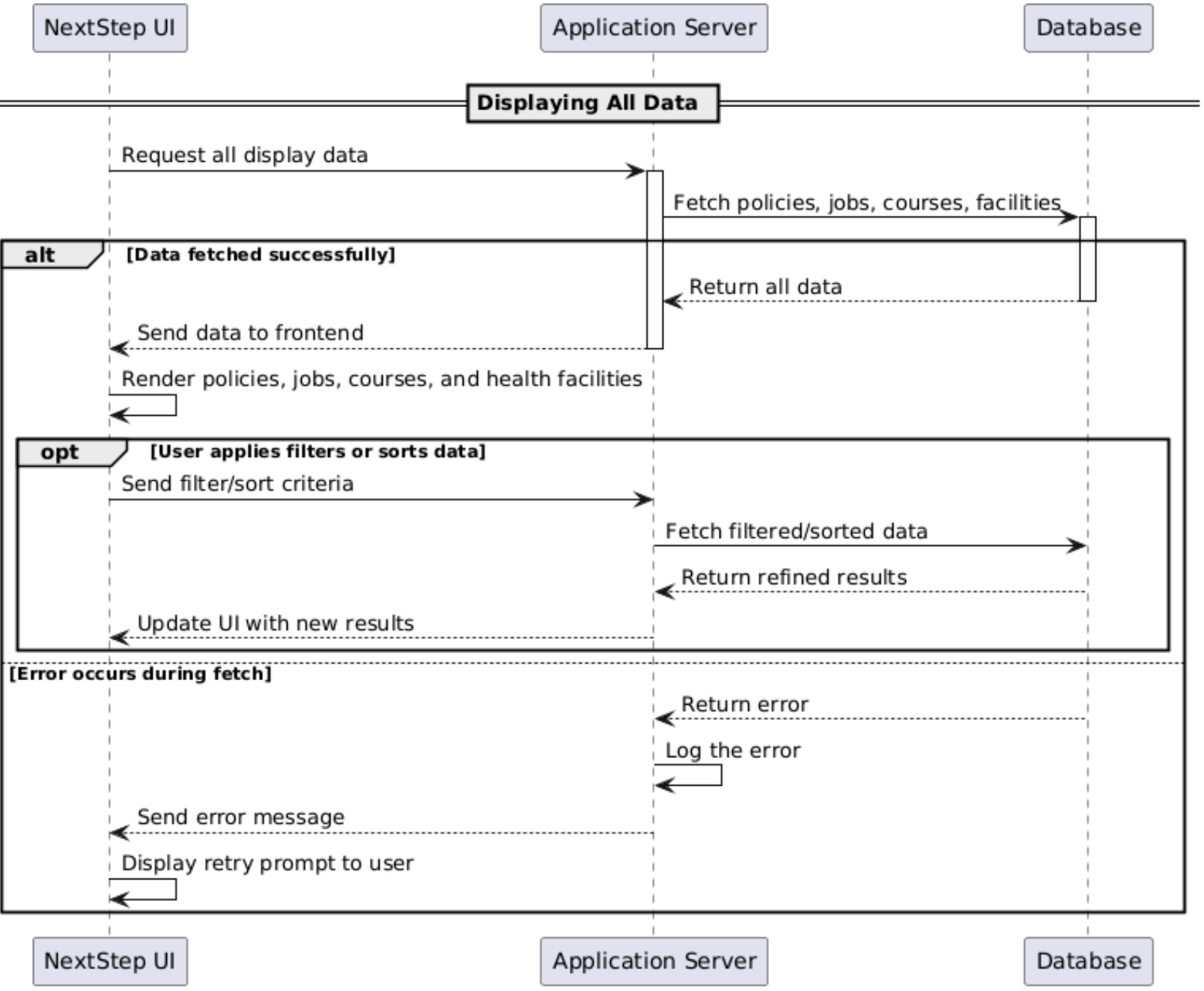
Use Case 8: Login/Registration



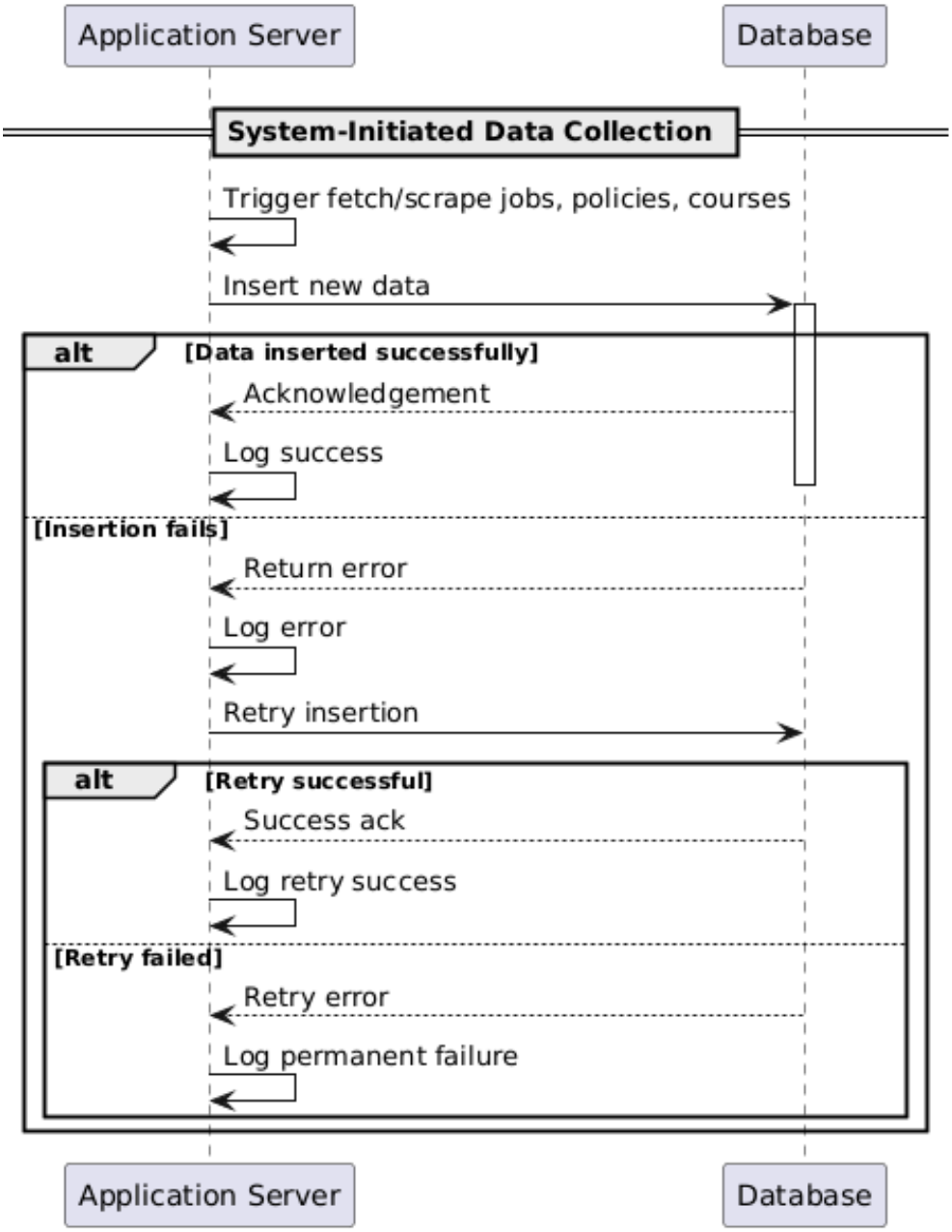
Use Case 9: Selecting Area



Use Case 10: Displaying Policies, Facilities, Jobs, and Courses


NextStep UI
Application Server
Database
Displaying All Data
Request all display data
Fetch policies, jobs, courses, facilities
alt
[Data fetched successfully]
Return all data
Send data to frontend
Render policies, jobs, courses, and health facilities
opt
[User applies filters or sorts data]
Send filter/sort criteria
Fetch filtered/sorted data
Return refined results
Update UI with new results
[Error occurs during fetch]
Return error
Log the error
Send error message
Display retry prompt to user
NextStep UI
Application Server
Database

Use Case 11: Search and Add to Database



Activity, State Diagrams and OCL

for

NextStep

Prepared by

Group ID: 21

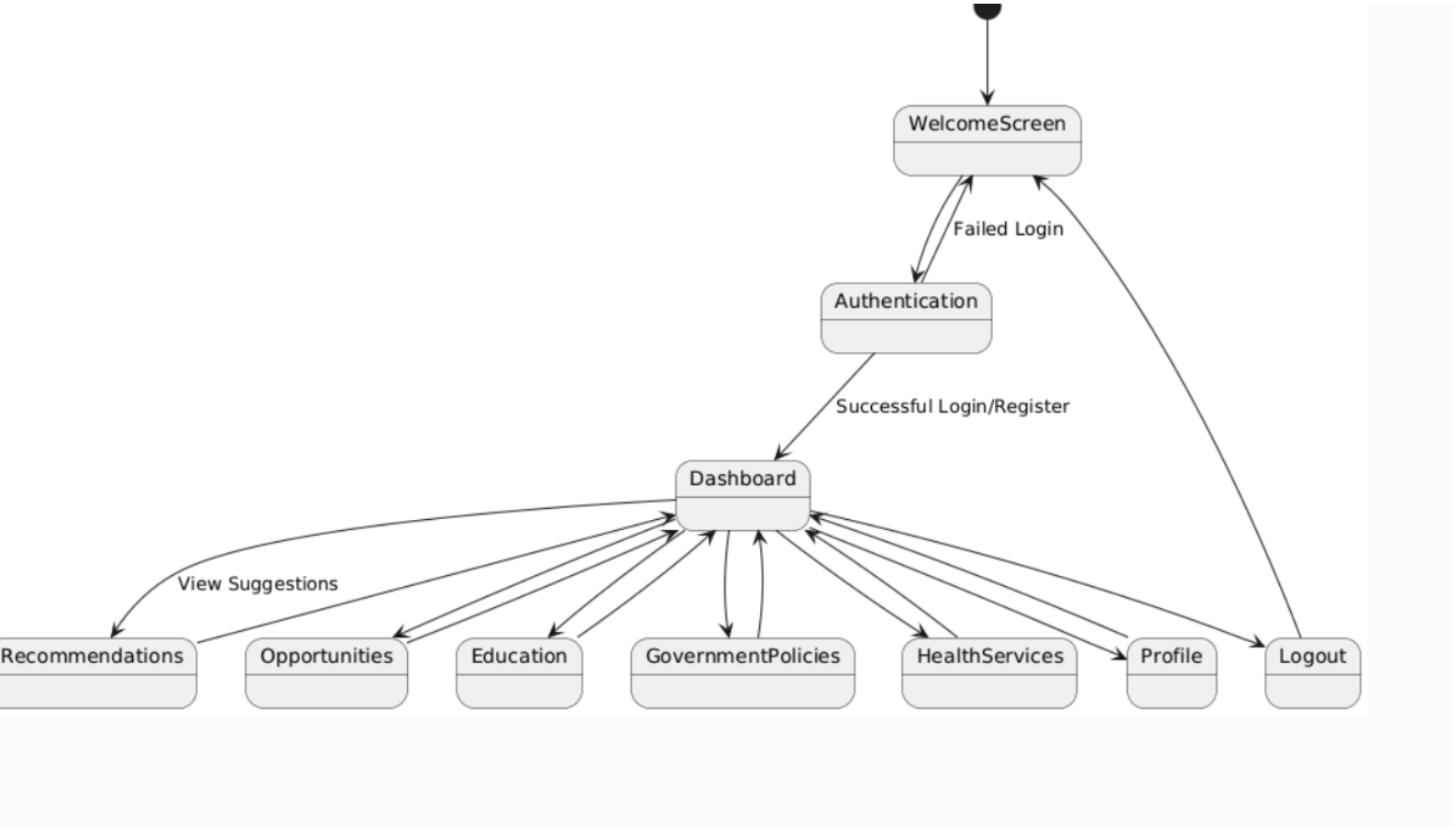
Aarushi Goel	**202412002**	**202412002@daiict.ac.in**
Mohsin Pathan	**202412070**	**202412070@daiict.ac.in**
Annirudh Pratap	**202412008**	**202412008@daiict.ac.in**

