Dharmsinh Desai University, Nadiad
Faculty of Technology

Computer Engineering Department
B.Tech - Semester – VI

Subject: Web Services Development

Project Title: <Hostel Management System>

Guided by: Prof. Ankit P. Vaishnav

Student Name (CE100) [20CEUOS052]

Report For Hostel Management System

Dharmsinh Desai University, Nadiad
Department of Computer Engineering
Faculty of Technology

CERTIFICATE

This is to certify that the project work carried out
in the subject of
Web Services Development
is the bonafide work of

Name of the student (CE100) [20CEUOS052]

of B. Tech. Semester-VI Computer Engineering during the academic year **2022-2023**.

Prof. Ankit P. Vaishnav	**Dr. C. K. Bhensdadia**
Assistant Professor	**Professor and HoD,**
Computer Engg. Department	Computer Engg. Department
Faculty of Technology	Faculty of Technology
D.D.U., Nadiad	D.D.U., Nadiad

Contents

Abstract

The Hostel Management Project is a web-based software system designed to automate the management and administration tasks of a hostel. This project aims to simplify the hostel management process, including room allocation, fee collection, student registration. The administrator is provided with better control over the transaction like adding the details of the student , deleting the student ,updating students fees status and also room details .The software's user-friendly interface allows hostel administrators to manage and monitor all hostel-related activities, and the system generates reports that aid in decision-making. With this Hostel Management System, hostel operations are streamlined, making it easier to manage and maintain hostels efficiently.

Brief Information

The Brief information about this project is that the admin takes care of all hostel tasks like adding, removing and updating student details , their room allocation details and their fee details.

Technology/Platform/Tools

1) ASP.NET Web API C#
2) React js
3) Visual studio code
4) Vs code

SRS:

Purpose

The hostel management system is computerized. Today is the era of computers .This software solves all the problems which are mentioned in the proposed section .The main objective of this system is to save money and time.

System provides the following features and functions.

Roles

In this system users can login as two type of roles

1) Administrator
2) Student

Software requirement specification:-

1)Authentication:

R.1.1 Sign Up:

Description: New user can sign up in system using basic details

Input: Email,Name,Password,Confirm password

Role:Student

Output: Confirmation of sign up.

R.1.2 Sign in:

Description: Once users sign up they sign in by correct email and password

Input: Email,Password.

Role:Student

Output: User redirected to home page.

2)Details Manage by Admin:

R.2.1 Update Hostel details and Room details:

Description: Admin can do crud operation on hostel details like

Hostel_name,hostel_id,hostel_room,number of student currently in hostel,room_status,how many student currently in particular room,room_type,room_number

Role:Admin

Output: updated details shown

R.2.2 Update Student details:

Description: Admin can do crud operation on Student details like

Student_id , student_name , student_email , student_address , student_mobile number , student_password

Role:Admin

Output: updated details shown

R.2.3 Update Fee details:

Description: Admin can do crud operation on Fee details like

Particular student payment_method , account_no , fee_status

Role:Admin

Output: updated details shown

R.2.4 Accept new student requests:

Description: Admin can accept or reject new student request

Role:Admin

Output: if accept, student added into system.. if reject,student is rejected

4) Student interface :

R.4.1 Student Profile:

Description: after login into system Student can see their fill up details and also see their fee status and hostel details

Input: Login as Student

Role:Student

Output: details shown on their profile

5) Admin interface :

R.5.1 Admin Dashboard:

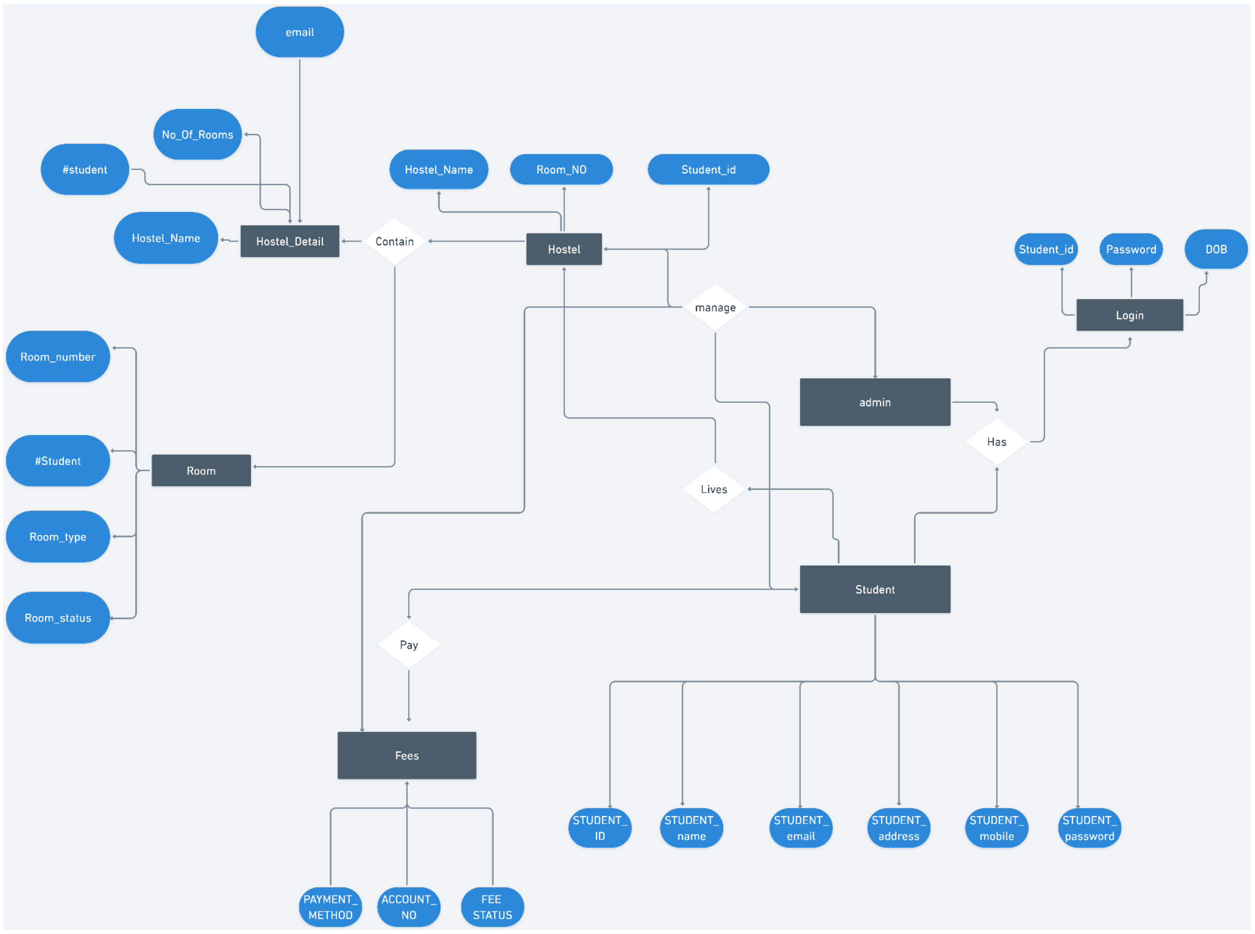
Description: after login into system admin can see all student details students fee status , club details , room details , hostel details

Input:Login as admin

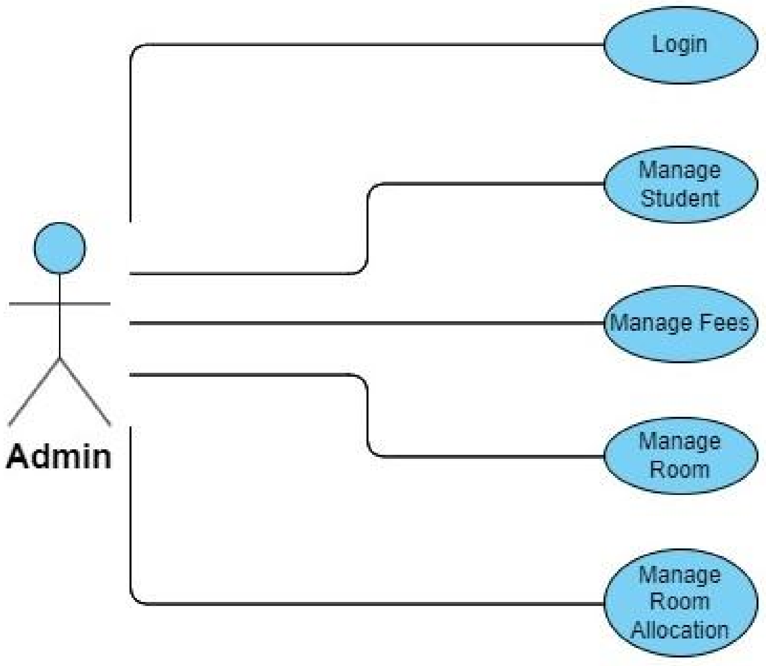
Role:Admin

Output: Dashboard which contain above details

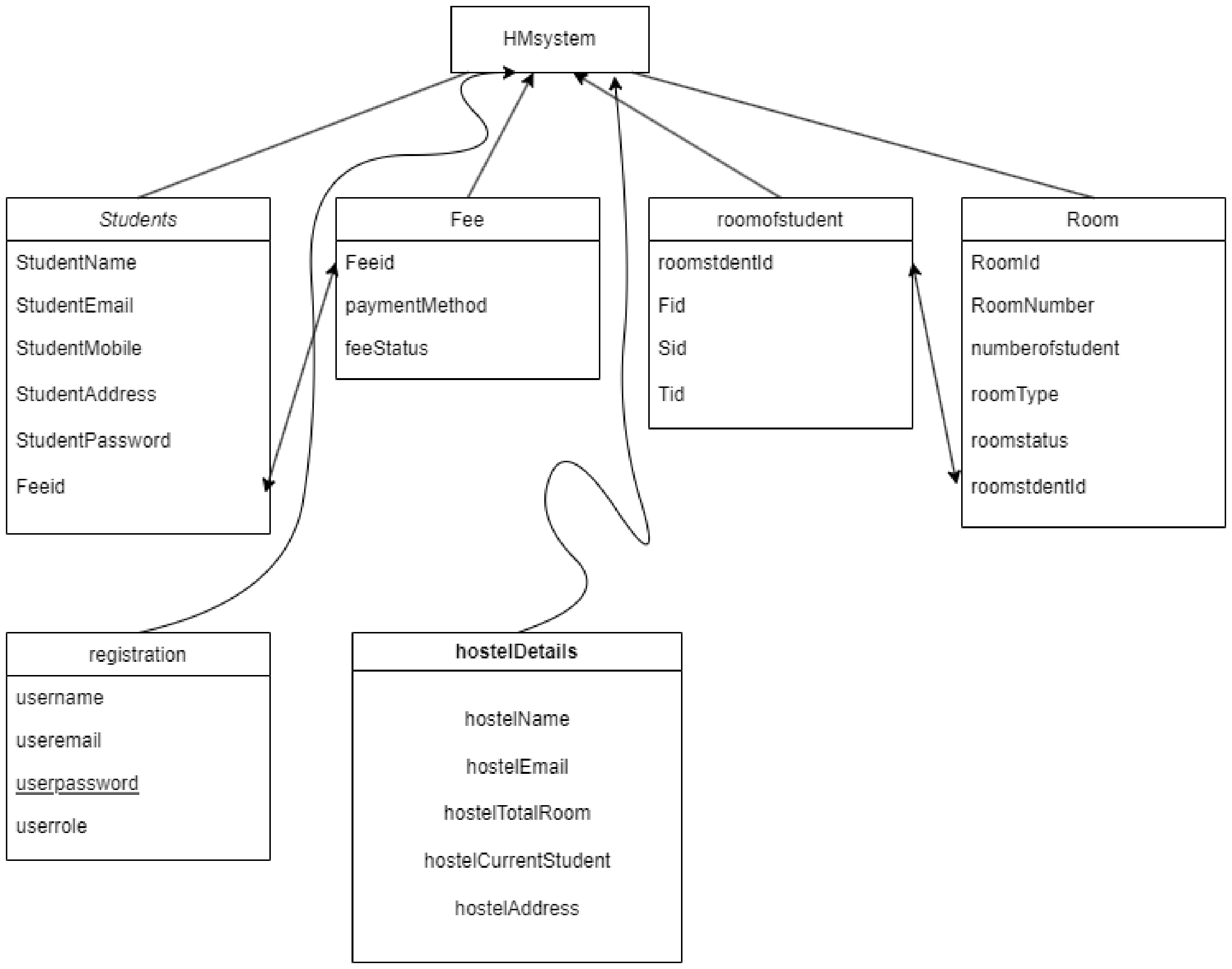
ER Diagram:-


email
No_Of_Rooms
#student
Hostel_Name
Hostel_Detail
Contain
Hostel_Name
Room_NO
Student_id
Hostel
Student_id
Password
DOB
Login
manage
admin
Has
Room_number
#Student
Room
Lives
Room_type
Student
Room_status
Pay
Fees
STUDENT_ ID
STUDENT_ name
STUDENT_ email
STUDENT_ address
STUDENT_ mobile
STUDENT_ password
PAYMENT_ METHOD
ACCOUNT_ NO
FEE STATUS

