HOSTELIAN

Project By:

Sr.No	Name	Email
1	Rohit Milind Kadam	dse21130185@git-india.edu.in
2	Amrut Laxman Gawade	dse21134575@git-india.edu.in
3	Sahil Deepak Pardhi	dse21112355@git-india.edu.in

ABSTRACT

As the name specifies “HOSTELIAN” is software developed for managing various activities in the hostel. For the past few years the number of educational institutions has been increasing rapidly. Thereby the number of hostels is also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who is running the hostel and software’s are not usually used in this context. This particular project deals with the problems of managing a hostel and avoids the problems which occur when carried manually. Identification of the drawbacks of the existing system leads to the development of a computerized hostel management system that will be compatible with the existing system with the system which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcoming the drawbacks of the existing hostel management system. Less human error, Strength and strain of manual labor can be reduced, High security, Data redundancy can be avoided to some extent, Data consistency, Easy to handle, Easy data updating, Easy record keeping, Backup data can be easily generated.

ACKNOWLEDGEMENT

In this project, we have made an effort. Nevertheless, it would not have been possible without the generous support and assistance of numerous people and organizations. We would like to express our gratitude towards Symbiosis Skills and Professional University. We would like to extend our sincere thanks to our trainer Mr. Chakradhar Shinde sir. We are highly indebted to our trainer for his kind cooperation, encouragement, guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

DECLARATION

We declare that the project entitled "**Hostelian**" submitted as part of our academic requirements for Java FullStack at Symbiosis Skills and Professional University, is entirely our original work. We further declare that this project has not been duplicated or submitted in any form, in full or in part, to any other university or educational institution for the award of any degree or qualification.

To the best of our knowledge and belief, other than ourselves, no individual has submitted this project to any other university or institution for academic credit or certification.

LIST OF FIGURES

CONTENT

Chapter 1
Introduction

In our current era of automated systems with it being either software or hardware, it's not advisable to be using manual systems. Hostels without a management system are usually done manually. Registration forms verification to other data saving processes are done manually and most at times, they are written on paper. Thus a lot of repetitions can be avoided with an automated system. The drawbacks of existing systems lead to the design of a computerized system that will help reduce a lot of manual inputs. With this system in place, we can improve the efficiency of the system, thus overcoming the drawbacks of the existing manual system. This system is designed in favor of the hostel management which helps them to save the records of the students about their rooms and other things. It helps them from the manual work from which it is very difficult to find the record of the students and the mess bills of the students, and the information about those who had left the hostel years before. This system gives an idea about how a student and fee details, room allocation, mess expenditure are maintained in a better way. The hostel management system will also contain special features like how many students are in a room, student's id and free rooms or space available. The administration has a unique identity for each member as well as student's details.

Chapter 2

Features and Functionality

Hostelian is a software that helps users to maintain the hostel details of every student very effectively.

An effective management of the hostel helps to maintain information of students like name, address, contact number, e-mail, department & room details etc. Here's how Hostelian can help you to achieve following benefits:

1) Register New User:

Here new users can register themselves to the system. Details of users are added to the database.

2) Book Hostel:

Here students can book hostel rooms by logging in to the HMS. Then the data will be saved to the database. And students can also view their room details from the database.

3) Update Existing Student Details:

Here students can only update their personal details. This feature will help to update the basic details of the student if necessary.

4) Delete Existing Student Details:

If the admin wants to delete a particular student's information from the database. Then he can simply drop that particular student from the database by using this feature.

Used Tools & Technologies

Technology/Domain - Java FullStack

Front-End - ReactJS

Database - MySQL

Back-End - Java

Chapter 3

Database Design

- **UserRegistration Table –**

Field Name	**DataType**
userId	Integer
firstName	String
middleName	String
lastName	String
course	String
gender	String
contactNo	Integer
email	String
password	String

- **Courses Table –**

Field Name	**DataType**
courseId	Integer
course_code	String
course_name	String

- **Admin Table –**

Field Name	**DataType**
adminId	Integer
userName	String
email	String
password	String