Instructor: Jayprakash Lalchandani

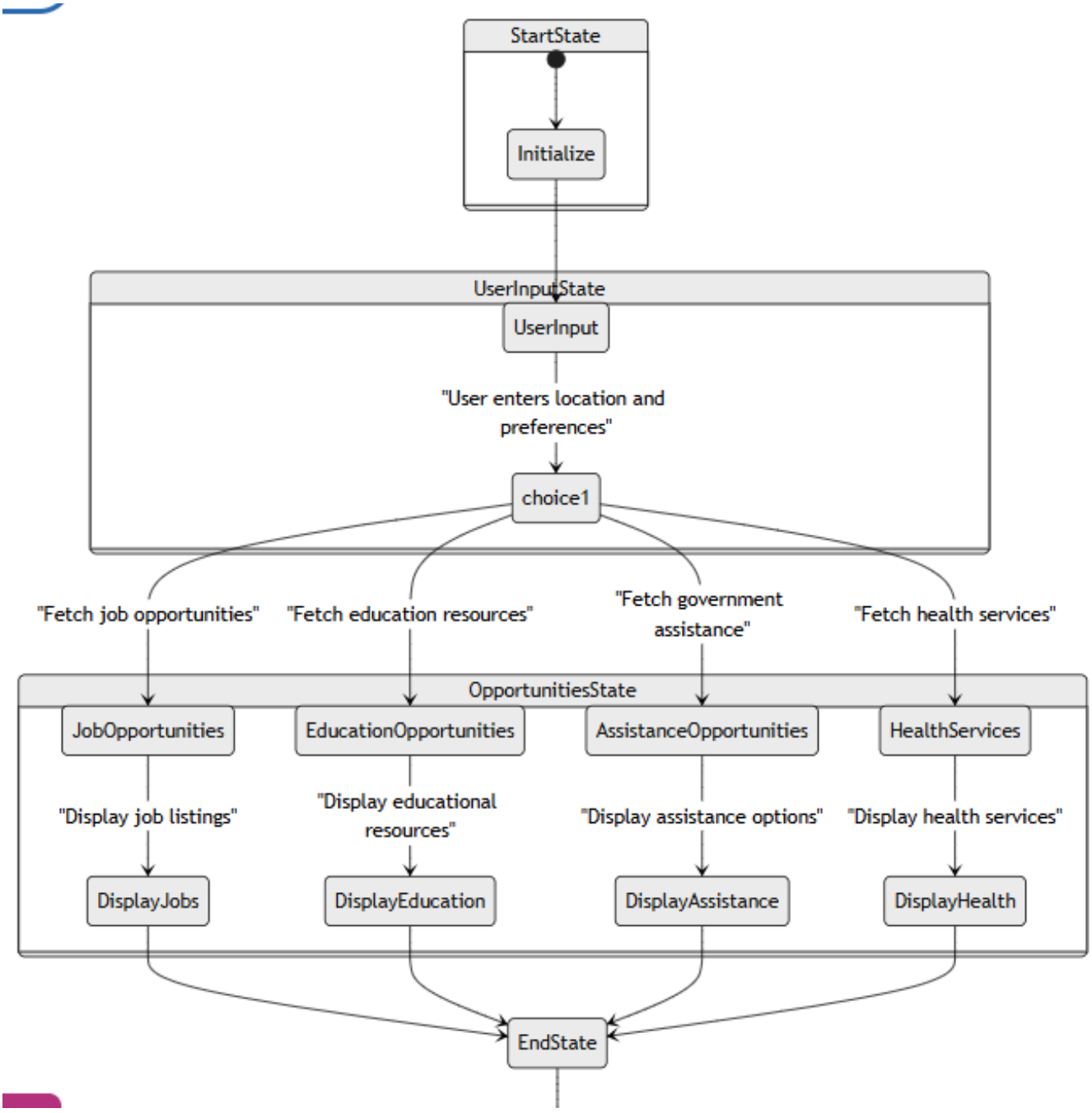
Date: 14th April 2025

Activity and State Diagram

Entire System Activity Diagram



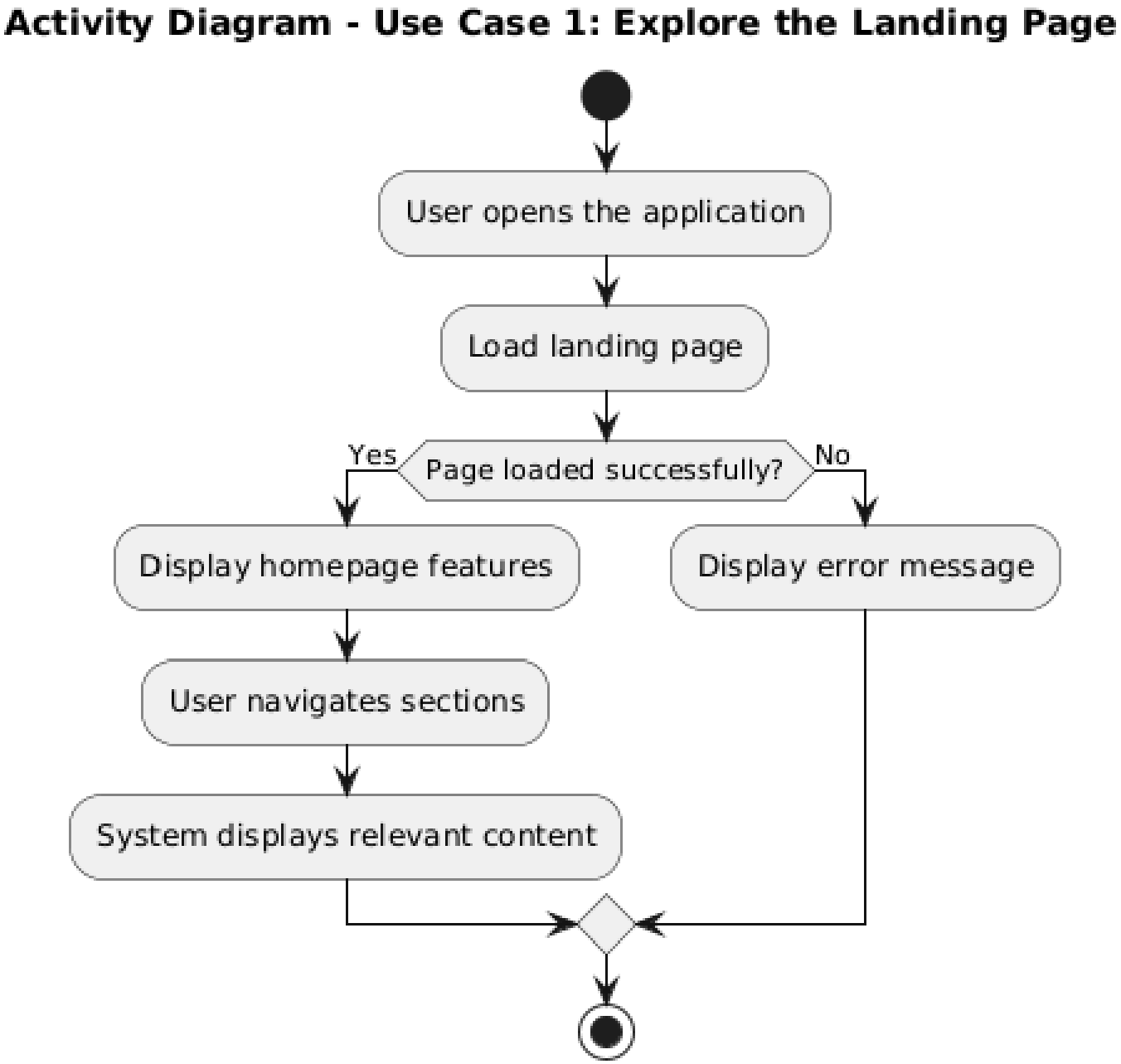
Entire System State Diagram



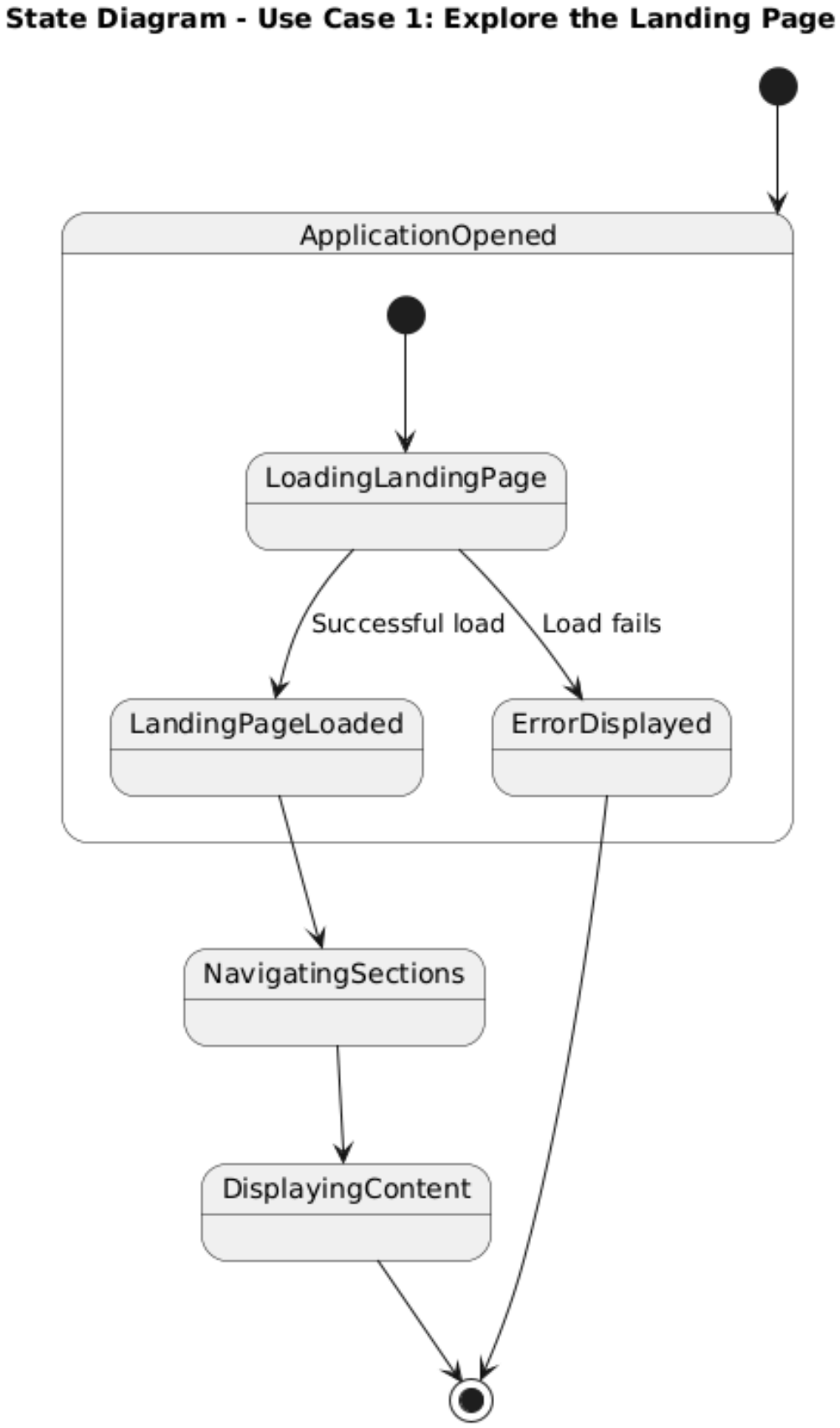
Use Case 1: Explore the Landing Page

Description: The user views and navigates the homepage of the NextStep application.

Activity Diagram

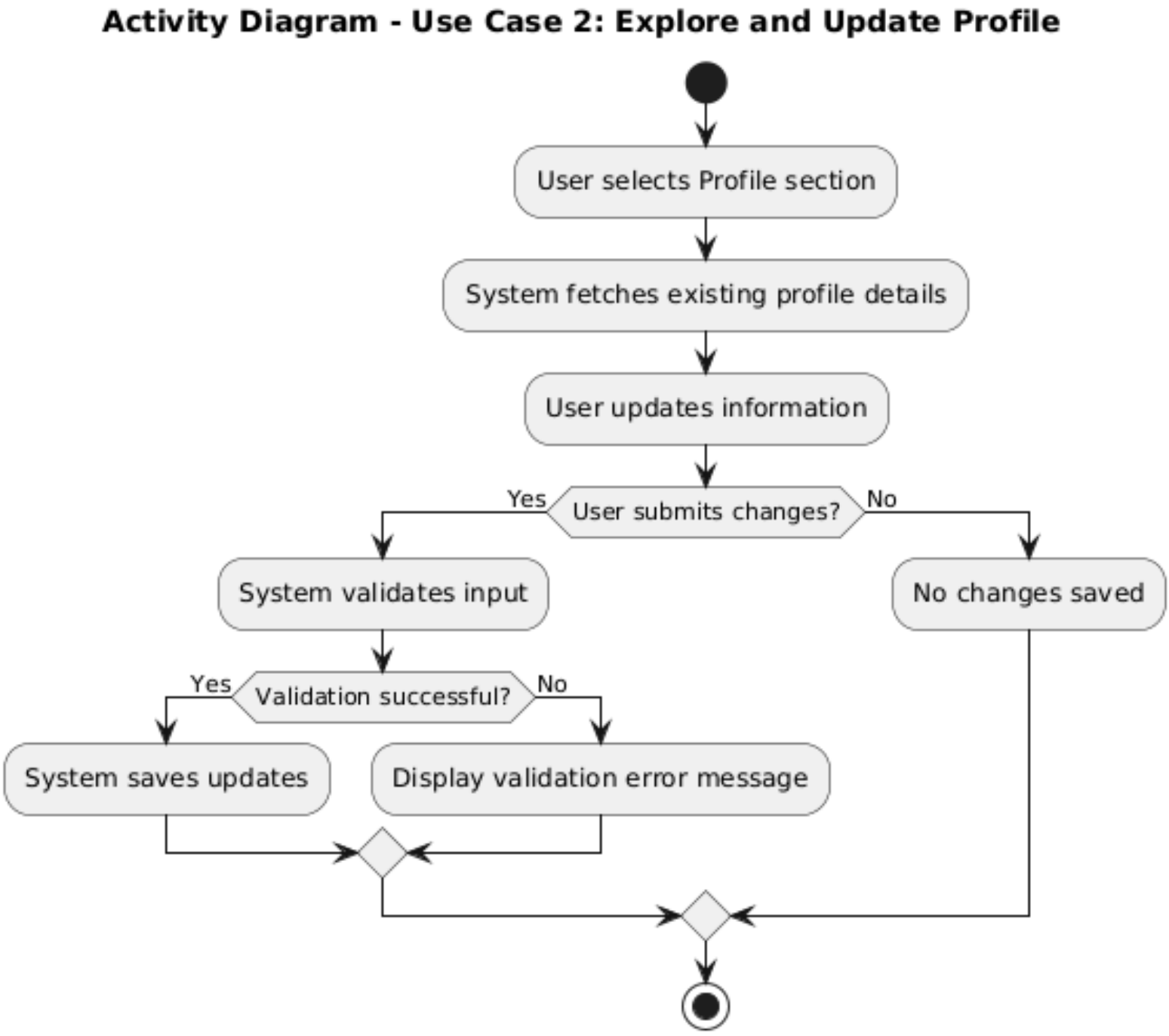


State Diagram

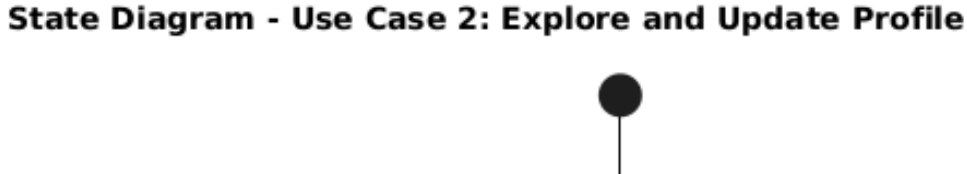


Use Case 2: Explore and Update Profile

Description: The user can update personal details such as location and interests.



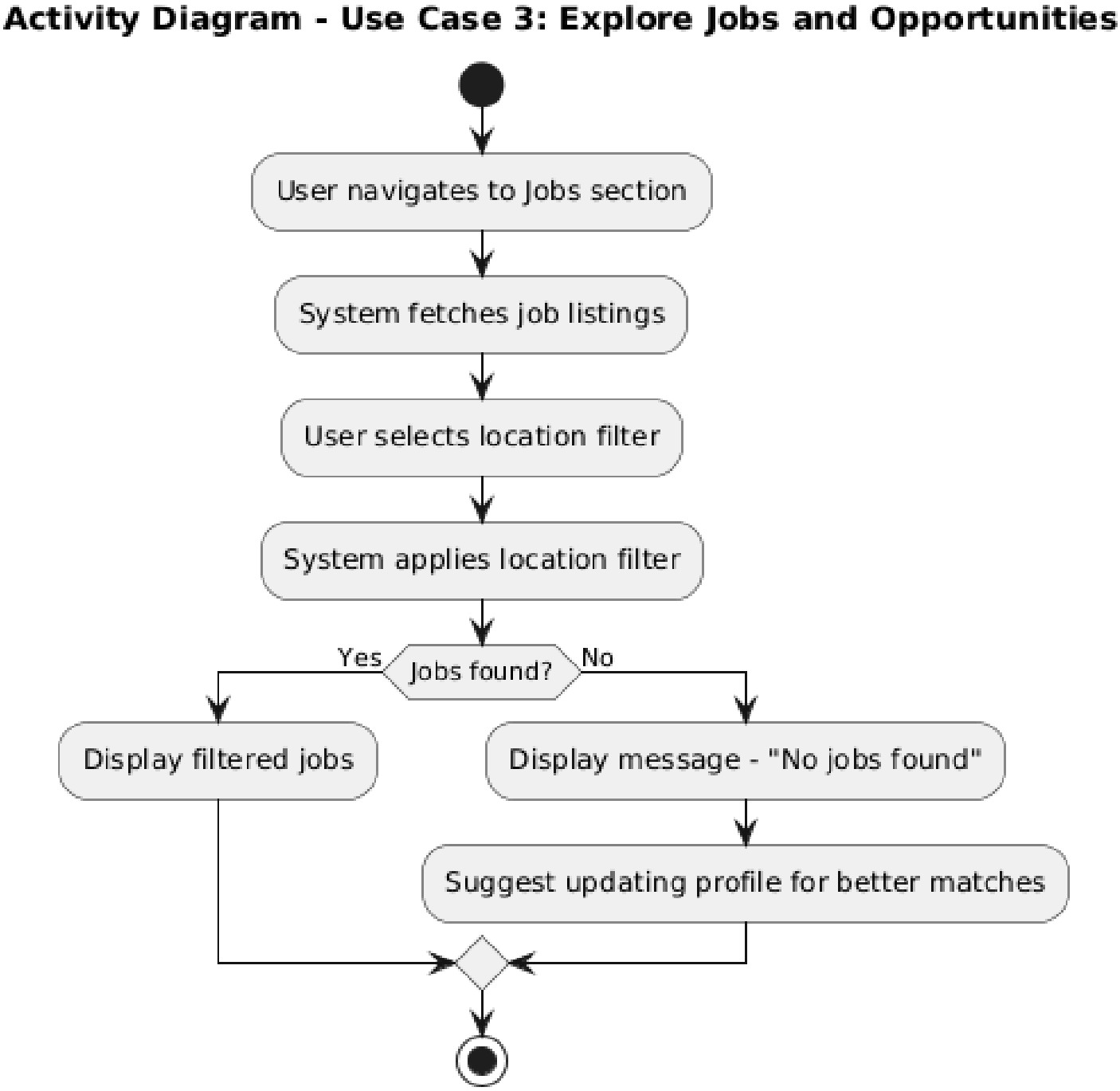
State diagram



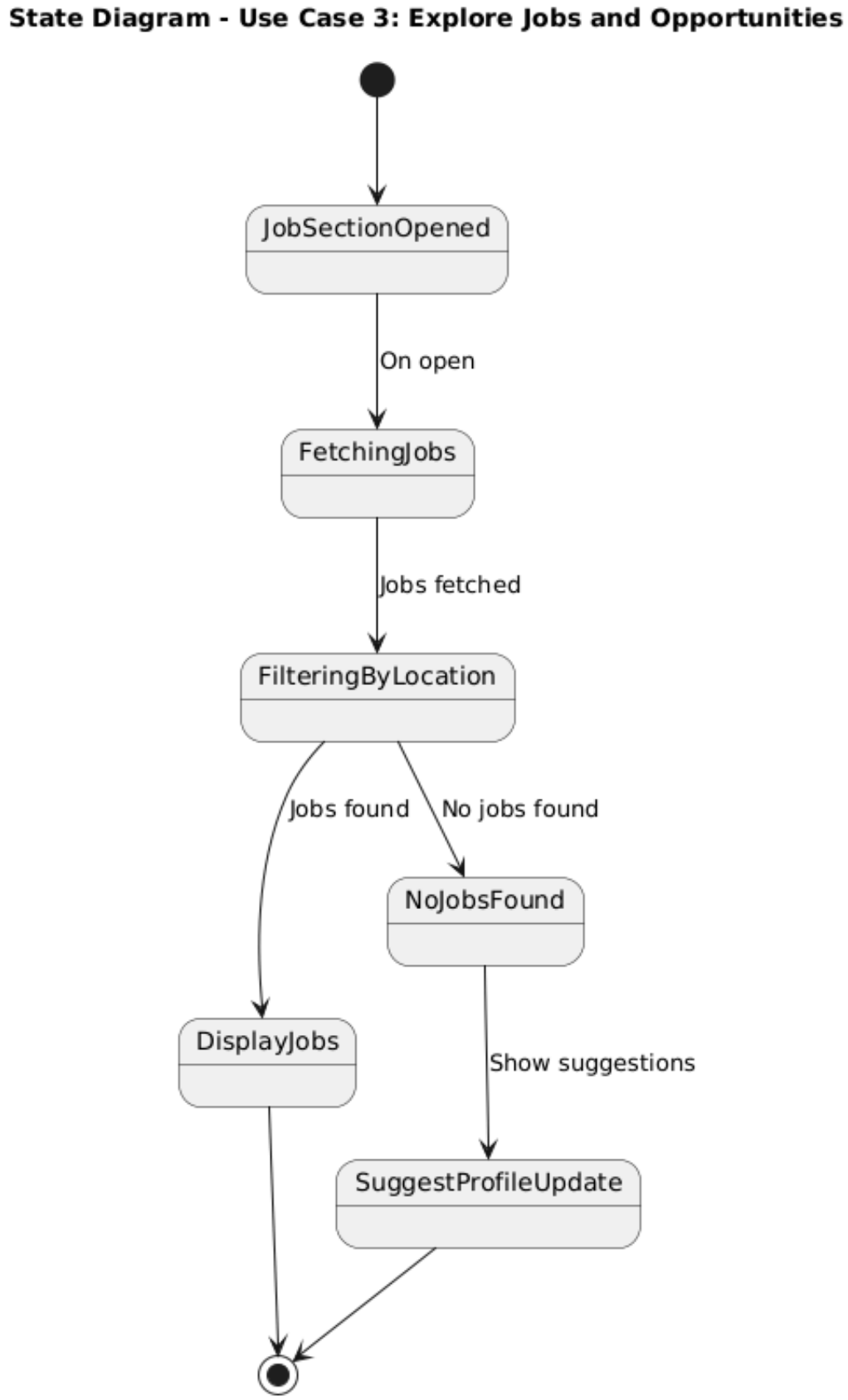
Use Case 3: Explore Jobs and Opportunities

Description: The user searches and browses job opportunities.

Activity Diagram



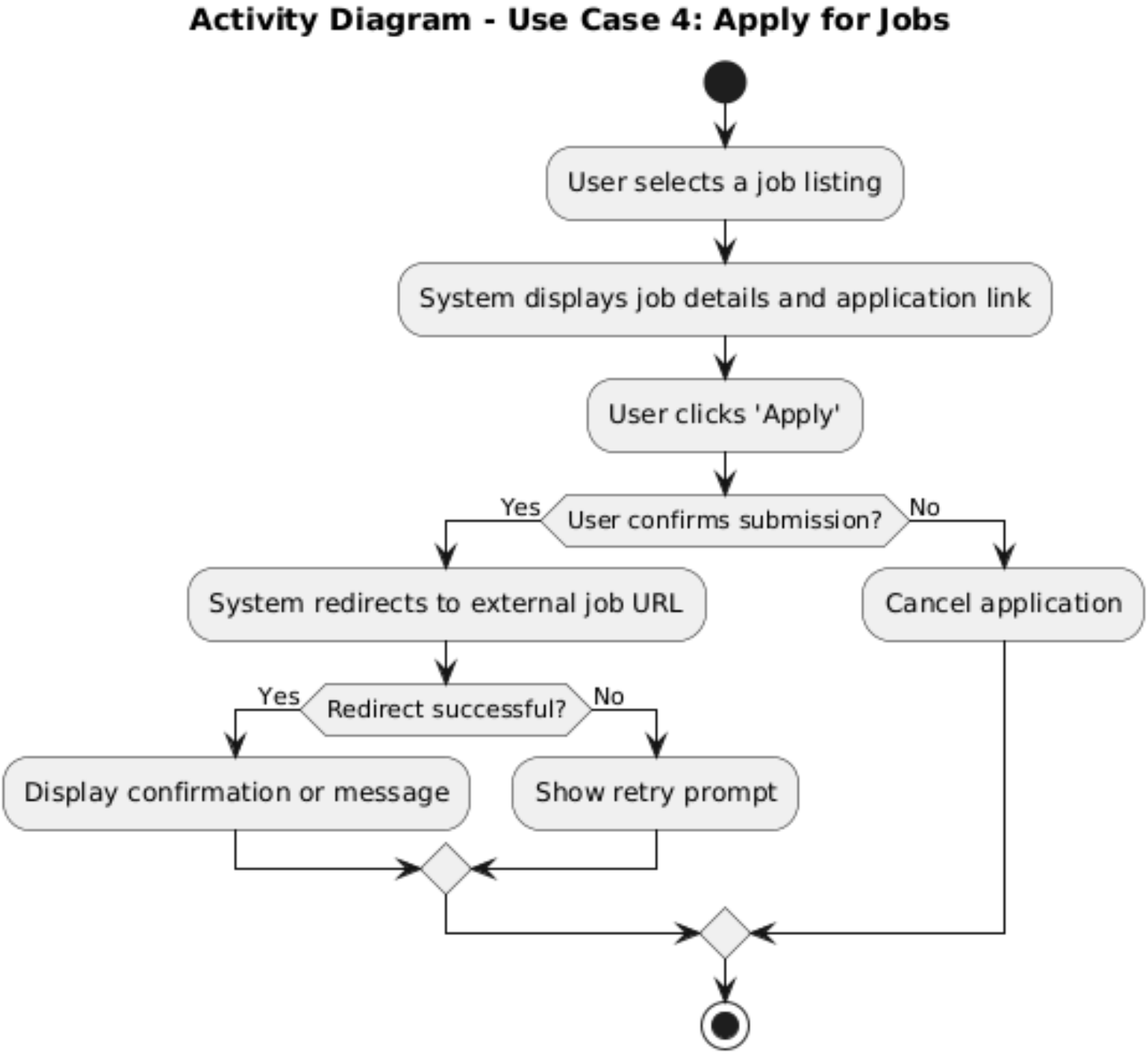
State diagram



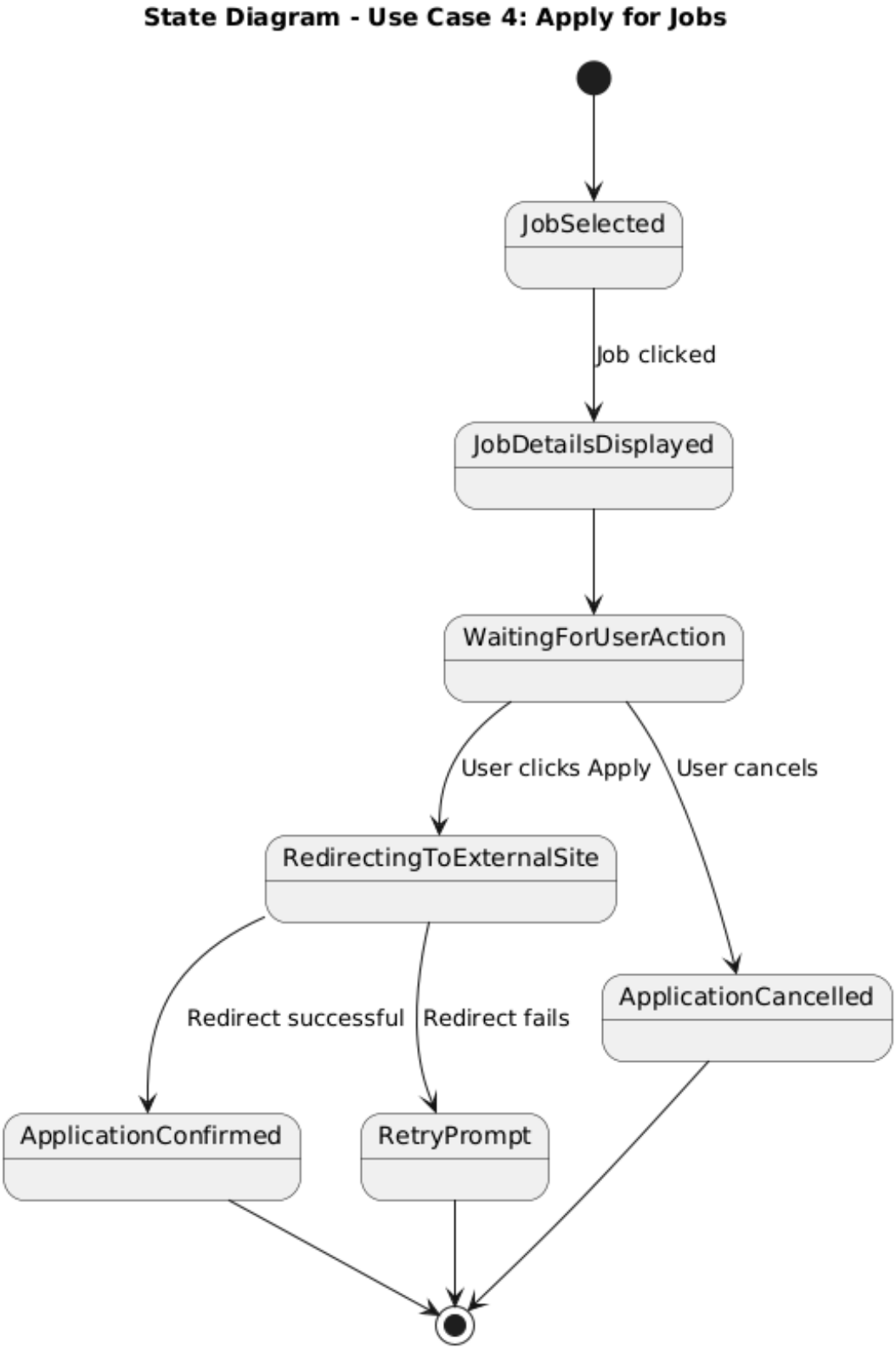
Use Case 4: Apply for Jobs

Description: The user applies for a selected job.