Use-Case Diagram:-

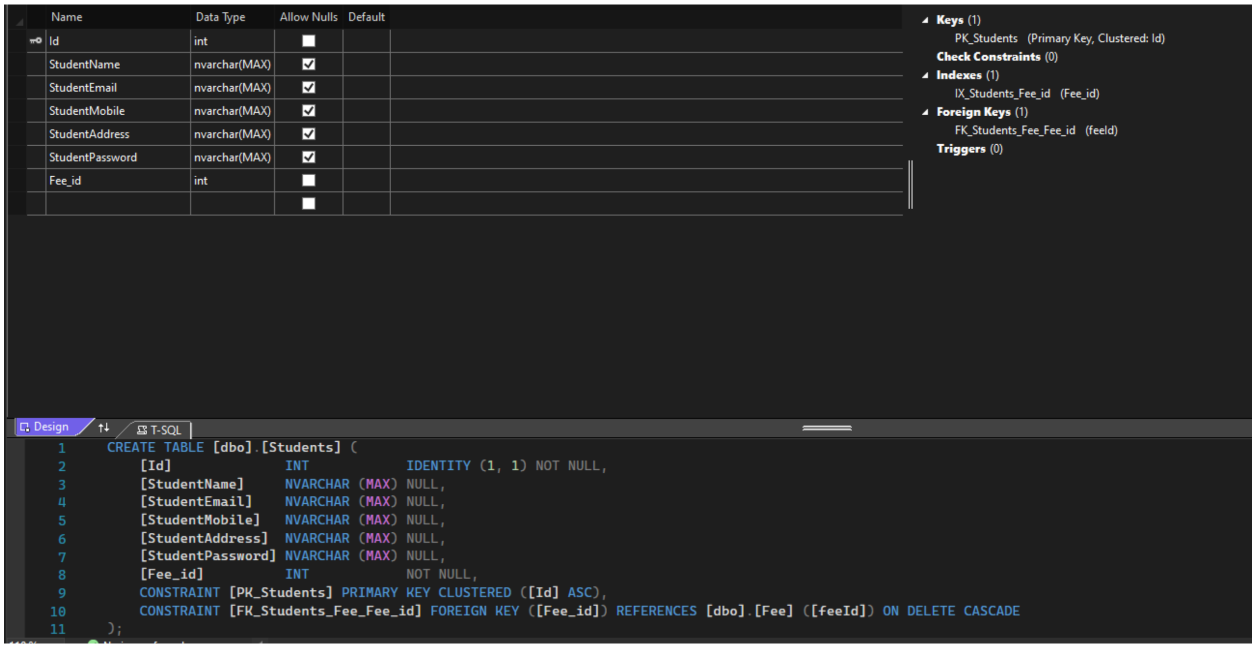


Database Design:-



Database Table

Student tabel:

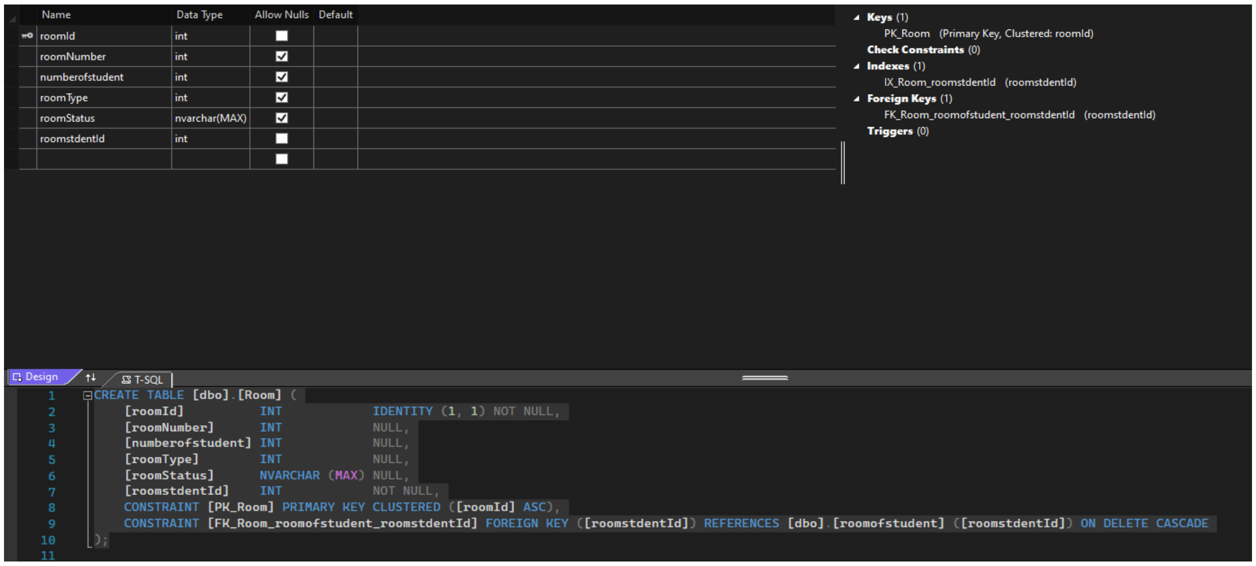


Name	Data Type	Allow Nulls	Default
Id	int	☐	
StudentName	nvarchar(MAX)	✓	
StudentEmail	nvarchar(MAX)	✓	
StudentMobile	nvarchar(MAX)	✓	
StudentAddress	nvarchar(MAX)	✓	
StudentPassword	nvarchar(MAX)	✓	
Fee_id	int	☐	
		☐	

Keys (1)
PK_Students (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (1)
IX_Students_Fee_id (Fee_id)
Foreign Keys (1)
FK_Students_Fee_Fee_id (feeId)
Triggers (0)

```sql
CREATE TABLE [dbo].[Students] (
    [Id]              INT            IDENTITY (1, 1) NOT NULL,
    [StudentName]     NVARCHAR (MAX) NULL,
    [StudentEmail]    NVARCHAR (MAX) NULL,
    [StudentMobile]   NVARCHAR (MAX) NULL,
    [StudentAddress]  NVARCHAR (MAX) NULL,
    [StudentPassword] NVARCHAR (MAX) NULL,
    [Fee_id]          INT            NOT NULL,
    CONSTRAINT [PK_Students] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_Students_Fee_Fee_id] FOREIGN KEY ([Fee_id]) REFERENCES [dbo].[Fee] ([feeId]) ON DELETE CASCADE
);
```

Room tabel:

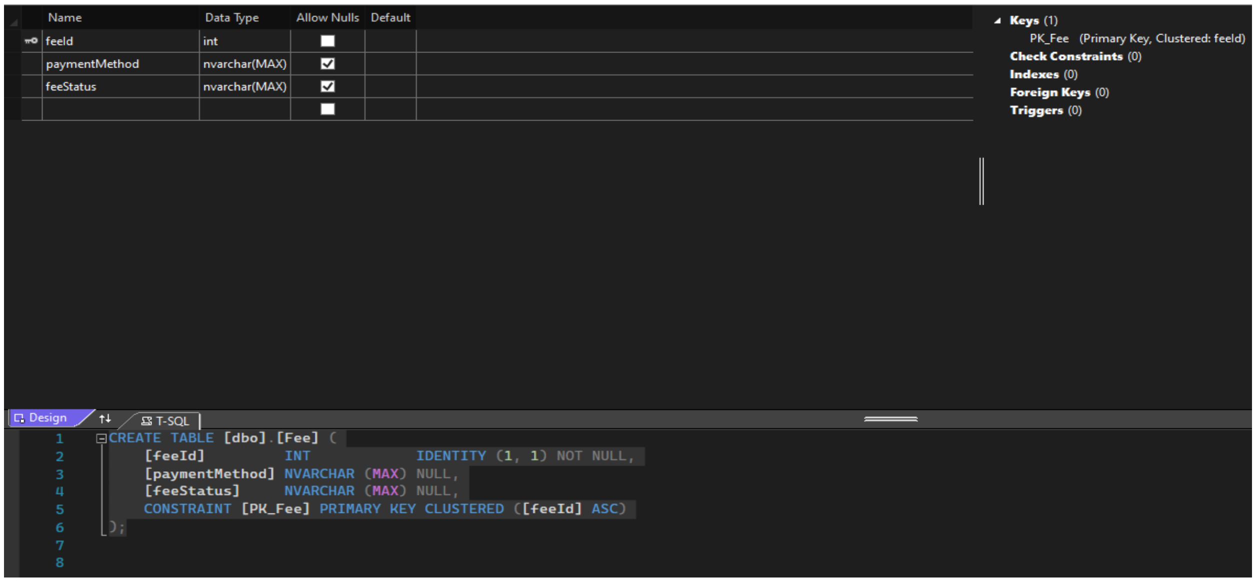


Name	Data Type	Allow Nulls	Default
roomId	int	☐	
roomNumber	int	✓	
numberofstudent	int	✓	
roomType	int	✓	
roomStatus	nvarchar(MAX)	✓	
roomstdentId	int	☐	
		☐	

Keys (1)
PK_Room (Primary Key, Clustered: roomId)
Check Constraints (0)
Indexes (1)
IX_Room_roomstdentId (roomstdentId)
Foreign Keys (1)
FK_Room_roomofstudent_roomstdentId (roomstdentId)
Triggers (0)

```sql
CREATE TABLE [dbo].[Room] (
    [roomId]          INT            IDENTITY (1, 1) NOT NULL,
    [roomNumber]      INT            NULL,
    [numberofstudent] INT            NULL,
    [roomType]        INT            NULL,
    [roomStatus]      NVARCHAR (MAX) NULL,
    [roomstdentId]    INT            NOT NULL,
    CONSTRAINT [PK_Room] PRIMARY KEY CLUSTERED ([roomId] ASC),
    CONSTRAINT [FK_Room_roomofstudent_roomstdentId] FOREIGN KEY ([roomstdentId]) REFERENCES [dbo].[roomofstudent] ([roomstdentId]) ON DELETE CASCADE
);
```

