RV COLLEGE OF ENGINEERING ®
BENGALURU-560059
(Autonomous Institution Affiliated to VTU, Belagavi)

"Hostel and Mess Database Management System"

Report

Database Design Laboratory Project

(18CS53)

Submitted By

Nehal N Shet (1RV18IS026) KS Harshavardhan (1RV18IS018)

Under the Guidance of

Prof.Vanishree K,

Assistant Professor

in partial fulfillment for the award of degree of

Bachelor of Engineering

in

INFORMATION SCIENCE AND ENGINEERING

2020-21

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Mini Project work entitled 'Hostel and Mess Database Management System' has been carried out as a part of Database Design Laboratory(18CS53) in partial fulfillment for the award of degree of **Bachelor of Engineering** in **Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year **2020-2021** by **Nehal N Shet(1RV18IS026), KS Harshavardhan (1RV18IS018),** who are bonafide students of **RV College of Engineering®**, Bengaluru. It is certified that all the corrections/suggestions indicated for the internal assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of work prescribed by the institution for the said degree.

Prof.Vanishree K	**Dr Sagar B M**
Assistant Professor.	**Head of the Department**
Department of ISE,	Department of ISE,
RVCE, Bengaluru-59	RVCE, Bengaluru-59

Name of the Examiners	Signature with Date
1. ______________	____________________
2. ______________	____________________

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DECLARATION

We **Nehal N Shet, KS Harshavardhan,** are students of Fifth Semester B.E Department of Information Science and Engineering, **RV College of Engineering ®,** bearing **USN: 1RV18IS026, 1RV18IS018,** hereby declare that the project titled "***Hostel and Mess Database Management System***" has been carried out as a part of Database Design (18CS53) by us and submitted in partial fulfillment of the program requirements for the award of degree in Bachelor of Engineering in Information Science and Engineering of the **Visvesvaraya Technological University, Belagavi** during the year **2020-2021.**

Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other University.

Place: Bengaluru	**Name**	**Signature with Date**
Date:	**Nehal N Shet**	
	KS Harshavardhan	

TABLE OF CONTENTS

INTRODUCTION

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 three years before. We design this system on the request of the hostel management, through this they cannot require such an efficient person to handle and calculate the things. This system automatically calculates all the bills and issues the notifications for those students who are against some rules.

1.1 Terminology

A) MongoDB: Cross-platform Document-Oriented Database: MongoDB is a NoSQL database where each record is a document consisting of key-value pairs that are similar to JSON (JavaScript Object Notation) objects. MongoDB is flexible and allows its users to create schema, databases, tables, etc. Documents that are identifiable by a primary key make up the basic unit of MongoDB. Once MongoDB is installed, users can make use of the Mongo shell as well. Mongo shell provides a JavaScript interface through which the users can interact and carry out operations (eg: querying, updating records, deleting records).

B) Express: Back-End Framework: Express is a Node.js framework. Rather than writing the code using Node.js and creating loads of Node modules, Express makes it simpler and easier to write the back-end code. Express helps in designing great web applications and APIs. Express supports many middlewares which makes the code shorter and easier to write.

C) React: Front-End Library: React is a JavaScript library that is used for building user interfaces. React is used for the development of single-page applications and mobile applications because of its ability to handle rapidly changing data. React allows users to code in JavasScript and create UI components.

4) Node.js: JS Runtime Environment: Node.js provides a JavaScript Environment which allows the user to run their code on the server (outside the browser). Node pack manager i.e. npm allows the user to choose from thousands of free packages (node modules) to download.

HMDMS – Hostel and Mess Database Management System
DBMS - Database Management System
UML - Unified Modelling Language
SRS – Software Requirements Specifications

1.2 Purpose

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 also helps them store details related to mess, like the ration consumed, food wastage 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.

1.3 Motivation

Hostel is not less than a home for students when staying away from their home. It has large well ventilated dormitories and single rooms and is situated in the school premises. Providing clean and calm hostel accommodation is one of the key responsible of school management.To manage the hostel facilities, a lot of data need to be maintained such as number of student hostel can accommodate, hostel rules and regulation, hostel fee, hostel in and out of student, guest and visitor record and so on. So, this needs the system which has an ability to capture all kinds of data and information and analyze it properly for smooth functioning of the hostel. Hostel wardens can easily maintain the data.

1.4 Problem Statement

After making an observation on the current process of hostel management at school, it found that every single thing is done completely by manual. Currently, the decision is to make a computerized system . It is because they are facing problems such as corruption of data. The data about student hostels are stored and kept not very well and systematically. In the current process, the data is stored into the file but not in the database which leads to data duplication, repetitive data, and isolation of data from one to another. It is also worried if something happens to the file, then all the data will be lost.In the current manual system, it will be very difficult to find the hostel records and other information of students manually. Because it has been kept on the paper and it is easy to lose. It also consumes time to search the paper of the student hostel record one by one.The manual system requires longer time for allocation the student to respective hostel, dorm, and bed.

1.5 Objective

The objectives of this project are (i)To develop an integrated system for the hostel management system.(ii)Providing a facility for wardens to manage hostel and mess data.(iii)To compare the efficiency of system design for a small IT project.

1.6 Scope and Relevance

The proposed system for “Hostel and Mess Database Management System” is computerized. Today is the era of computers. This software project solves all the problems discussed above in the present system. The main objective of developing this project is to save time and efforts. The proposed system provides features on different tasks like, All the details related to hostelites can be found at one place like the admission details, fees details, room details , parent details ,mess details etc. All the details related to a mess employees and mess can be found here and managed by the wardens. The product will be useful to - Warden – To be able to alter and view student, employee, mess details. Mess Employee – To be able to view student details and view, alter mess details.

REQUIREMENT SPECIFICATION

2.1 Specific Requirements

2.1.1 Functional Requirements

- New Admin Registration – Wardens will be the admins for this system. New wardens can register themselves by providing necessary details.
- New Student Registration - Any student who is willing to join hostel must be registered anytime using the web application by the admin (chief warden).
- New Employee Registration - Any new employee who joins the mess must be registered into the portal.
- New Room Addition - All the rooms in the hostel must be added to the database by the admin.
- Admin / Mess worker Login - All the existent users must login to the website to avail the services.
- Mess details request – Admin will be able to view all the eatables and food items consumed , purchased and wasted in a particular interval of time. (weekly basis)
- Room Allotment – Admin will be able to allot rooms to students and maintain all the records for future use.
- Adding new Dish – Admin can change the menu according to food wastage so as to reduce the same.

2.1.2 Non-Functional Requirements

- Security - The security features have been designed to restrict unauthorized access to other accounts.
- Availability - 24 X 7 availability.
- Reliability - The web application is more reliable as every student and employee is verified by admins before adding them to the database. The student and employee data is used by the institution only and hence their data is safe. Resolving issues for records that are already added by the users ensures correction to any details of the records.
- Portability- The web application must be portable to different operating systems and environments so that it can be accessed by users from different platoforms.
- Quality- The system must be visually pleasing and all the functionality required should be provided in the application itself.
- Readability- The application should be easy to use and read as the systems stakeholders come from various backgrounds who might not have much knowledge about web development and applications.
- Maintainability- The application should require minimum maintenance as it will be used on a daily basis.

