

**USER MANAGEMENT SYSTEM FOR STUDENTS**

**CS23333-OBJECT ORIENTED PROGRAMMING USING JAVA**

*Submitted by*

JASPER SAVIO M - 231001071

JAYABALAJI S - 231001073

*Of*

**BACHELOR OF TECHNOLOGY IN**

**INFORMATION TECHNOLOGY**

**RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM**  
**(An Autonomous Institution)**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**RAJALAKSHMI ENGINEERING COLLEGE**

**THANDALAM , CHENNAI - 600025**

**NOVEMBER 2024**

## **BONAFIDECERTIFICATE**

Certified that this project titled “Student Mark Analysis System” is the Bonafide work of “**JASPER SAVIO M. (231001071)**,” **JAYABALAJI S.(231001073)**” who carried out the project work under my supervision.

**SIGNATURE**

**Dr. P. Valarmathie**

**HEAD OF THE DEPARTMENT**

Information Technology

Rajalakshmi Engineering College,  
Rajalakshmi Nagar, Thandalam  
Chennai – 602105

**SIGNATURE**

**Mrs.T Sangeetha**

**COURSE INCHARGE**

Information Technology

Rajalakshmi Engineering College  
Rajalakshmi Nagar, Thandalam  
Chennai – 602105

This project is submitted for IT19341 – Introduction to Oops and Java held on

\_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## TABLE OF CONTENTS

### 1. USER MANAGEMENT SYSTEM FOR STUDENTS

1.1. Abstract-----05

1.2. Introduction-----05

1.3. Purpose-----05

1.4. Scope of Project-----06

1.5. Software Requirement Specification-----06

2. System Flow Diagrams-----12

2.1. Use Case Diagram-----12

2.2. Entity-relationship Diagrams-----12

3. Module Description-----13

4. Implementation-----14

4.1. Design-----14

4.2. Database Design-----15

<b>4.3. Code</b>	<b>16</b>
<b>5. Conclusion</b>	<b>20</b>

# **USER MANAGEMENT SYSTEM FOR STUDENTS**

## **1.1 Abstract**

The User Management System of Students is a Java-based application utilizing JDBC to connect to an SQL database. This system enables authorized users to store, edit, and view comprehensive student information, including first name, last name, email ID, phone number, and address. The secure login mechanism ensures that only authorized personnel can access and manage the data, enhancing the security and integrity of the information. Designed for educational institutions, this system aims to streamline the management of student data, making it more efficient and reliable.

## **1.2 Introduction**

In today's educational landscape, effective management of student information is essential for maintaining streamlined administrative processes. This Java-based application developed to address this need. By leveraging JDBC to interface with an SQL database, the system ensures seamless storage, retrieval, and updating of student data, including first name, last name, email ID, phone number, and address. This user-friendly application enables authorized personnel to efficiently manage student information, thereby enhancing the accuracy and accessibility of data.

A key feature of this system is its secure login mechanism, which ensures that only authorized users can access and manage sensitive student data. This security measure not only protects the integrity of the information but also complies with privacy regulations. By implementing this system, educational institutions can significantly improve their data management capabilities, leading to more efficient and effective administrative operations.

## **1.3 Purpose**

The purpose of this project is to create an efficient and user-friendly student management system that benefits both students and educators.

The system aims to:

- Provide a comprehensive tool for managing student information, including personal details such as first name, last name, email ID, phone number, and address.
- Facilitate easy data entry, editing, and viewing to keep student records up to date.

- Ensure data security through a secure login mechanism, restricting access to authorized users only.
- Enhance administrative efficiency and accuracy in managing student information.
- Support better decision-making and administrative processes through reliable data management.

## 1.4 Scope of the Project

The scope of the User Management System of Students encompasses a range of functionalities designed to enhance the management and accessibility of student information. The system aims to streamline the processes of storing, editing, and retrieving student data. It provides educators and administrative staff with a robust platform to manage detailed student profiles, including first name, last name, email ID, phone number, and address.

A core component of this project is its integration with an SQL database using JDBC (Java Database Connectivity). JDBC ensures efficient and secure interaction between the application and the database, allowing for seamless data storage and retrieval. The system supports multiple features such as adding new student records, updating existing information, and providing a user-friendly interface for viewing student profiles. Additionally, the secure login mechanism ensures that only authorized users can access and manage the data, enhancing data security and administrative efficiency. This system ultimately facilitates informed decision-making and improves the overall management of student information within educational institutions.