Fee table

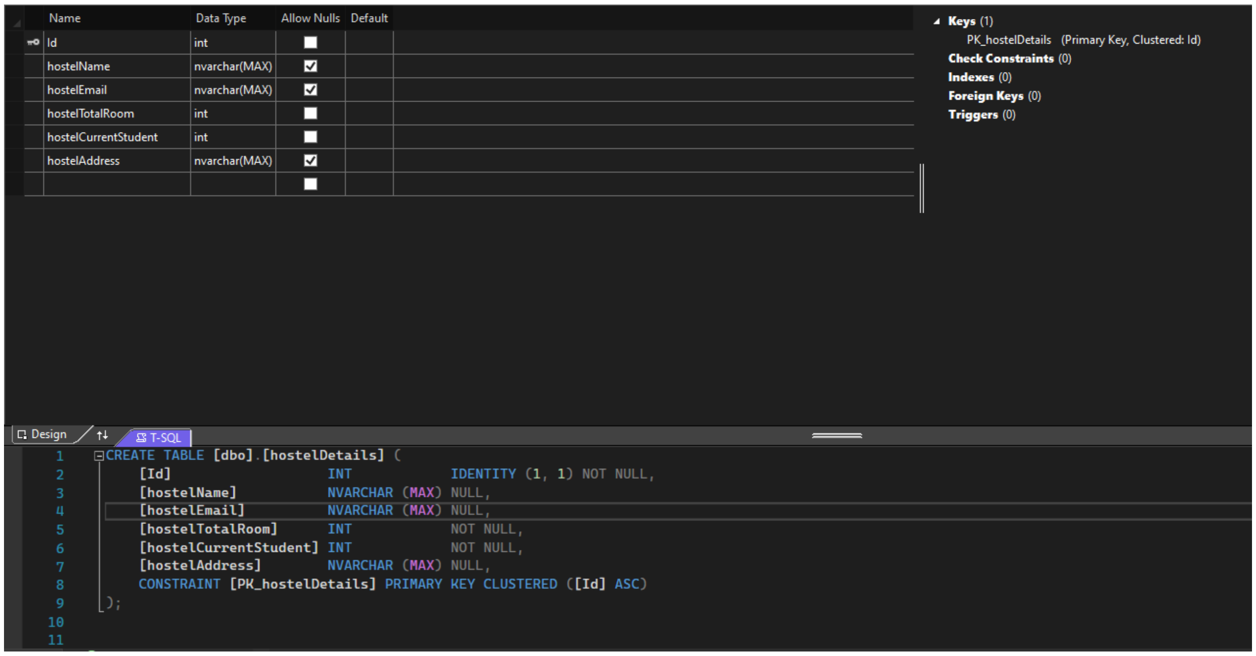


Name	Data Type	Allow Nulls	Default
feeId	int	☐	
paymentMethod	nvarchar(MAX)	✓	
feeStatus	nvarchar(MAX)	✓	
		☐	

Keys (1)
PK_Fee (Primary Key, Clustered: feeId)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```sql
CREATE TABLE [dbo].[Fee] (
    [feeId]         INT            IDENTITY (1, 1) NOT NULL,
    [paymentMethod] NVARCHAR (MAX) NULL,
    [feeStatus]     NVARCHAR (MAX) NULL,
    CONSTRAINT [PK_Fee] PRIMARY KEY CLUSTERED ([feeId] ASC)
);
```

hostelDetails table

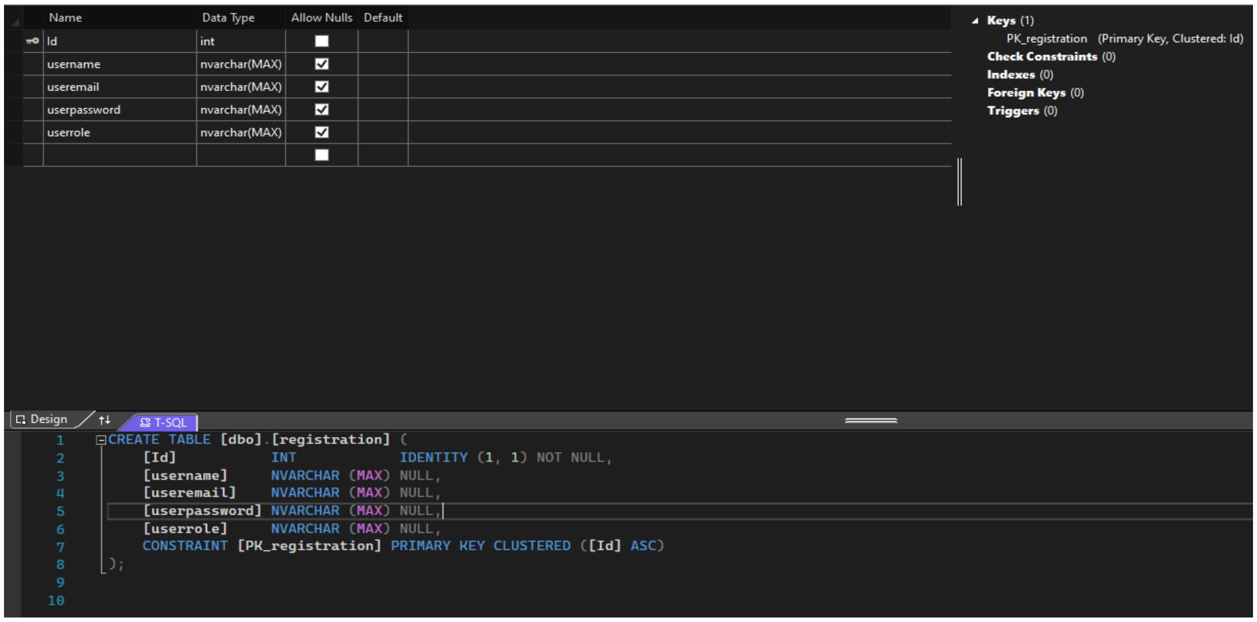


Name	Data Type	Allow Nulls	Default
Id	int	☐	
hostelName	nvarchar(MAX)	✓	
hostelEmail	nvarchar(MAX)	✓	
hostelTotalRoom	int	☐	
hostelCurrentStudent	int	☐	
hostelAddress	nvarchar(MAX)	✓	
		☐	

Keys (1)
PK_hostelDetails (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```sql
CREATE TABLE [dbo].[hostelDetails] (
    [Id]                   INT            IDENTITY (1, 1) NOT NULL,
    [hostelName]           NVARCHAR (MAX) NULL,
    [hostelEmail]          NVARCHAR (MAX) NULL,
    [hostelTotalRoom]      INT            NOT NULL,
    [hostelCurrentStudent] INT            NOT NULL,
    [hostelAddress]        NVARCHAR (MAX) NULL,
    CONSTRAINT [PK_hostelDetails] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Registration table

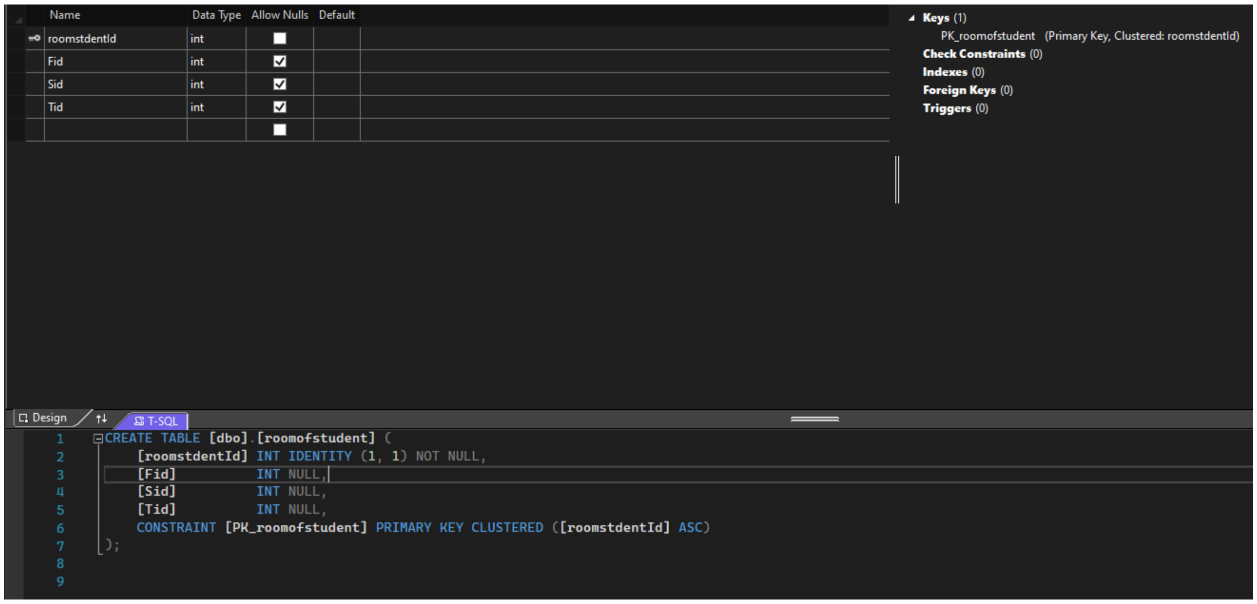


Name	Data Type	Allow Nulls	Default
Id	int	☐	
username	nvarchar(MAX)	✓	
useremail	nvarchar(MAX)	✓	
userpassword	nvarchar(MAX)	✓	
userrole	nvarchar(MAX)	✓	
		☐	

Keys (1)
PK_registration (Primary Key, Clustered: Id)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```sql
CREATE TABLE [dbo].[registration] (
    [Id]           INT            IDENTITY (1, 1) NOT NULL,
    [username]     NVARCHAR (MAX) NULL,
    [useremail]    NVARCHAR (MAX) NULL,
    [userpassword] NVARCHAR (MAX) NULL,
    [userrole]     NVARCHAR (MAX) NULL,
    CONSTRAINT [PK_registration] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

roomofstudent table



Name	Data Type	Allow Nulls	Default
roomstdentId	int	☐	
Fid	int	✓	
Sid	int	✓	
Tid	int	✓	
		☐	

Keys (1)
PK_roomofstudent (Primary Key, Clustered: roomstdentId)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```sql
CREATE TABLE [dbo].[roomofstudent] (
    [roomstdentId] INT IDENTITY (1, 1) NOT NULL,
    [Fid]          INT NULL,
    [Sid]          INT NULL,
    [Tid]          INT NULL,
    CONSTRAINT [PK_roomofstudent] PRIMARY KEY CLUSTERED ([roomstdentId] ASC)
);
```

Implementation details:

Models:

The Model represents the data and business logic of an application. It is responsible for retrieving, storing, and manipulating data, and for enforcing the business rules that govern the application.

Models which are used in this project

1) Student
2) Rooms
3) Roomofstudent
4) Fees
5) registrations
6) hostelDetails

Controller:

A Web API Controller is a component of a web application that handles incoming HTTP requests and generates corresponding HTTP responses. It serves as an interface between the client-side user interface and the server-side application. A Web API Controller is responsible for receiving the incoming HTTP request, processing it, and returning the appropriate HTTP response, typically in JSON format. It may also interact with various data sources to retrieve or update data.

Web API Controllers are commonly used in web applications to implement a service-oriented architecture (SOA) and to provide web services that can be consumed by other applications. They can be used to provide various functionalities, such as data retrieval, data insertion, data update, and data deletion. Additionally, Web API Controllers are highly customizable and can be extended to implement custom functionality as per the application's needs.

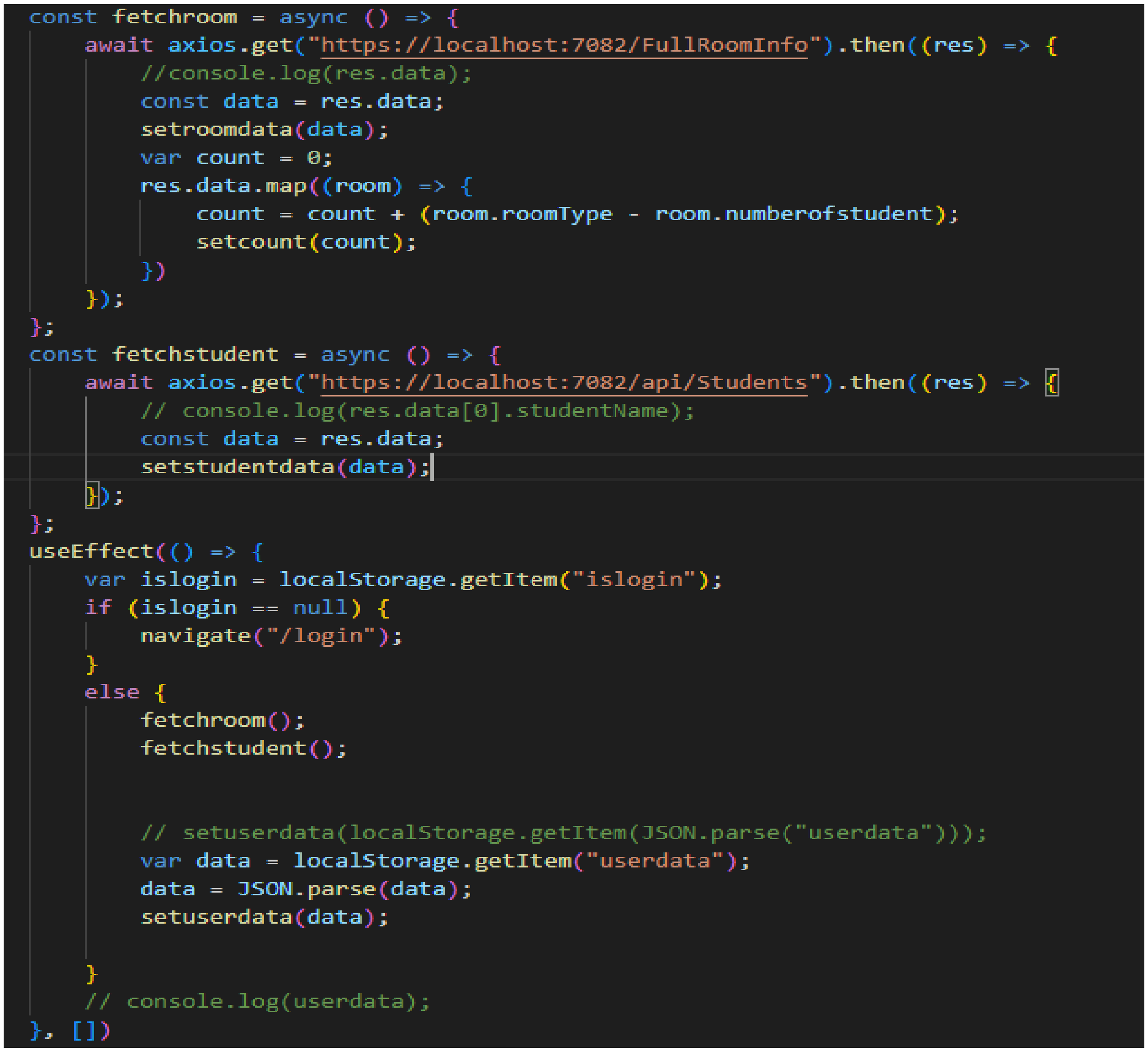
Web API Controllers are typically implemented in languages such as C#, Java, or Python, using frameworks such as ASP.NET, Spring, or Django.