Activity Diagram



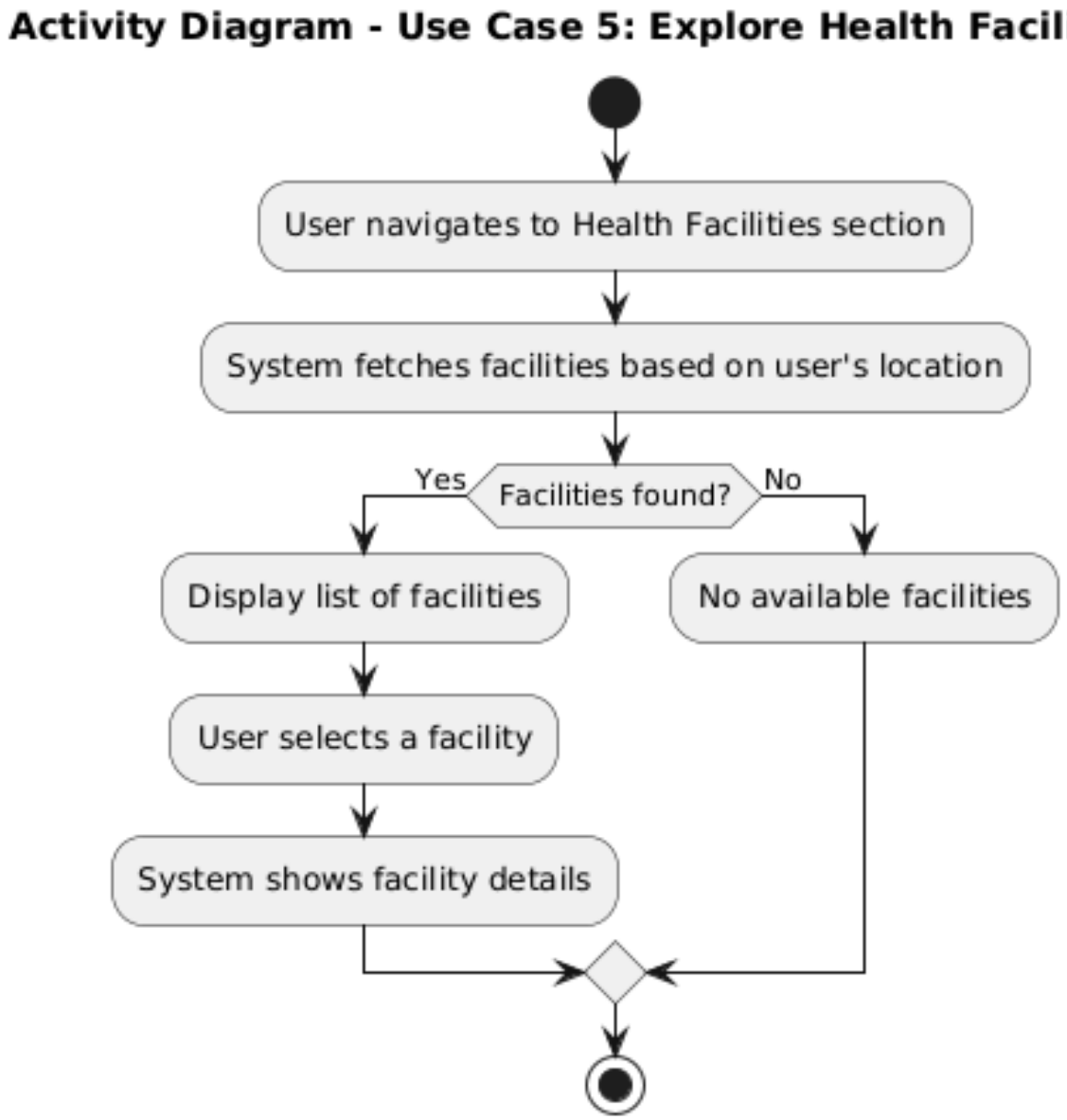
State Diagram



Use Case 5: Explore Health Facilities

Description: The user searches for available health facilities in their region.

Activity Diagram

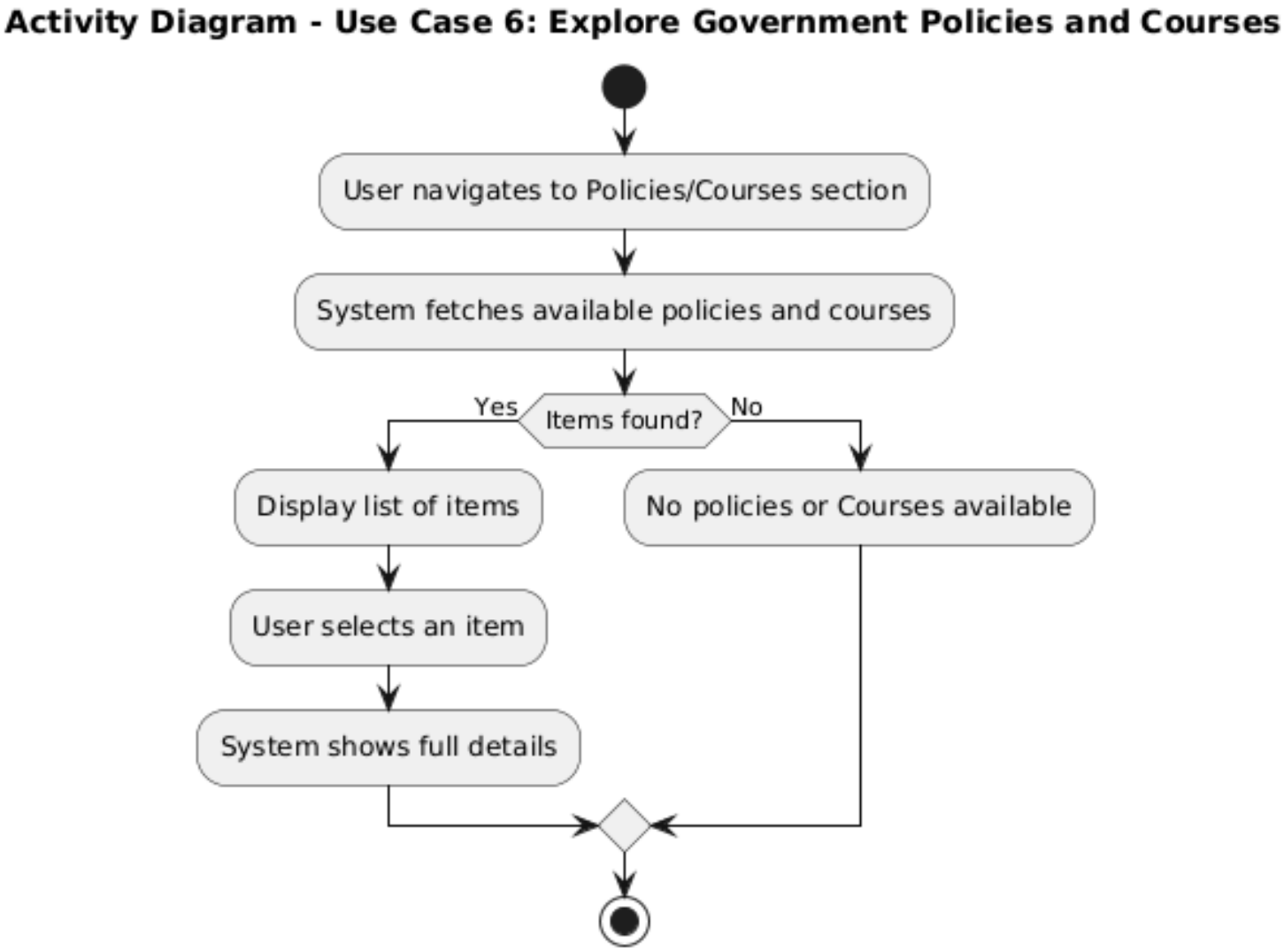


State Diagram

Use Case 6: Explore Government Policies and Courses

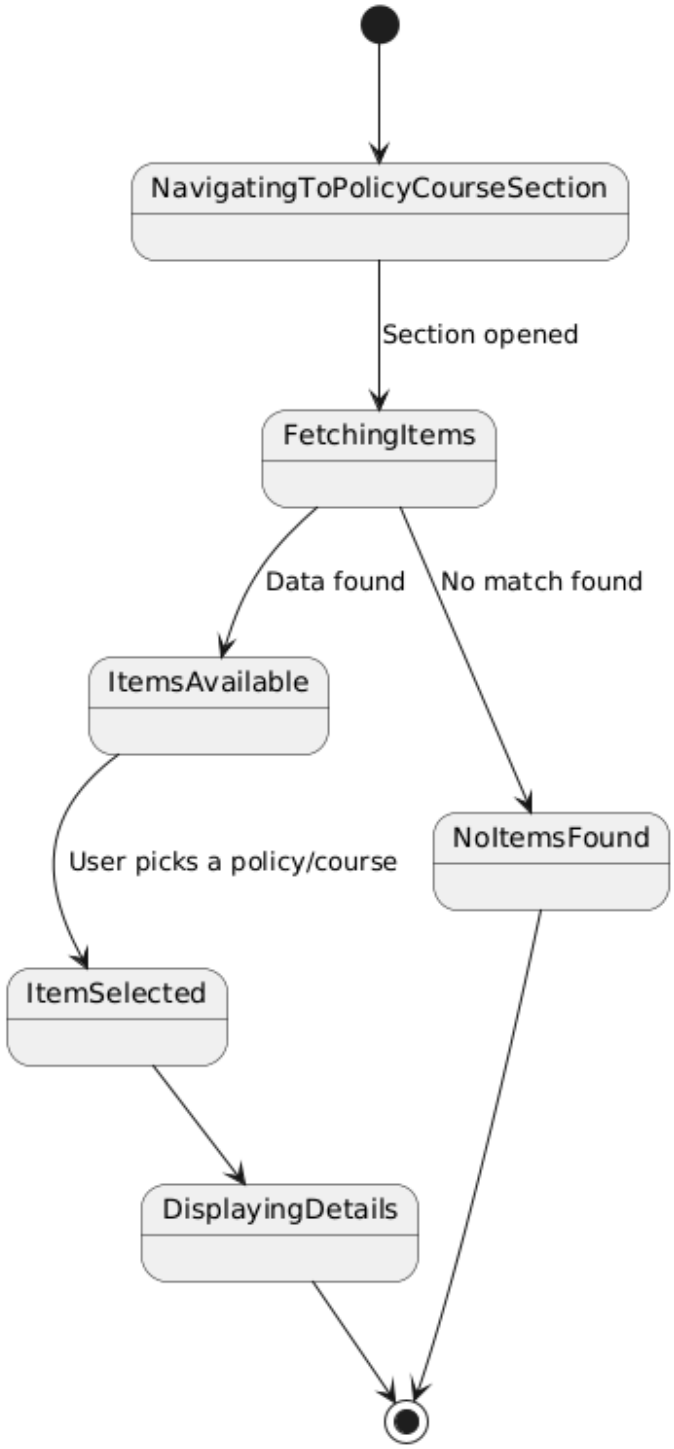
Description: The user explores available policies and educational courses.

Activity Diagram



State Diagram

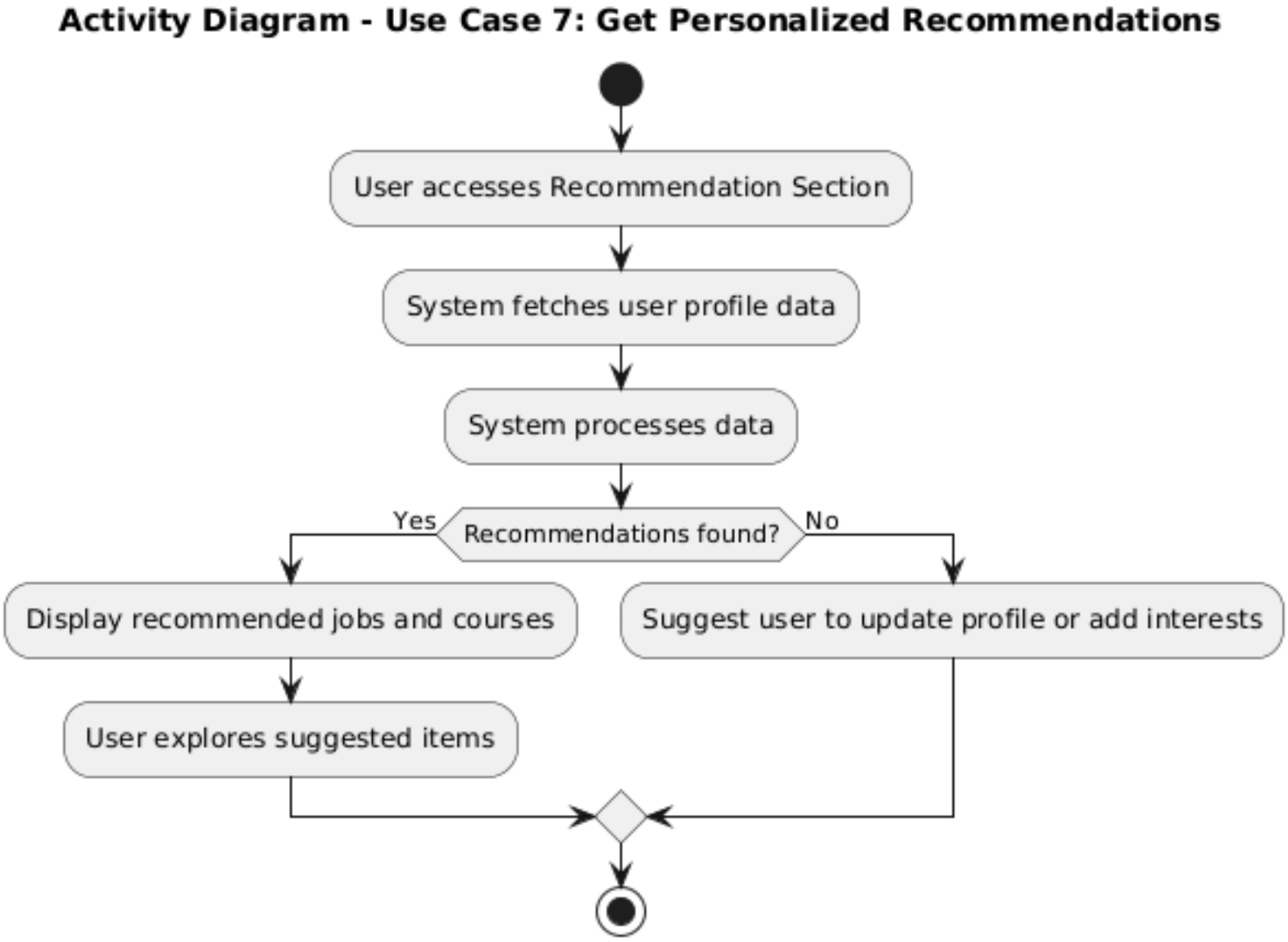
State Diagram - Use Case 6: Explore Government Policies and Courses



Use Case 7: Get Personalized Recommendations

Description: The system suggests jobs and courses based on the user's profile.