Chapter 4

Diagram

- **Class Diagram -**

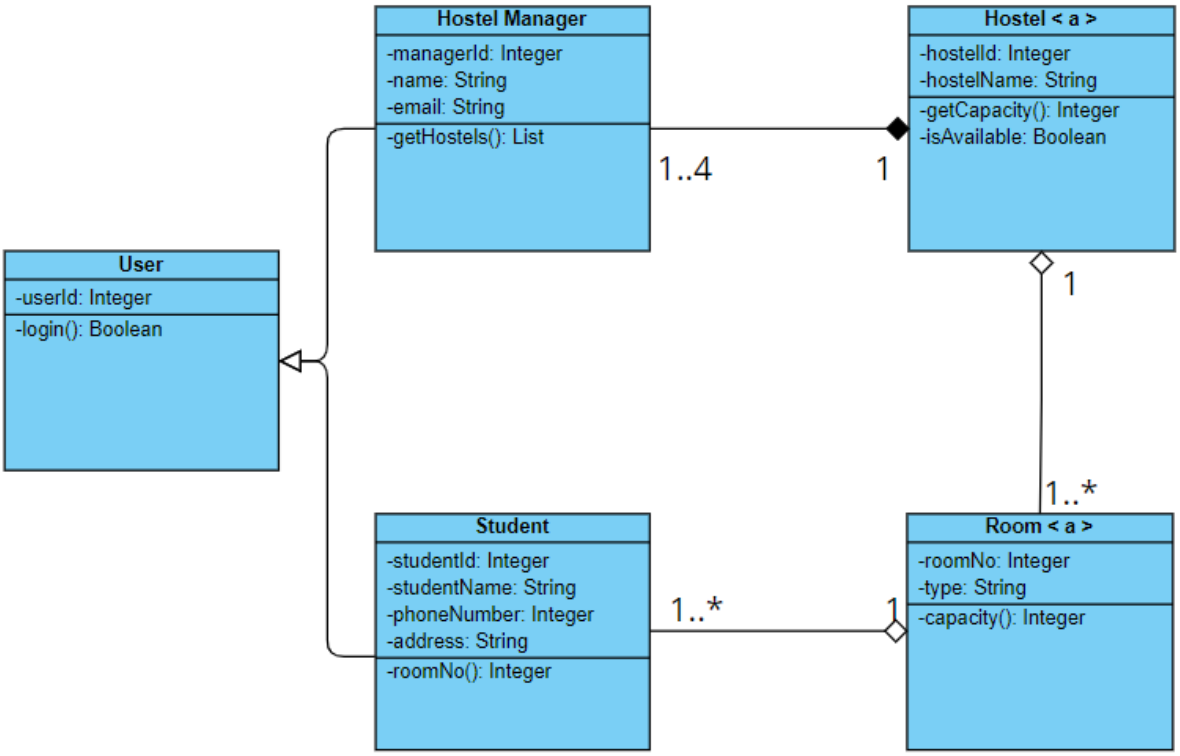


