Student Management System Project is a Spring Boot-based web application designed to manage student data. It allows for adding, updating, deleting, searching, and listing students using a MySQL database for storage. Here's a detailed explanation of the project and the role of each component:

Project Overview

The application provides a web interface with pages for:

- Adding new student records.
- Updating existing student records.
- Deleting a student using their unique ID.
- Searching for a specific student by their unique ID.
- Fetching and displaying all student records.

The key components include:

Controllers: Handle HTTP requests and map them to services or views.

Services: Contain the business logic for operations.

Entities: Represent the data model in the database.

Repositories: Handle data interaction with the database.

The project uses the following dependencies:

1. Spring Data JPA for database operations.
2. Thymeleaf for rendering views.
3. Spring Web for building the RESTful web application.
4. Spring Dev Tools for live reload during development.
5. MySQL Connector for connecting to the MySQL database.

Code Breakdown

Controllers

1.StudentController

- Manages student-related requests such as adding, updating, deleting, searching, and fetching students.
- Maps specific endpoints (/reg, /upd, /del, /search, /fetchAll) to corresponding service methods.
- Returns appropriate views (e.g., index, displayStudent, displayAllStudents).

Example:

- /reg Endpoint: Calls the addStudent method in the service layer to add a new student.
- /fetchAll Endpoint: Calls the fetchAllStudents service to retrieve all students and passes them to the displayAllStudents view.

2.NavigationController

- Handles navigation between different pages of the application (e.g., home page, registration page, search page).

- Maps endpoints like /, /registerPage, /searchPage to return corresponding view templates such as index, addStudent, searchStudent.

Services

1.StudentServices (Interface)

Defines the contract for all student-related operations (e.g., addStudent, updateStudent).

2.StudentServiceImplementation

- Implements the StudentServices interface.
- Contains the business logic for managing students:

addStudent: Saves a new student to the database using repo.save.

fetchAllStudents: Retrieves all students using repo.findAll.

searchStudent: Finds a student by ID using repo.findById.

updateStudent: Updates an existing student using repo.save.

deleteStudent: Removes a student using repo.deleteById.

Example:

addStudent Method: Saves the student entity to the database and returns a success message.

Entities

1.Student

- A JPA entity representing the student table in the database.
- Fields: univId (Primary Key), name, email, branch, address.
- Includes getters and setters for all fields.
- Annotated with @Entity to define it as a database table.
- @Id Annotation: Marks univId as the primary key.

Repositories

1.StudentRepository

- Extends JpaRepository to leverage built-in database operations (e.g., save, findAll, findById, deleteById).
- Provides a seamless way to interact with the database without writing SQL queries.

How It Works

- When a user interacts with the application via the web interface:
- For example, submitting a form to add a student:
- The form data is sent to the /reg endpoint.
- The StudentController receives the request and invokes the addStudent method in the StudentServiceImplementation.

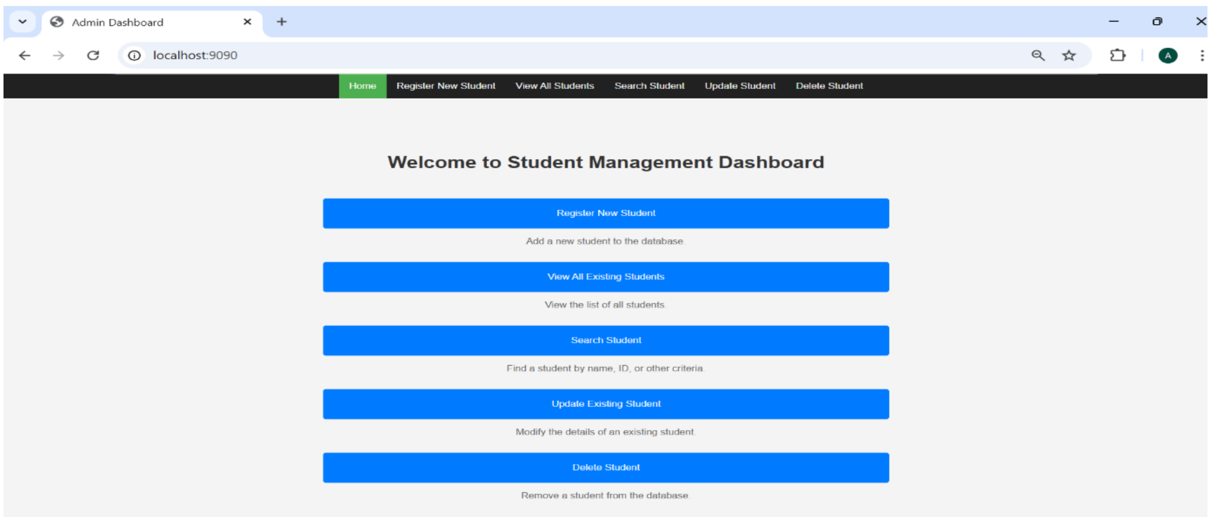
- The service saves the data to the database using the repository.
- The Thymeleaf templates render the user interface for actions such as displaying a list of students, a specific student's details, or forms for adding/updating/deleting records.
- The application uses Spring Data JPA to abstract database operations, making it easy to manage student records without writing boilerplate SQL.