Activity Diagram

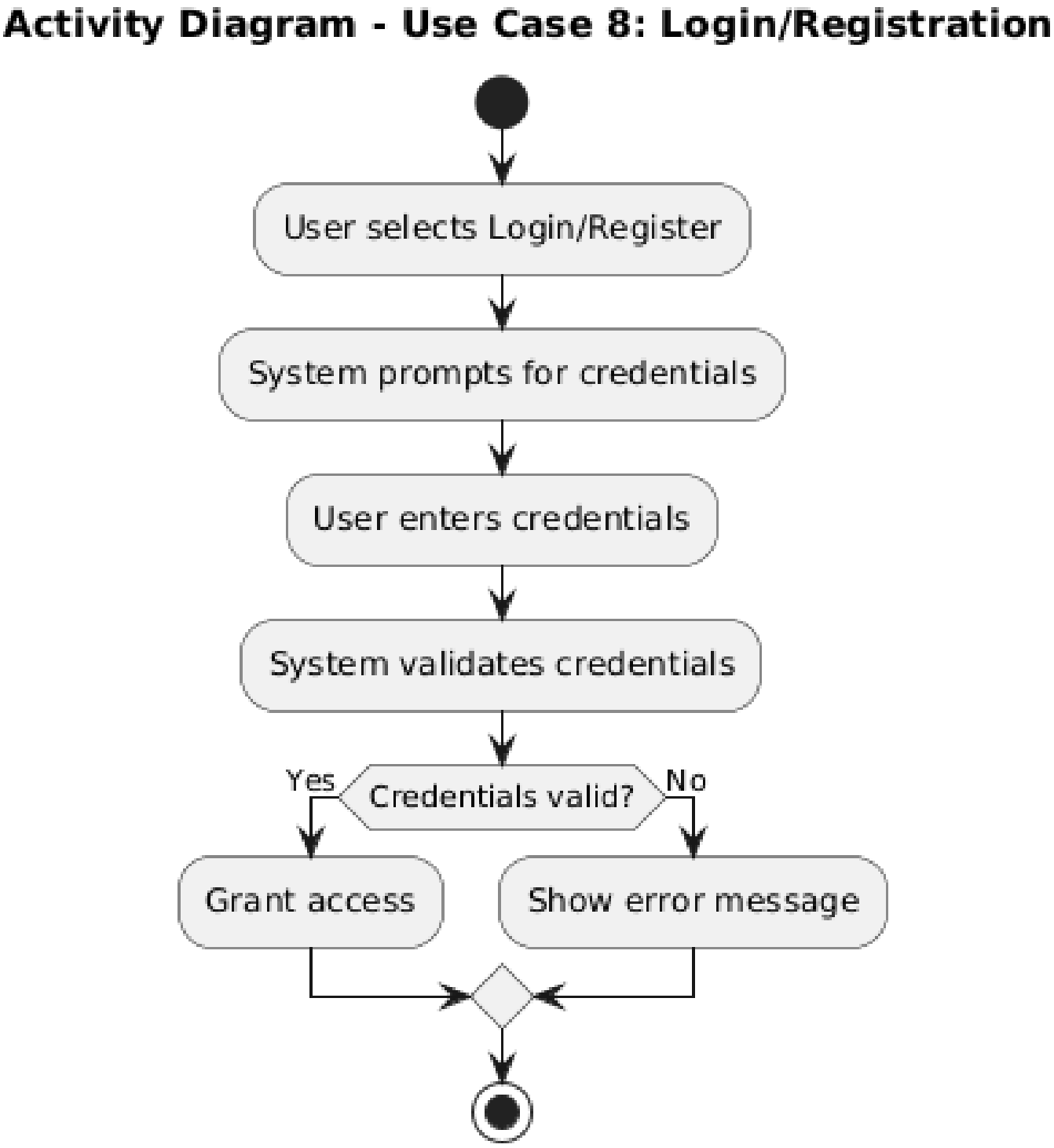


State Diagram

Use Case 8: Login/Registration

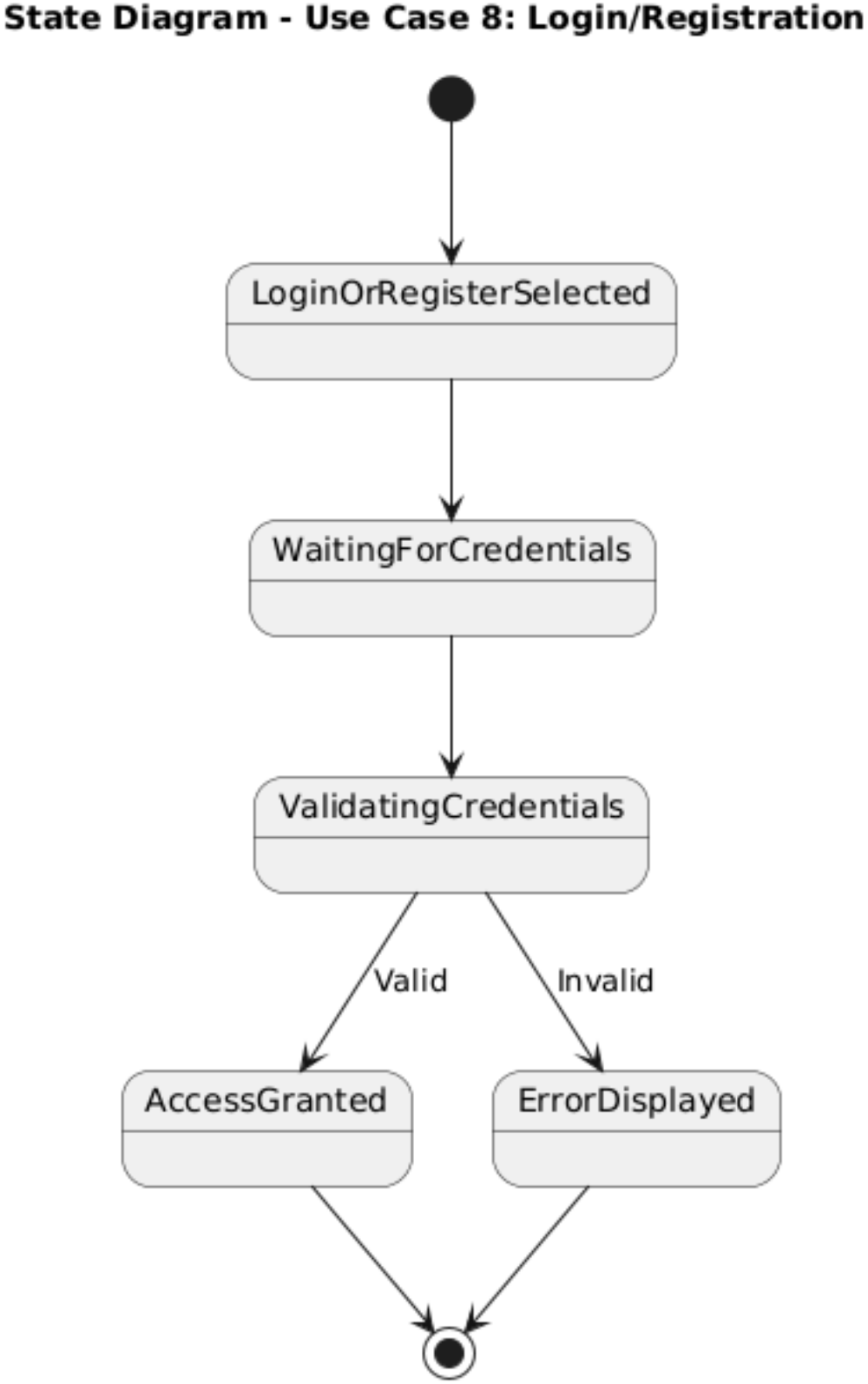
Description: A user registers or logs into their account.

Activity Diagram



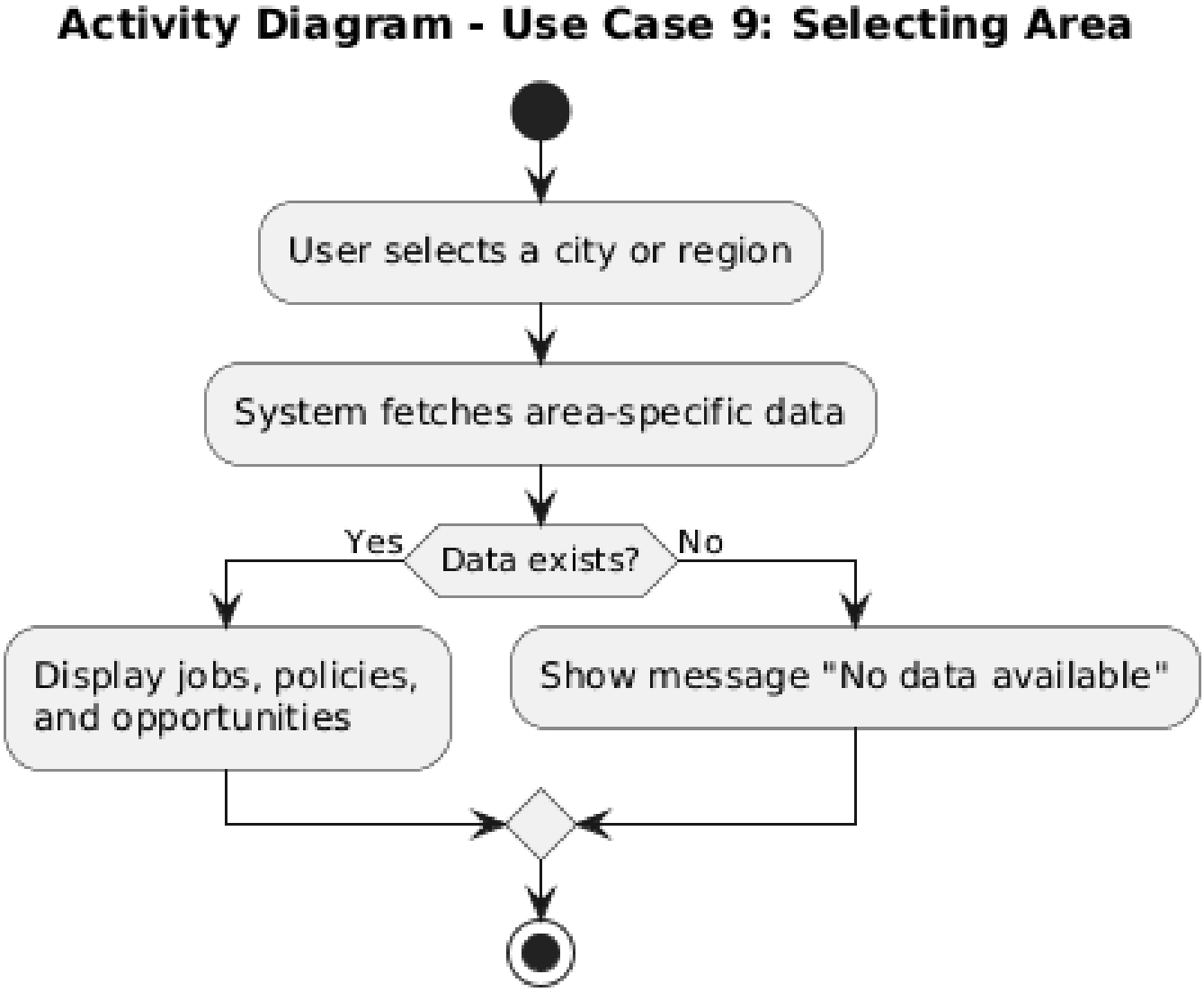
State Diagram

Use Case 9: Selecting Area

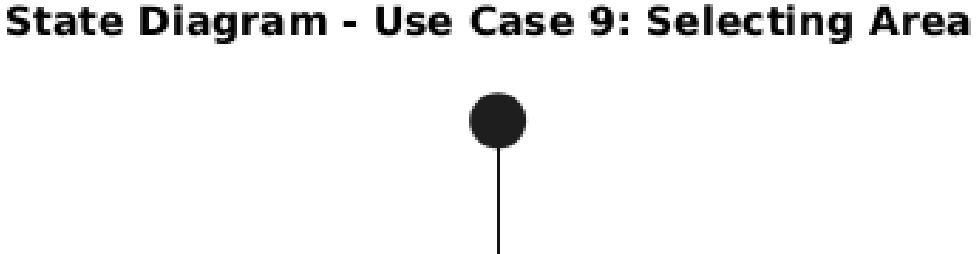


Description: The user selects a specific area to view related data.

Activity Diagram



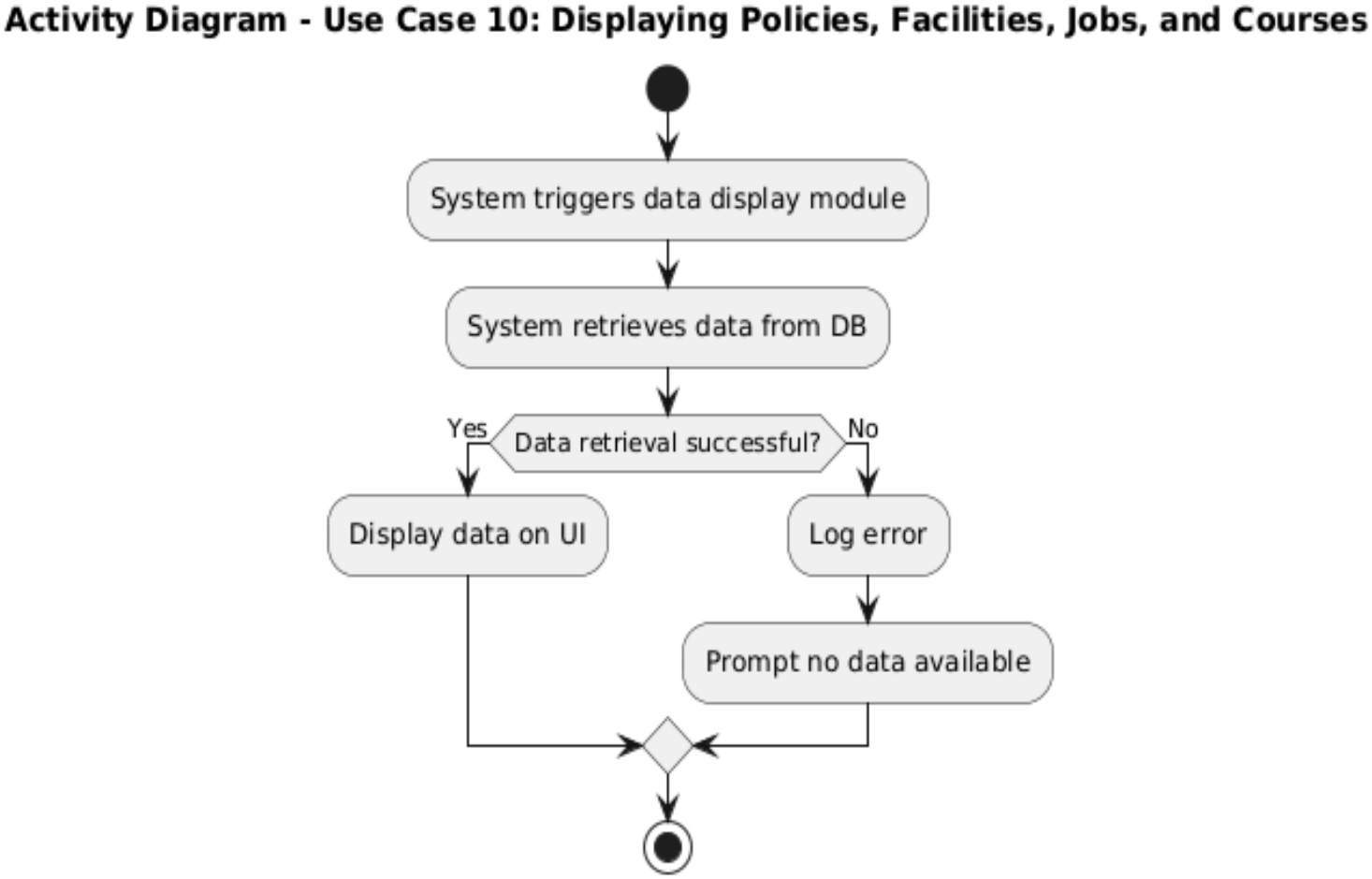
State Diagram



Use Case 10: Displaying Policies, Facilities, Jobs, and Courses

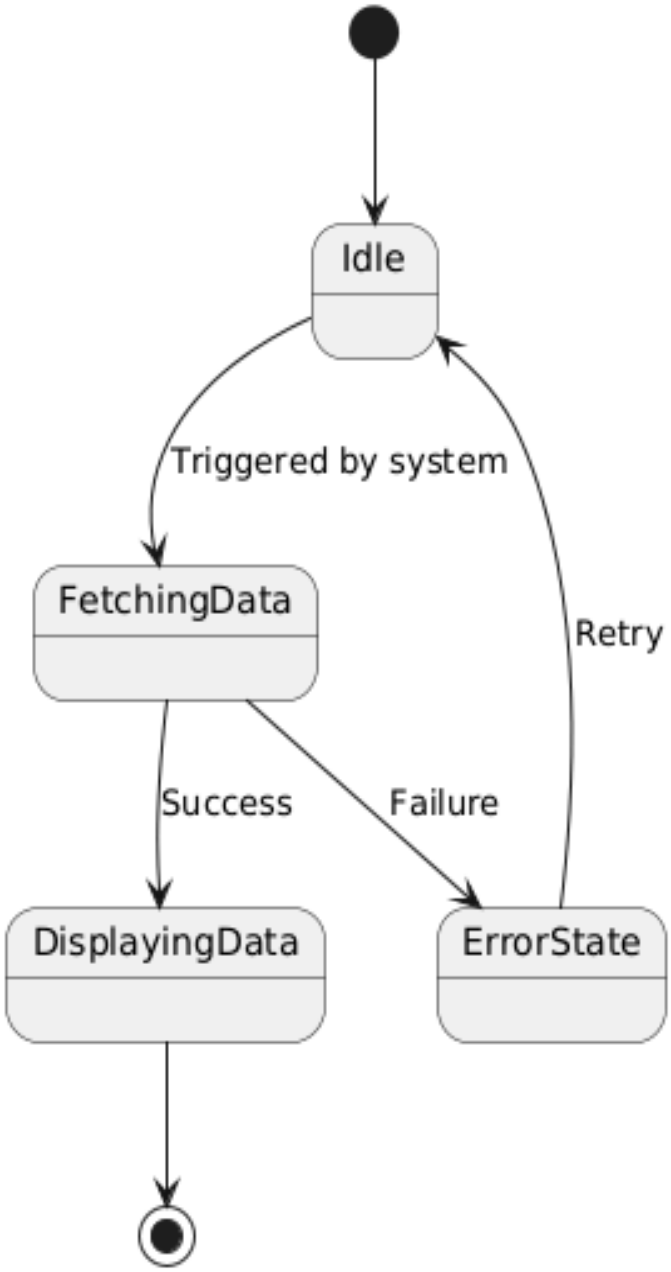
Description: The system fetches and displays relevant data for users.

Activity Diagram



State Diagram

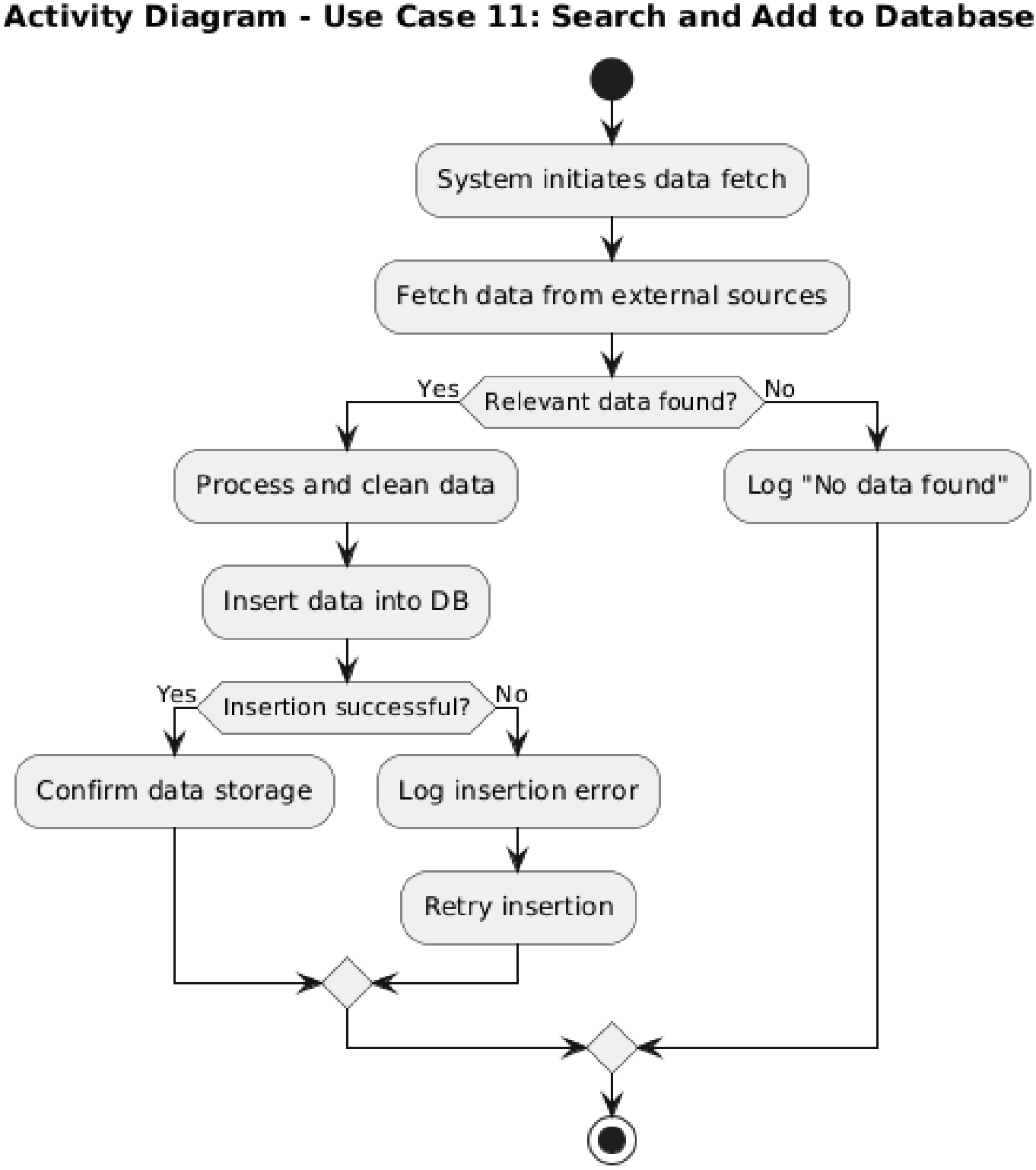
State Diagram - Use Case 10: Displaying Policies, Facilities, Jobs, and Courses