Fig 4.1: Class Diagram

- **Data Flow Diagram -**

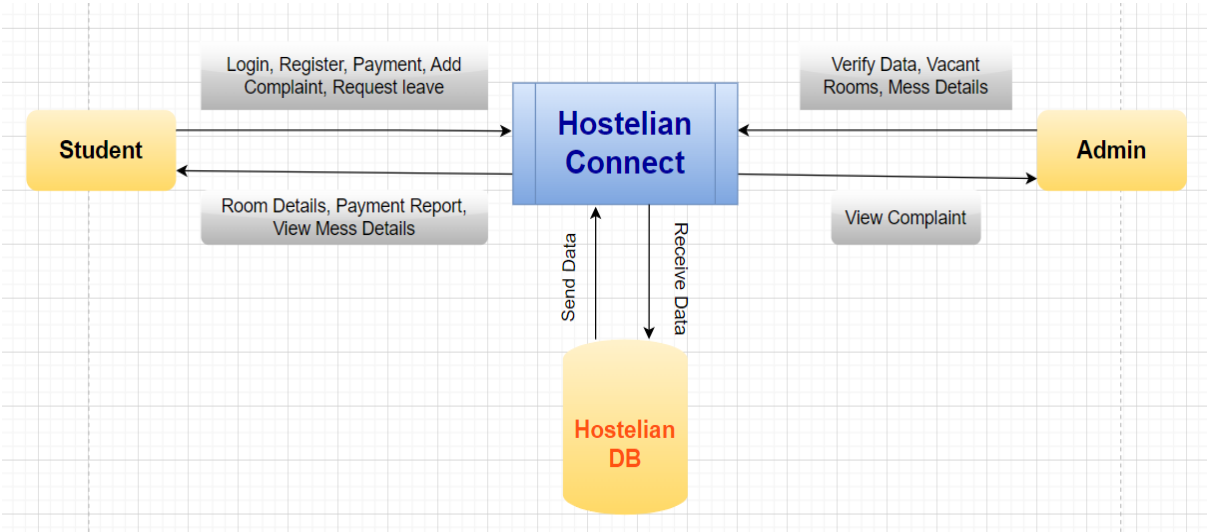