Controllers which are used in this project:

1) StudentController
2) RoomsController
3) RoomofstudentController
4) FeesController
5) registrationsController
6) hostelDetailsController ….This all controller are used to do CRUD operations in its own entity

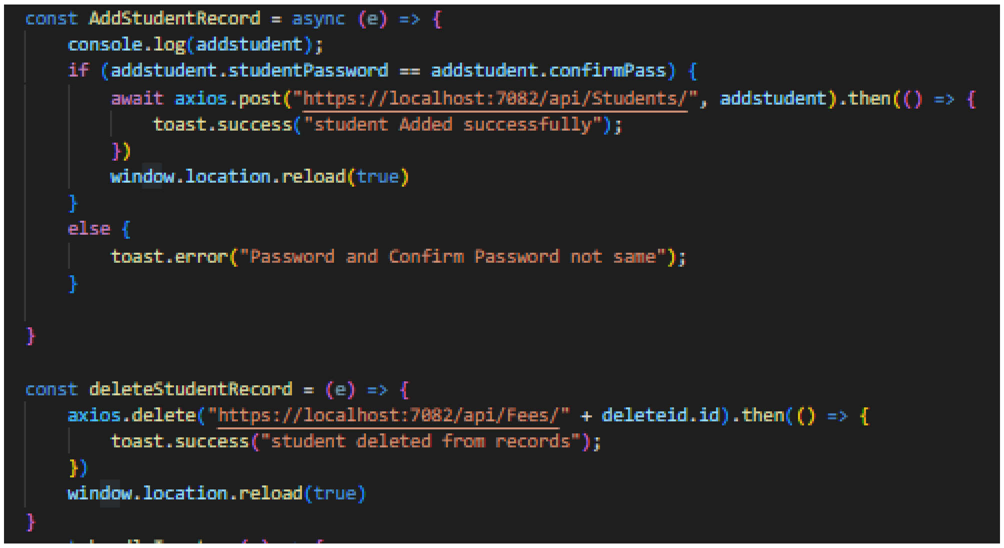
Major Functionality

It is used to display Total student,total room,available space on first page of dashboard

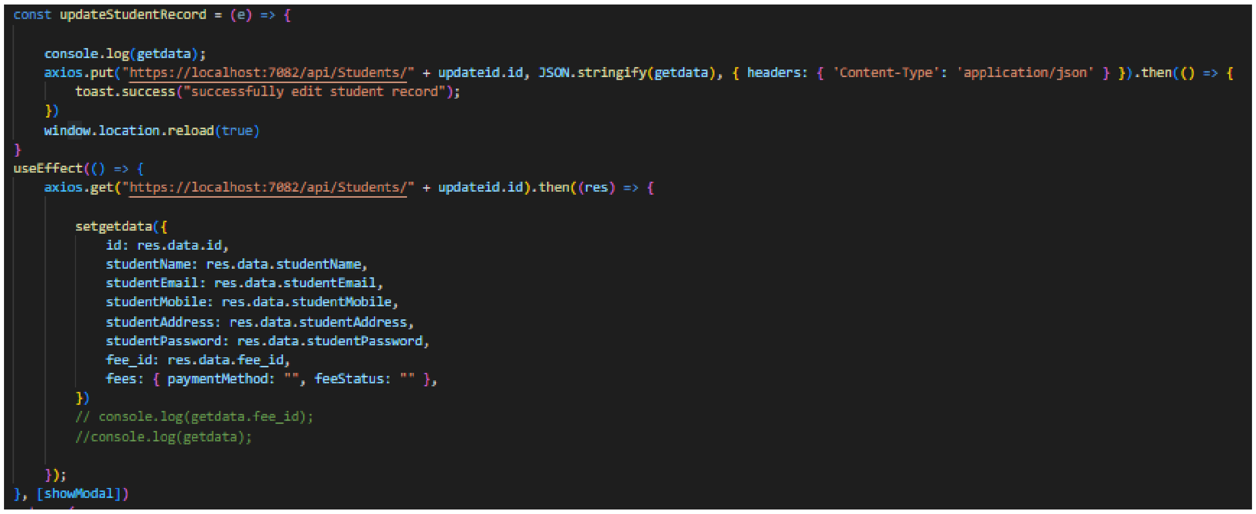


```
const fetchroom = async () => {
    await axios.get("https://localhost:7082/FullRoomInfo").then((res) => {
        //console.log(res.data);
        const data = res.data;
        setroomdata(data);
        var count = 0;
        res.data.map((room) => {
            count = count + (room.roomType - room.numberofstudent);
            setcount(count);
        })
    });
};
const fetchstudent = async () => {
    await axios.get("https://localhost:7082/api/Students").then((res) => {
        // console.log(res.data[0].studentName);
        const data = res.data;
        setstudentdata(data);
    });
};
useEffect(() => {
    var islogin = localStorage.getItem("islogin");
    if (islogin == null) {
        navigate("/login");
    }
    else {
        fetchroom();
        fetchstudent();


        // setuserdata(localStorage.getItem(JSON.parse("userdata")));
        var data = localStorage.getItem("userdata");
        data = JSON.parse(data);
        setuserdata(data);

    }
    // console.log(userdata);
}, [])
```

It is used to display each student details and used to add , delete and update student details

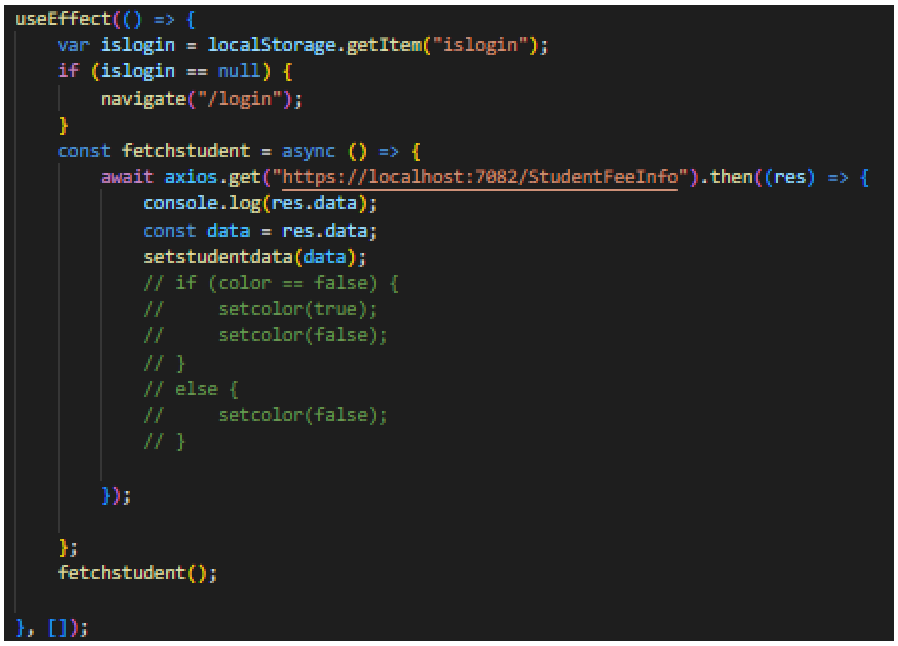


```
const AddStudentRecord = async (e) => {
    console.log(addstudent);
    if (addstudent.studentPassword == addstudent.confirmPass) {
        await axios.post("https://localhost:7082/api/Students/", addstudent).then(() => {
            toast.success("student Added successfully");
        })
        window.location.reload(true)
    }
    else {
        toast.error("Password and Confirm Password not same");
    }

}

const deleteStudentRecord = (e) => {
    axios.delete("https://localhost:7082/api/Fees/" + deleteid.id).then(() => {
        toast.success("student deleted from records");
    })
    window.location.reload(true)
}
```

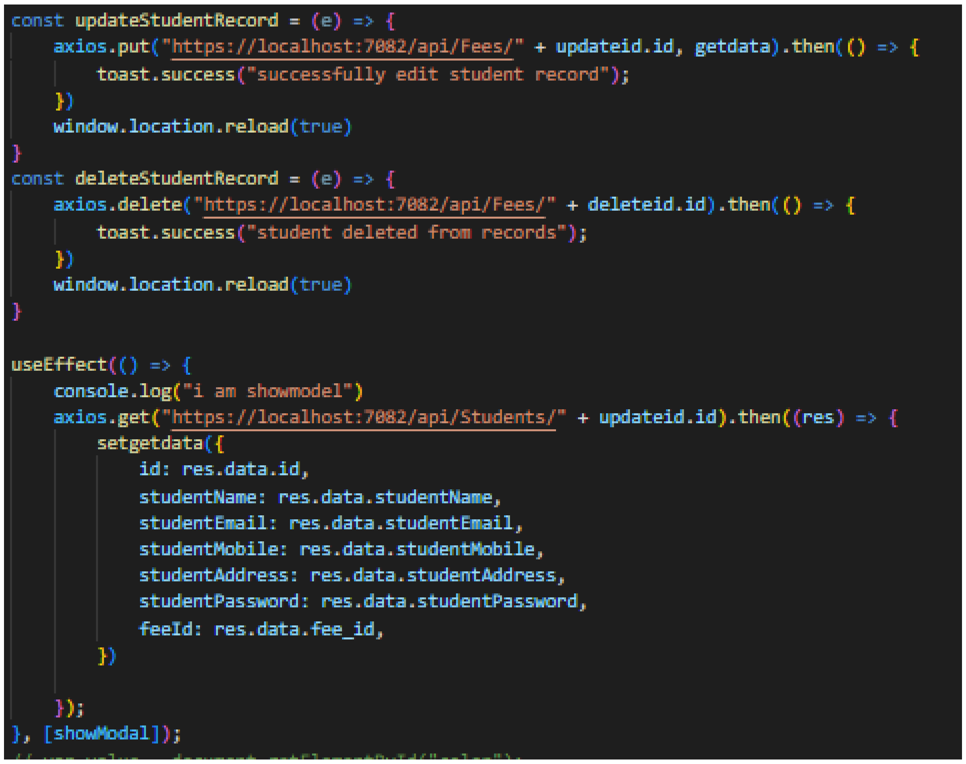


```
const updateStudentRecord = (e) => {

    console.log(getdata);
    axios.put("https://localhost:7082/api/Students/" + updateid.id, JSON.stringify(getdata), { headers: { 'Content-Type': 'application/json' } }).then(() => {
        toast.success("successfully edit student record");
    })
    window.location.reload(true)
}
useEffect(() => {
    axios.get("https://localhost:7082/api/Students/" + updateid.id).then((res) => {

        setgetdata({
            id: res.data.id,
            studentName: res.data.studentName,
            studentEmail: res.data.studentEmail,
            studentMobile: res.data.studentMobile,
            studentAddress: res.data.studentAddress,
            studentPassword: res.data.studentPassword,
            fee_id: res.data.fee_id,
            fees: { paymentMethod: "", feeStatus: "" },
        })
        // console.log(getdata.fee_id);
        //console.log(getdata);

    });
}, [showModal])
```