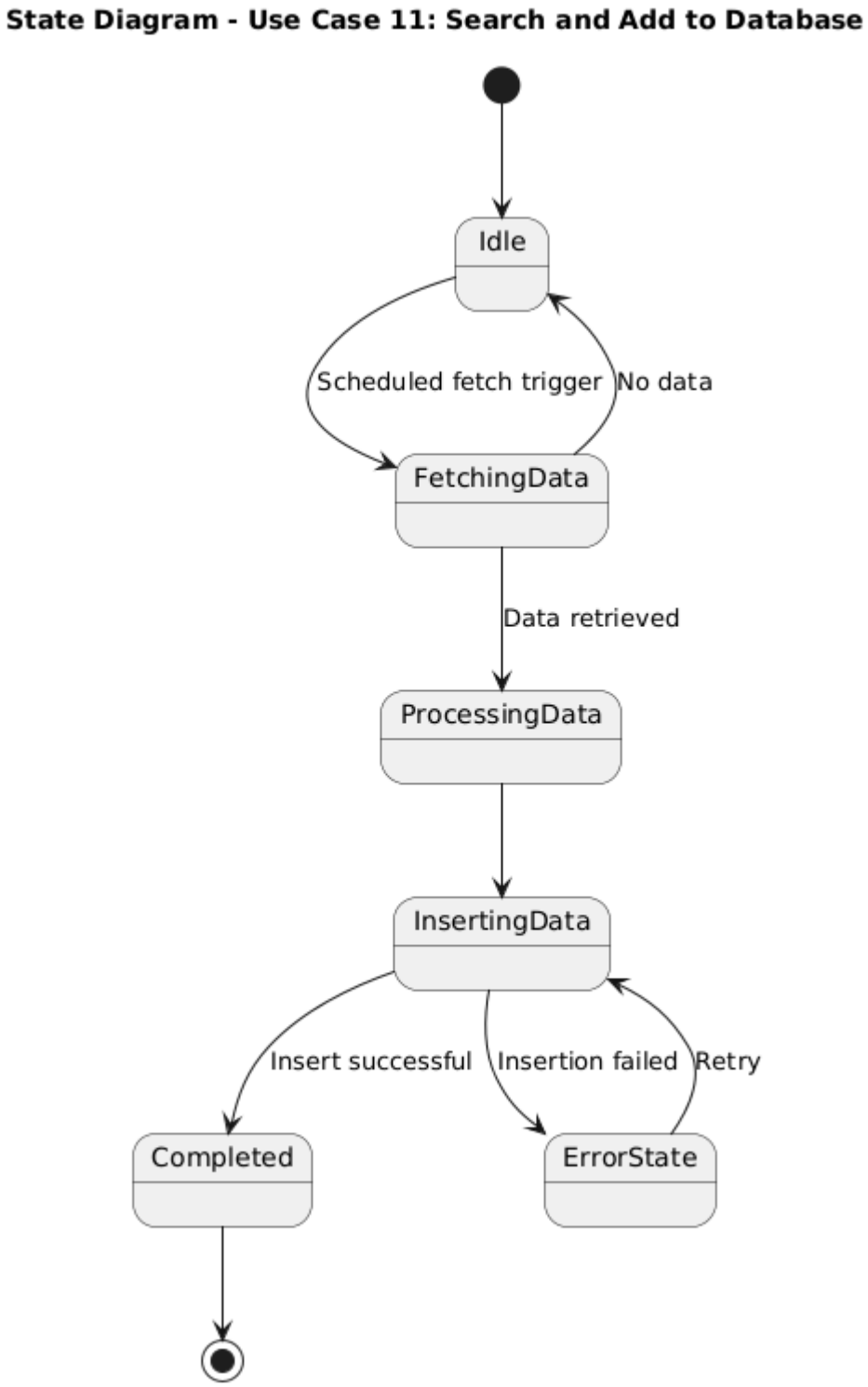
Use Case 11: Search and Add to Database

Description: The system searches for and stores job/policy/course data in the database.

Activity Diagram



State Diagram



OCL Constraints

General constraints

- context User

  inv ValidAge: self.age >0 and self.age <= 100

- context User

  inv ValidProfile: not self.name.oclIsUndefined() and not self.location.oclIsUndefined()

- context Job

  inv SalaryPositive: self.salary > 0

- context Course

  inv DurationPositive: self.duration > 0

  Inv Rating: self.rating >=0 and self.rating <=5

- context Hospital

  inv Rating: self.rating >=0 and self.rating <=5

- context Institute

  inv Establishment_year: self.establishment_year > 0

- context Employer

  inv Average_salary: self.average_salary >= 0

- context Policy

  inv ValidRegionReference: self.region.oclIsKindOf(State)

- context City

  inv StateReference: not self.stateId.oclIsUndefined()

Login/registration use case

context User::login(email: String, password: String)

pre: not email.oclIsUndefined() and not password.oclIsUndefined()

post: self.isLoggedIn = true

ProfileUpdate use case

context User::updateProfile(newLocation: City, newInterests: Set(String))

pre: self.isLoggedIn

post: self.location = newLocation and self.interests = newInterests

Explore Jobs / Apply for Job use case

context User::applyForJob(job: Job)

pre: self.isLoggedIn and not self.resume.oclIsUndefined()

post: job.applications->includes(self)

context Job

inv MustBelongToCity: not self.location.oclIsUndefined() and self.location.oclIsKindOf(City)

Explore Health Facilities use case

context Hospital

inv MustHaveCityReference: not self.cityId.oclIsUndefined()

Explore Courses and Policies

context Course

inv HasInstituteReference: not self.instituteId.oclIsUndefined()

context Policy

inv PolicyAppliesToState: self.region.oclIsKindOf(State)

Personalized Recommendations

context Recommendation

inv RelevantToUser: self.user.age >= 15 and self.user.location = self.recommendedItem.location

Area Selection

context User::selectArea(newCity: City)

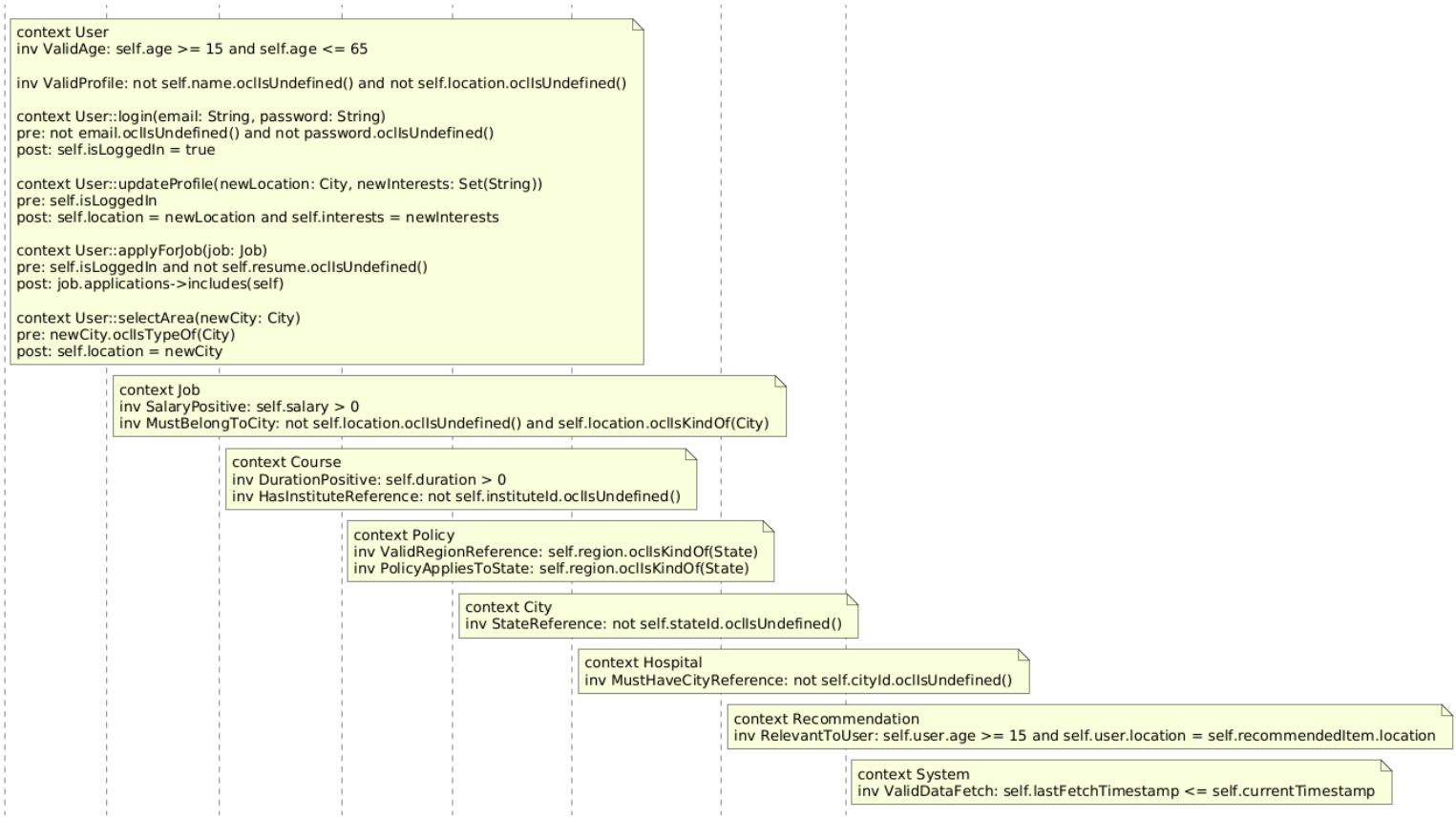
pre: newCity.oclIsTypeOf(City)

post: self.location = newCity

Database Data Fetching

context System

inv ValidDataFetch: self.lastFetchTimestamp <= self.currentTimestamp


context User
inv ValidAge: self.age >= 15 and self.age <= 65
inv ValidProfile: not self.name.oclIsUndefined() and not self.location.oclIsUndefined()
context User::login(email: String, password: String)
pre: not email.oclIsUndefined() and not password.oclIsUndefined()
post: self.isLoggedIn = true
context User::updateProfile(newLocation: City, newInterests: Set(String))
pre: self.isLoggedIn
post: self.location = newLocation and self.interests = newInterests
context User::applyForJob(job: Job)
pre: self.isLoggedIn and not self.resume.oclIsUndefined()
post: job.applications->includes(self)
context User::selectArea(newCity: City)
pre: newCity.oclIsTypeOf(City)
post: self.location = newCity
context Job
inv SalaryPositive: self.salary > 0
inv MustBelongToCity: not self.location.oclIsUndefined() and self.location.oclIsKindOf(City)
context Course
inv DurationPositive: self.duration > 0
inv HasInstituteReference: not self.instituteId.oclIsUndefined()
context Policy
inv ValidRegionReference: self.region.oclIsKindOf(State)
inv PolicyAppliesToState: self.region.oclIsKindOf(State)
context City
inv StateReference: not self.stateId.oclIsUndefined()
context Hospital
inv MustHaveCityReference: not self.cityId.oclIsUndefined()
context Recommendation
inv RelevantToUser: self.user.age >= 15 and self.user.location = self.recommendedItem.location
context System
inv ValidDataFetch: self.lastFetchTimestamp <= self.currentTimestamp

1. User Class Constraints

The User class includes essential attributes such as location, isAuthenticated, and currentState.

Constraint 1: Location is Required for Users

context User

inv LocationIsRequired:

self.location->notEmpty()

This constraint ensures that the location attribute for a User is always set. A user must have a location specified in order to get personalized recommendations.

Constraint 2: User Can Access Dashboard Only if Authenticated

context User

inv CanAccessDashboard:

self.isAuthenticated = true implies self.currentState = 'Dashboard'

This constraint ensures that only authenticated users can access the Dashboard. If the user is authenticated (isAuthenticated = true), then their current state must be set to Dashboard.

2. Job Class Constraints

The JobListing class represents job opportunities, and it contains attributes like location, description, and preferences.

Constraint 1: Job Listing Must Have a Location and Description

context JobListing

inv ValidJobData:

self.location->notEmpty() and self.description->notEmpty()

This constraint ensures that a Job Listing must have both a location and a description. It validates that these fields are filled out before the job listing is considered valid.

Constraint 2: Job Preferences Should Match User Preferences

context JobListing

inv JobPreferencesMatch:

self.jobPreferences->exists(p | p = self.user.preferences)

This constraint ensures that the job preferences of the listing must match the user preferences. It checks if there is any preference in the job listing that matches the user's preferences.

3. Institute Class Constraints

The Institute class holds attributes related to courses and institutions, such as location, isAccredited, and institution.

Constraint 1:Institute Must Be Accredited

context Institute

inv MustBeAccredited:

self.isAccredited = true

This constraint ensures that all Institutes listed in the system are accredited. The isAccredited attribute must be true for a program to be displayed.

Constraint 2: Institute Must Have a Location

context Institute

inv LocationRequired:

self.location->notEmpty()

This constraint ensures that the location for an Education Program must be specified before it is valid.

4. Health Class Constraints

The Health class represents healthcare facilities, including facilityType, location, and available wellness programs.

Constraint 1: Health Must Have a Facility Type

context Health

inv MustHaveFacilityType:

    self.facilityType->notEmpty()

This constraint ensures that every Health listed must have a facility type (e.g., hospital, clinic, pharmacy).

Constraint 2: Health Must Be Located in User's Region

context Health

inv FacilityLocationMatch:

    self.location = self.user.location

This constraint checks that the location of a Health matches the location of the User. It ensures that users only see healthcare services available in their own region.

5. Recommendation Engine Constraints

The Recommendation class serves to deliver personalized recommendations for jobs, education, and health services.

Constraint 1: Recommendations Must Match User's Location

context Recommendation

inv MustBeRelevantToUserLocation:

    self.user.location = self.recommendationLocation

This constraint ensures that all Recommendations given to a user are based on their location. The recommended service, job, or education program should be geographically relevant to the user.

Constraint 2: Recommendation Must Not Be Duplicated

context Recommendation

inv UniqueRecommendation:

    self.user.recommendations->forAll(r | r != self)

This constraint ensures that each recommendation is unique and that there are no duplicate recommendations for a user. This helps maintain a clean and relevant list of recommendations for each user.

6. GovernmentPolicy Class Constraints

The GovernmentPolicy class includes various government support programs, such as grants, scholarships, and healthcare subsidies.

Constraint 1: Government Policy Must Be Applicable to User's Region

context GovernmentPolicy

inv PolicyRegionMatch:

    self.region = self.user.region

This constraint ensures that a Government Policy is only applicable to users in the same region. It prevents users from accessing policies that are not available in their location.

Constraint 2: Policy Must Have a Valid End Date

context GovernmentPolicy

inv ValidEndDate:

    self.endDate >= self.startDate

This constraint ensures that a Government Policy has a valid end date that is not earlier than its start date.

7. General Data Integrity Constraints

These constraints ensure that data integrity is maintained across all classes.

Constraint 1: User Profile Must Be Complete

context User

inv ProfileComplete:

self.firstName->notEmpty() and self.lastName->notEmpty() and self.email->notEmpty() and self.location->notEmpty()

This constraint ensures that a User's profile is complete before they can access the system. It checks that the first name, last name, email, and location are not empty.

Constraint 2: No Duplicate Job Listings

context JobListing

inv UniqueJobListing:

self.description->forAll(d | self != d)

This constraint ensures that job listings are unique by description. It prevents duplicate job listings from being added to the system.

Test Plan

for

NextStep

Version 1.0

Prepared by

Aarushi Goel	**202412002**	**202412002@daiict.ac.in**
Mohsin Pathan	**202412070**	**202412070@daiict.ac.in**
Annirudh Pratap	**202412008**	**202412008@daiict.ac.in**

Instructor: Prof. Jayprakash Lalchandani

Course: Software Engineering (IT 632)

Date: 4th May 2025

1. Test Plan Identifier

TP-NextStep-001

2. Introduction

The NextStep application is a modern web-based platform that delivers personalized career, education, healthcare, and policy recommendations. This test plan covers functional and UI testing, defining the purpose, scope, objectives, and key deliverables to ensure the quality and reliability of core features.

3. Test Items

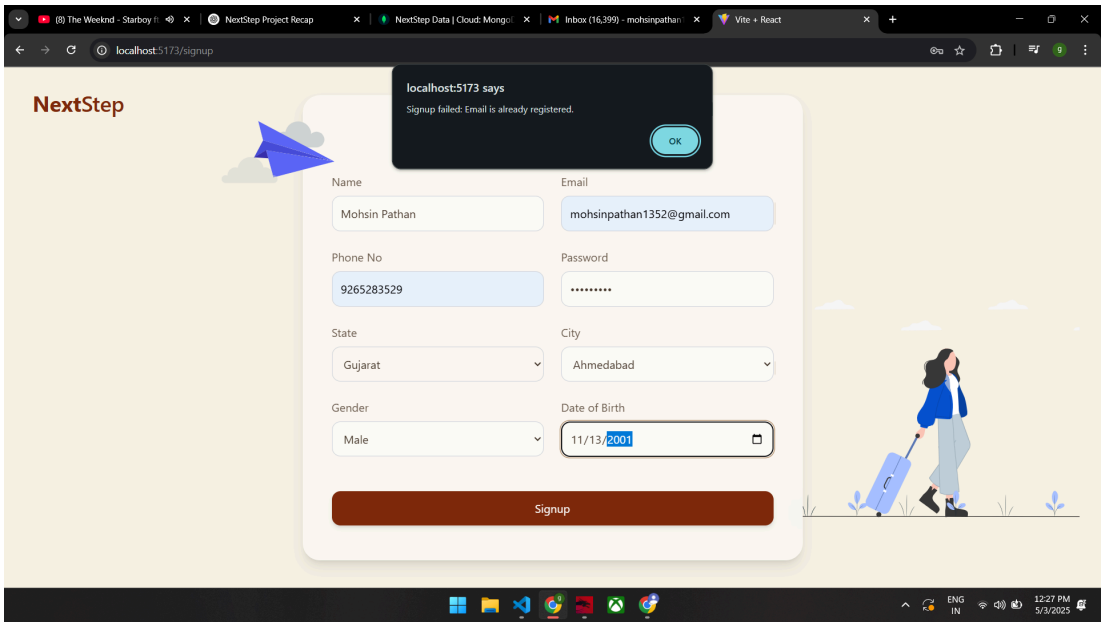
Modules and features to be tested include:

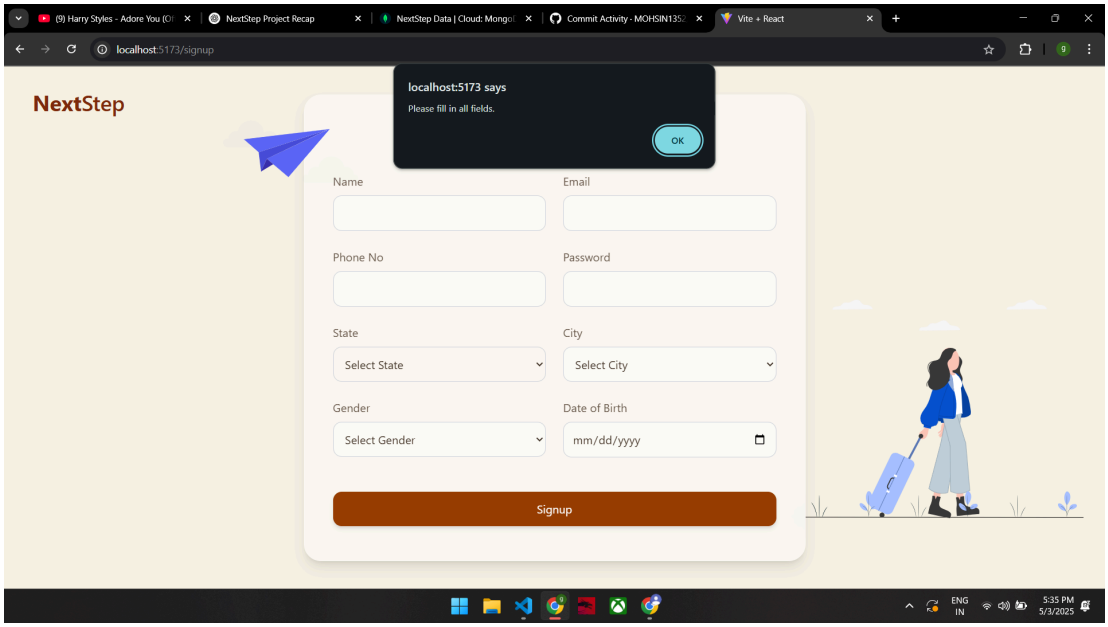
- User Registration
- Authentication
- User Profile Management
- Government Policies
- Health Facilities
- Opportunity Search
- Recommendation Engine
- All User Interface Screens (Registration, Login, Dashboard, Search & Filters, Profile, Opportunity Details, Job/Education/Healthcare/Policy Interfaces)