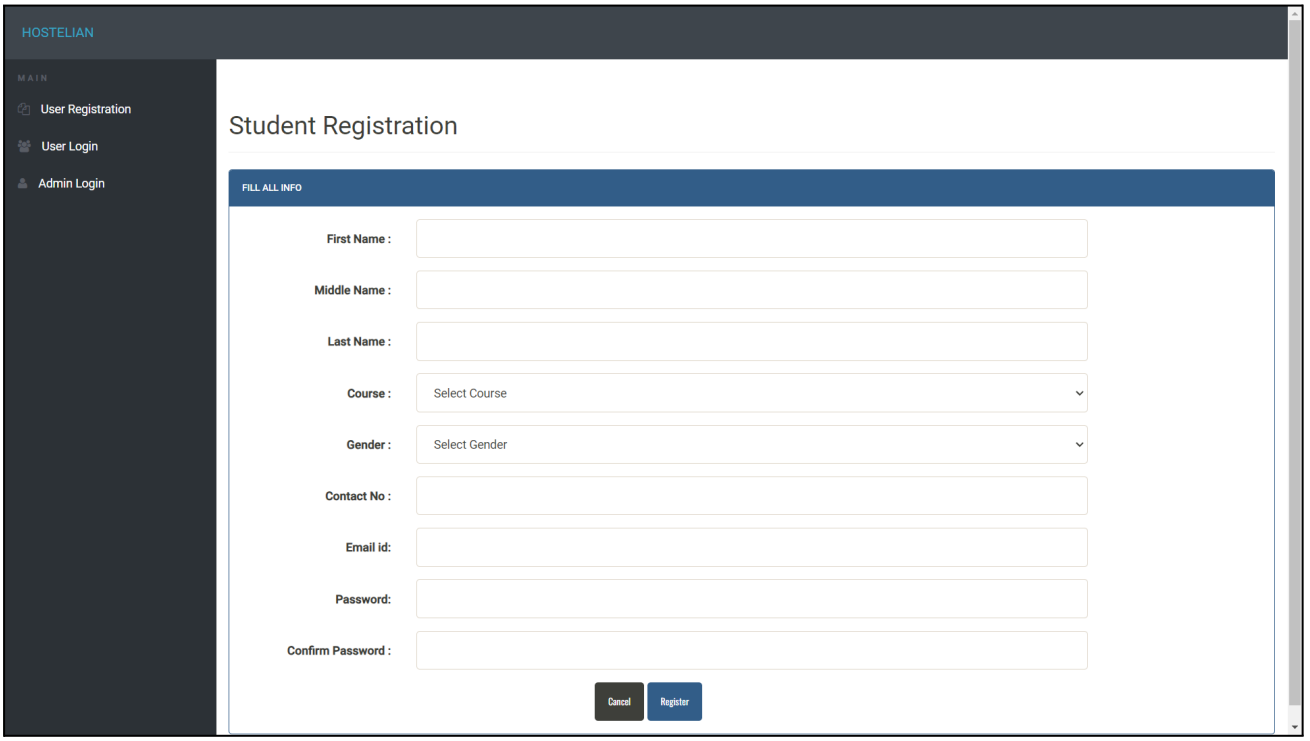
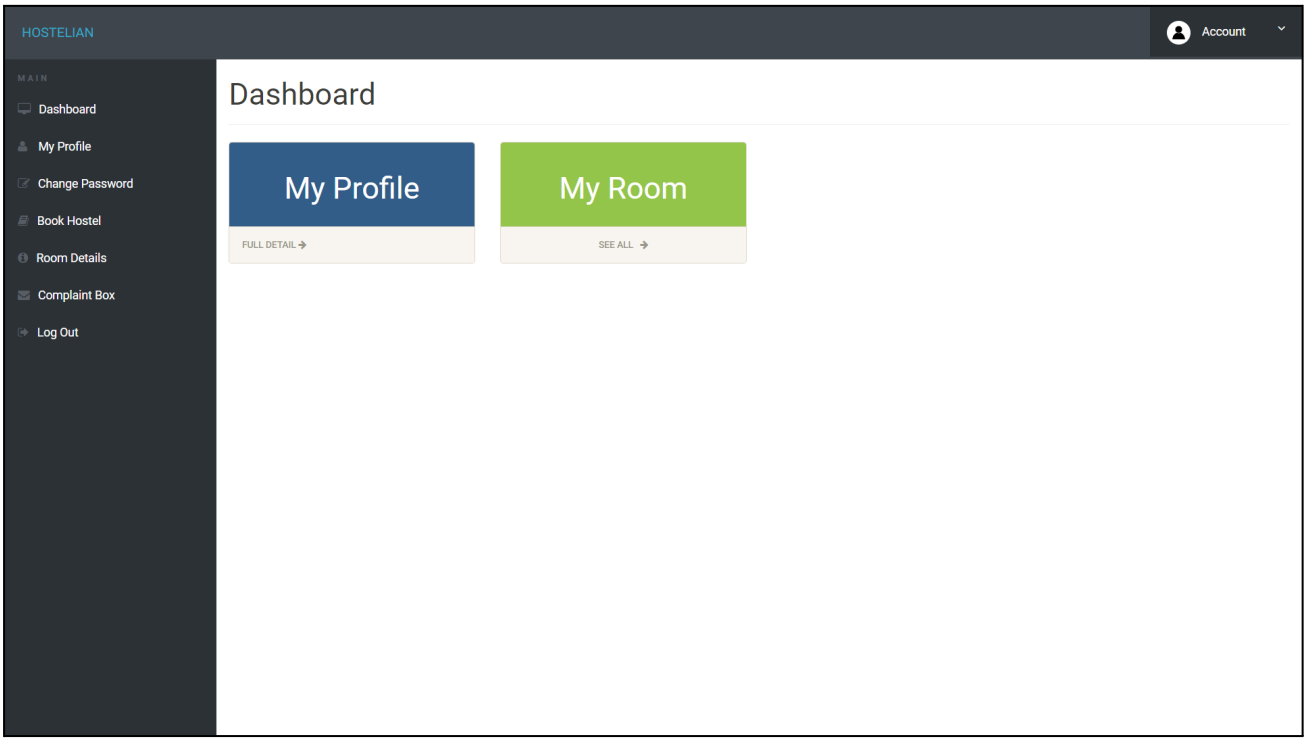