2.1.3 Hardware Requirements

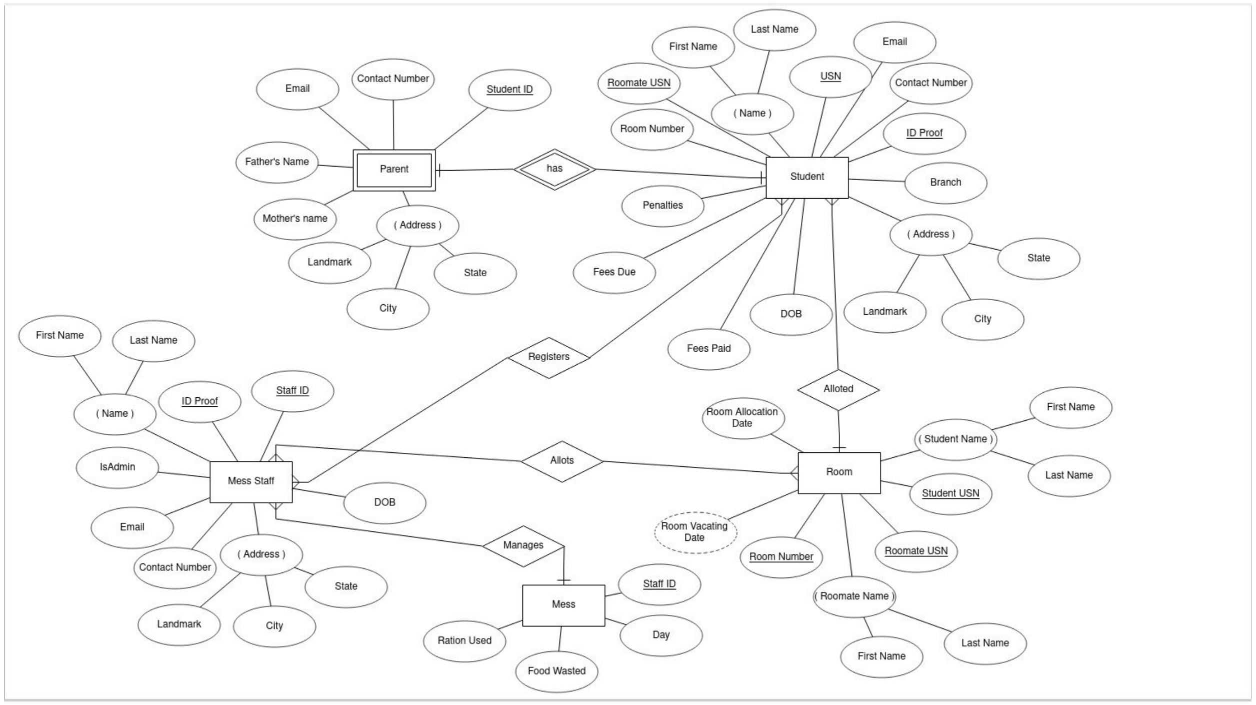
- Operating System: Ubuntu 16.04 LTS or higher, Windows 7 or higher.
- Processor: 1.8 Ghz or faster processor
- RAM: 0.6GB or more
- Internet: 200kbps or more

2.1.4 Software Requirements

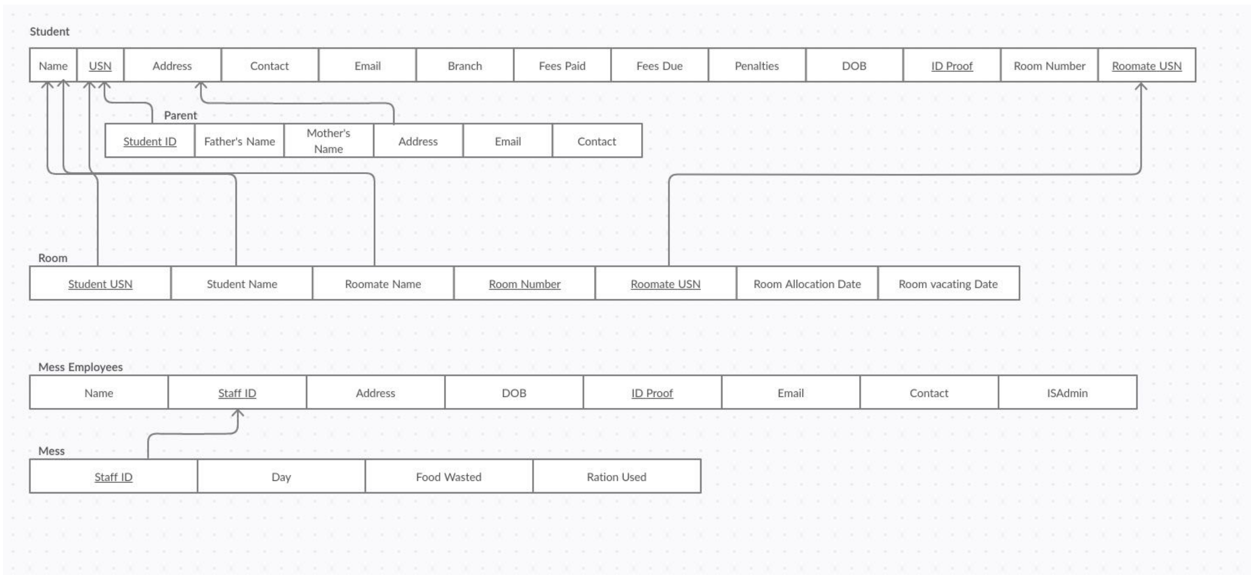
- Front End: ReactJS, Javascript, Bootstrap
- Back End: Node.js
- database: MongoDB
- Google Chrome / Firefox
- Ubuntu 18.04 Operating system