## 1.5 Software Requirement Specification

### 1. Database Integration:

- **SQL Database:** The system uses a SQL database to store student data, including names and contact details. This database is connected using JDBC (Java Database Connectivity), ensuring seamless interaction between the application and the database.
- **Schema Design:** The database schema includes a table named students with columns for first name, last name, email ID, phone number, and address.

## 2. Backend Development:

- **Server Setup:** The backend server is implemented using Java with the `HttpServer` class, handling various endpoints for different functionalities.
- **Endpoints:**
  - `/api/insert`: Handles the insertion of student records.
  - `/api/update`: Manages updates to existing student records.
  - `/api/view`: Retrieves and displays student records.
  - `/api/login`: Manages user authentication for accessing the system.

## 3. Frontend Development:

- **HTML and CSS:** The frontend is developed using HTML for structure and CSS for styling, providing a user-friendly interface for managing student information.
- **JavaScript:** Utilizes JavaScript to dynamically update the frontend, handle form submissions, and ensure interactive user experiences.

## 4. Functionality:

- **Data Input:** Administrators can input student details through a web form.
- **Data Management:** The system supports adding new records, editing existing information, and viewing student profiles.
- **User Authentication:** Secure login mechanism ensures that only authorized personnel can access the system.

## 5. Technology Stack:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Java, SQL, JDBC
- **Tools and Libraries:** JSON handling libraries for data exchange, `HttpServer` for handling HTTP requests and responses.

## Overall Description

### Product Perspective

The system is built using a client/server architecture, compatible with various operating systems. The frontend is developed with HTML, CSS, and JavaScript, incorporating Chart.js for data visualization. The backend is powered by Java with the `HttpServer` class for handling HTTP requests, and SQL.

### Product Functionality

- a) **Insert Student Data:** Allows users to input and store student information such as first name, last name, email ID, phone number, and address in the database.

- b) **Update Student Data:** Enables users to update existing student records to ensure data accuracy and relevance.
- c) **View Student Data:** Provides the capability to retrieve and display detailed student profiles.
- d) **Search Functionality:** Allows users to search for students based on various criteria such as name or email ID.
- e) **User Authentication:** Ensures secure access to the system, allowing only authorized users to manage student information.

## **User and Characteristics**

### **Qualification:**

- Users should have a basic understanding of educational processes and a qualification equivalent to matriculation.

### **Experience:**

- Familiarity with student information management and basic data handling is beneficial.

### **Technical Skills:**

- Users are expected to have elementary knowledge of computers and the ability to interact with web-based applications.

### **Operating Environment:**

#### **Hardware Requirements:**

- **Processor:** Any processor over i3
- **Operating System:** Windows 8, 10, 11
- **Processor Speed:** 2.0 GHz
- **RAM:** 4GB
- **Hard Disk:** 500GB

#### **Software Requirements:**

- **Database:** SQL
- **Frontend:** HTML, CSS, JavaScript (with Chart.js)
- **Backend:** Java



## Constraints

- **Authorized Access:** System access is limited to authorized users, such as administrators and teachers.
- **Delete Operation:** The delete operation is restricted to administrators, designed without additional checks for simplicity.
- **Data Consistency:** Administrators must exercise caution during deletion to ensure data consistency and prevent unintended data loss.

## Assumptions and Dependencies

- System administrators are responsible for creating and securely communicating login IDs and passwords to users.
- Users are assumed to have basic knowledge of using web-based applications.

## Specific Requirements

### User Interface Features for the User Management System of Students Are:

- **Login:** Secure login for authorized users to maintain data security.
- **Insert Student Data:** Input and store new student information accurately and efficiently.
- **View Student Data:** Display and manage detailed student information seamlessly.
- **Update Student Data:** Modify existing student records to keep data up to date.
- **Search Student Data:** Retrieve specific student details quickly based on search criteria.
- **User Authentication:** Ensure secure management of user access to protect sensitive information.