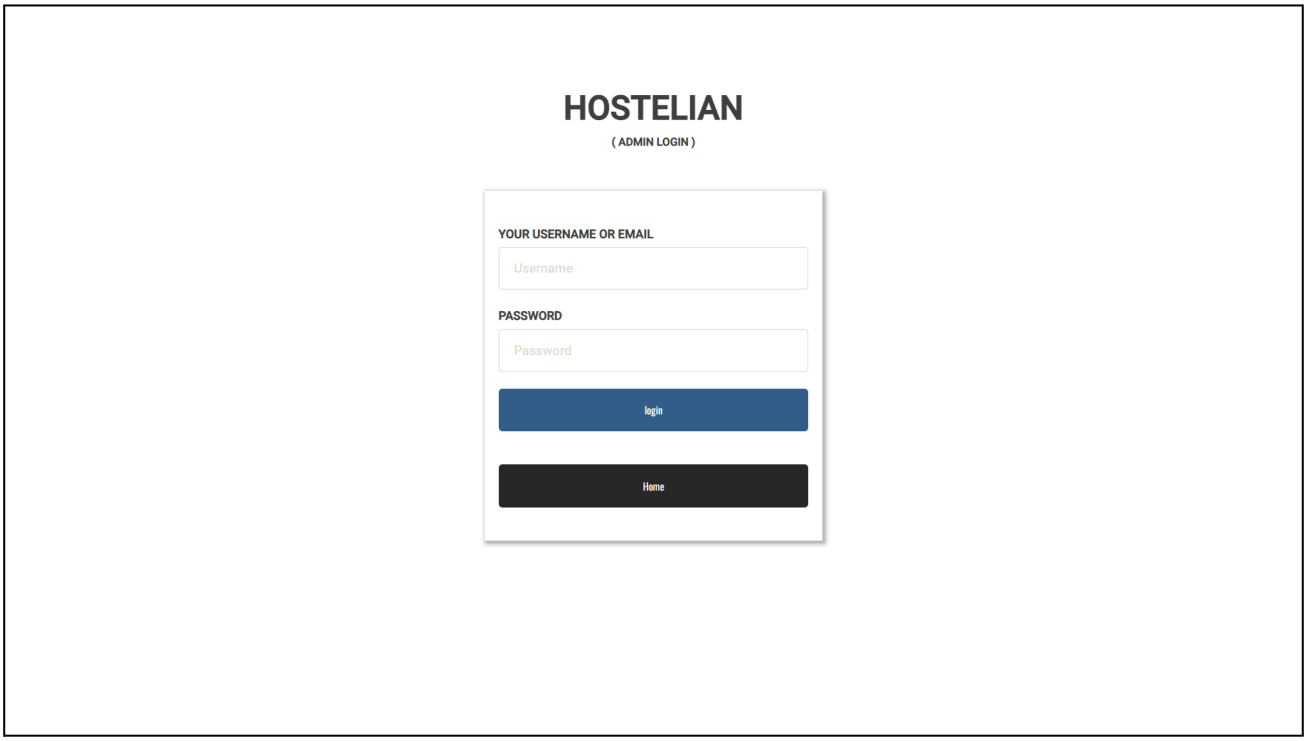
Fig 4.2: Data Flow Diagram

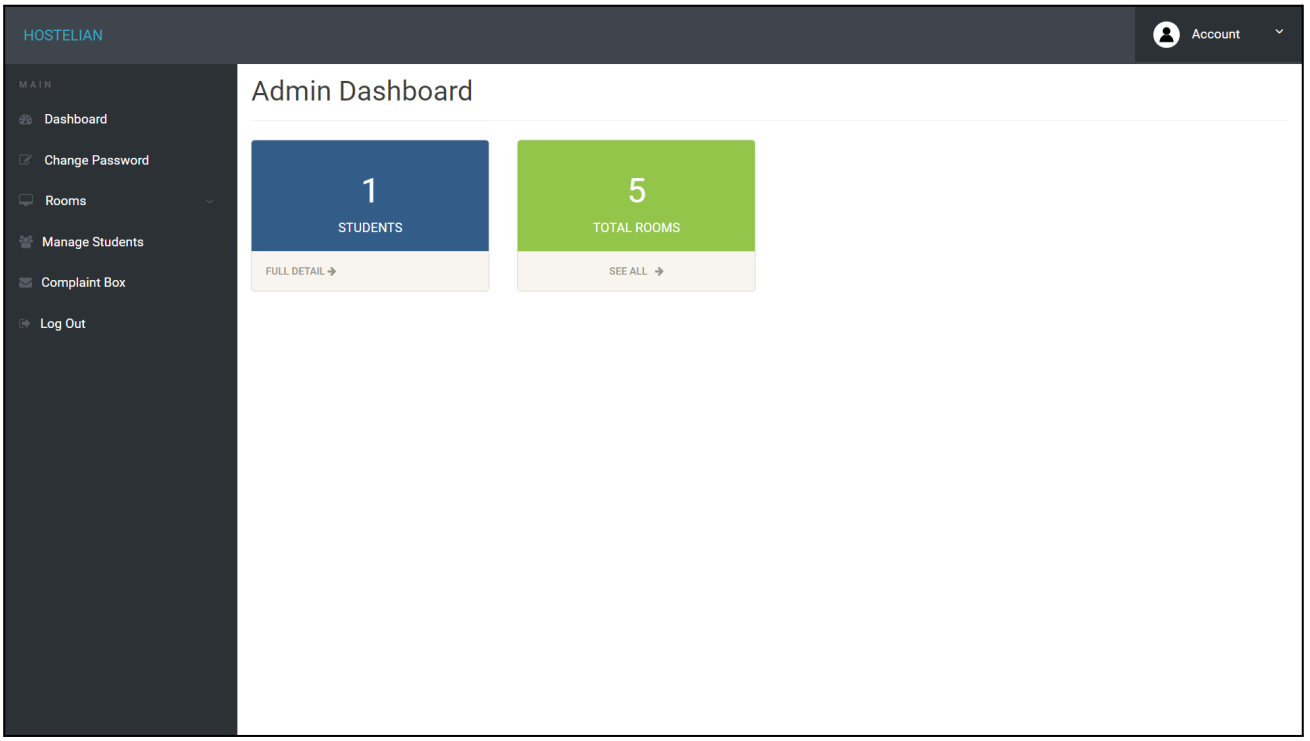
Chapter 5

Design and Implementation






HOSTELIAN
(ADMIN LOGIN)
YOUR USERNAME OR EMAIL
Username
PASSWORD
Password
login
Home