DESIGN

3.1 E-R Diagram

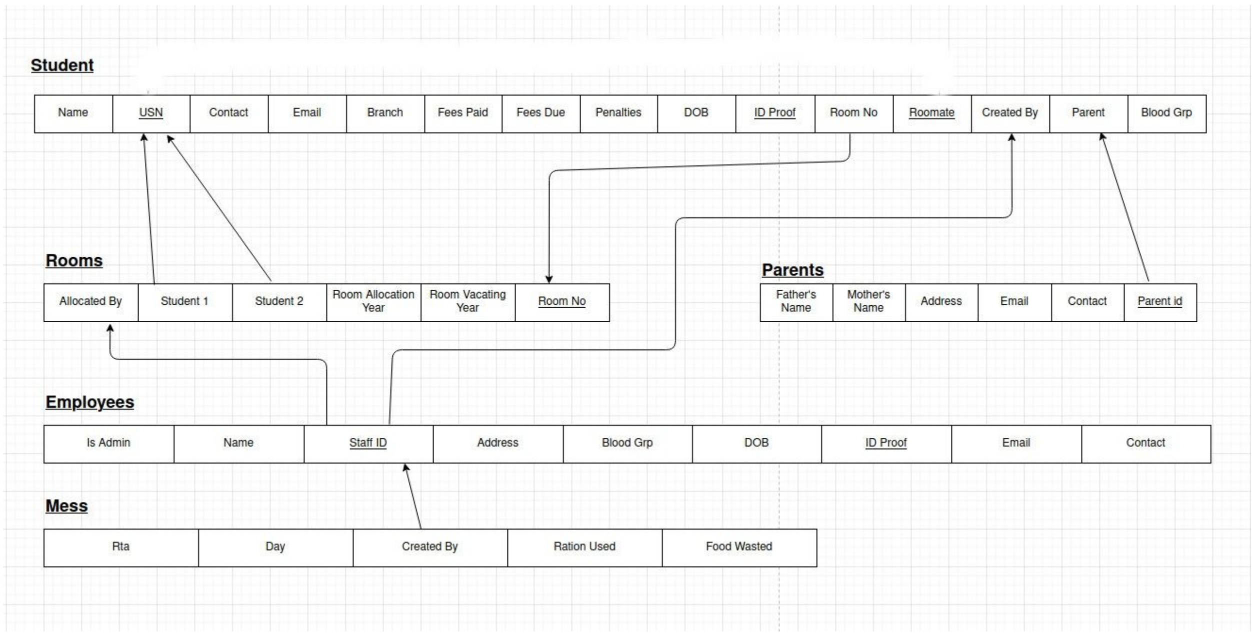


3.1.1 Schema Representation



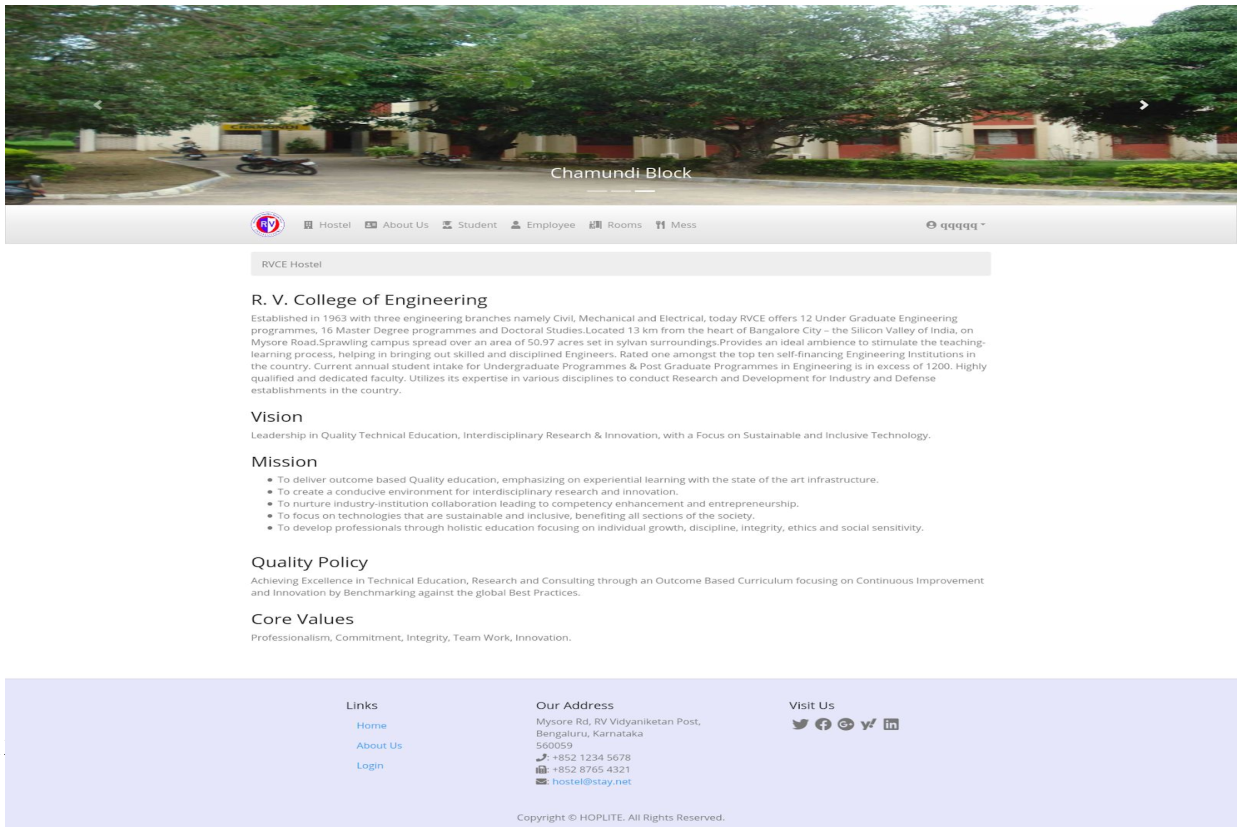
3.2 Normalization

3.2.1 Schema after Normalization



3.3 Front End Design

Front end is built using React. Screenshots of frontend is displayed below:



RVCE Hostel

R. V. College of Engineering

Established in 1963 with three engineering branches namely Civil, Mechanical and Electrical, today RVCE offers 12 Under Graduate Engineering programmes, 16 Master Degree programmes and Doctoral Studies.Located 13 km from the heart of Bangalore City – the Silicon Valley of India, on Mysore Road.Sprawling campus spread over an area of 50.97 acres set in sylvan surroundings.Provides an ideal ambience to stimulate the teaching-learning process, helping in bringing out skilled and disciplined Engineers. Rated one amongst the top ten self-financing Engineering Institutions in the country. Current annual student intake for Undergraduate Programmes & Post Graduate Programmes in Engineering is in excess of 1200. Highly qualified and dedicated faculty. Utilizes its expertise in various disciplines to conduct Research and Development for Industry and Defense establishments in the country.

Vision

Leadership in Quality Technical Education, Interdisciplinary Research & Innovation, with a Focus on Sustainable and Inclusive Technology.

Mission

- To deliver outcome based Quality education, emphasizing on experiential learning with the state of the art infrastructure.
- To create a conducive environment for interdisciplinary research and innovation.
- To nurture industry-institution collaboration leading to competency enhancement and entrepreneurship.
- To focus on technologies that are sustainable and inclusive, benefiting all sections of the society.
- To develop professionals through holistic education focusing on individual growth, discipline, integrity, ethics and social sensitivity.

Quality Policy

Achieving Excellence in Technical Education, Research and Consulting through an Outcome Based Curriculum focusing on Continuous Improvement and Innovation by Benchmarking against the global Best Practices.

Core Values

Professionalism, Commitment, Integrity, Team Work, Innovation.

Links
Home
About Us
Login

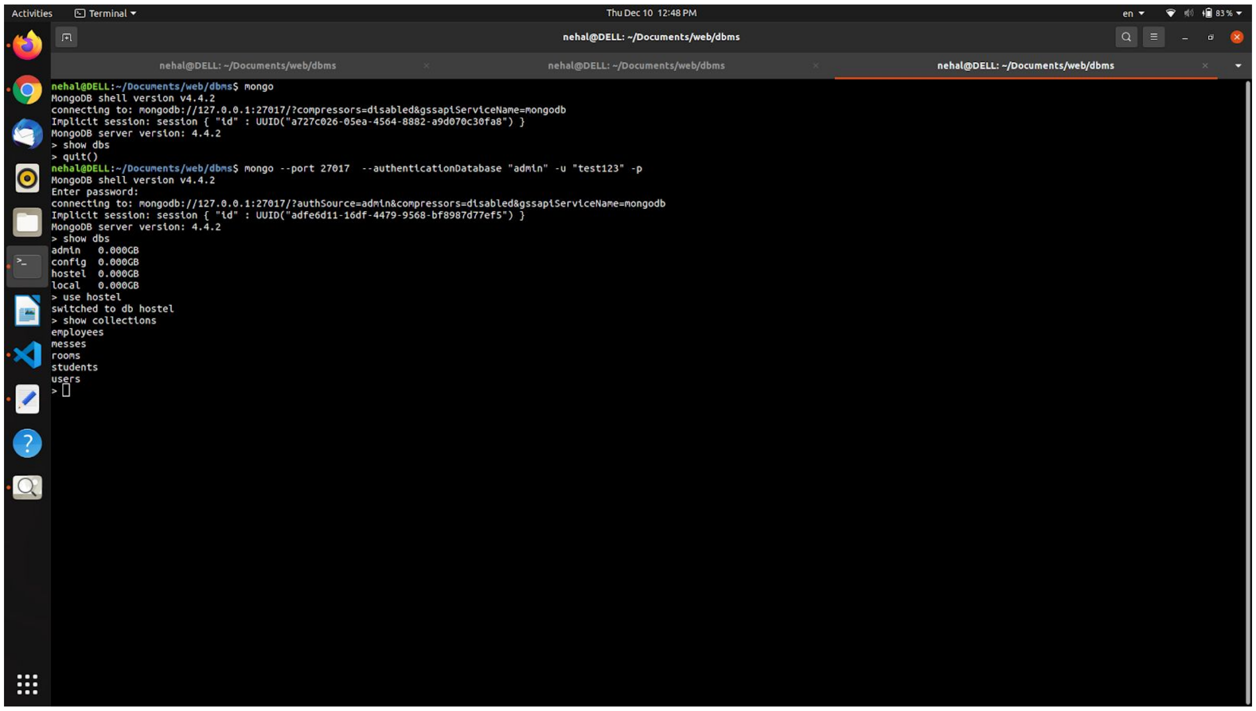
Our Address
Mysore Rd, RV Vidyaniketan Post,
Bengaluru, Karnataka
560059
: +852 1234 5678
: +852 8765 4321
: hostel@stay.net

Visit Us

Copyright © HOPLITE. All Rights Reserved.