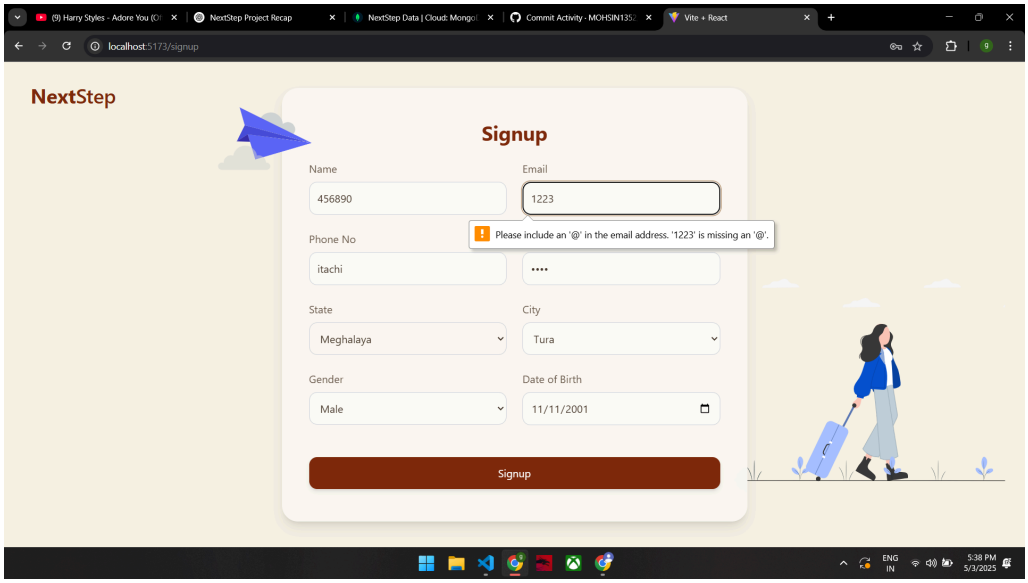
4. Features to be Tested

- **User Authentication Flows**: Registration, email verification, login/logout, password reset

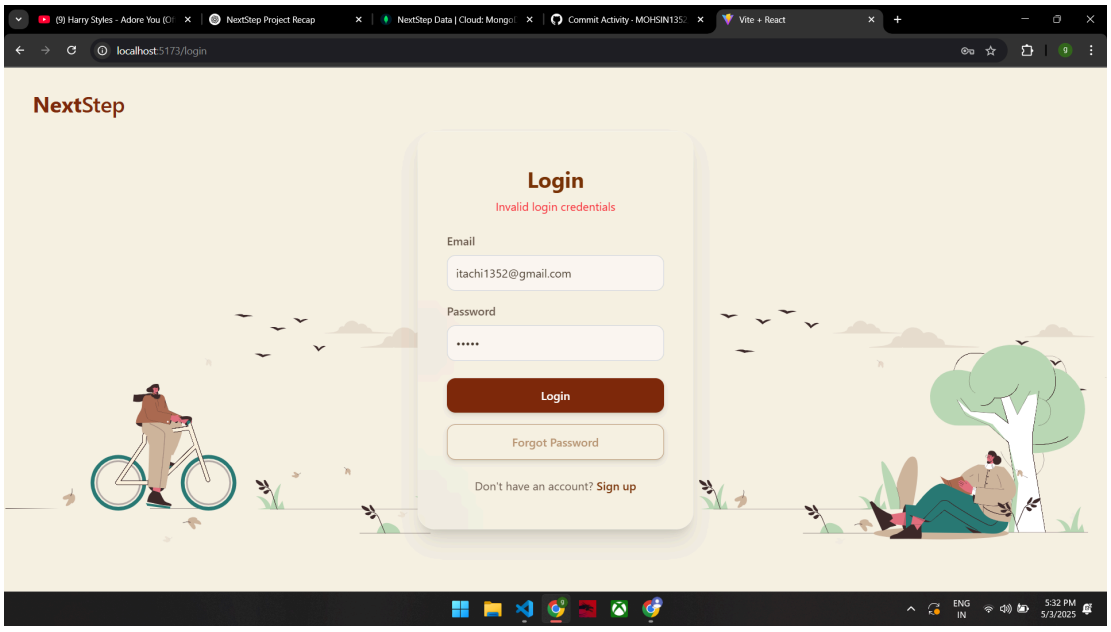
1. Registration not possible if already registered, or empty registration

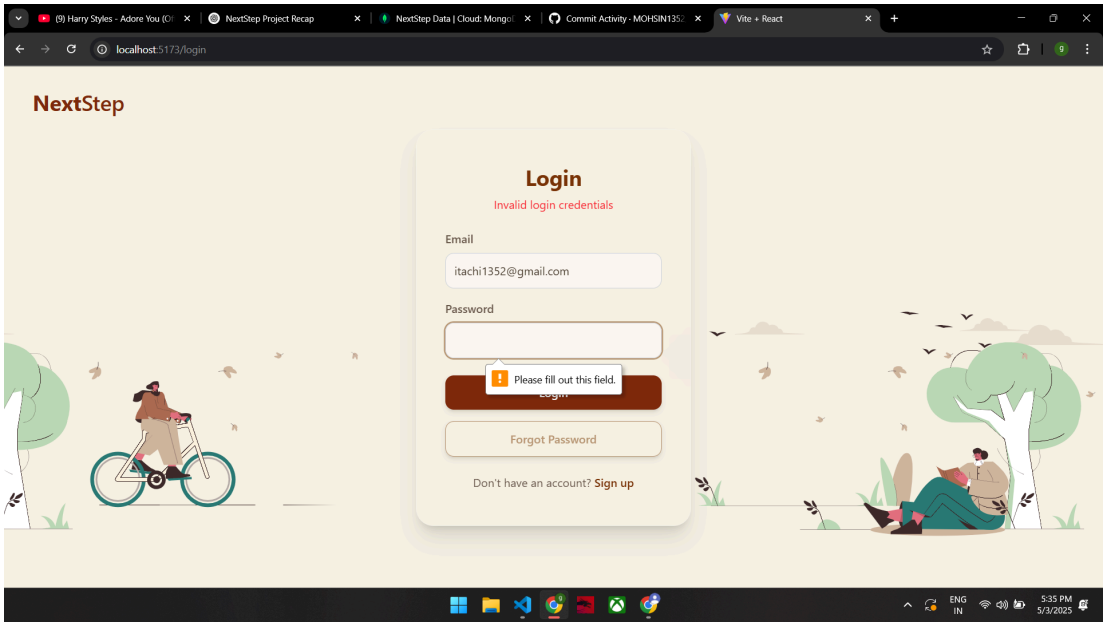




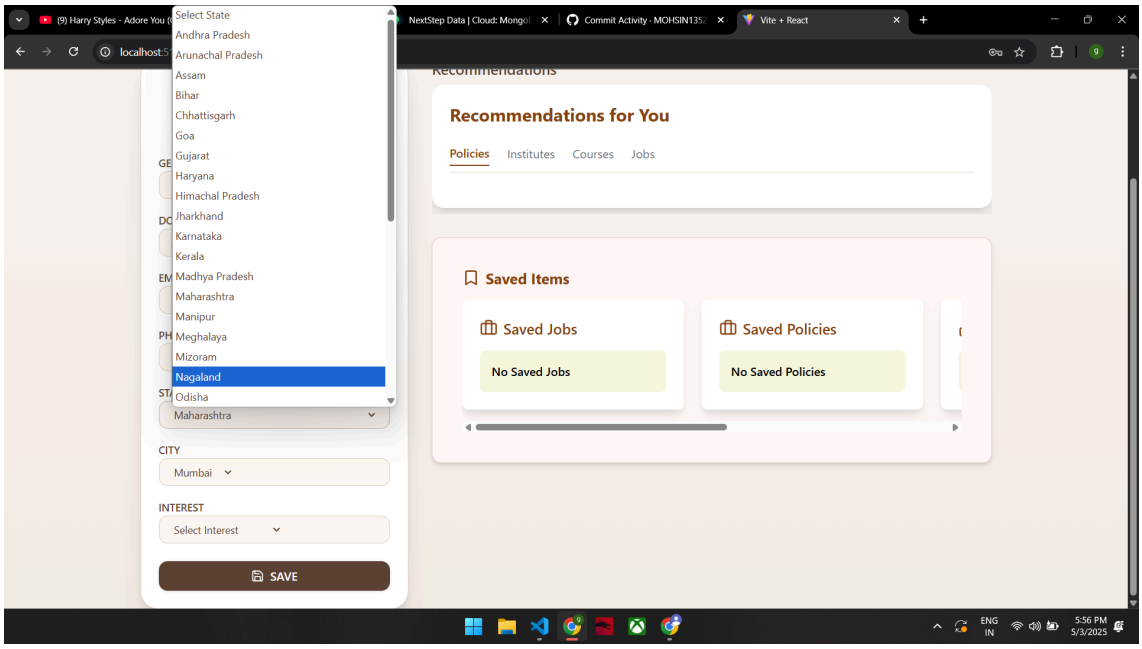


2. Login without registration, without no credentials or with wrong credentials not possible

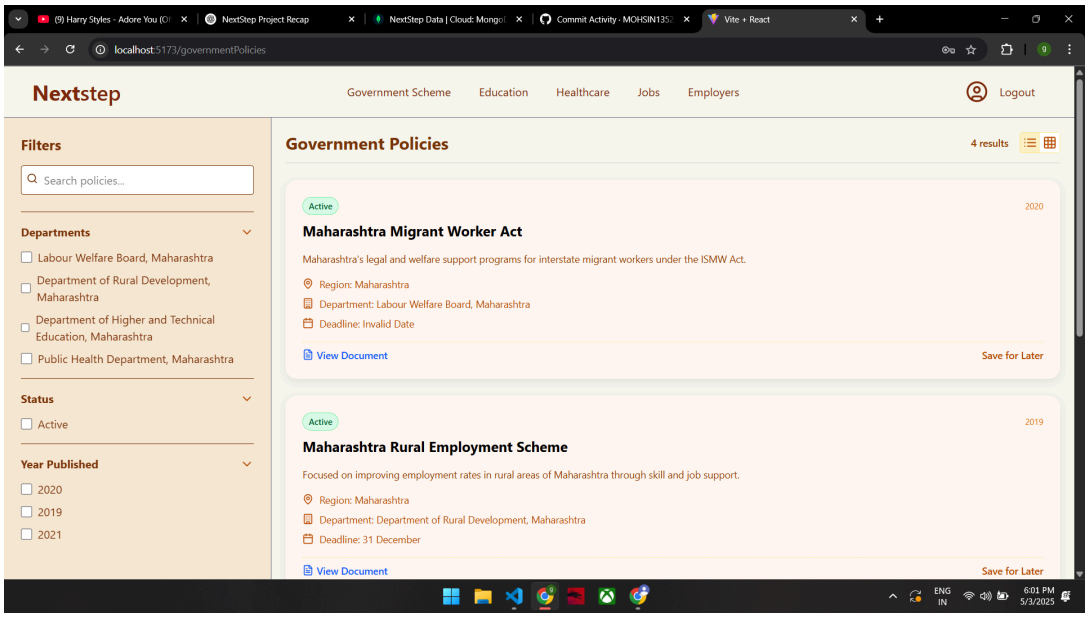




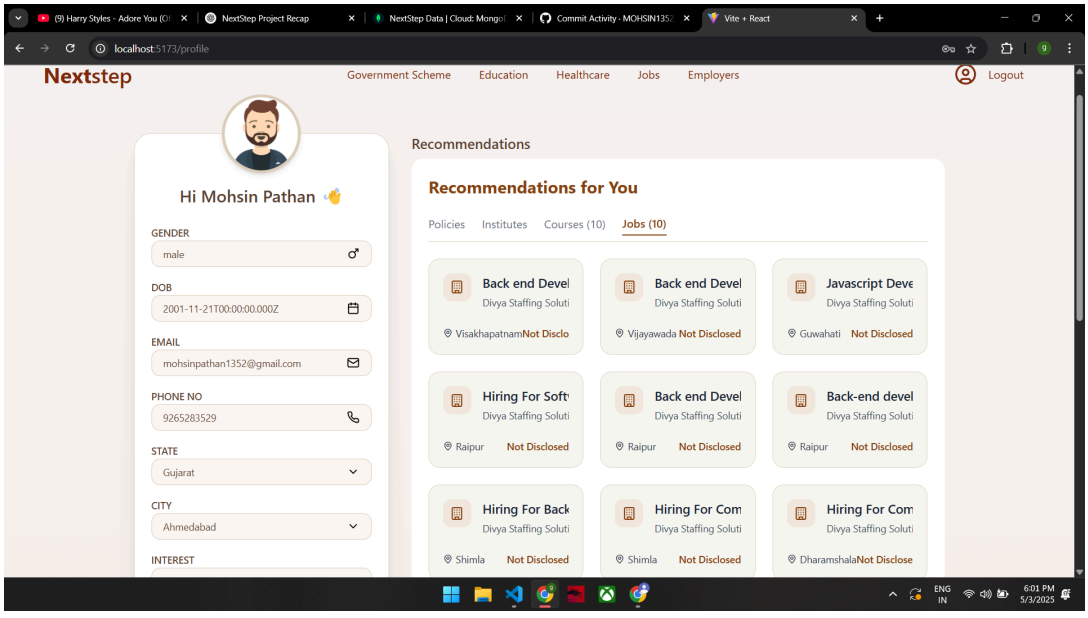
- **Profile Management**: Update profile data



- **Policy & Health Data Display**: Location-based policies and nearby facilities

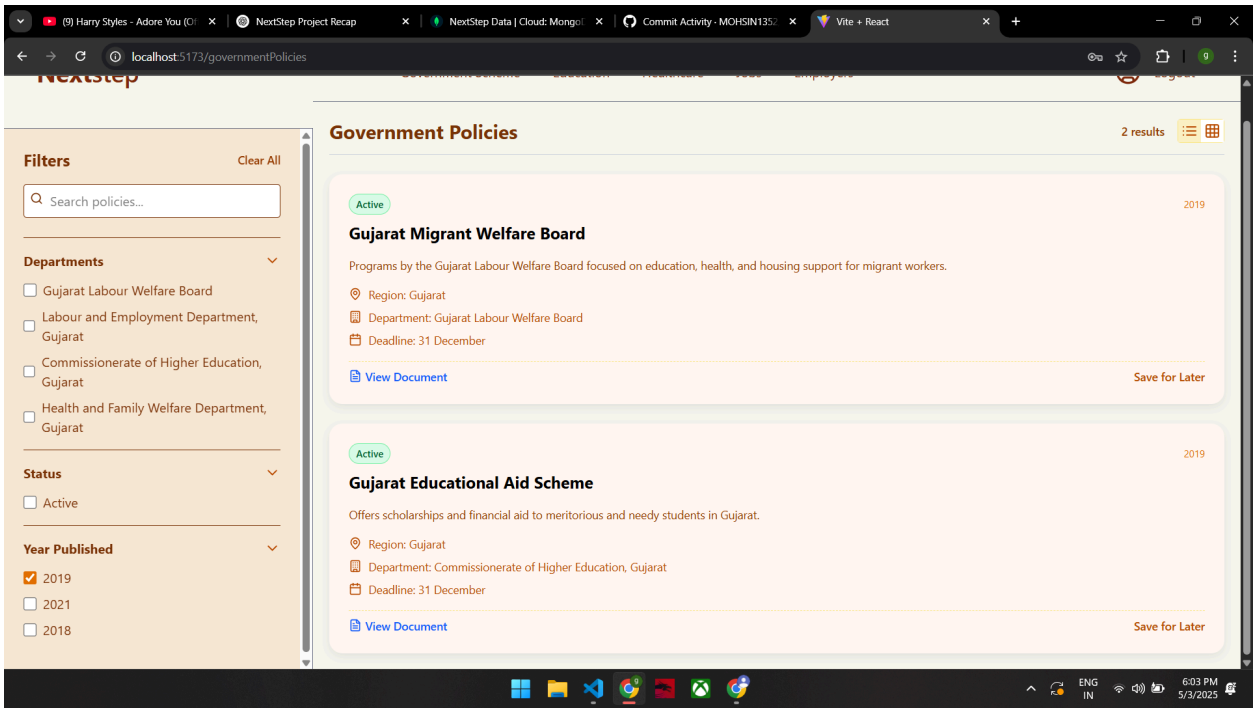


- **Opportunity Listings & Recommendations**: Region-filtered job and education opportunities



- **UI Functionality**: Field validations, navigation flows, search filters, pagination, detail page links.

  Filters

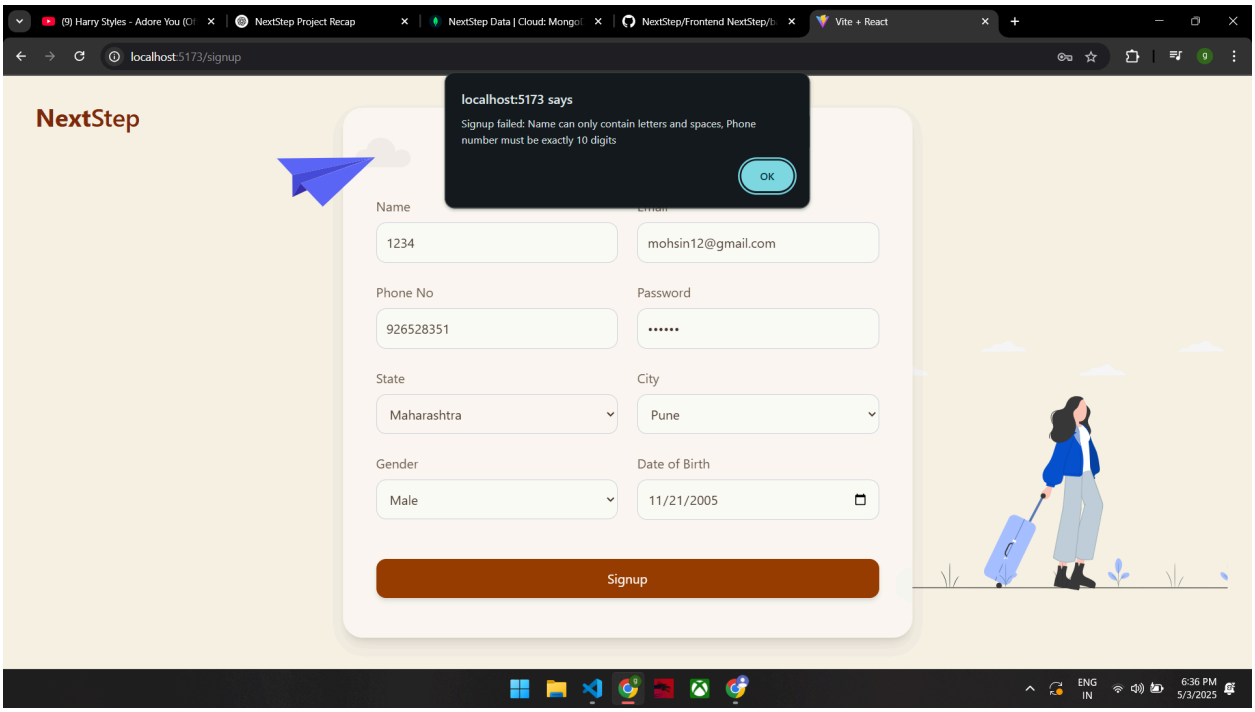


5. Features Not to be Tested

- Third-party payment gateways
- Performance/load/stress testing beyond basic responsiveness checks
- Legacy browser compatibility (IE11 or below)
- Non-core UI theming (color schemes, fonts beyond accessibility checks)

6. Approach

- **Manual Testing**: Execute test cases against UI screens and workflows for all modules
- **API Automation**: Postman collections for key endpoints (F1–F5)



- **UI Automation**: Selenium scripts covering critical scenarios (login → search → save → logout)
- **Black-Box Techniques**: Equivalence partitioning and boundary value analysis for form inputs
- **Traceability**: Map each test case to functional requirements and UI components

7. Item Pass/Fail Criteria

- **Pass**: Actual results match expected outcomes defined in test cases
- **Fail**: Any deviation from requirement, data inconsistency, or UI malfunction

8. Suspension Criteria and Resumption Requirements

- **Suspend Testing**:
  - Critical API unavailability or schema changes breaking core flows
  - Major UI navigation failure (e.g., inability to reach dashboard)
- **Resume Testing**:
  - Smoke tests for authentication and core UI workflows pass successfully
  - Blockers are fixed and retested

9. Test Deliverables

- Approved Test Plan document
- Detailed Test Cases & Scripts (manual and automated)
- Traceability Matrix
- Automated Test Suites (Postman, Selenium)
- Defect Reports and Logs
- Test Execution Summary and Metrics