HOSTELIAN
Account
MAIN
Dashboard
Change Password
Rooms
Manage Students
Complaint Box
Log Out
Admin Dashboard
1
STUDENTS
FULL DETAIL
5
TOTAL ROOMS
SEE ALL

Chapter 6

Source Code

- **Login Page -**

```
import React, { useState } from 'react';

const LoginForm = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleEmailChange = (e) => {
    setEmail(e.target.value);
  };

  const handlePasswordChange = (e) => {
    setPassword(e.target.value);
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    // Perform login logic here with email and password
    // For simplicity, let's just log the values
    console.log('Email:', email);
    console.log('Password:', password);

    // Reset the form
    setEmail('');
    setPassword('');
  };

  return (
    <form onSubmit={handleSubmit}>
      <div>
        <label htmlFor="email">Email:</label>
        <input
          type="email"
          id="email"
          value={email}
          onChange={handleEmailChange}
          required
        />
      </div>
      <div>
        <label htmlFor="password">Password:</label>
        <input
          type="password"
          id="password"
          value={password}
          onChange={handlePasswordChange}
```

```
          required
        />
      </div>
      <button type="submit">Login</button>
    </form>
  );
};

export default LoginForm;
```

- **Registration Page -**

```
import React, { useState } from 'react';

const RegistrationForm = () => {
  const [firstName, setFirstName] = useState('');
  const [middleName, setMiddleName] = useState('');
  const [lastName, setLastName] = useState('');
  const [course, setCourse] = useState('');
  const [gender, setGender] = useState('');
  const [contact, setContact] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleFirstNameChange = (e) => {
    setFirstName(e.target.value);
  };

  const handleMiddleNameChange = (e) => {
    setMiddleName(e.target.value);
  };

  const handleLastNameChange = (e) => {
    setLastName(e.target.value);
  };

  const handleCourseChange = (e) => {
    setCourse(e.target.value);
  };

  const handleGenderChange = (e) => {
    setGender(e.target.value);
  };

  const handleContactChange = (e) => {
    setContact(e.target.value);
  };

  const handleEmailChange = (e) => {
    setEmail(e.target.value);
  };
```

```
const handlePasswordChange = (e) => {
  setPassword(e.target.value);
};

const handleSubmit = (e) => {
  e.preventDefault();

  // Perform registration logic here with form values
  // For simplicity, let's just log the values
  console.log('First Name:', firstName);
  console.log('Middle Name:', middleName);
  console.log('Last Name:', lastName);
  console.log('Course:', course);
  console.log('Gender:', gender);
  console.log('Contact:', contact);
  console.log('Email:', email);
  console.log('Password:', password);

  // Reset the form
  setFirstName('');
  setMiddleName('');
  setLastName('');
  setCourse('');
  setGender('');
  setContact('');
  setEmail('');
  setPassword('');
};

return (
  <form onSubmit={handleSubmit}>
    <div>
      <label htmlFor="firstName">First Name:</label>
      <input
        type="text"
        id="firstName"
        value={firstName}
        onChange={handleFirstNameChange}
        required
      />
    </div>
    <div>
      <label htmlFor="middleName">Middle Name:</label>
      <input
        type="text"
        id="middleName"
        value={middleName}
        onChange={handleMiddleNameChange}
      />
    </div>
    <div>
      <label htmlFor="lastName">Last Name:</label>
      <input
        type="text"
```

```
      id="lastName"
      value={lastName}
      onChange={handleLastNameChange}
      required
     />
    </div>
    <div>
     <label htmlFor="course">Course:</label>
     <select id="course" value={course} onChange={handleCourseChange}>
      <option value="">Select a Course</option>
      <option value="Computer">Computer</option>
      <option value="Mechanical">Mechanical</option>
      <option value="Civil">Civil</option>
      <option value="Chemical">Chemical</option>
      <option value="Electronics">Electronics</option>
      <option value="AI&ML">AI&ML</option>
     </select>
    </div>
    <div>
     <label htmlFor="gender">Gender:</label>
     <input
      type="radio"
      id="male"
      name="gender"
      value="Male"
      checked={gender === 'Male'}
      onChange={handleGenderChange}
     />
     <label htmlFor="male">Male</label>
     <input
      type="radio"
      id="female"
      name="gender"
      value="Female"
      checked={gender === 'Female'}
      onChange={handleGenderChange}
     />
     <label htmlFor="female">Female</label>
     <input
      type="radio"
      id="other"
      name="gender"
      value="Other"
      checked={gender === 'Other'}
      onChange={handleGenderChange}
     />
     <label htmlFor="other">Other</label>
    </div>
    <div>
     <label htmlFor="contact">Contact:</label>
     <input
      type="text"
      id="contact"
      value={contact}
```

```
        onChange={handleContactChange}
        required
      />
    </div>
    <div>
      <label htmlFor="email">Email:</label>
      <input
        type="email"
        id="email"
        value={email}
        onChange={handleEmailChange}
        required
      />
    </div>
```