IMPLEMENTATION DETAILS

4.1.1 Table Creation



```
nehal@DELL:~/Documents/web/dbms$ mongo
MongoDB shell version v4.4.2
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a727c026-05ea-4564-8882-a9d070c30fa8") }
MongoDB server version: 4.4.2
> show dbs
> quit()
nehal@DELL:~/Documents/web/dbms$ mongo --port 27017  --authenticationDatabase "admin" -u "test123" -p
MongoDB shell version v4.4.2
Enter password:
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("adfe6d11-16df-4479-9568-bf8987d77ef5") }
MongoDB server version: 4.4.2
> show dbs
admin   0.000GB
config  0.000GB
hostel  0.000GB
local   0.000GB
> use hostel
switched to db hostel
> show collections
employees
messes
rooms
students
users
> 
```

4.1.2 Table Population

<u>Student Collection</u>

```
const parentSchema = mongoose.Schema({
fname: {type: String,required: true,},
mname: {type: String,required: true,},
address: {type: String,required: true,},
contact: {type: String,required: true,},
email: {type: String,required: true,},
});
const studentSchema = mongoose.Schema(
{
user: {type: mongoose.Schema.Types.ObjectId,ref: "User",required: true,},
name: {type: String,required: true,},
usn: {type: String,required: true,unique: true,},
image: {type: String,required: true,},
branch: {type: String,required: true,},
year: {type: Number,required: true,min: 1,max: 4,default: 1,},
roomno: {type: String,},
roomatename: {type: String,},
roomateusn: {type: String,},
dob: {type: String,required: true,},
idproof: {type: String,required: true,unique: true,},
contact: {type: String,required: true,},
email: {type: String,required: true,},
address: {type: String,required: true,},
feespaid: {type: Number,required: true,default: 25000.0,},
feesdue: {type: Number,required: true,default: 0.0,},
penalties: {type: Number,required: true,default: 0.0,},
firstyear: {type: Number,required: true,},
finalyear: {type: Number,required: true,},
bloodgrp: {type: String,required: true,},
parents: parentSchema,
ispassedout: {type: Boolean,required: true,default: false,},
}, {timestamps: true,}
);
```

Employee Collection

```
const employeeSchema = mongoose.Schema(
{
user: {type: mongoose.Schema.Types.ObjectId,ref: "User",},
name: {type: String,required: true,},
staffid: {type: String,required: true,unique: true,},
image: {type: String,required: true,},
dob: {type: String,required: true,},
idproof: {type: String,required: true,unique: true,},
contact: {type: String,required: true,},
email: {type: String,required: true,},
address: {type: String,required: true,},
bloodgrp: {type: String,required: true,},
role: {type: String,required: true,},
isadmin: {type: Boolean,required: true,default: false,},
},{timestamps: true,}
);
```

Mess Collection

```
const messSchema = mongoose.Schema({
user: {type: mongoose.Schema.Types.ObjectId,ref: 'User',required: true},
date: {type: String,required: true},
day: {type: String,required: true},
rationused: {type: Number,required: true,default: 0.0},
foodwasted: {type: Number,required: true,default: 0.0}
},{timestamps: true}
);
```

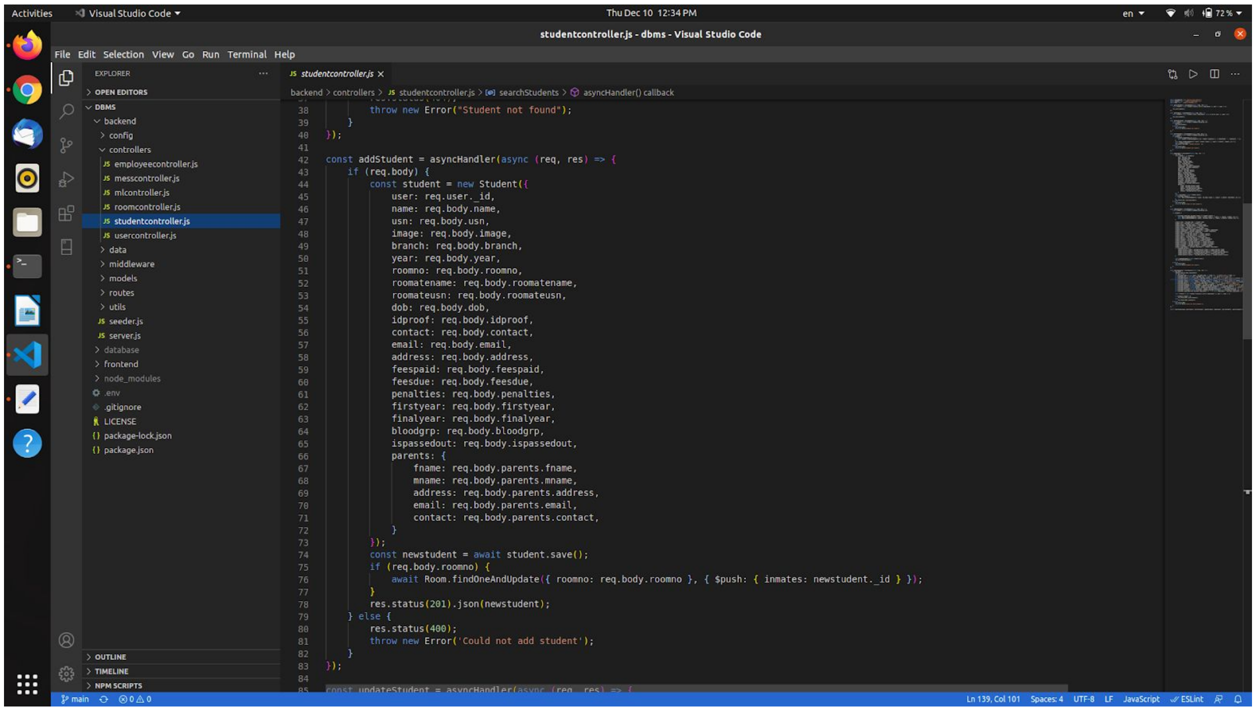
Room Collection

```
const roomSchema = mongoose.Schema(
{
user: {type: mongoose.Schema.Types.ObjectId,ref: "User",required: true},
roomno: {type: String,required: true,},
inmates: [{type: mongoose.Schema.Types.ObjectId,ref: "Student"}],
roomallocationyear: {type: Number,required: true,},
roomvacatingyear: {type: Number,required: true,},
```

},{timestamps: true,}
);

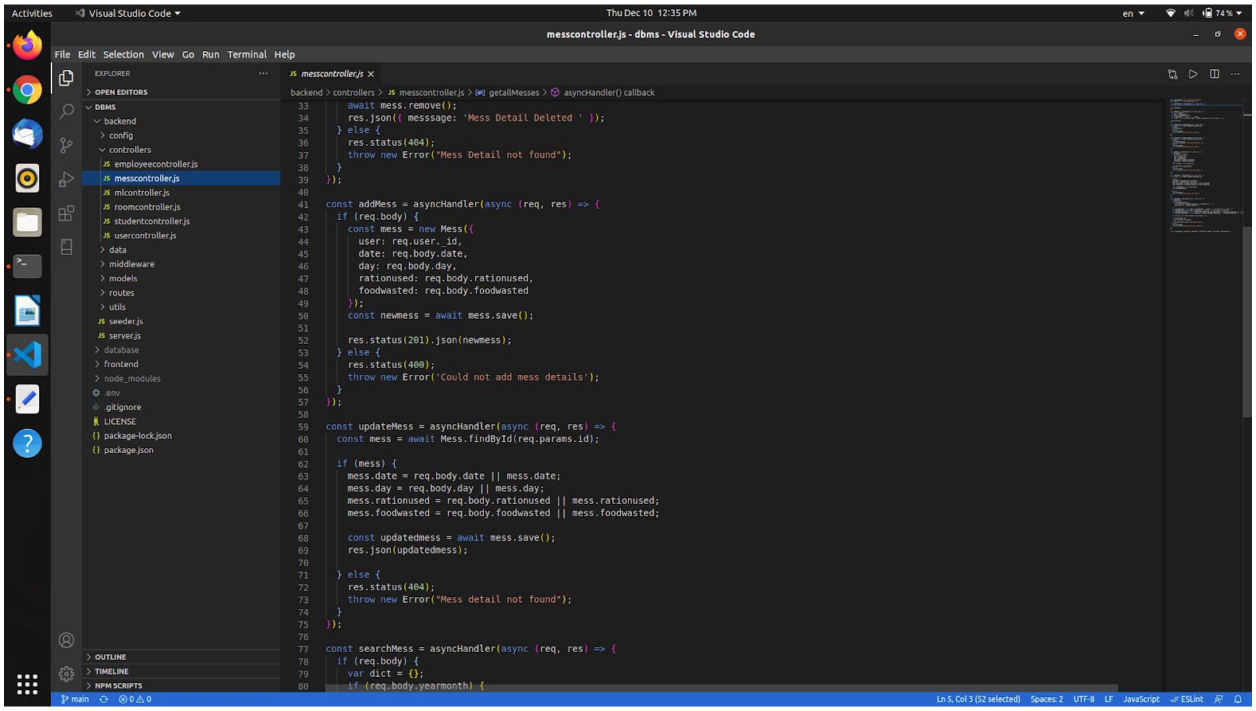
User Collection
const userSchema = mongoose.Schema({
name: {type: String,required: true},
id: {type: String,required: true,unique: true},
password: {type: String,required: true},
isadmin: {type: Boolean,required: true,default: false},
employeeid: {type: mongoose.Schema.Types.ObjectId,ref: "Employee",required: true},
},{timestamps: true}
);