### Hardware Interface:

- Screen resolution of at least 640 x 480 or above.
- Compatible with any version of Windows 8, 10,11

## Software Interface for Student Mark Analysis System

- **Operating System:** MS-Windows (Windows 8, 10, 11)
- **Frontend Development:** HTML, CSS, JavaScript (with Chart.js)
- **Backend Development:** Java
- **Database:** SQL

## **Functional Requirements for User Management System of Students**

- **Login Module (LM):**

- Users (admins) can access the Login Module via a secure login page.
- The system supports login using a username and password, with passwords masked for security.
- Only authorized admins, whose credentials match those in the database, are granted access.

- **Registered Users Module (RUM):**

- After successful login, users (admins) are granted access to the main features.
- Users can view detailed information about students and their marks.
- Admins can update student marks, insert new data, and perform various administrative tasks.

- **Administrator Module (AM):**

- The system displays administrative functions after a successful login.
- Admins can manage student data: adding new records, updating existing records, and deleting unused data.
- The "Add" function allows admins to input new student details and marks, while the "Update" function allows modifications to existing information.
- All add, update, or delete actions trigger communication with the backend (via the Server Module) to make necessary changes in the database.

- **Server Module (SM):**

- Acts as an intermediary between the frontend modules and the database.
- Receives and processes requests from various modules, ensures proper data formatting, and manages the system's functionality.
- Handles communication with the database to validate and execute requests, ensuring data consistency and integrity.

## **Non-functional Requirements**

### **Performance:**

- The system must efficiently handle student data analysis requests, ensuring that calculations like mean, standard deviation, and other statistics are completed in under 2 seconds.

- The system should handle a large number of concurrent requests for data input and report generation without significant delays, ensuring high responsiveness for administrators.

**Reliability:**

- The system must be robust and capable of recovering gracefully from failures. In case of data corruption or abnormal shutdown, the system should provide mechanisms for data recovery and ensure minimal data loss.
- The system should be thoroughly tested to handle edge cases such as incorrect or missing data entries, ensuring smooth operation under various conditions.

**Availability:**

- The system should be available 24/7 for administrators to perform tasks like adding, updating, or deleting student data and generating reports.
- It should maintain high availability and perform critical operations, like querying student marks, with minimal downtime or service interruptions.

**Security:**

- The system must implement strong authentication mechanisms to ensure that only authorized administrators can access the backend and modify sensitive student data.
- Passwords and sensitive user data must be encrypted in both transit (using SSL/TLS) and at rest in the database.
- The system should regularly audit user activities to prevent unauthorized access or modifications to student records.

**Maintainability**

- **24/7 Availability:** The system should be available around the clock for administrators to perform tasks like adding, updating, or deleting student data and generating reports.
- **High Availability:** It should maintain high availability, performing critical operations like querying student marks with minimal downtime or service interruptions.