Dependencies and Their Roles

1. Spring Data JPA: Simplifies database interactions.
2. Thymeleaf: Enables dynamic HTML rendering for views.
3. Spring Web: Provides the foundation for building the web application and handling HTTP requests.
4. DevTools: Speeds up development with live reload features.
5. MySQL Connector: Bridges the application with the MySQL database.

- This is a well-structured project showcasing the classic MVC (Model-View-Controller) design pattern in a Spring Boot application.

Output:



Welcome to Student Management Dashboard

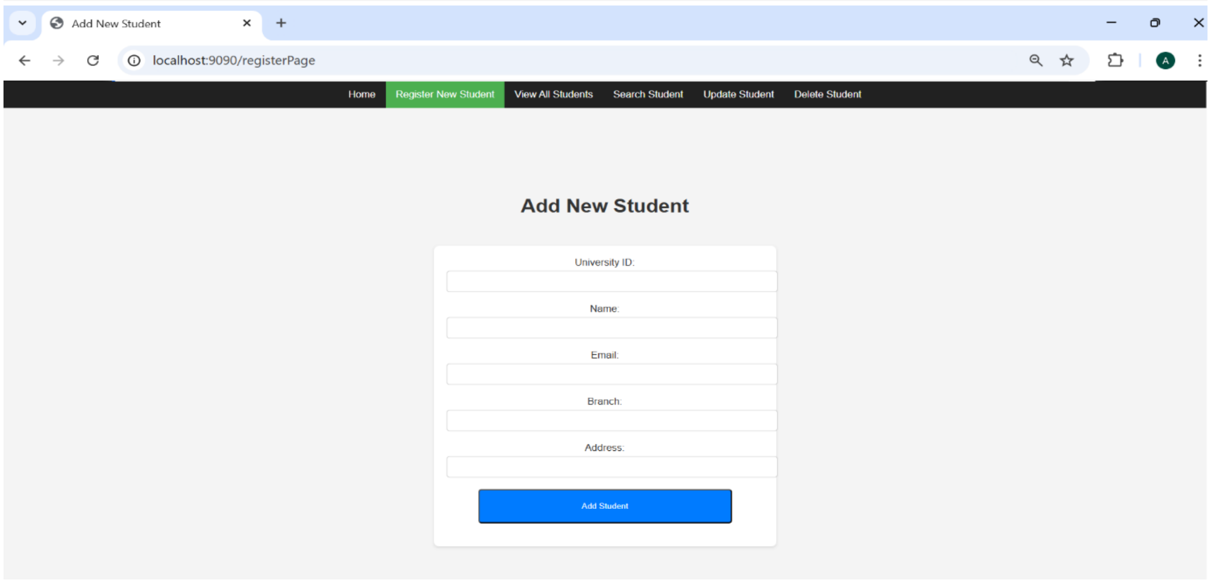
Add a new student to the database

View the list of all students

Find a student by name, ID, or other criteria

Modify the details of an existing student

Remove a student from the database



Add New Student

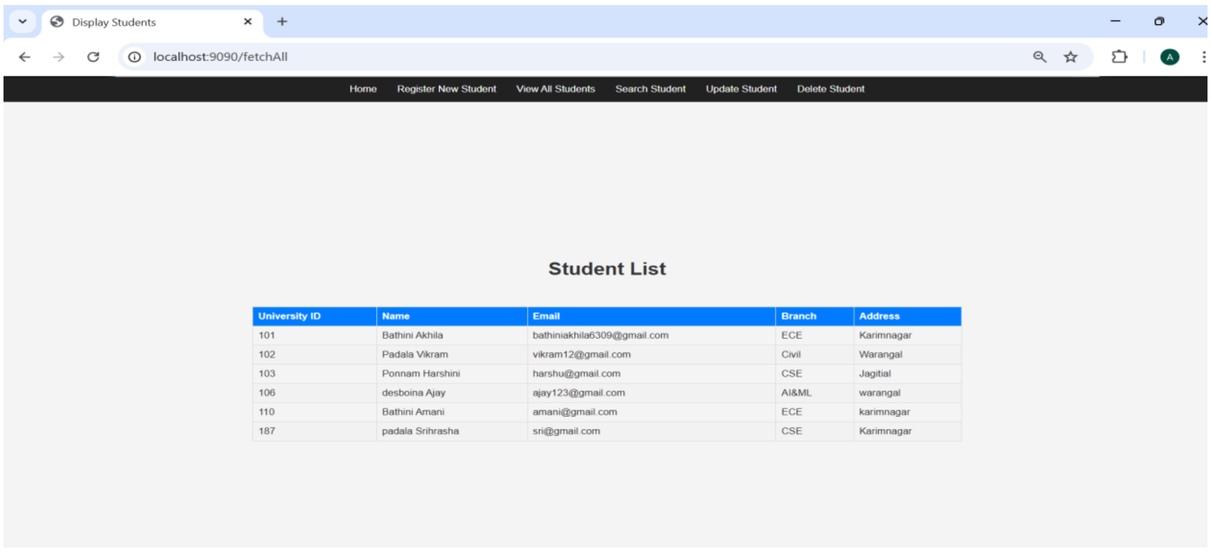
University ID:

Name:

Email:

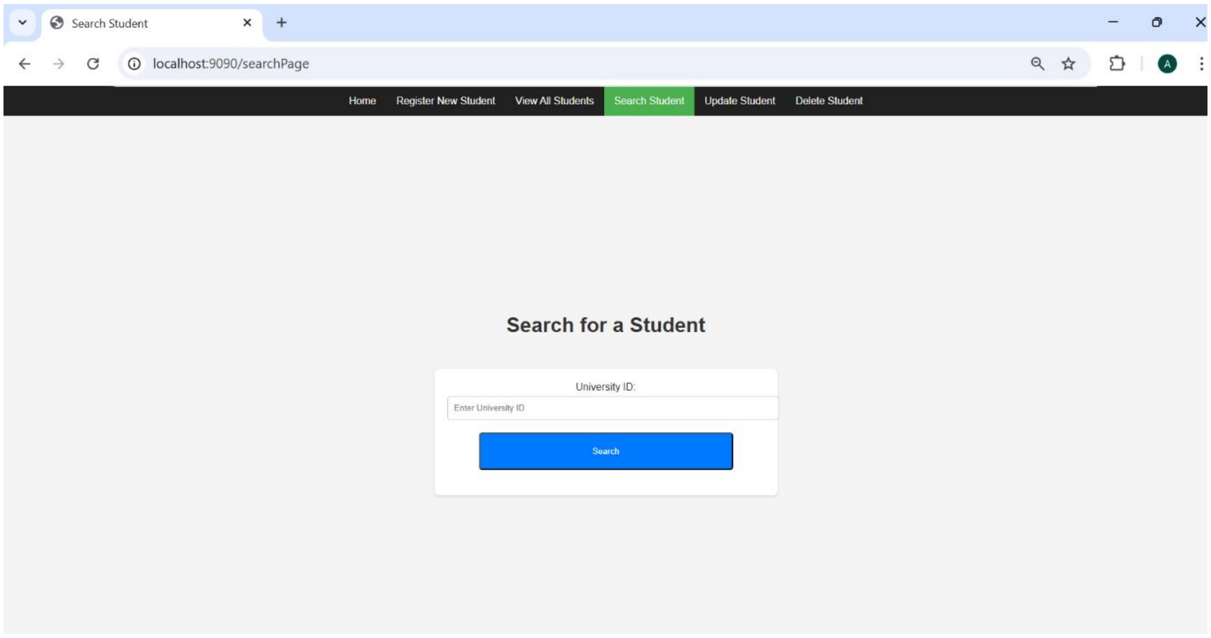
Branch:

Address:



Student List

University ID	Name	Email	Branch	Address
101	Bathini Akhila	bathiniakhila6309@gmail.com	ECE	Karimnagar
102	Padala Vikram	vikram12@gmail.com	Civil	Warangal
103	Ponnam Harshini	harshu@gmail.com	CSE	Jagitial
106	desboina Ajay	ajay123@gmail.com	AI&ML	warangal
110	Bathini Amani	amani@gmail.com	ECE	karimnagar
187	padala Srihrasha	sri@gmail.com	CSE	Karimnagar