4.1.3 Query Execution and Output



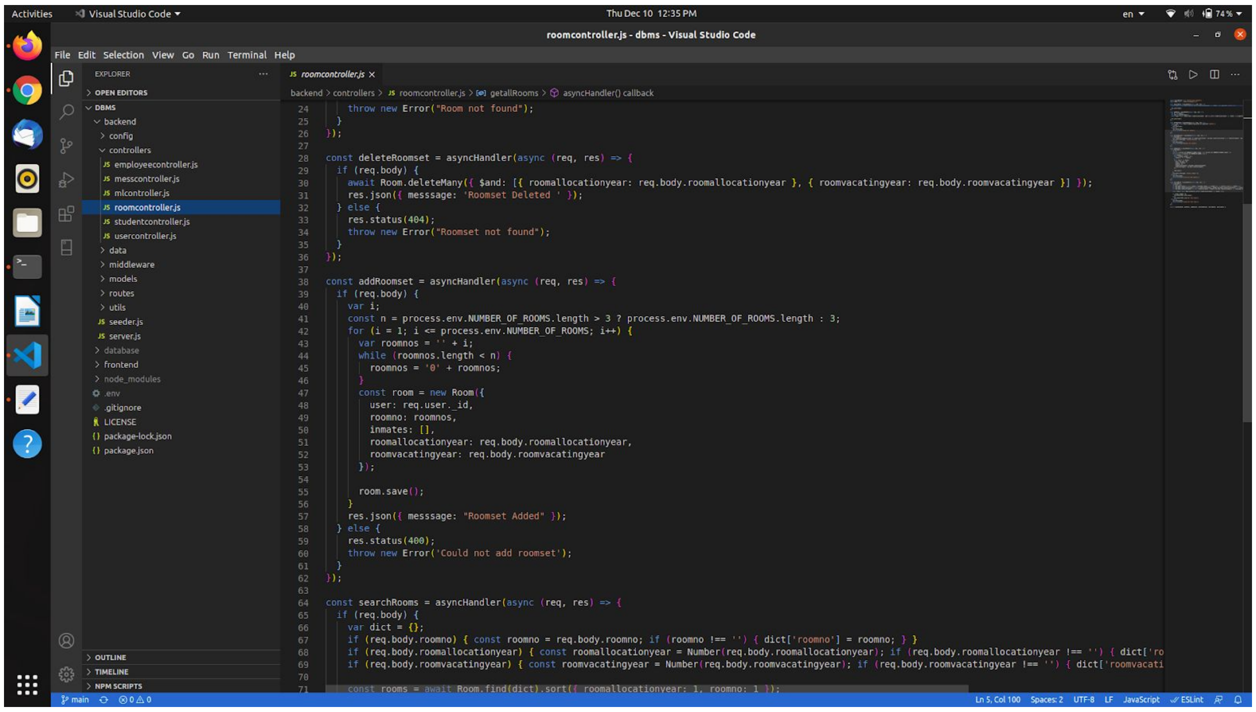
```
        throw new Error("Student not found");
    }
});

const addStudent = asyncHandler(async (req, res) => {
    if (req.body) {
        const student = new Student({
            user: req.user._id,
            name: req.body.name,
            usn: req.body.usn,
            image: req.body.image,
            branch: req.body.branch,
            year: req.body.year,
            roomno: req.body.roomno,
            roomatename: req.body.roomatename,
            roomateusn: req.body.roomateusn,
            dob: req.body.dob,
            idproof: req.body.idproof,
            contact: req.body.contact,
            email: req.body.email,
            address: req.body.address,
            feespaid: req.body.feespaid,
            feesdue: req.body.feesdue,
            penalties: req.body.penalties,
            firstyear: req.body.firstyear,
            finalyear: req.body.finalyear,
            bloodgrp: req.body.bloodgrp,
            ispassedout: req.body.ispassedout,
            parents: {
                fname: req.body.parents.fname,
                mname: req.body.parents.mname,
                address: req.body.parents.address,
                email: req.body.parents.email,
                contact: req.body.parents.contact,
            }
        });
        const newstudent = await student.save();
        if (req.body.roomno) {
            await Room.findOneAndUpdate({ roomno: req.body.roomno }, { $push: { inmates: newstudent._id } });
        }
        res.status(201).json(newstudent);
    } else {
        res.status(400);
        throw new Error('Could not add student');
    }
});

const updateStudent = asyncHandler(async (req, res) => {
```

Student Query



```
    await mess.remove();
    res.json({ messsage: 'Mess Detail Deleted ' });
  } else {
    res.status(404);
    throw new Error("Mess Detail not found");
  }
});

const addMess = asyncHandler(async (req, res) => {
  if (req.body) {
    const mess = new Mess({
      user: req.user._id,
      date: req.body.date,
      day: req.body.day,
      rationused: req.body.rationused,
      foodwasted: req.body.foodwasted
    });
    const newmess = await mess.save();

    res.status(201).json(newmess);
  } else {
    res.status(400);
    throw new Error('Could not add mess details');
  }
});

const updateMess = asyncHandler(async (req, res) => {
  const mess = await Mess.findById(req.params.id);

  if (mess) {
    mess.date = req.body.date || mess.date;
    mess.day = req.body.day || mess.day;
    mess.rationused = req.body.rationused || mess.rationused;
    mess.foodwasted = req.body.foodwasted || mess.foodwasted;

    const updatedmess = await mess.save();
    res.json(updatedmess);

  } else {
    res.status(404);
    throw new Error("Mess detail not found");
  }
});

const searchMess = asyncHandler(async (req, res) => {
  if (req.body) {
    var dict = {};
    if (req.body.yearmonth) {
```

Employee Query



```
    throw new Error("Room not found");
  }
});

const deleteRoomset = asyncHandler(async (req, res) => {
  if (req.body) {
    await Room.deleteMany({ $and: [{ roomallocationyear: req.body.roomallocationyear }, { roomvacatingyear: req.body.roomvacatingyear }] });
    res.json({ messsage: 'Roomset Deleted ' });
  } else {
    res.status(404);
    throw new Error("Roomset not found");
  }
});

const addRoomset = asyncHandler(async (req, res) => {
  if (req.body) {
    var i;
    const n = process.env.NUMBER_OF_ROOMS.length > 3 ? process.env.NUMBER_OF_ROOMS.length : 3;
    for (i = 1; i <= process.env.NUMBER_OF_ROOMS; i++) {
      var roomnos = '' + i;
      while (roomnos.length < n) {
        roomnos = '0' + roomnos;
      }
      const room = new Room({
        user: req.user._id,
        roomno: roomnos,
        inmates: [],
        roomallocationyear: req.body.roomallocationyear,
        roomvacatingyear: req.body.roomvacatingyear
      });

      room.save();
    }
    res.json({ messsage: "Roomset Added" });
  } else {
    res.status(400);
    throw new Error('Could not add roomset');
  }
});

const searchRooms = asyncHandler(async (req, res) => {
  if (req.body) {
    var dict = {};
    if (req.body.roomno) { const roomno = req.body.roomno; if (roomno !== '') { dict['roomno'] = roomno; } }
    if (req.body.roomallocationyear) { const roomallocationyear = Number(req.body.roomallocationyear); if (req.body.roomallocationyear !== '') { dict['ro
    if (req.body.roomvacatingyear) { const roomvacatingyear = Number(req.body.roomvacatingyear); if (req.body.roomvacatingyear !== '') { dict['roomvacati

    const rooms = await Room.find(dict).sort({ roomallocationyear: 1, roomno: 1 });
```