10. Testing Tasks

- Review & Finalize Test Plan
- Design Manual Test Cases
- Prepare Test Environment & Data
- Execute Manual Tests (Core)
- Execute Manual Tests (UI)
- Develop & Run API Automation
- Develop & Run UI Automation
- Defect Retesting & Regression
- Compile Summary Report & Sign-off

11. Environmental Needs

- Node.js & MongoDB test servers
- Browsers: Chrome & Firefox (latest)
- Tools: Postman, Selenium WebDriver, Jira/TestRail

12. Responsibilities

- **Test Lead**: Plan coordination, reporting
- **QA Engineers**: Test design, execution, automation
- **Developers**: Defect resolution
- **DevOps**: Test environment provisioning

13. Risks and Contingencies

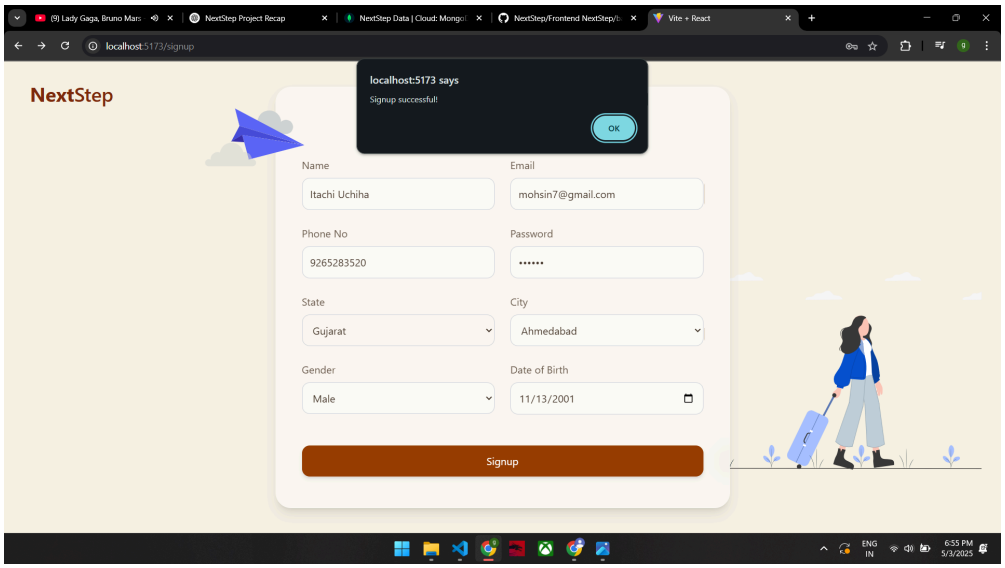
- **API Stability**: Use mock services if endpoints are unstable
- **Data Variability**: Generate synthetic location-based data sets

14. Requirement & UI Traceability Matrix

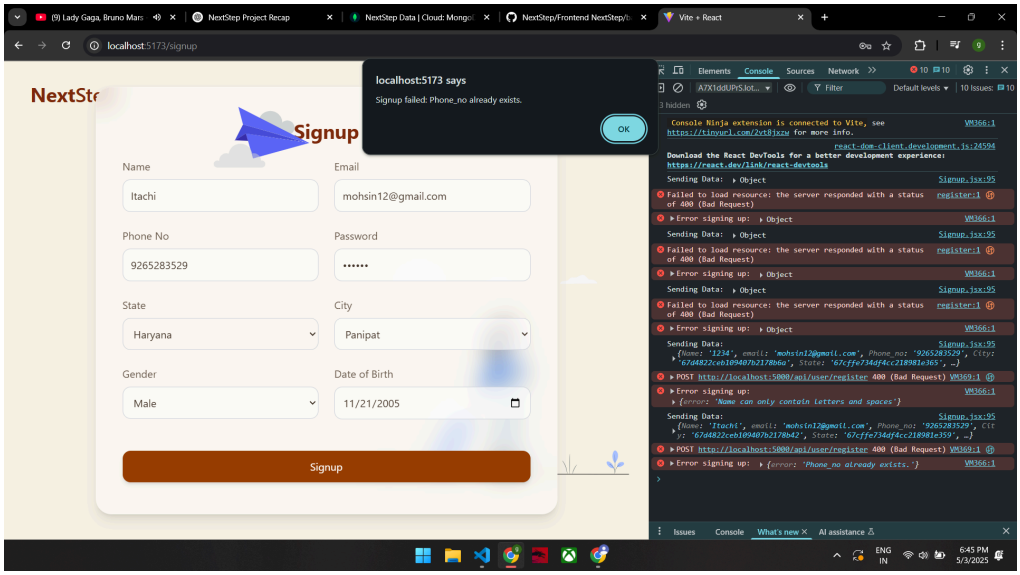
ID	Test Case IDs	Status
F1	TC-Reg-01- TC-Reg-05 (Registration)	[Successful, duplicate (fail), empty details(fail), not correct length for number(fail), wrong characters(fail)]
F2	TC-Auth-01 - TC-Auth-03 (Login)	[Successful, Empty credentials (fail), Wrong credentials (fail)]
F3	TC-Prof-01 - TC-Prof-03	[Normal, Changed, Changed]
F4	TC-Heal-01 - TC-Heal-02	[Available, Not available]
F5	TC-Pol-01 - TC-Pol-02	[Available, Filtered]
UI-1	TC-UI-1	Responsive
UI-2	TC-UI-2	Saved items Section
…	….	….

Manual Testing

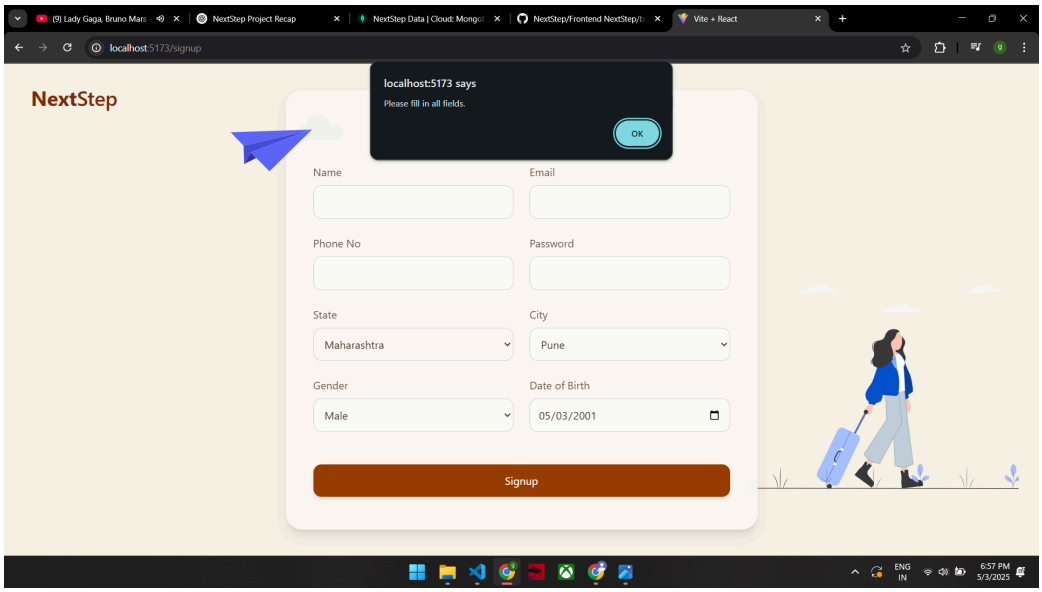
TC-Reg-01 - {Name: 'Itachi Uchiha', email: 'mohsin7@gmail.com', Phone_no: '9265283520', City: '67d4822ceb109407b2178b42', State: '67cffe734df4cc218981e359',}



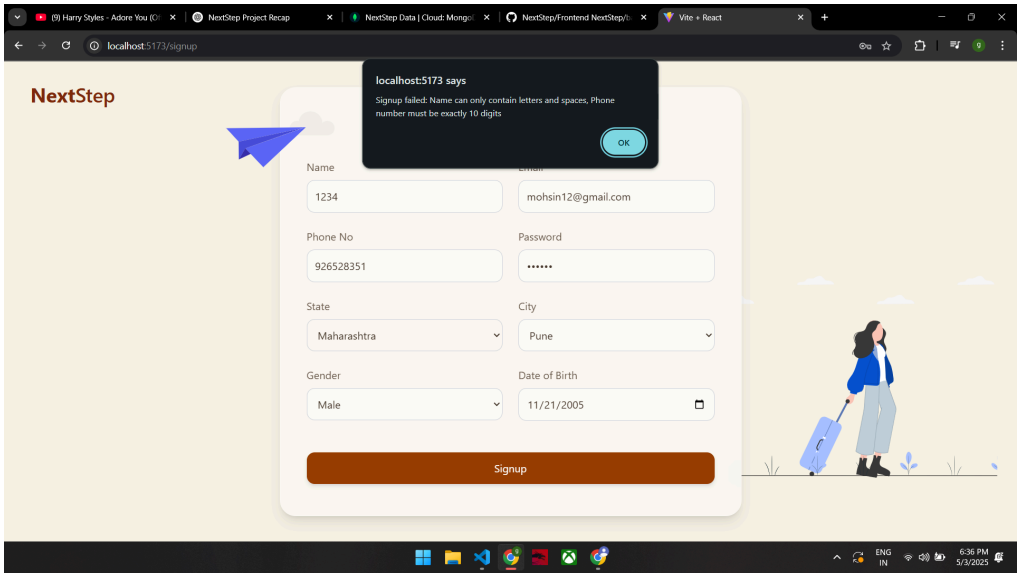
TC-Reg-02 - {Name: 'Itachi', email: 'mohsin3@gmail.com', Phone_no: '9265283520', City: '67d4822ceb109407b2178b42', State: '67cffe734df4cc218981e359',



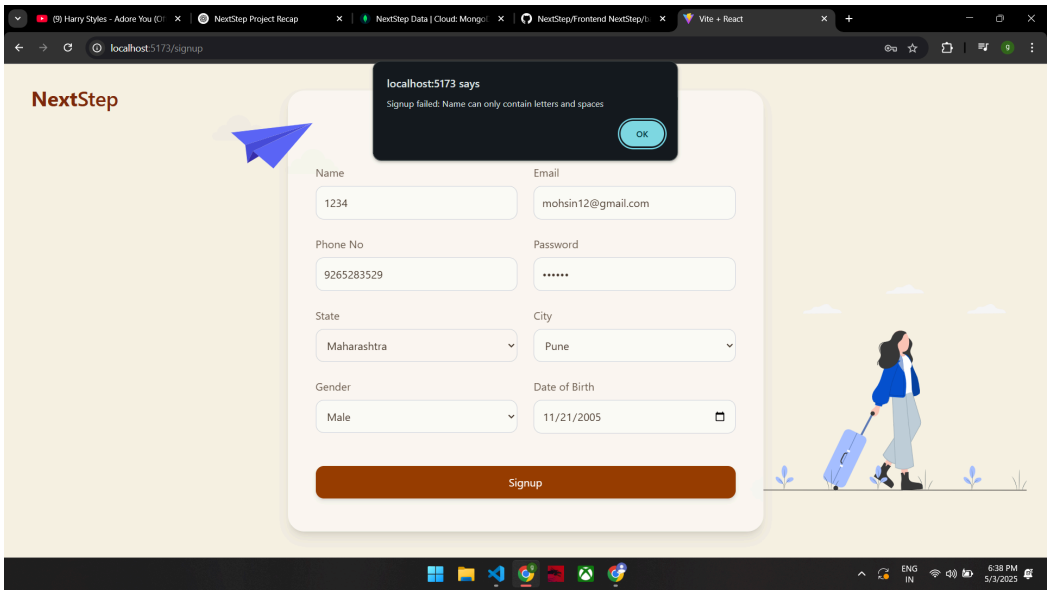
TC-Reg-03 - {Name: ' ', email: ' ', Phone_no: ' ', City: '67d4822ceb109407b2178b42', State: '67cffe734df4cc218981e359', …}



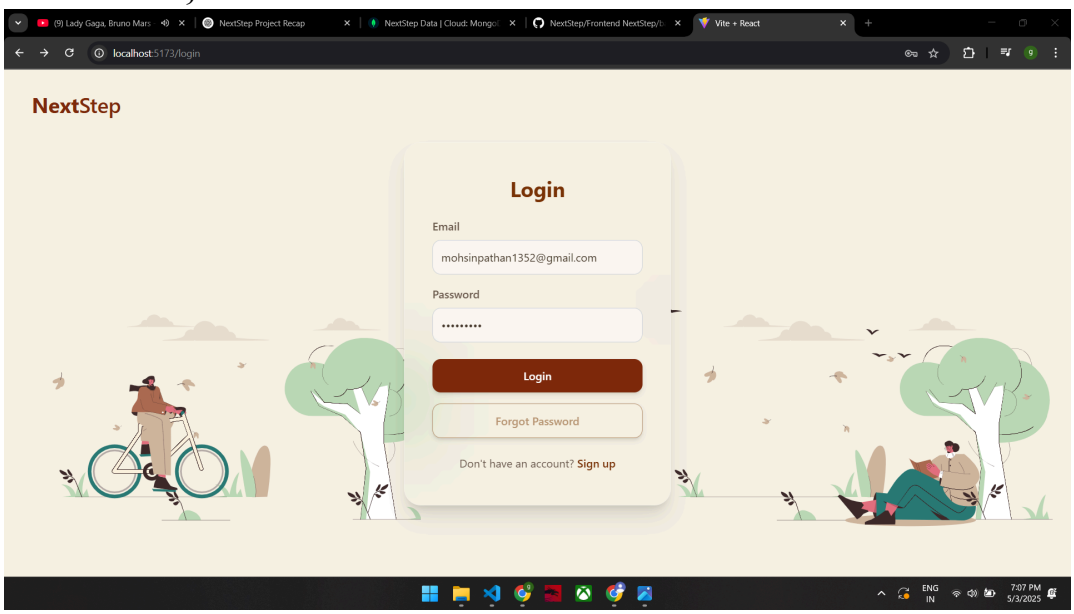
TC-Reg-04 - {Name: 'Hiii', email: 'mohsin42@gmail.com', Phone_no: '9265281', City: '67d4822ceb109407b2178b42', State: '67cffe734df4cc218981e359', …}



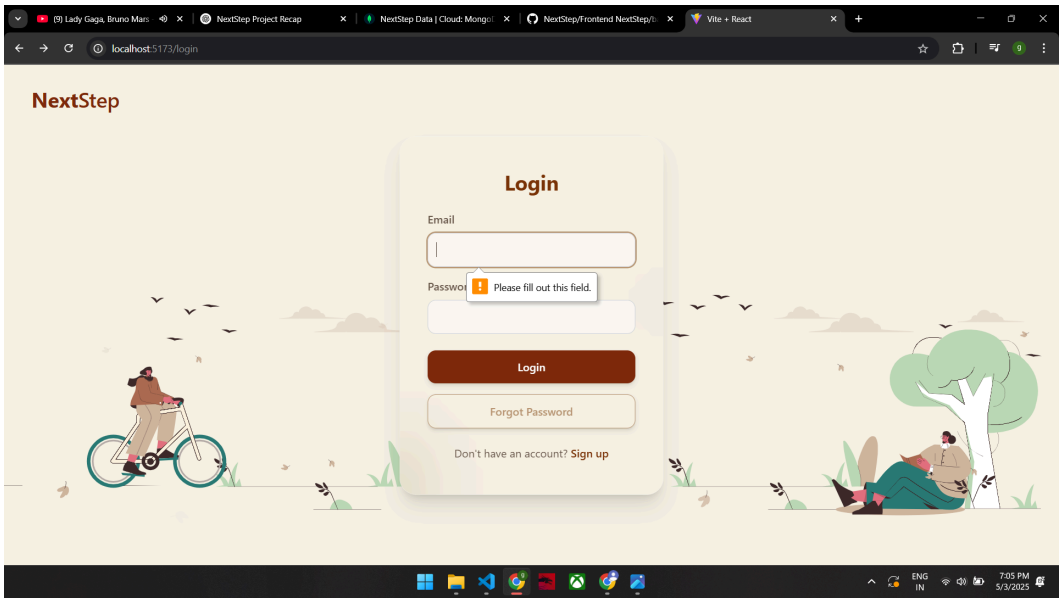
TC-Reg-05 - {Name: '1234', email: 'mohsin42@gmail.com', Phone_no: '9265281231', City: '67d4822ceb109407b2178b42', State: '67cffe734df4cc218981e359', …}



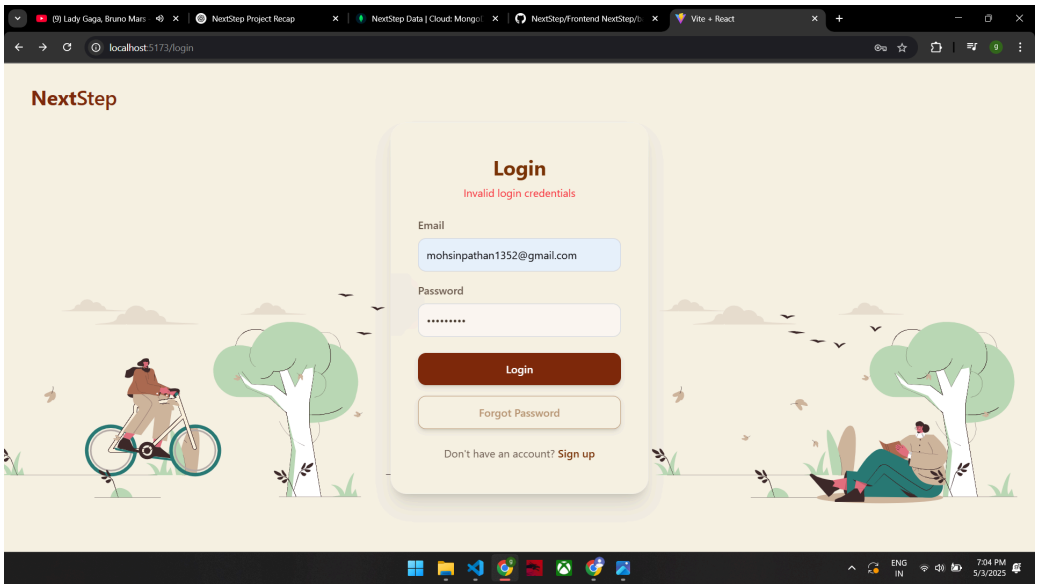
TC-Auth-01 - {Email : 'mohsinpathan1352@gmail.com' , Password: 'Mo*****'}



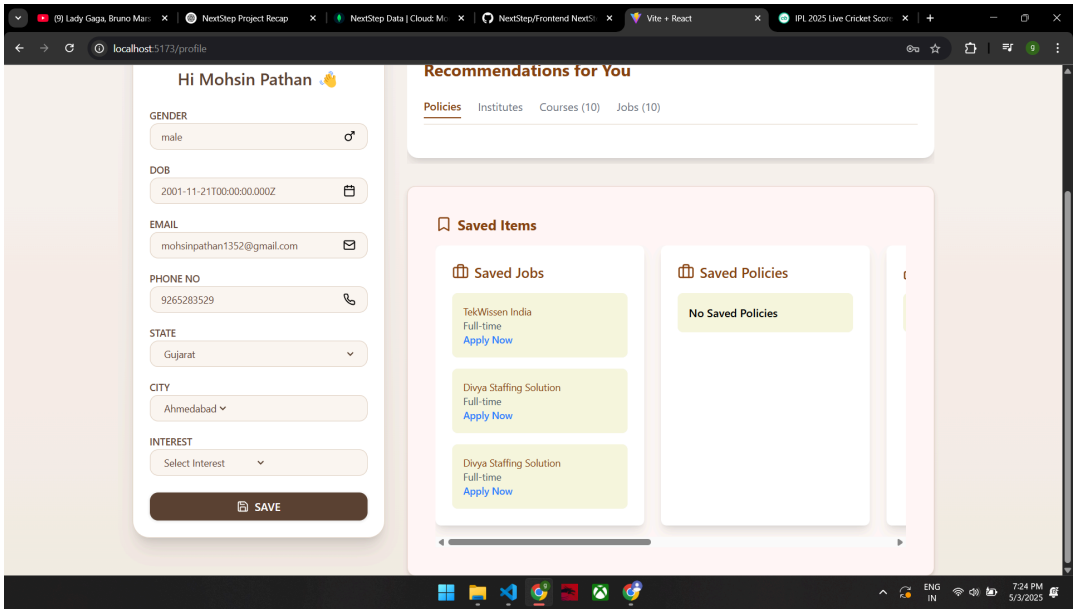
TC-Auth-02 - {Email : ‘’ , Password: ‘’}