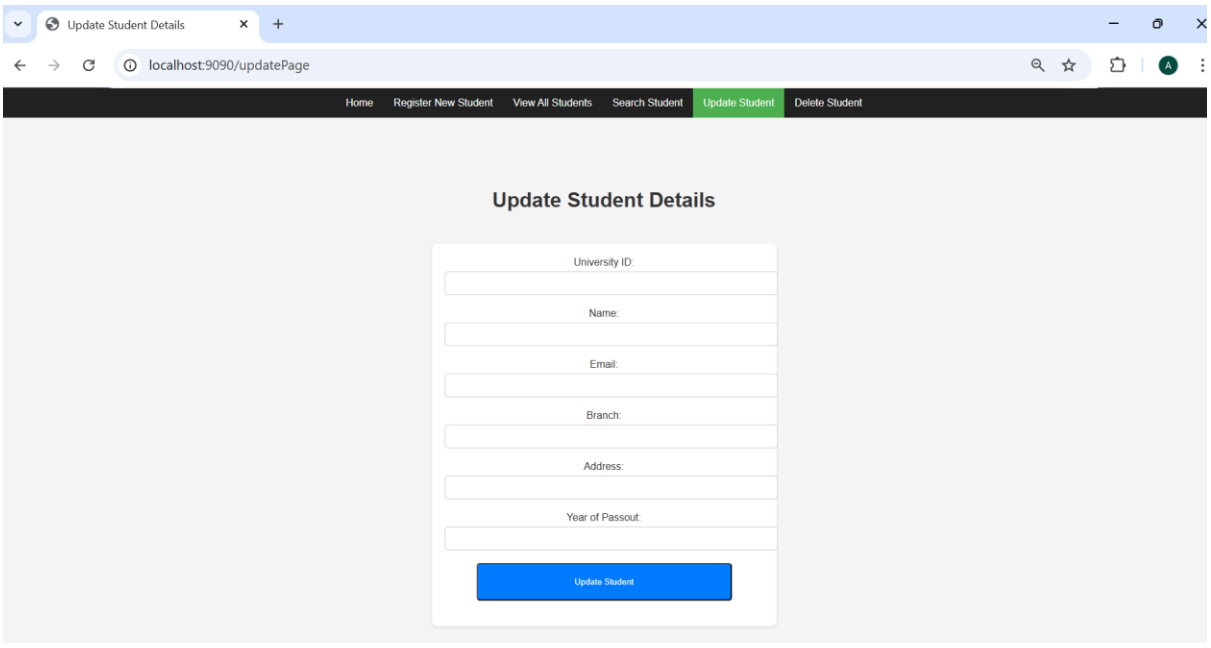


Search for a Student

University ID:

Enter University ID

Search



Update Student Details

University ID:

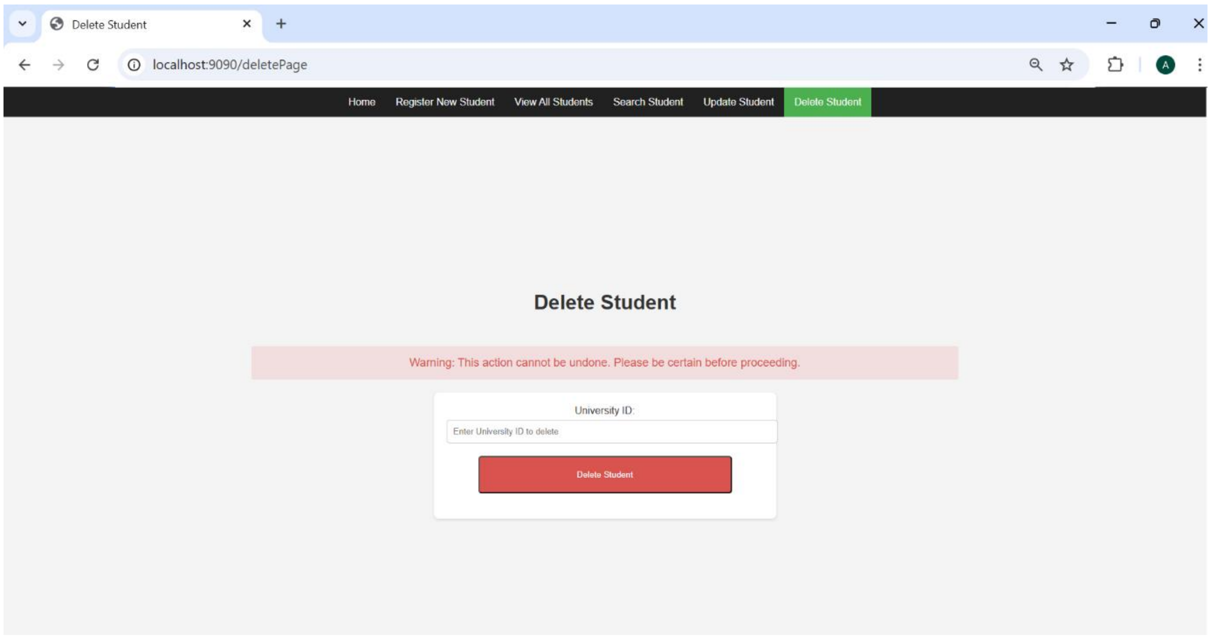
Name:

Email:

Branch:

Address:

Year of Passout:



Delete Student

Warning: This action cannot be undone. Please be certain before proceeding.

University ID:

Enter University ID to delete