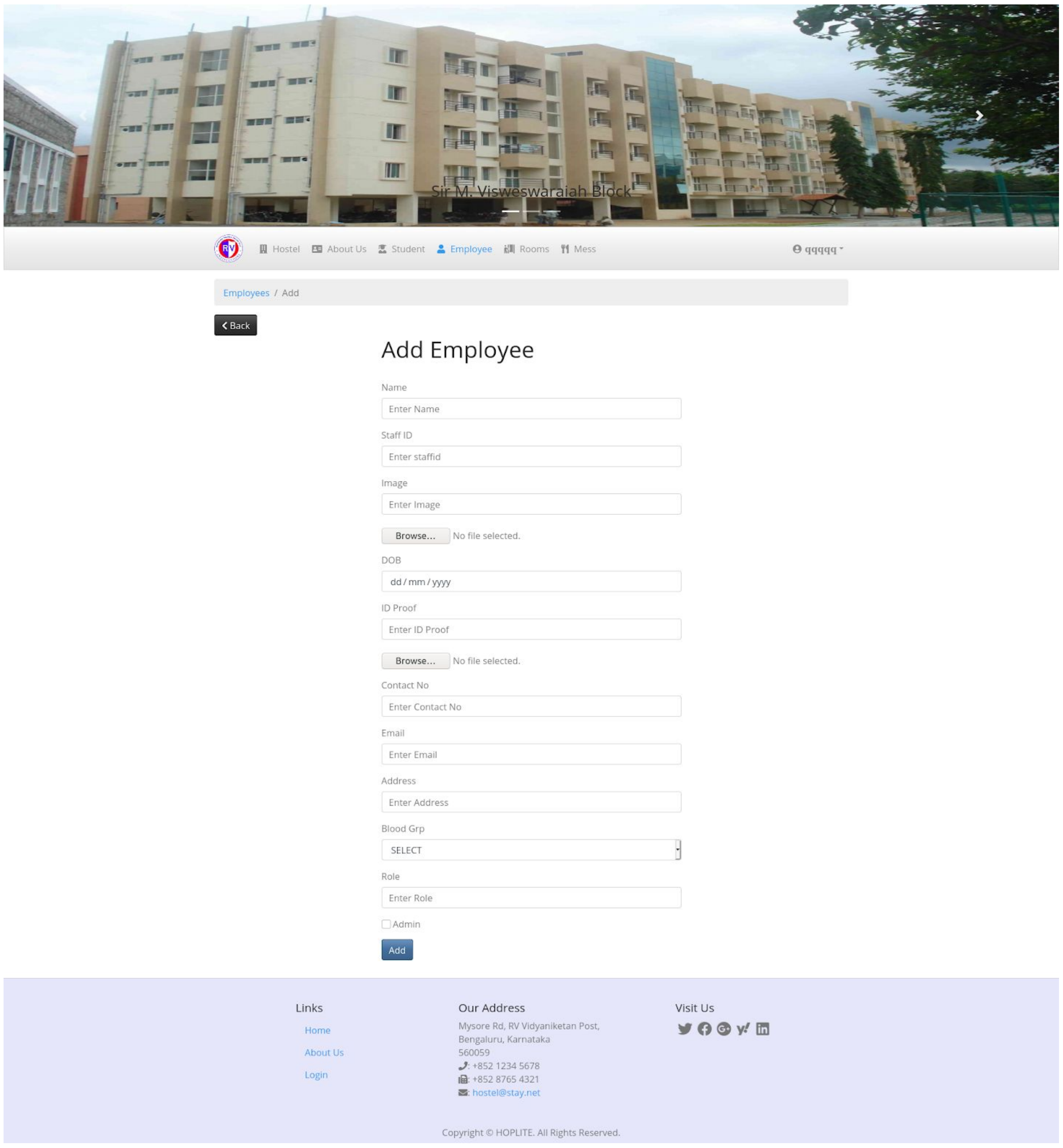
Room Querry

4.1.4 Security features

MongoDB is used, so the data will be directly sent to the database securely and any person without the database credentials cannot access the data in the database. The attributes like passwords are encrypted and stored in our database and on the frontend, passwords are not visible.

4.2 Front End implementation

4.2.1 Form Creation



Add Employee

Name	Enter Name
Staff ID	Enter staffid
Image	Enter Image
DOB	dd / mm / yyyy
ID Proof	Enter ID Proof
Contact No	Enter Contact No
Email	Enter Email
Address	Enter Address
Blood Grp	SELECT
Role	Enter Role

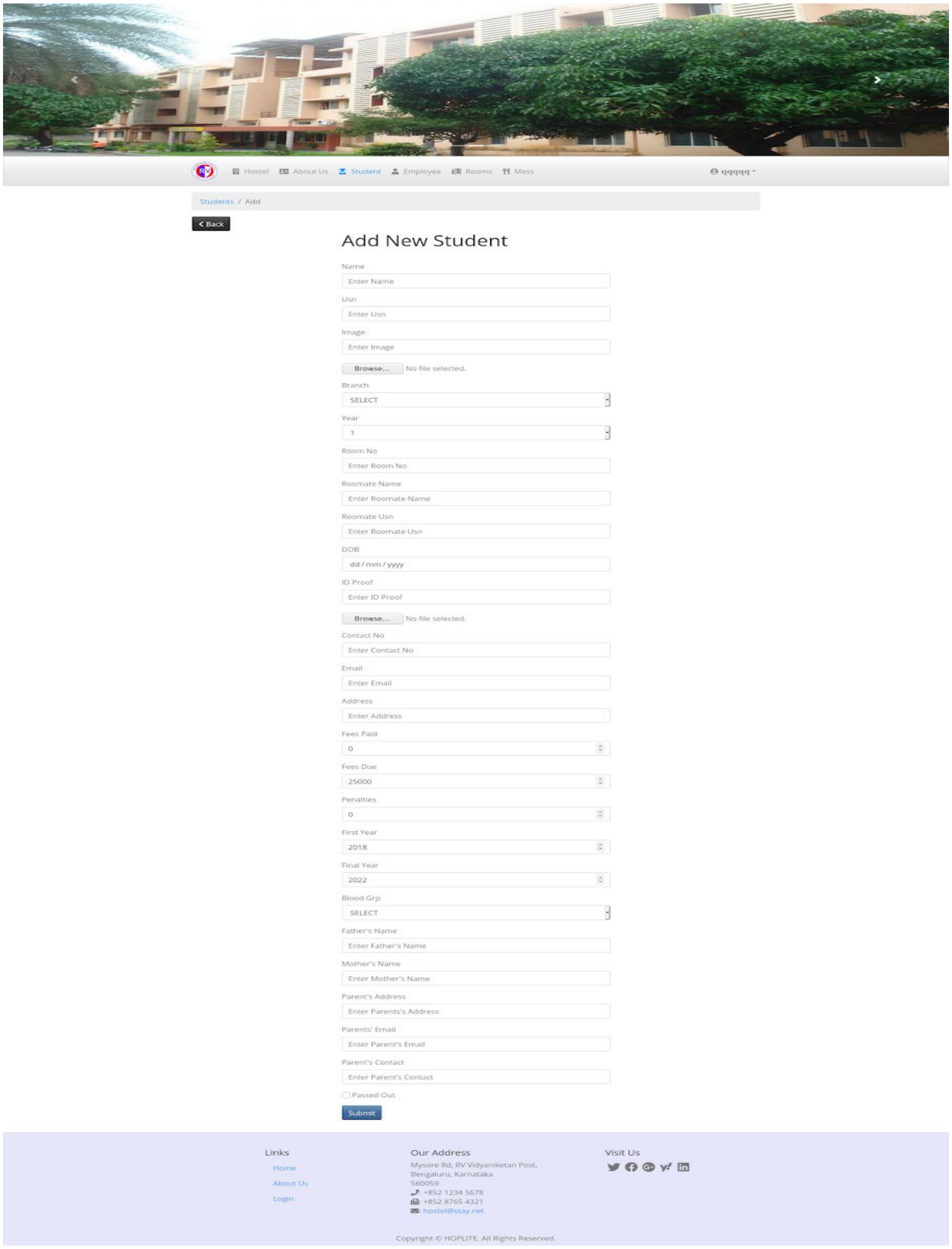
Links
Home
About Us
Login

Our Address
Mysore Rd, RV Vidyaniketan Post,
Bengaluru, Karnataka
560059
: +852 1234 5678
: +852 8765 4321
: hostel@stay.net