It is used to display fees information of each student and update fees information

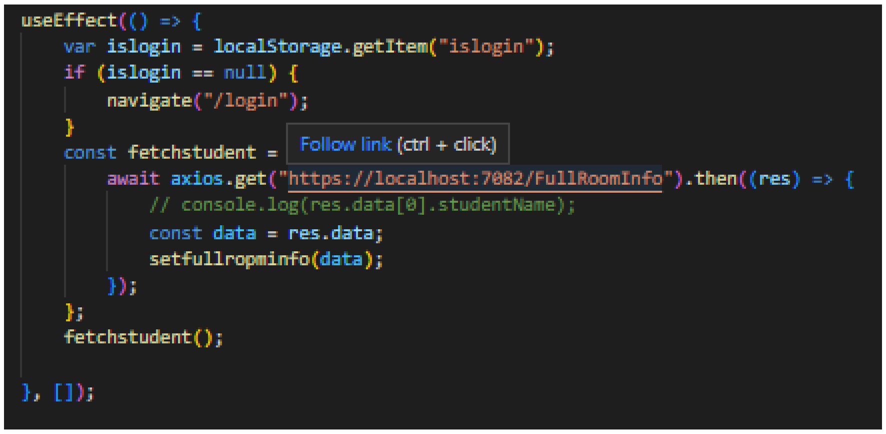


```
useEffect(() => {
    var islogin = localStorage.getItem("islogin");
    if (islogin == null) {
        navigate("/login");
    }
    const fetchstudent = async () => {
        await axios.get("https://localhost:7082/StudentFeeInfo").then((res) => {
            console.log(res.data);
            const data = res.data;
            setstudentdata(data);
            // if (color == false) {
            //     setcolor(true);
            //     setcolor(false);
            // }
            // else {
            //     setcolor(false);
            // }

        });

    };
    fetchstudent();

}, []);
```



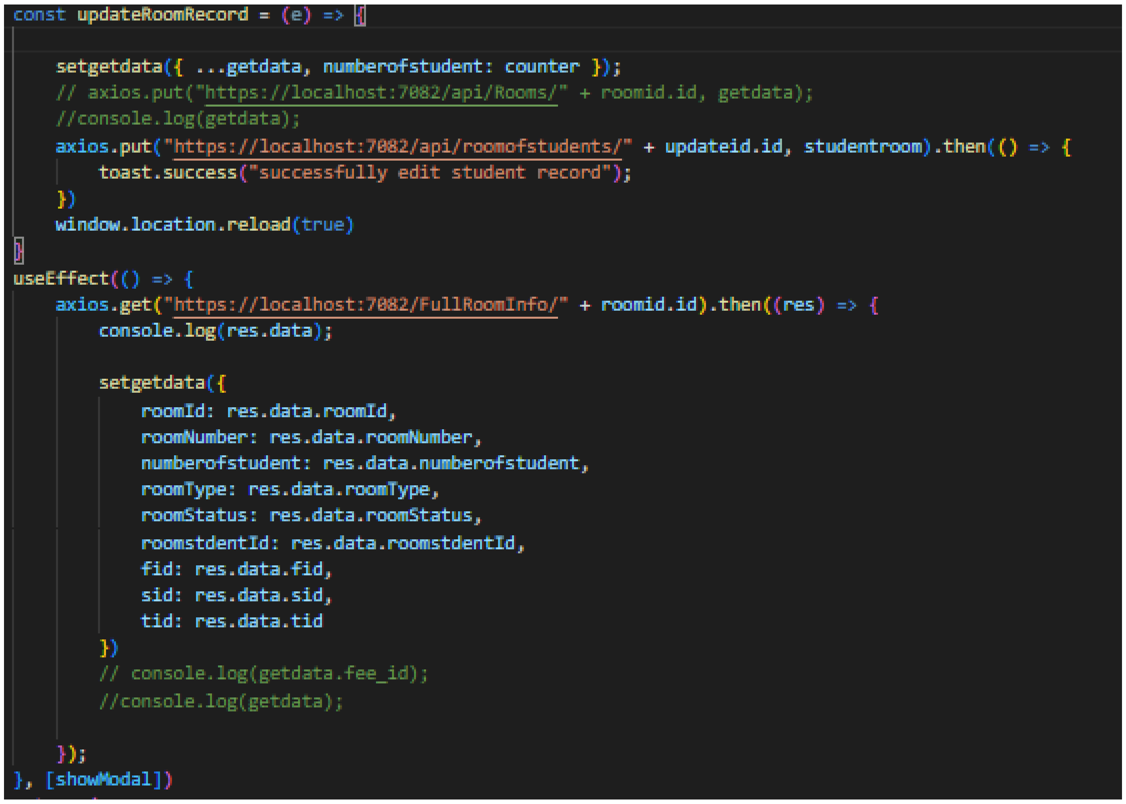
```
const updateStudentRecord = (e) => {
    axios.put("https://localhost:7082/api/Fees/" + updateid.id, getdata).then(() => {
        toast.success("successfully edit student record");
    })
    window.location.reload(true)
}
const deleteStudentRecord = (e) => {
    axios.delete("https://localhost:7082/api/Fees/" + deleteid.id).then(() => {
        toast.success("student deleted from records");
    })
    window.location.reload(true)
}

useEffect(() => {
    console.log("i am showmodel")
    axios.get("https://localhost:7082/api/Students/" + updateid.id).then((res) => {
        setgetdata({
            id: res.data.id,
            studentName: res.data.studentName,
            studentEmail: res.data.studentEmail,
            studentMobile: res.data.studentMobile,
            studentAddress: res.data.studentAddress,
            studentPassword: res.data.studentPassword,
            feeId: res.data.fee_id,
        })

    });
}, [showModal]);
```

It is used to display which student is allocated in which room with room number



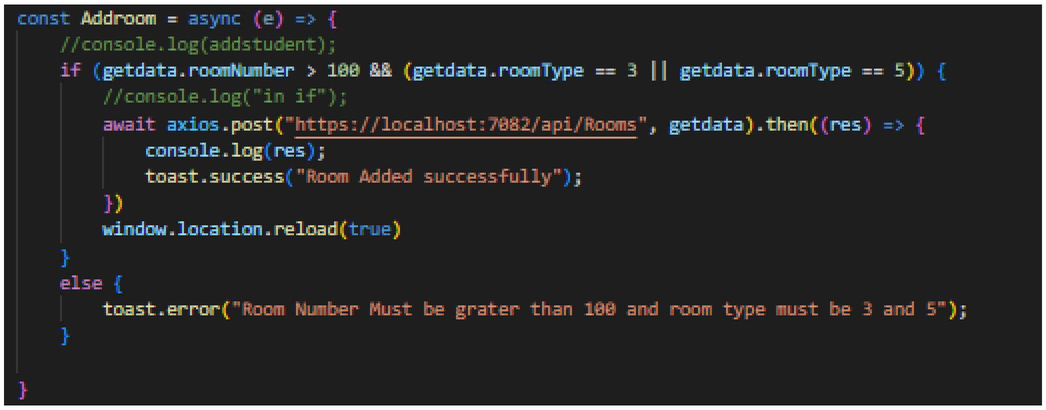
```
useEffect(() => {
    var islogin = localStorage.getItem("islogin");
    if (islogin == null) {
        navigate("/login");
    }
    const fetchstudent = 
        await axios.get("https://localhost:7082/FullRoomInfo").then((res) => {
            // console.log(res.data[0].studentName);
            const data = res.data;
            setfullropminfo(data);
        });
    };
    fetchstudent();

}, []);
```

Follow link (ctrl + click)

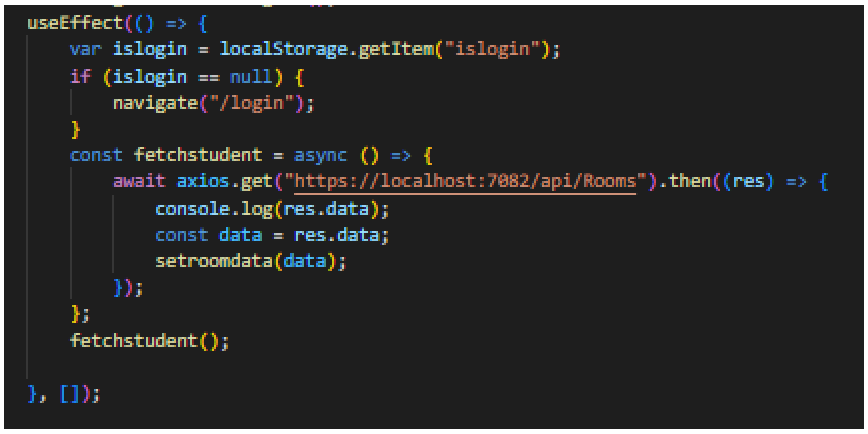


```
const updateRoomRecord = (e) => {

    setgetdata({ ...getdata, numberofstudent: counter });
    // axios.put("https://localhost:7082/api/Rooms/" + roomid.id, getdata);
    //console.log(getdata);
    axios.put("https://localhost:7082/api/roomofstudents/" + updateid.id, studentroom).then(() => {
        toast.success("successfully edit student record");
    })
    window.location.reload(true)
}
useEffect(() => {
    axios.get("https://localhost:7082/FullRoomInfo/" + roomid.id).then((res) => {
        console.log(res.data);

        setgetdata({
            roomId: res.data.roomId,
            roomNumber: res.data.roomNumber,
            numberofstudent: res.data.numberofstudent,
            roomType: res.data.roomType,
            roomStatus: res.data.roomStatus,
            roomstdentId: res.data.roomstdentId,
            fid: res.data.fid,
            sid: res.data.sid,
            tid: res.data.tid
        })
        // console.log(getdata.fee_id);
        //console.log(getdata);

    });
}, [showModal])
```

It is used to display how many students are currently in each room and used to add room

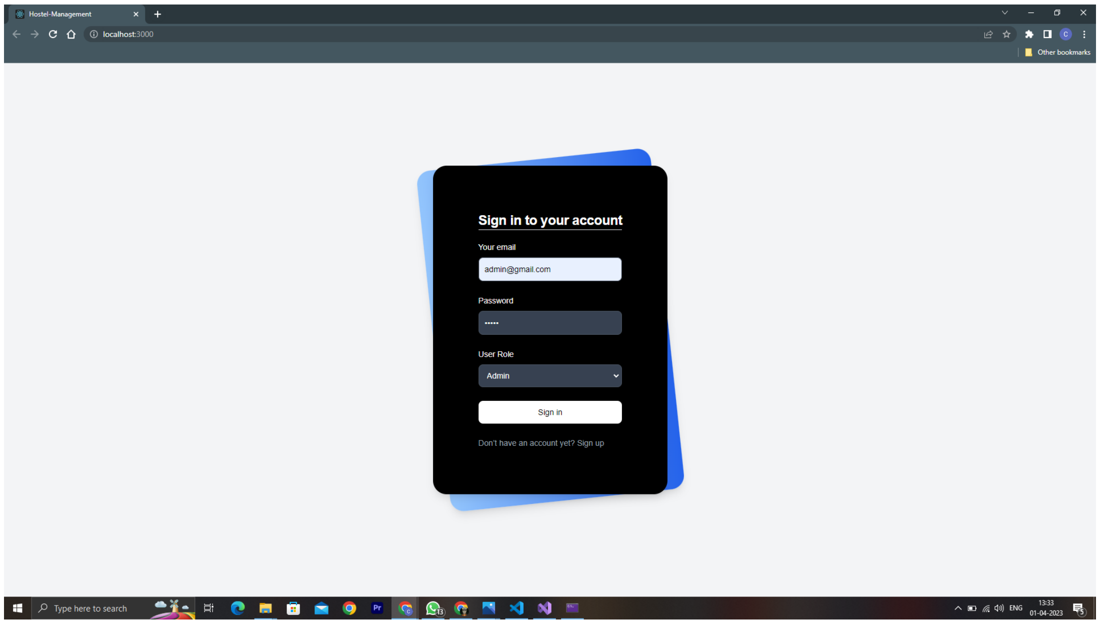


```
const Addroom = async (e) => {
    //console.log(addstudent);
    if (getdata.roomNumber > 100 && (getdata.roomType == 3 || getdata.roomType == 5)) {
        //console.log("in if");
        await axios.post("https://localhost:7082/api/Rooms", getdata).then((res) => {
            console.log(res);
            toast.success("Room Added successfully");
        })
        window.location.reload(true)
    }
    else {
        toast.error("Room Number Must be grater than 100 and room type must be 3 and 5");
    }

}
```



```
useEffect(() => {
    var islogin = localStorage.getItem("islogin");
    if (islogin == null) {
        navigate("/login");
    }
    const fetchstudent = async () => {
        await axios.get("https://localhost:7082/api/Rooms").then((res) => {
            console.log(res.data);
            const data = res.data;
            setroomdata(data);
        });
    };
    fetchstudent();

}, []);
```