## SYSTEM FLOW DIAGRAMS

### 2.1 Use Case Diagrams :

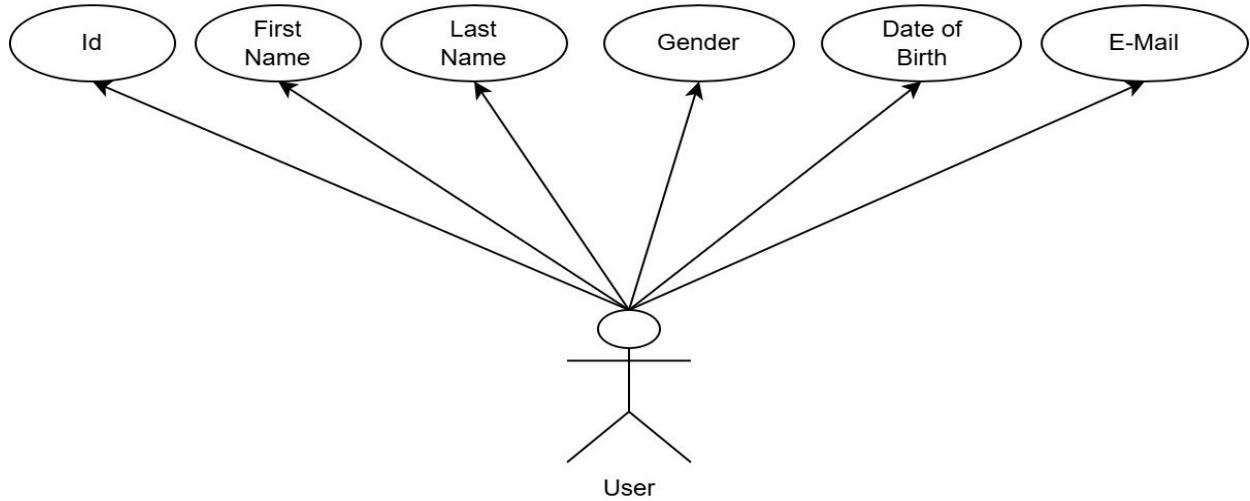


FIG 2.1 CASE DIAGRAM

### 2.2 Entity-relationship diagram:

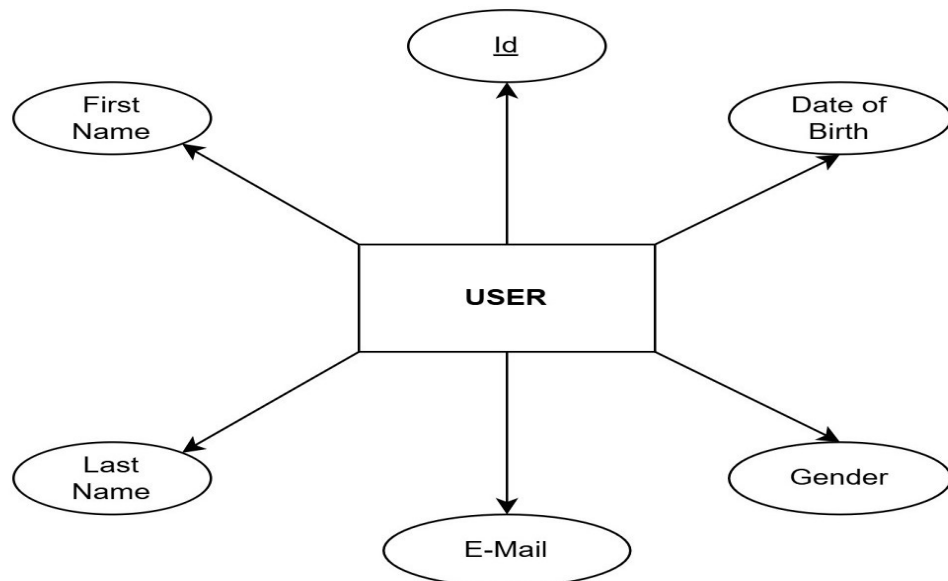


Fig 2.2 ER DIAGRAM OF UMS

## MODULE DESCRIPTION

### Register:

- Admins can register an account by providing a unique username and secure password. The system securely stores these credentials, ensuring only authorized administrators can access the backend.

### Login:

- Admins log in using their username and password. The system verifies credentials against the database, granting access to administrative functions upon successful login.

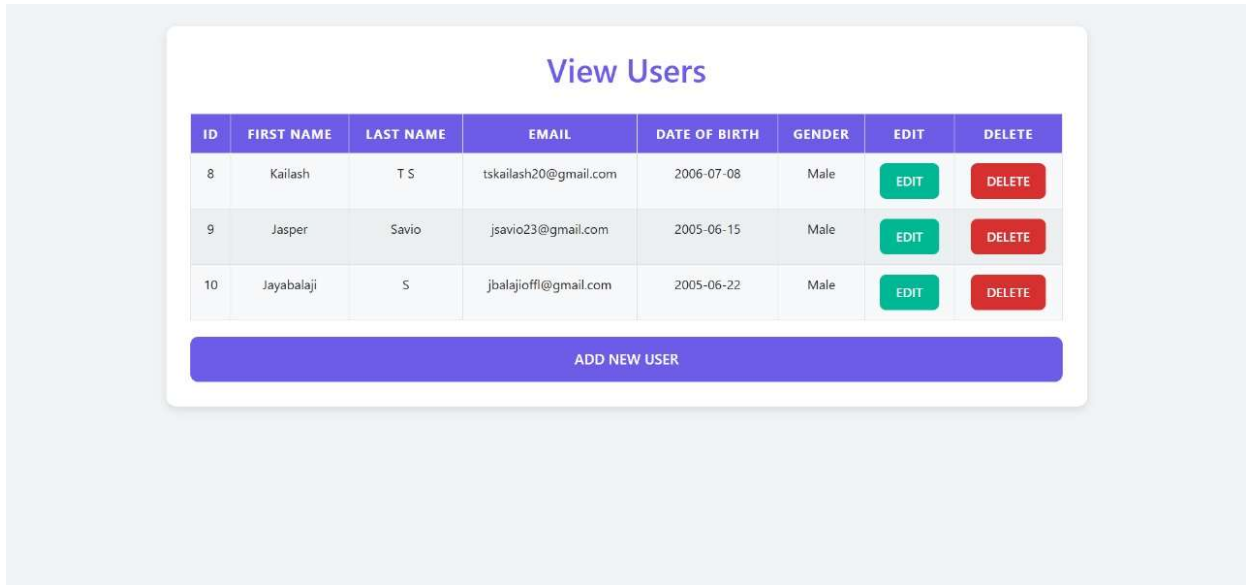
### After Login:

- **Add Student Data:** Admins can input student details, including names and marks. The system saves this data for future analysis and report generation.
- **View Student Data:** Admins can view detailed information about students' marks in a user-friendly format, with options to update or modify records as needed.
- **Update Student Data:** Admins can modify student marks, allowing for necessary corrections or updates.
- **Delete Student Data:** Admins can delete student marks when no longer needed, with restrictions to prevent accidental deletion of important data.
- **Generate Statistical Reports:** Admins can request statistical analysis of student marks, including mean, variance, and standard deviation. The system calculates and displays results for easier decision-making.
- **Generate Performance Reports:** Admins can generate reports showing top performers and lowest scorers, aiding in the assessment and decision-making process.
- **Remove Admin:** Admins can remove other administrators from the system, ensuring only authorized personnel have access to sensitive operations.

## IMPLEMENTATION

### 4.1 Design:

#### View User:



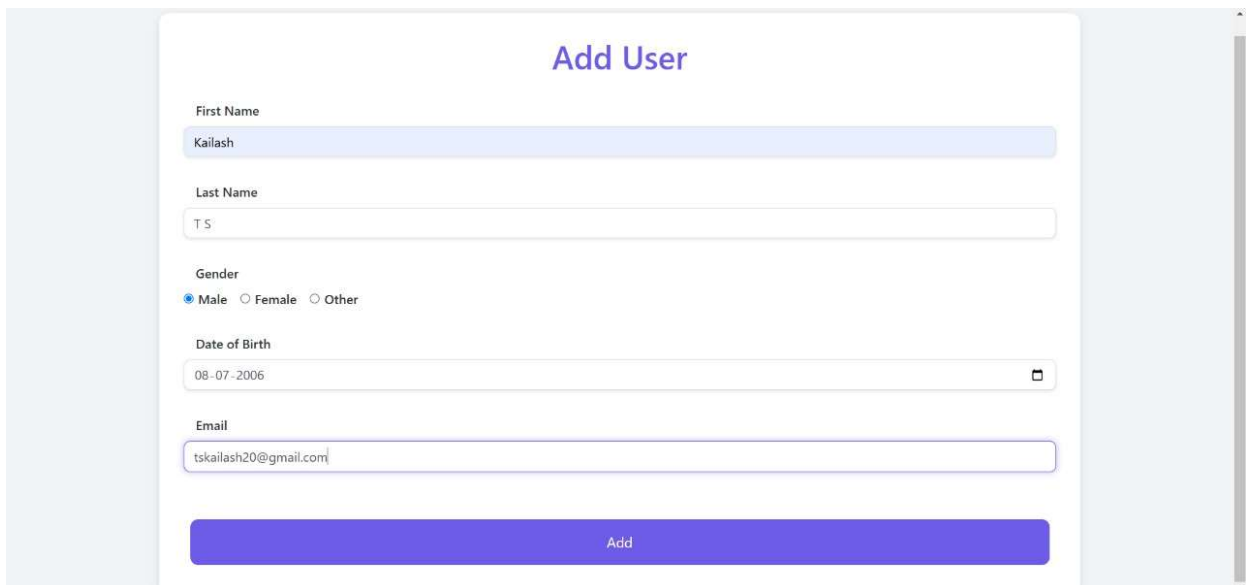
The 'View Users' interface displays a table with the following data:

ID	FIRST NAME	LAST NAME	EMAIL	DATE OF BIRTH	GENDER	EDIT	DELETE
8	Kailash	T S	tskailash20@gmail.com	2006-07-08	Male	<button>EDIT</button>	<button>DELETE</button>
9	Jasper	Savio	jsavio23@gmail.com	2005-06-15	Male	<button>EDIT</button>	<button>DELETE</button>
10	Jayabalaji	S	jbalajioffi@gmail.com	2005-06-22	Male	<button>EDIT</button>	<button>DELETE</button>


Below the table is a blue button labeled 'ADD NEW USER'.

FIG 4.1 VIEW USER

#### Add User:



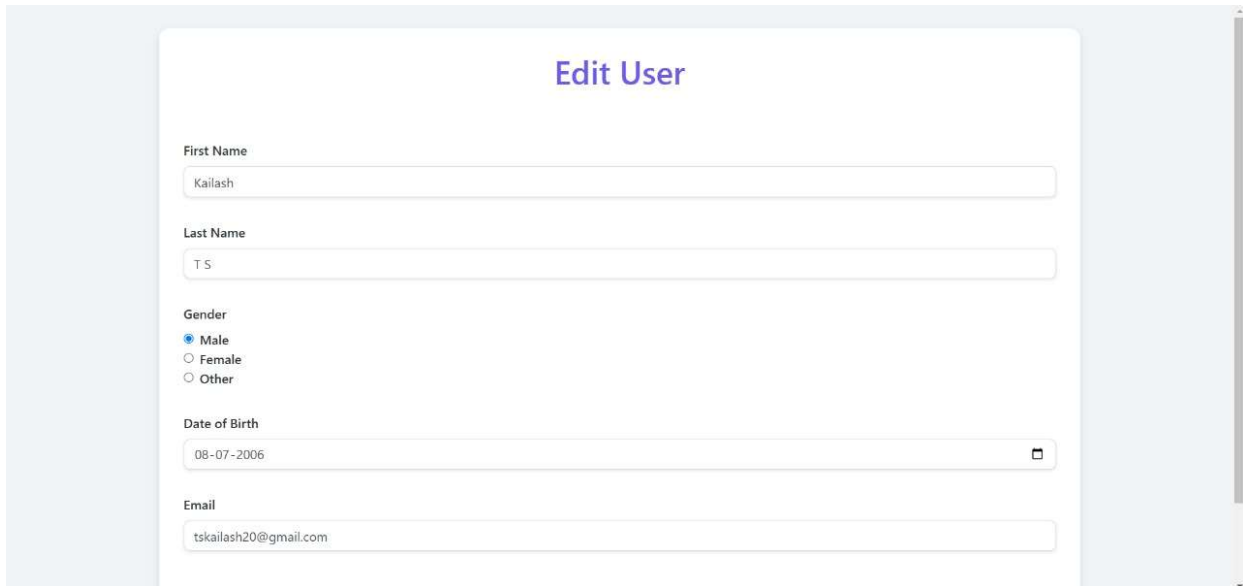
The 'Add User' interface contains the following form fields:

- First Name:
- Last Name:
- Gender: ☒ Male ☐ Female ☐ Other
- Date of Birth:  
- Email:

At the bottom is a blue button labeled 'Add'.

FIG 4.2 ADD USER

### Edit User:



The screenshot shows a web application interface with a light blue background. In the center is a white rounded rectangle titled "Edit User" in purple text. Below the title are several form fields: "First Name" with the value "Kailash", "Last Name" with the value "T S", "Gender" with radio buttons for "Male" (selected), "Female", and "Other", "Date of Birth" with the value "08-07-2006" and a calendar icon, and "Email" with the value "tskailash20@gmail.com".

FIG 4.3 EDIT USER

## 4.2 Database Design:

For the Student Mark Analysis System, the data is stored and retrieved from a MySQL database, which is chosen for its ability to handle structured data efficiently. The system uses a database to store student marks, perform statistical analyses, and manage administrator activities.