Visit Us

Copyright © HOPLITE. All Rights Reserved.

Student Form



Add New Student

Name	Enter Name
Usn	Enter Usn
Image	Enter Image
	Browse... No file selected.
Branch	SELECT
Year	1
Room No	Enter Room No
Roomate Name	Enter Roomate Name
Roomate Usn	Enter Roomate Usn
DOB	dd / mm / yyyy
ID Proof	Enter ID Proof
	Browse... No file selected.
Contact No	Enter Contact No
Email	Enter Email
Address	Enter Address
Fees Paid	0
Fees Due	25000
Penalties	0
First Year	2018
Final Year	2022
Blood Grp	SELECT
Father's Name	Enter Father's Name
Mother's Name	Enter Mother's Name
Parent's Address	Enter Parents's Address
Parents' Email	Enter Parent's Email
Parent's Contact	Enter Parent's Contact

☐ Passed Out

Submit

Links
Home
About Us
Login

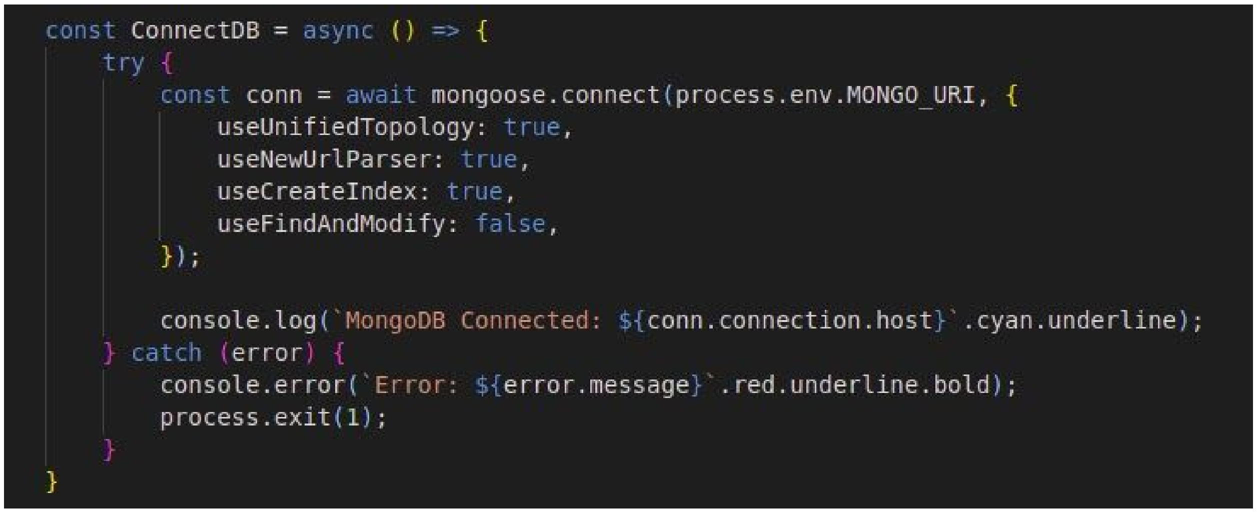
Our Address
Mysore Rd, RV Vidyaniketan Post,
Bengaluru, Karnataka
560059
: +852 1234 5678
: +852 8765 4321
: hostel@stay.net

Visit Us

Copyright © HOPLITE. All Rights Reserved.

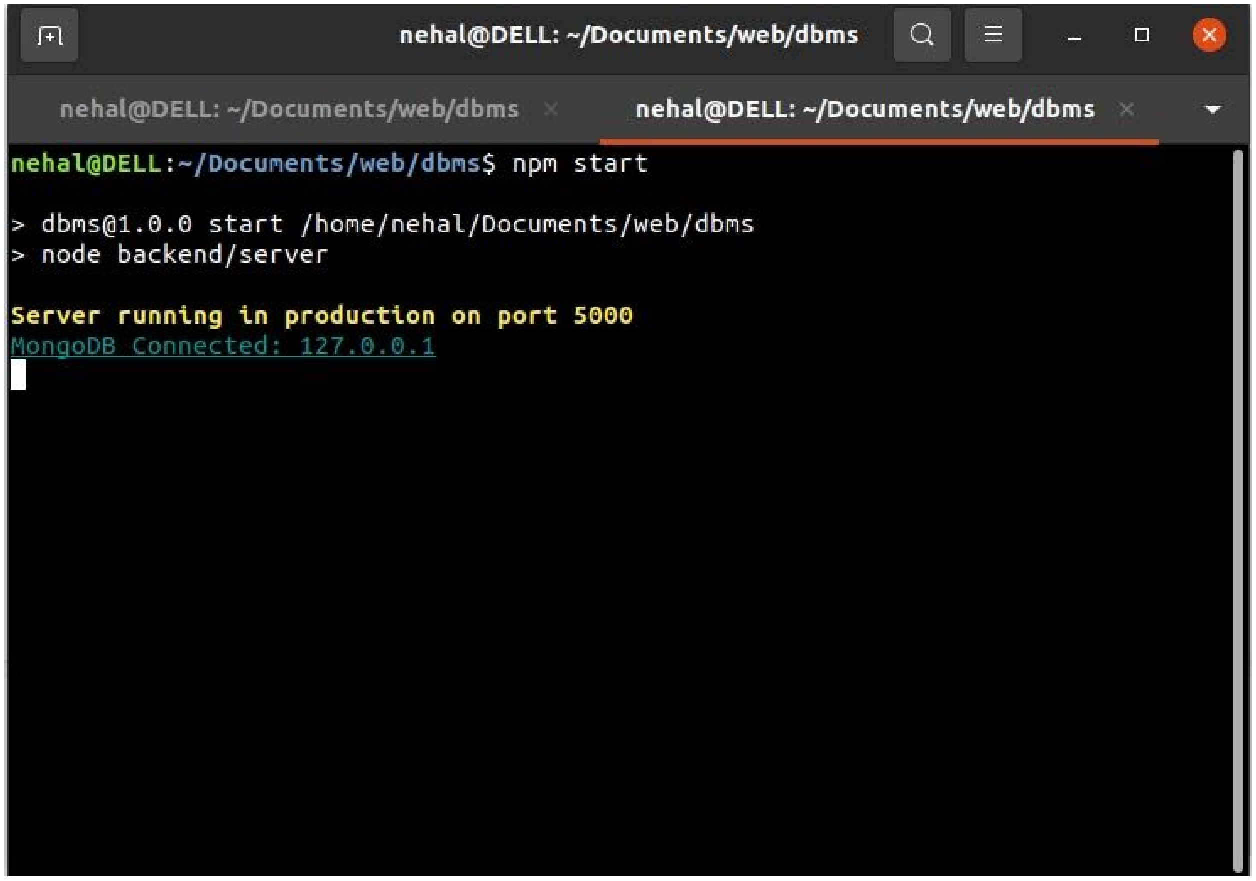
Employee form

4.2.2 Connectivity to the Database



```
const ConnectDB = async () => {
    try {
        const conn = await mongoose.connect(process.env.MONGO_URI, {
            useUnifiedTopology: true,
            useNewUrlParser: true,
            useCreateIndex: true,
            useFindAndModify: false,
        });

        console.log(`MongoDB Connected: ${conn.connection.host}`.cyan.underline);
    } catch (error) {
        console.error(`Error: ${error.message}`.red.underline.bold);
        process.exit(1);
    }
}
```