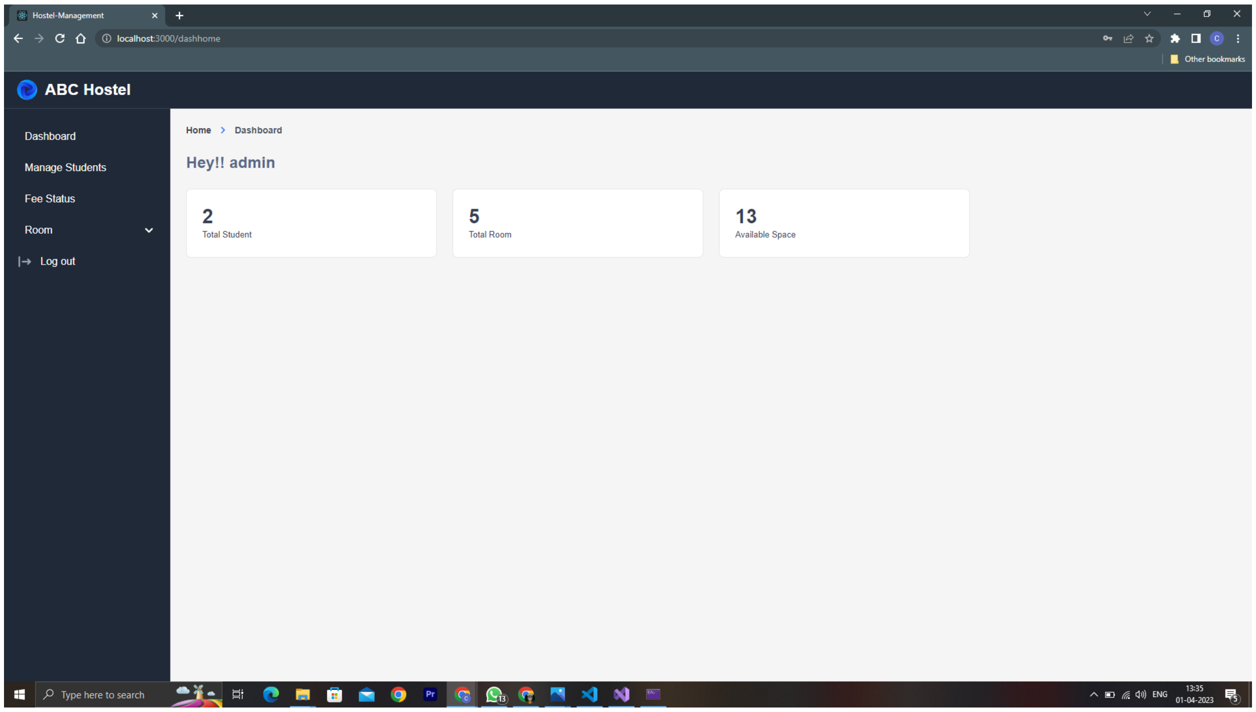
ScreenShorts


Sign in to your account

Your email: admin@gmail.com

Password: ••••••

User Role: Admin

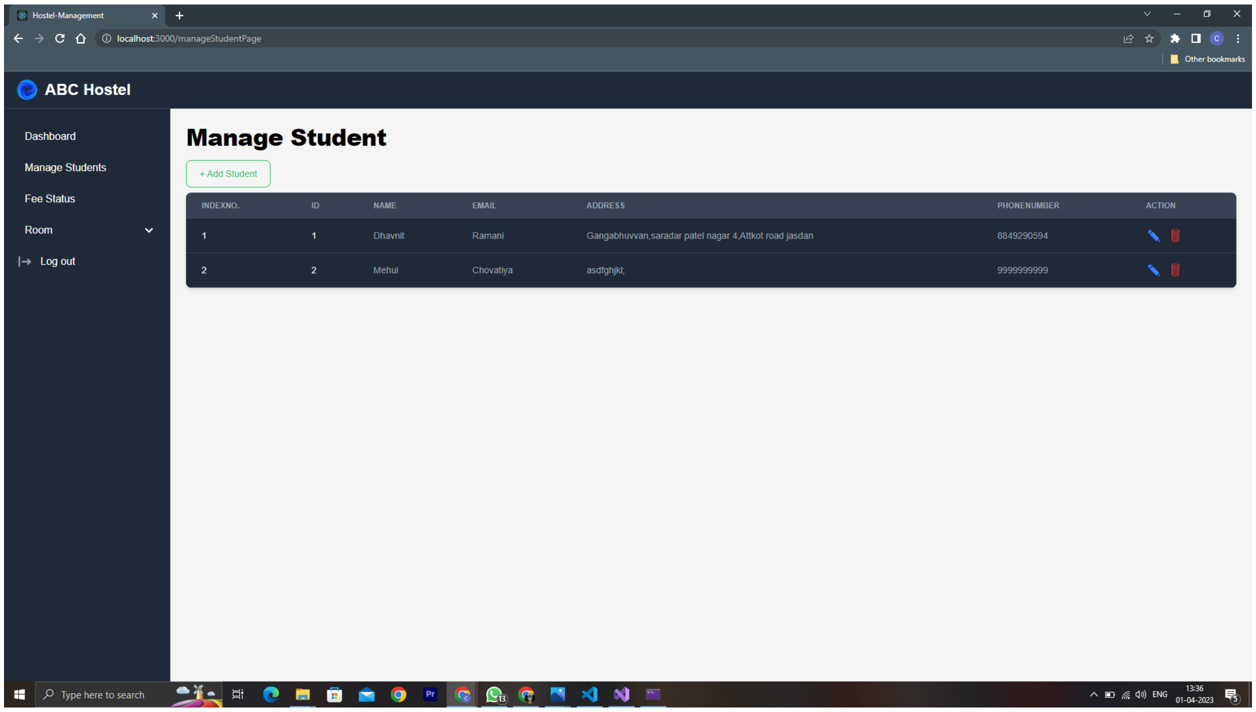


Hey!! admin

2 Total Student

5 Total Room

13 Available Space



Hostel-Management

localhost:3000/manageStudentPage

ABC Hostel

Dashboard

Manage Students

Fee Status

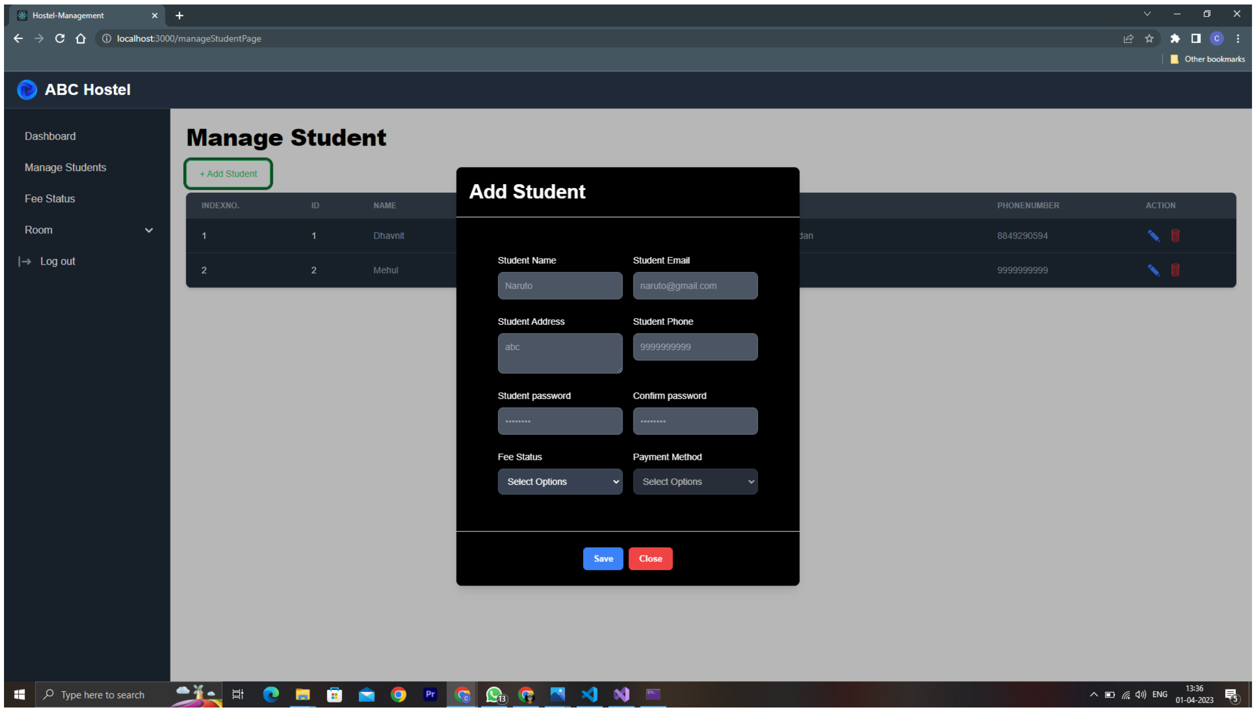
Room

Log out

Manage Student

+ Add Student

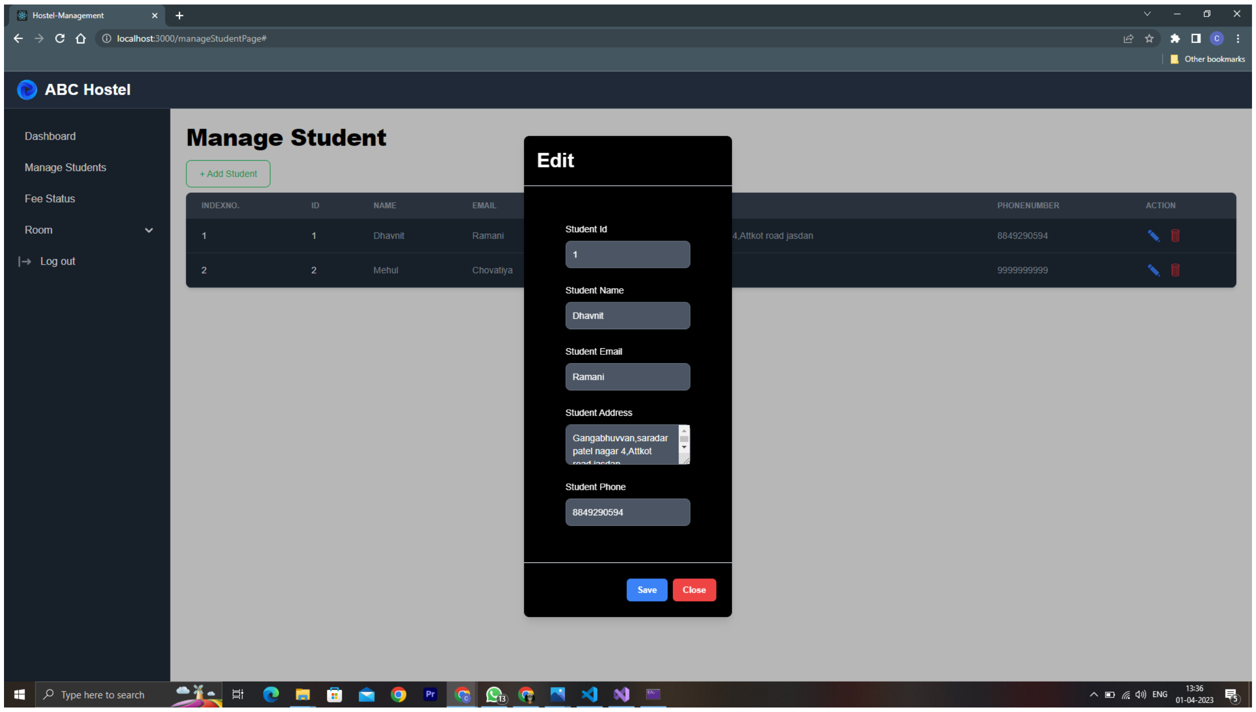
INDEXNO.	ID	NAME	EMAIL	ADDRESS	PHONENUMBER	ACTION
1	1	Dhavnit	Ramani	Gangabhuvvan,saradar patel nagar 4,Attkot road jasdan	8849290594	
2	2	Mehul	Chovatiya	asdfghjkl;	9999999999	