### Database Design

**Data Elements and Structures:** At the analysis stage, the required data elements are identified, such as student names, marks, and statistical analysis results. These elements are structured and organized to facilitate storage and retrieval.

**Normalization:** The database schema undergoes normalization to ensure internal consistency, minimize redundancy, and optimize data storage. This process helps avoid unnecessary duplication of data and ensures that the database is scalable and efficient.

**Data Integrity:** Relationships between various data items are established, ensuring that data integrity is maintained. The normalization process helps in minimizing the chances of data inconsistencies and allows for easier updates.

**MySQL Database:** MySQL is selected due to its widespread use, efficiency, and flexibility in handling relational data. It supports quick, reliable, and flexible access to the data for various users (administrators) while minimizing data inconsistencies.

### Student Mark Analysis System SQL Tables

	id	dob	email	first_name	gender	last_name
▶	8	2006-07-08	tskailash20@gmail.com	Kailash	Male	T S
	9	2005-06-15	jsavio23@gmail.com	Jasper	Male	Savio
	10	2005-06-22	jbalajioff@gmail.com	Jayabalaji	Male	S
*	NULL	NULL	NULL	NULL	NULL	NULL

Fig 4.4 SQL Database

### 4.3 Code:

#### application.properties:

```
spring.application.name=user-management-portal
spring.datasource.url = jdbc:mysql://localhost:3306/user_mgmt
spring.datasource.username = root
spring.datasource.password = tsailashts0807@@
spring.jpa.hibernate.ddl-auto = update
spring.mvc.hiddenmethod.filter.enabled=true
```

#### UserController.java:

```
package com.example.user_management_portal.controller;
```



```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;

import com.example.user_management_portal.entity.User;
import com.example.user_management_portal.service.UserService;

@Controller
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping("/getUsers")
    public String getUsers(Model model) {
        model.addAttribute("usersList", userService.getAllUsers());
        return "ViewUsers";
    }

    @GetMapping("/addUser")
    public String addUser(Model model) {
        model.addAttribute("user", new User());
        return "AddUser";
    }

    @PostMapping("/saveUser")
    public String saveUser(User user) {
        userService.saveOrUpdateUser(user);
        return "redirect:/getUsers";
    }

    @GetMapping("/editUser/{id}")
    public String editUser(@PathVariable Long id, Model model){
        model.addAttribute("user", userService.getUserById(id));
        return "EditUser";
    }

    @PutMapping("/editSaveUser")
    public String editSaveUser(@ModelAttribute User user) {

```

```

        userService.saveOrUpdateUser(user);
        return "redirect:/getUsers";
    }

    @GetMapping("/deleteUser/{id}")
    public String deleteUser(@PathVariable Long id) {
        userService.deleteUser(id);
        return "redirect:/getUsers";
    }
}

```

### **User.java:**

```

package com.example.user_management_portal.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column
    private String firstName;

    @Column
    private String lastName;

    @Column
    private String gender;

    @Column
    private String dob;

    @Column
    private String email;
}

```

```

public User() {
    super();
}

public User(Long id, String firstName, String lastName, String gender, String dob, String
email) {
    super();
    this.id = id;
    this.firstName = firstName;
    this.lastName = lastName;
    this.gender = gender;
    this.dob = dob;
    this.email = email;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}

```

```
public String getDob() {  
    return dob;  
}  
public void setDob(String dob) {  
    this.dob = dob;  
}  
  
public String getEmail() {  
    return email;  
}  
  
public void setEmail(String email) {  
    this.email = email;  
}  
}
```

## CONCLUSION

The "User Management System of Students" project, developed with meticulous attention to detail, emphasizes a seamless user experience while ensuring efficient functionality for administrators. Key features include adding, viewing, and managing student information, with robust security measures for sensitive operations like modifying and removing student records to maintain data integrity and protect against unauthorized access. This system is designed to meet the current needs of educational institutions, offering long-term effectiveness and adaptability for future improvements, ensuring efficient management and reliability of student data.

## Reference links

[Student Management System - Javatpoint](#)

[Educational Data Management - ResearchGate](#)

[Java HttpServer - Oracle Documentation](#)

[Chart.js - Official Documentation](#)