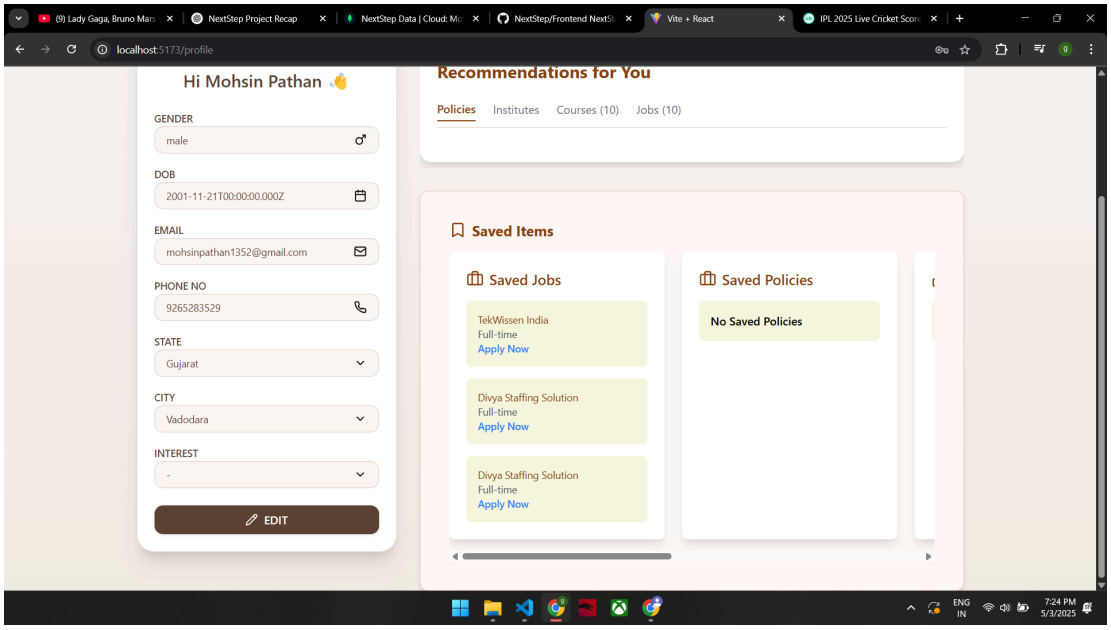
TC-Auth-03 - {Email : ‘mohsinpathan1352@gmail.com’ , Password: ‘bbq’}



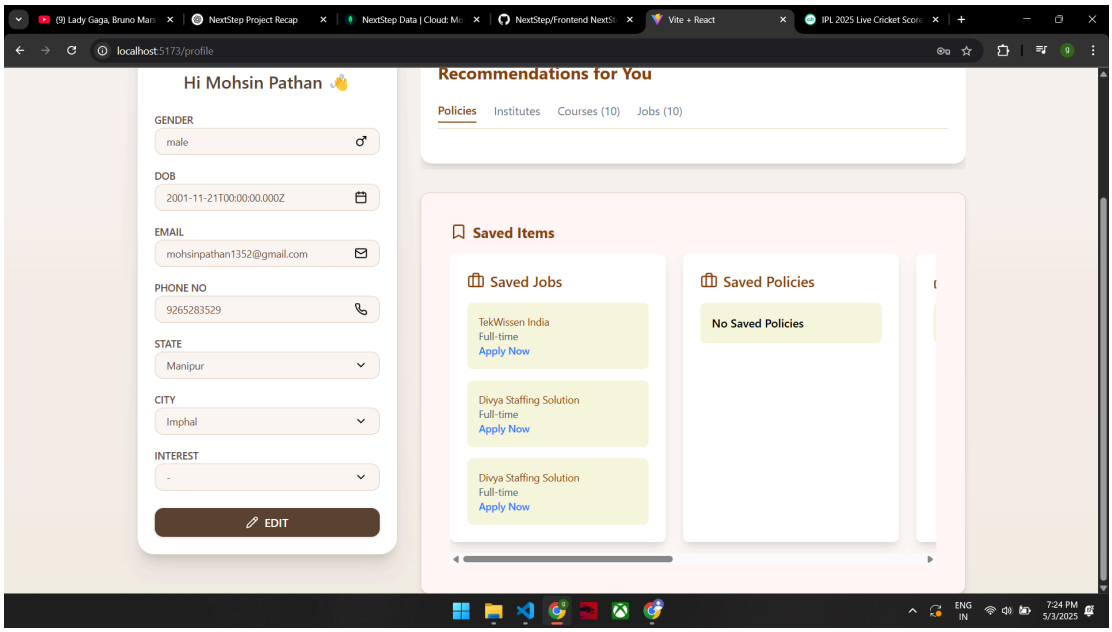
TC-Prof-01



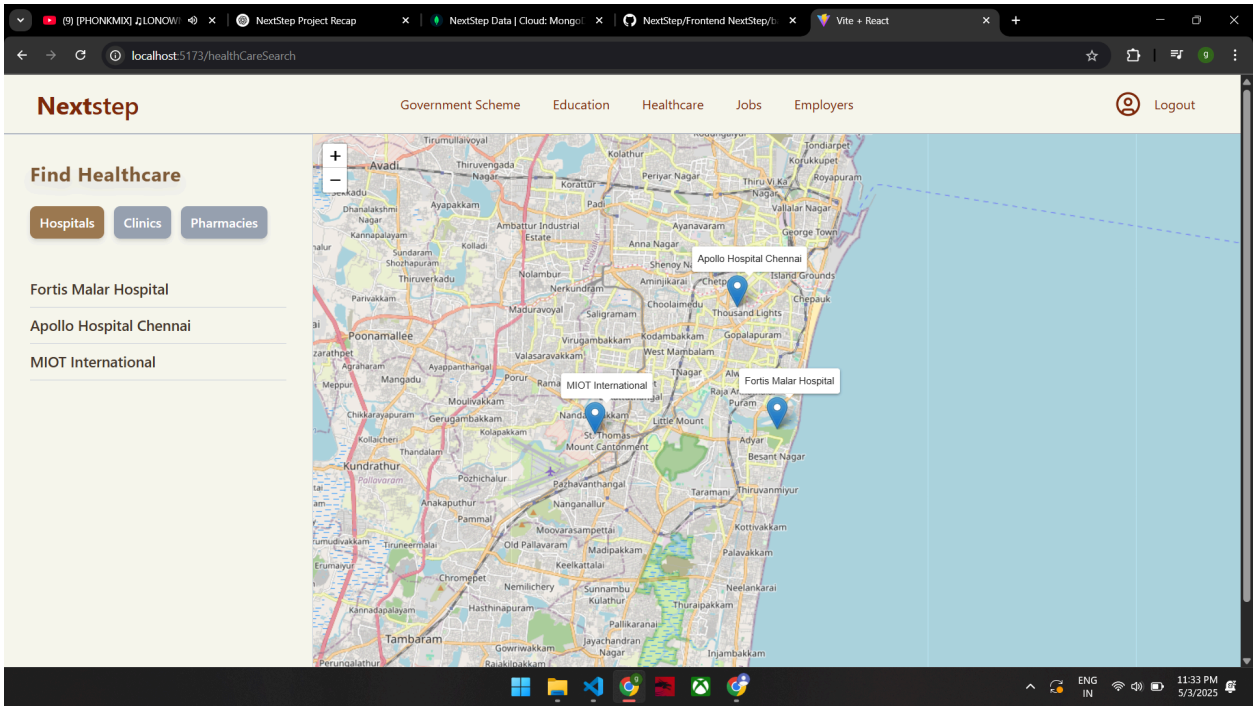
TC-Prof-02 (City Change)



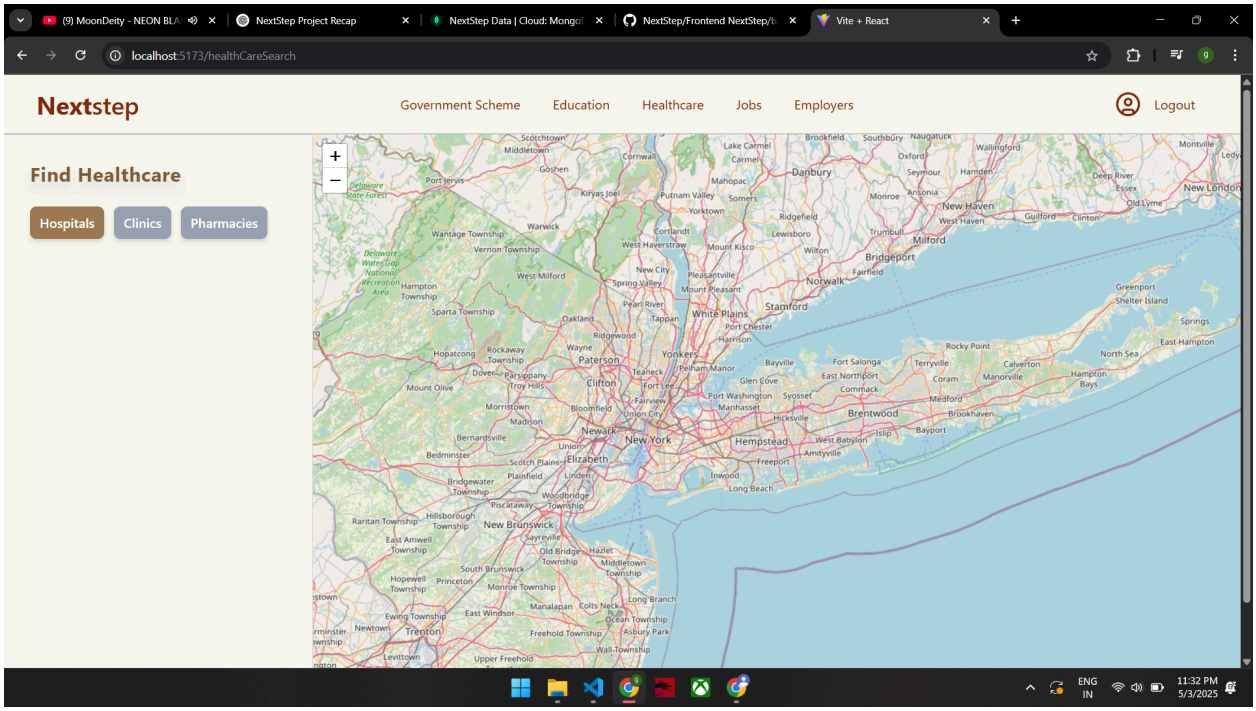
TC-Prof-03 (State Change)



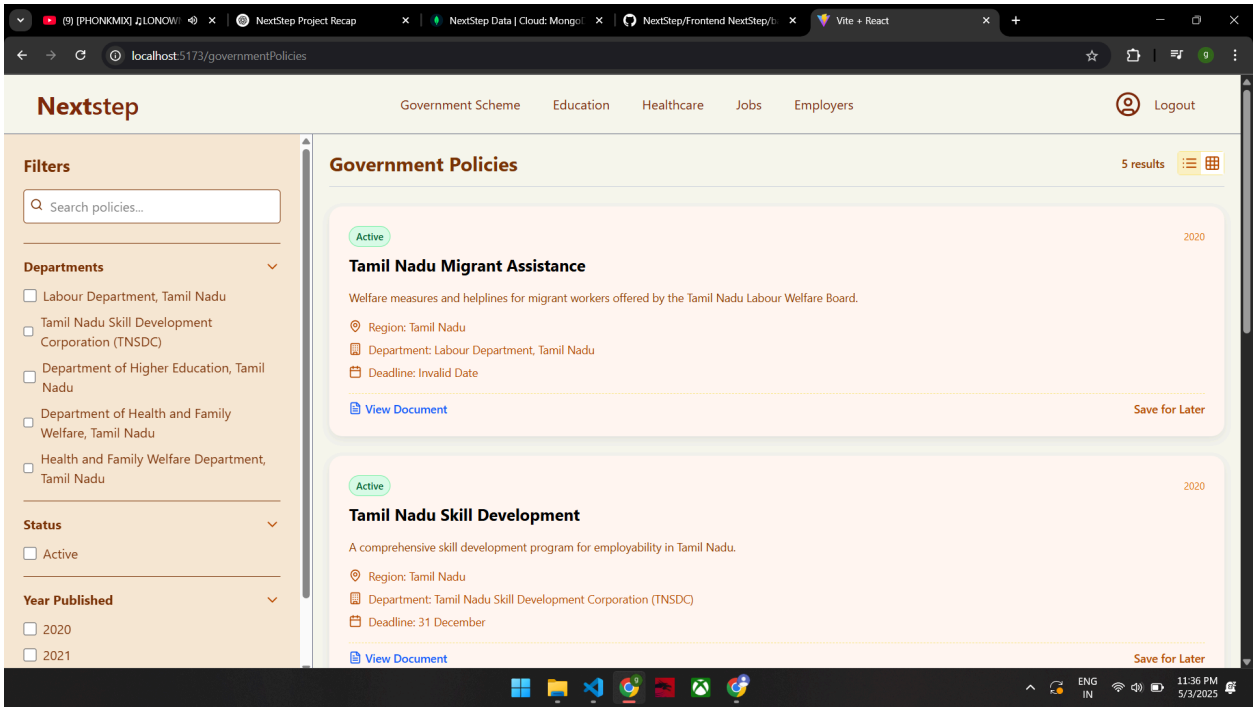
TC-Heal-01 (Available)



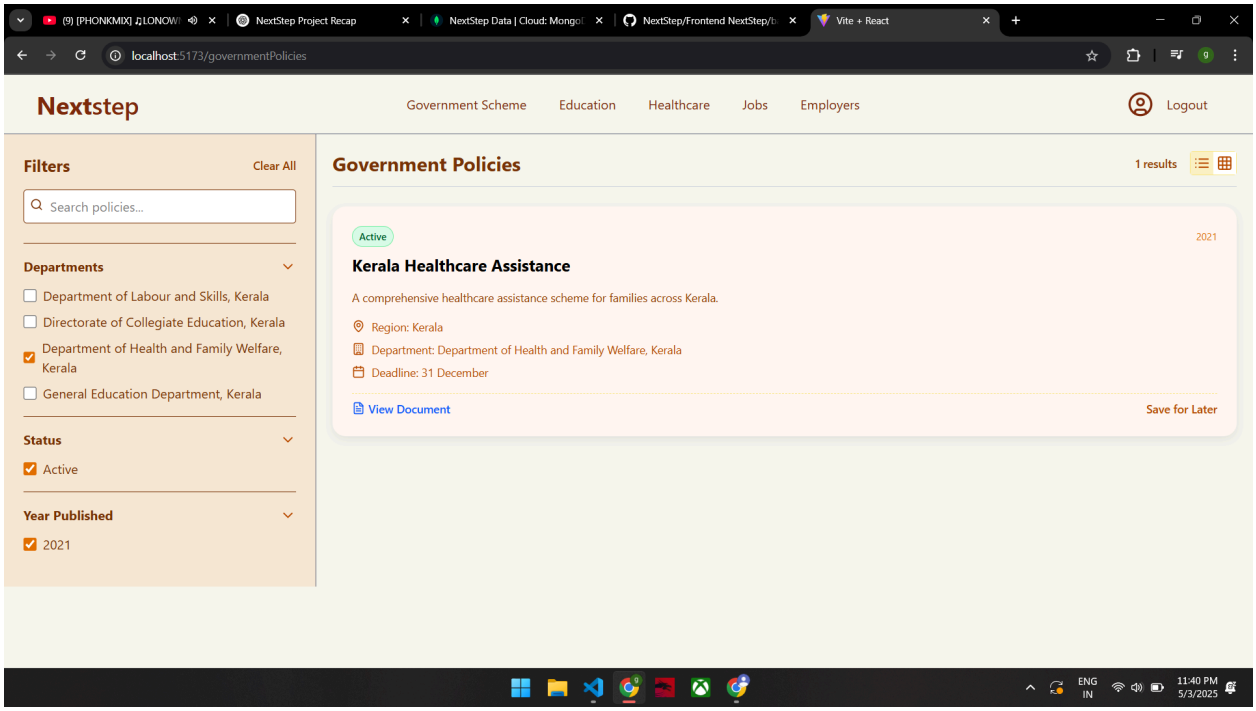
TC-Heal-02 (Not available)



TC-Pol-01 [Available]

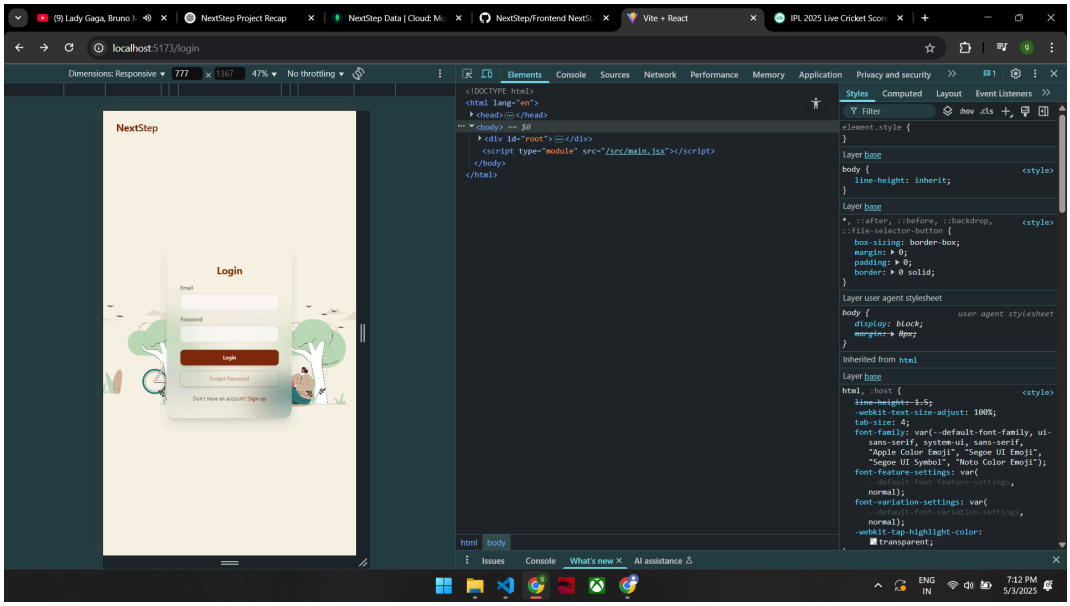


TC-Pol-02

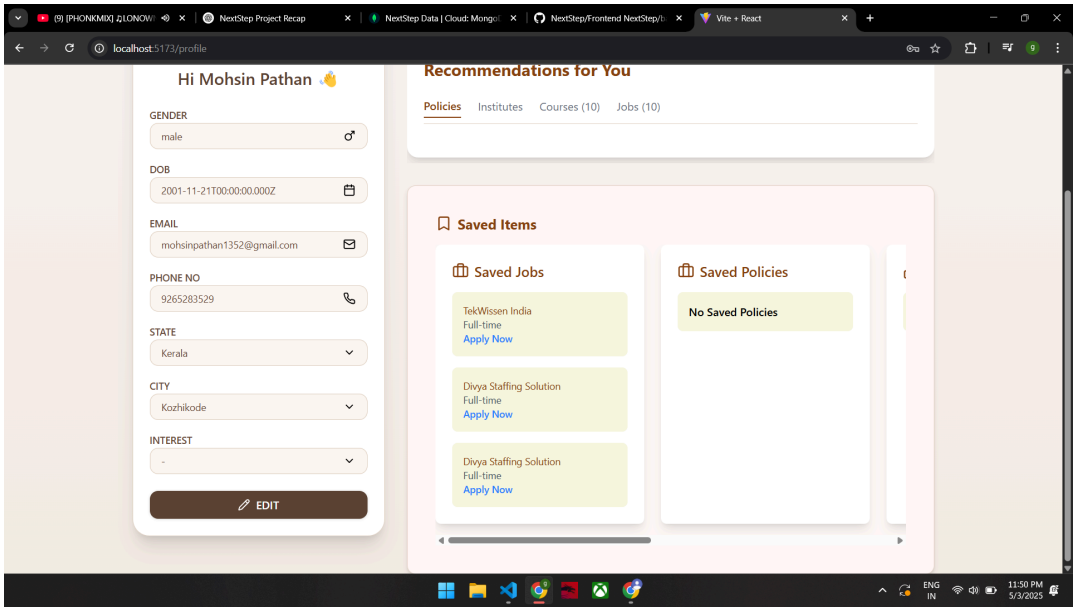


User Interface test cases

TC-UI-1



TC-UI-2



15. Approvals

QA Lead: Mohsin Pathan
Project Manager: Aarushi Goel
Product Owner: Annirudh Pratap