Add Student

Student Name: Naruto	Student Email: naruto@gmail.com
Student Address: abc	Student Phone: 9999999999
Student password: ········	Confirm password: ········
Fee Status: Select Options	Payment Method: Select Options

Save Close



Hostel-Management

localhost:3000/manageStudentPage#

ABC Hostel

Dashboard
Manage Students
Fee Status
Room
Log out

Manage Student

+ Add Student

INDEXNO.	ID	NAME	EMAIL		PHONENUMBER	ACTION
1	1	Dhavnit	Ramani	4,Attkot road jasdan	8849290594	
2	2	Mehul	Chovatiya		9999999999	

Edit

Student Id: 1

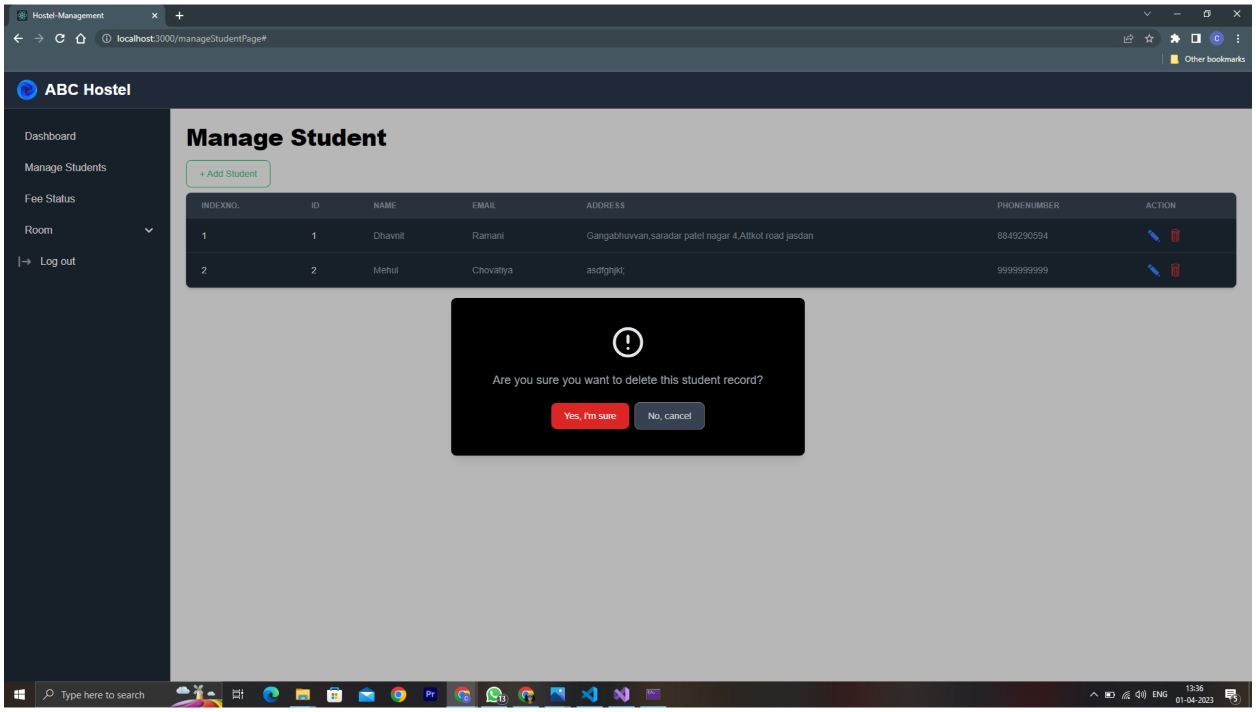
Student Name: Dhavnit

Student Email: Ramani

Student Address: Gangabhuvvan,saradar patel nagar 4,Attkot road jasdan

Student Phone: 8849290594

Save Close



Hostel-Management

localhost:3000/manageStudentPage#

ABC Hostel

Dashboard
Manage Students
Fee Status
Room
Log out

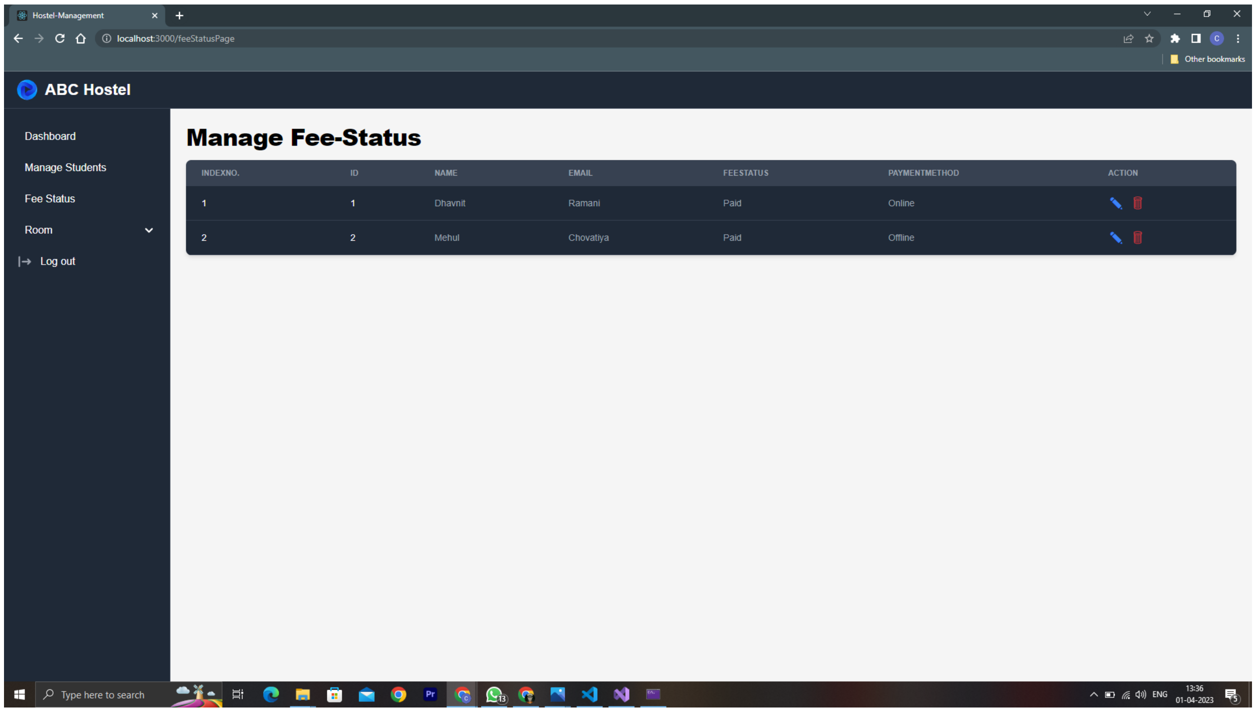
Manage Student

+ Add Student

INDEXNO.	ID	NAME	EMAIL	ADDRESS	PHONENUMBER	ACTION
1	1	Dhavnit	Ramani	Gangabhuvvan,saradar patel nagar 4,Attkot road jasdan	8849290594	
2	2	Mehul	Chovatiya	asdfghjkl;	9999999999	

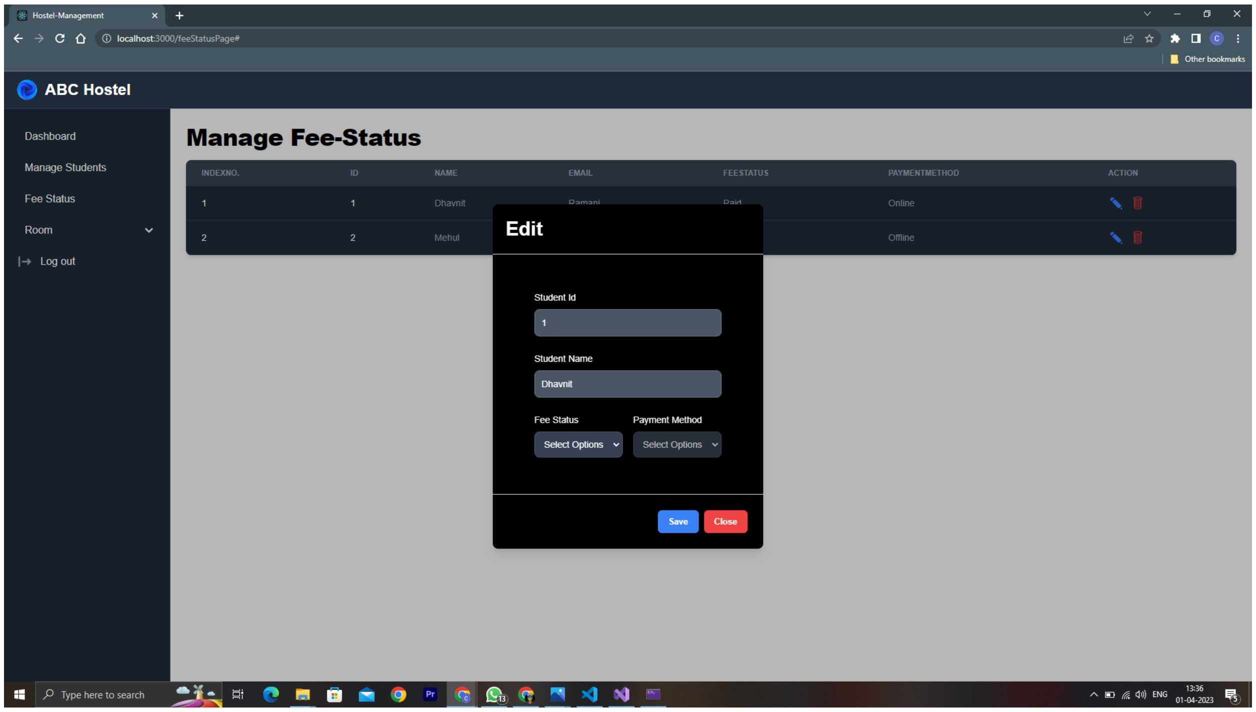
Are you sure you want to delete this student record?