- **Student Controller -**

```
package com.fullstacktraining.hostelian.controllers;

import com.fullstacktraining.hostelian.models.Student;
import com.fullstacktraining.hostelian.services.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@CrossOrigin
@RequestMapping("/student/")
public class StudentController {
    @Autowired
    private StudentService studentService;

    @PostMapping("/register")
    public String registerUser(@PathVariable Student student) {
        return studentService.register(student);
    }
}
```

- **Student Entity Model -**

```
package com.fullstacktraining.hostelian.models;

import jakarta.persistence.*;

@Entity
@Table(name = "student")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String middleName;
```

```
    private String lastName;
    private String course;
    private char gender;
    private Long phoneNumber;
    private String emailId;
    private String password;
    @Transient
    private String confirmPassword;

    // Getters and setters

    public Student() {
    }

    public Student(String firstName, String middleName, String lastName, String
course, char gender, Long phoneNumber, String emailId, String password, String
confirmPassword) {
        this.firstName = firstName;
        this.middleName = middleName;
        this.lastName = lastName;
        this.course = course;
        this.gender = gender;
        this.phoneNumber = phoneNumber;
        this.emailId = emailId;
        this.password = password;
        this.confirmPassword = confirmPassword;
    }
    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 getMiddleName() {
        return middleName;
    }

    public void setMiddleName(String middleName) {
        this.middleName = middleName;
    }

    public String getLastName() {
        return lastName;
    }
```

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

public String getCourse() {
    return course;
}

public void setCourse(String course) {
    course = course;
}

public char getGender() {
    return gender;
}

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

public Long getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(Long phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public String getEmailId() {
    return emailId;
}

public void setEmailId(String emailId) {
    this.emailId = emailId;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getConfirmPassword() {
    return confirmPassword;
}

public void setConfirmPassword(String confirmPassword) {
    this.confirmPassword = confirmPassword;
}

@Override
```

```
    public String toString() {
        return "User{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", middleName='" + middleName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", Course='" + course + '\'' +
                ", gender=" + gender +
                ", phoneNumber=" + phoneNumber +
                ", emailId='" + emailId + '\'' +
                ", password='" + password + '\'' +
                ", confirmPassword='" + confirmPassword + '\'' +
                '}';
    }
}
```

- **Student Service -**

```
package com.fullstacktraining.hostelian.services;

import com.fullstacktraining.hostelian.models.Student;
import com.fullstacktraining.hostelian.repositories.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class StudentServiceImpl implements StudentService {
    @Autowired
    private StudentRepository studentRepository;

    @Override
    public String register(Student student) {
        if(!student.getPassword().equals(student.getConfirmPassword()))
            return "Password Does not matches..!";

        studentRepository.save(student);
        return "Data inserted successfully..!";
    }
}
```

Chapter 7

Conclusion & Future Work

● Conclusion

To conclude the description about the project: The project developed using JAVA and MySQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement. The expanded functionality of today's software requires an appropriate approach towards software development. This hostel management software is designed for people who want to manage various activities in the hostel. For the past few years the number of educational institutions has been increasing rapidly. Thereby the number of hostels is also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who is running the hostel and software's are not usually used in this context. This particular project deals with the problems of managing a hostel and avoids the problems which occur when carried manually. Identification of the drawbacks of the existing system leads to the designing of a computerized system that will be compatible with the existing system with the system which is more user friendly and more GUI oriented.

● Future Work

- ☐ Our software will help users to save the data and manage their student details.
- ☐ Helps to maintain databases.
- ☐ We will update the GUI for the project in future.
- ☐ We will implement more new features in future.