Yes, I'm sure No, cancel



Manage Fee-Status

INDEXNO.	ID	NAME	EMAIL	FEESTATUS	PAYMENTMETHOD	ACTION
1	1	Dhavnit	Ramani	Paid	Online	
2	2	Mehul	Chovatiya	Paid	Offline	



Manage Fee-Status

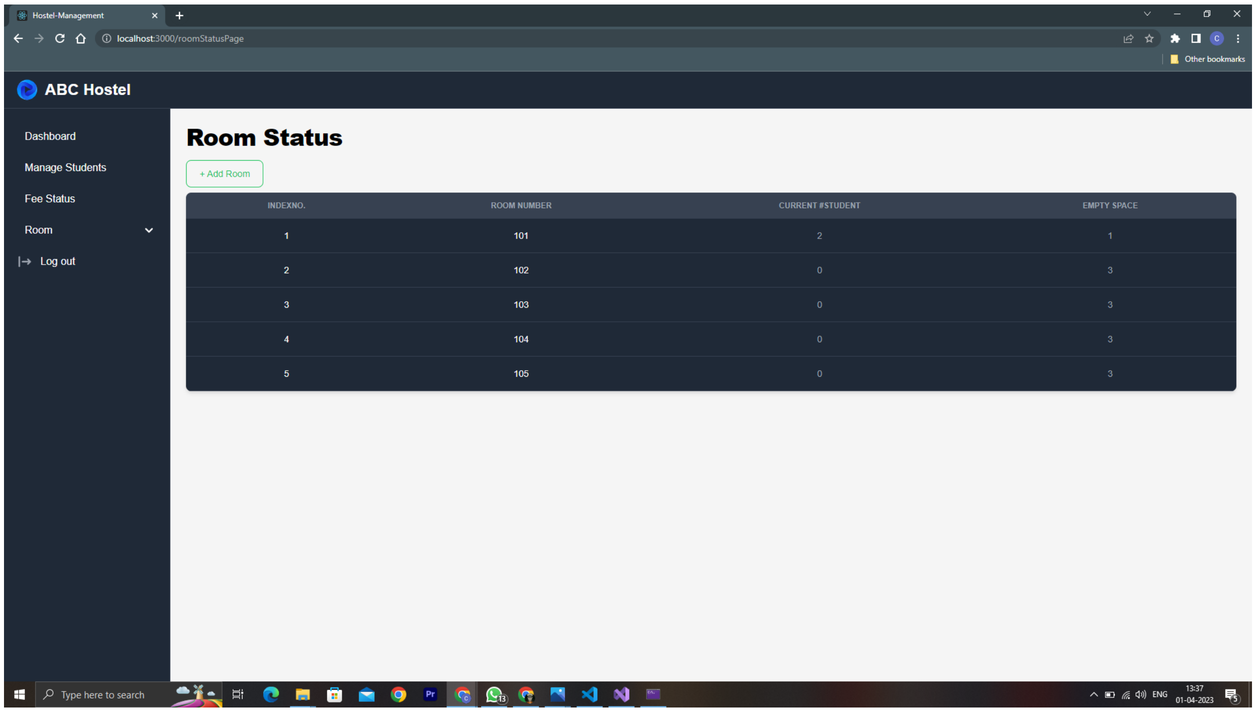
Edit

Student Id: 1

Student Name: Dhavnit

Fee Status: Select Options

Payment Method: Select Options



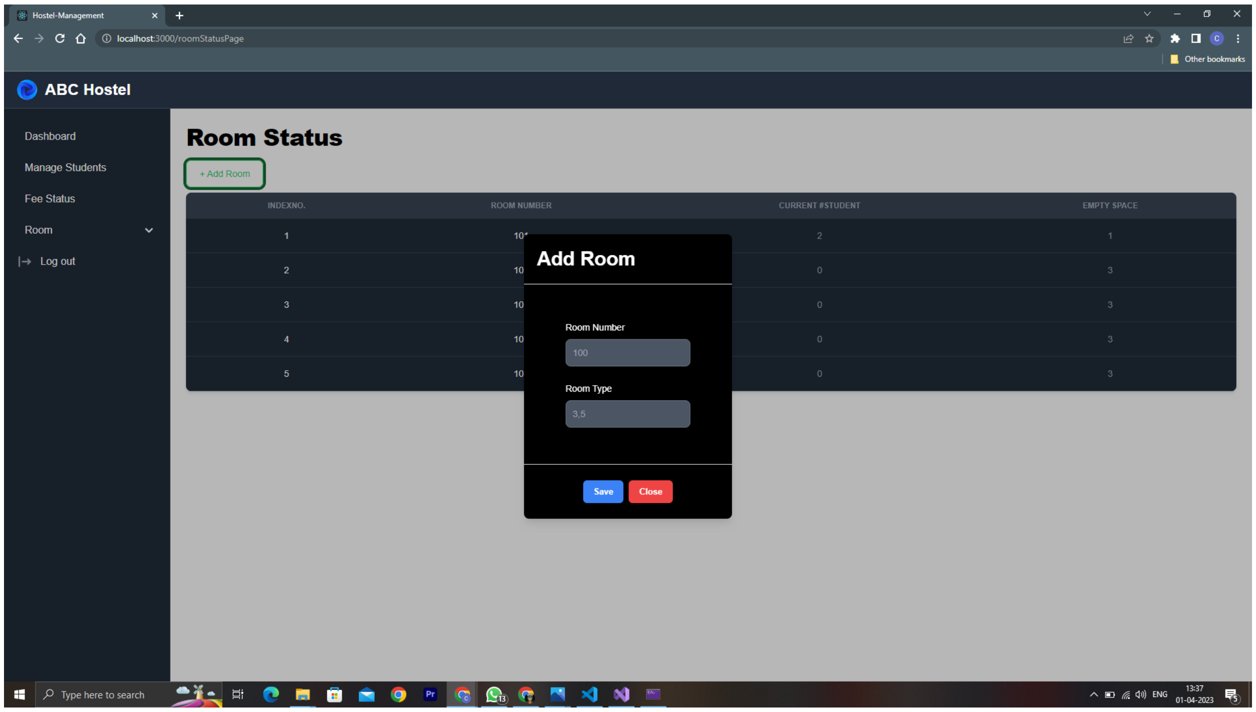
ABC Hostel

Dashboard
Manage Students
Fee Status
Room
Log out

Room Status

+ Add Room

INDEXNO.	ROOM NUMBER	CURRENT #STUDENT	EMPTY SPACE
1	101	2	1
2	102	0	3
3	103	0	3
4	104	0	3
5	105	0	3

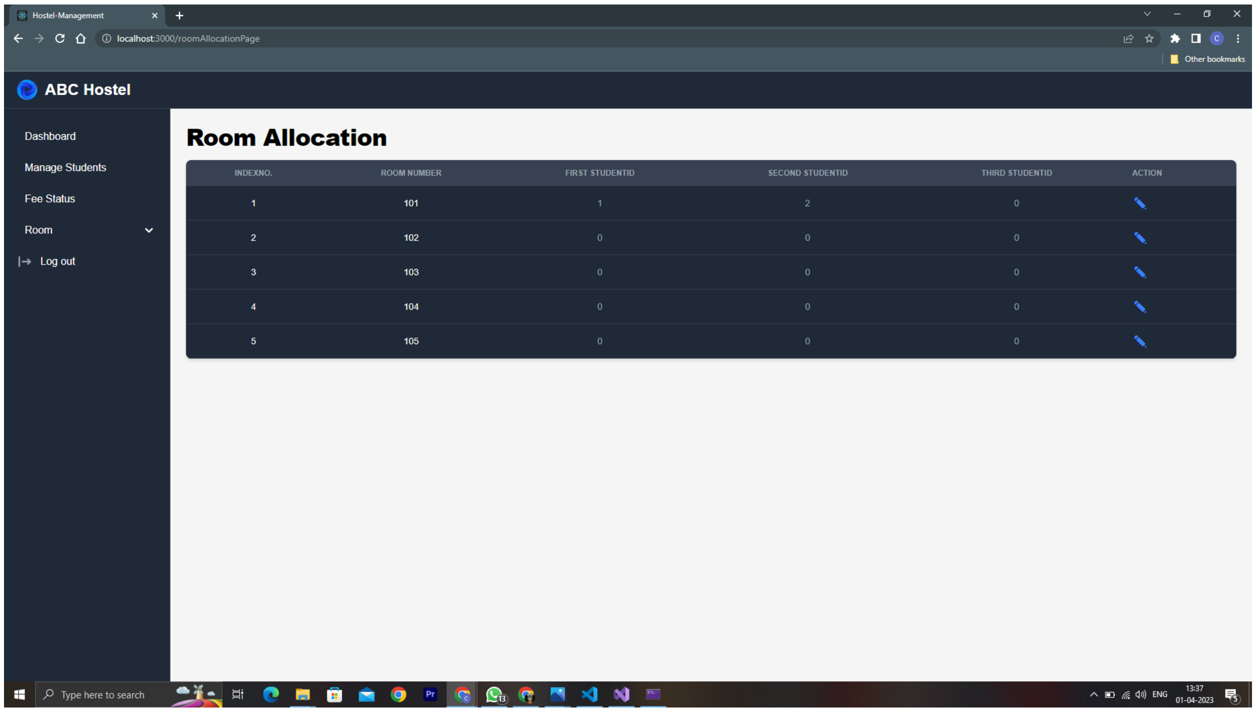


Add Room

Room Number: 100

Room Type: 3,5

Save Close



ABC Hostel

Dashboard

Manage Students

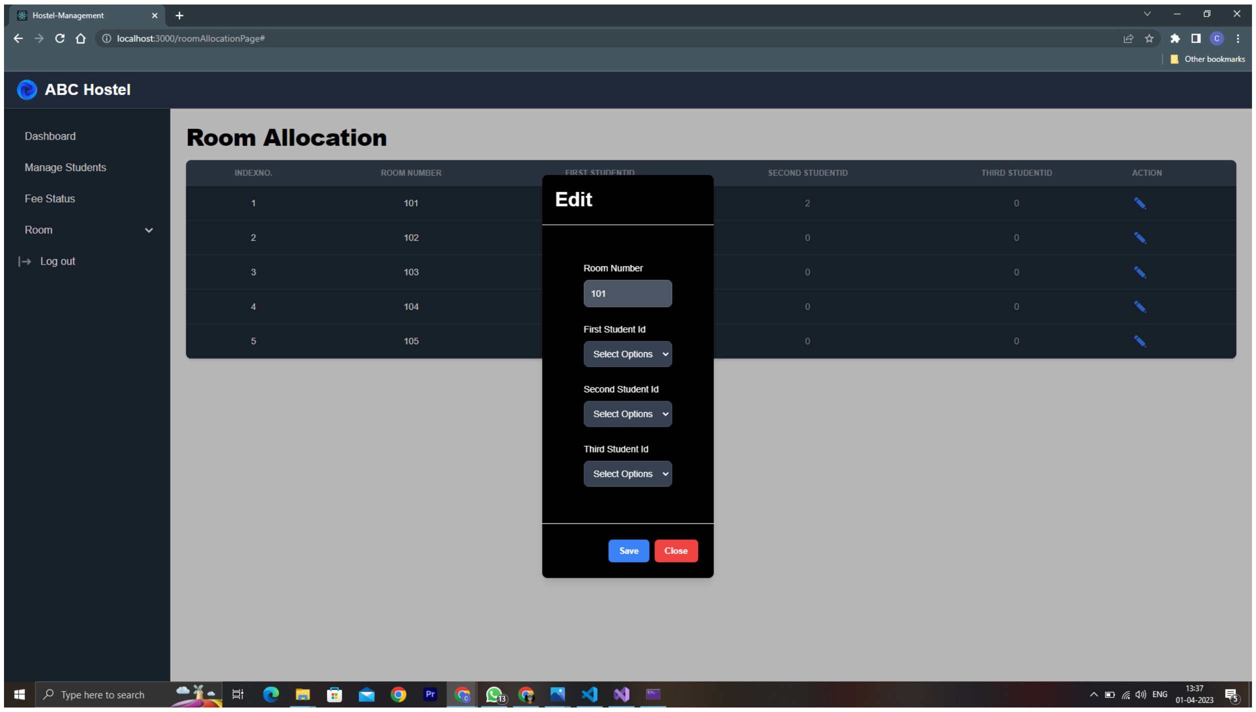
Fee Status

Room

Log out

Room Allocation

INDEXNO.	ROOM NUMBER	FIRST STUDENTID	SECOND STUDENTID	THIRD STUDENTID	ACTION
1	101	1	2	0	
2	102	0	0	0	
3	103	0	0	0	
4	104	0	0	0	
5	105	0	0	0	



ABC Hostel

Dashboard

Manage Students

Fee Status

Room

Log out

Room Allocation

Edit

Room Number: 101

First Student Id: Select Options

Second Student Id: Select Options

Third Student Id: Select Options

Save | Close

Limitation

1) The proposed system is not a client server system

Future Expansion

1) We can add clubs entity and add more user like accountant

Bibliography

-> Patel Pratham A.

-> Created->1-02-2023.....Last Updated->10-03-2023

-> Hostel management

-> Url : http://localhost:3000/

Reference

1) Stackoverflow
2) React js
3) Tutorial : creating web api with asp.net core from learn.microsoft.com