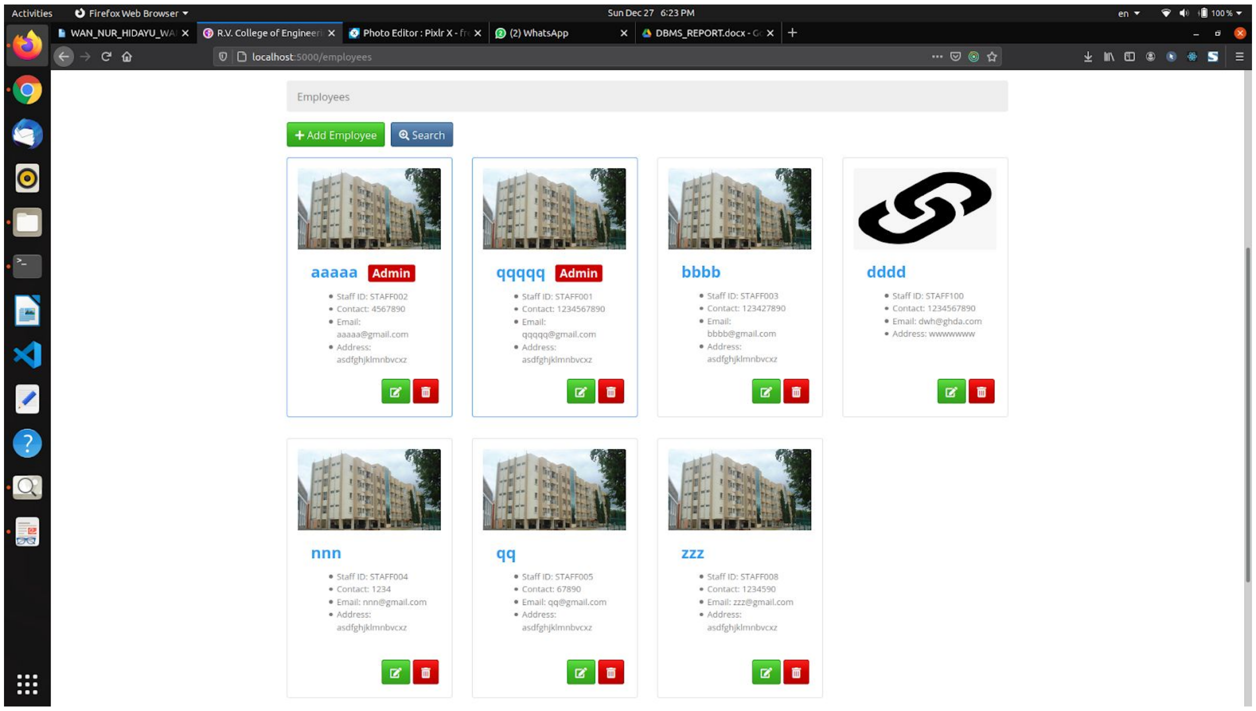
Code for connectivity


nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms

```
nehal@DELL:~/Documents/web/dbms$ npm start

> dbms@1.0.0 start /home/nehal/Documents/web/dbms
> node backend/server

Server running in production on port 5000
MongoDB Connected: 127.0.0.1
```

Terminal

4.2.3 Report generation



aaaaa Admin
- Staff ID: STAFF002
- Contact: 4567890
- Email: aaaaa@gmail.com
- Address: asdfghjklmnbvcxz

qqqqq Admin
- Staff ID: STAFF001
- Contact: 1234567890
- Email: qqqqq@gmail.com
- Address: asdfghjklmnbvcxz

bbbb
- Staff ID: STAFF003
- Contact: 123427890
- Email: bbbb@gmail.com
- Address: asdfghjklmnbvcxz

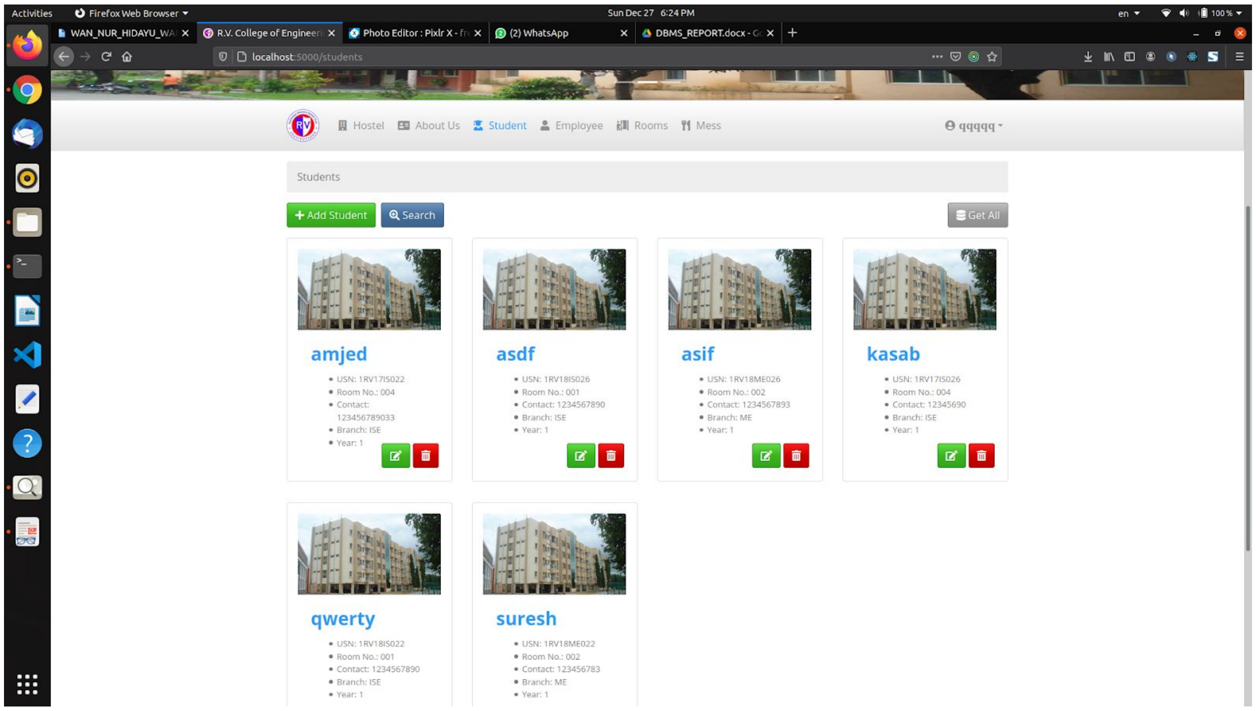
dddd
- Staff ID: STAFF100
- Contact: 1234567890
- Email: dwh@ghda.com
- Address: wwwwwww

nnn
- Staff ID: STAFF004
- Contact: 1234
- Email: nnn@gmail.com
- Address: asdfghjklmnbvcxz

qq
- Staff ID: STAFF005
- Contact: 67890
- Email: qq@gmail.com
- Address: asdfghjklmnbvcxz

zzz
- Staff ID: STAFF008
- Contact: 1234590
- Email: zzz@gmail.com
- Address: asdfghjklmnbvcxz

Employee Reports



amjed
- USN: 1RV17IS022
- Room No.: 004
- Contact: 123456789033
- Branch: ISE
- Year: 1

asdf
- USN: 1RV18IS026
- Room No.: 001
- Contact: 1234567890
- Branch: ISE
- Year: 1

asif
- USN: 1RV18ME026
- Room No.: 002
- Contact: 1234567893
- Branch: ME
- Year: 1

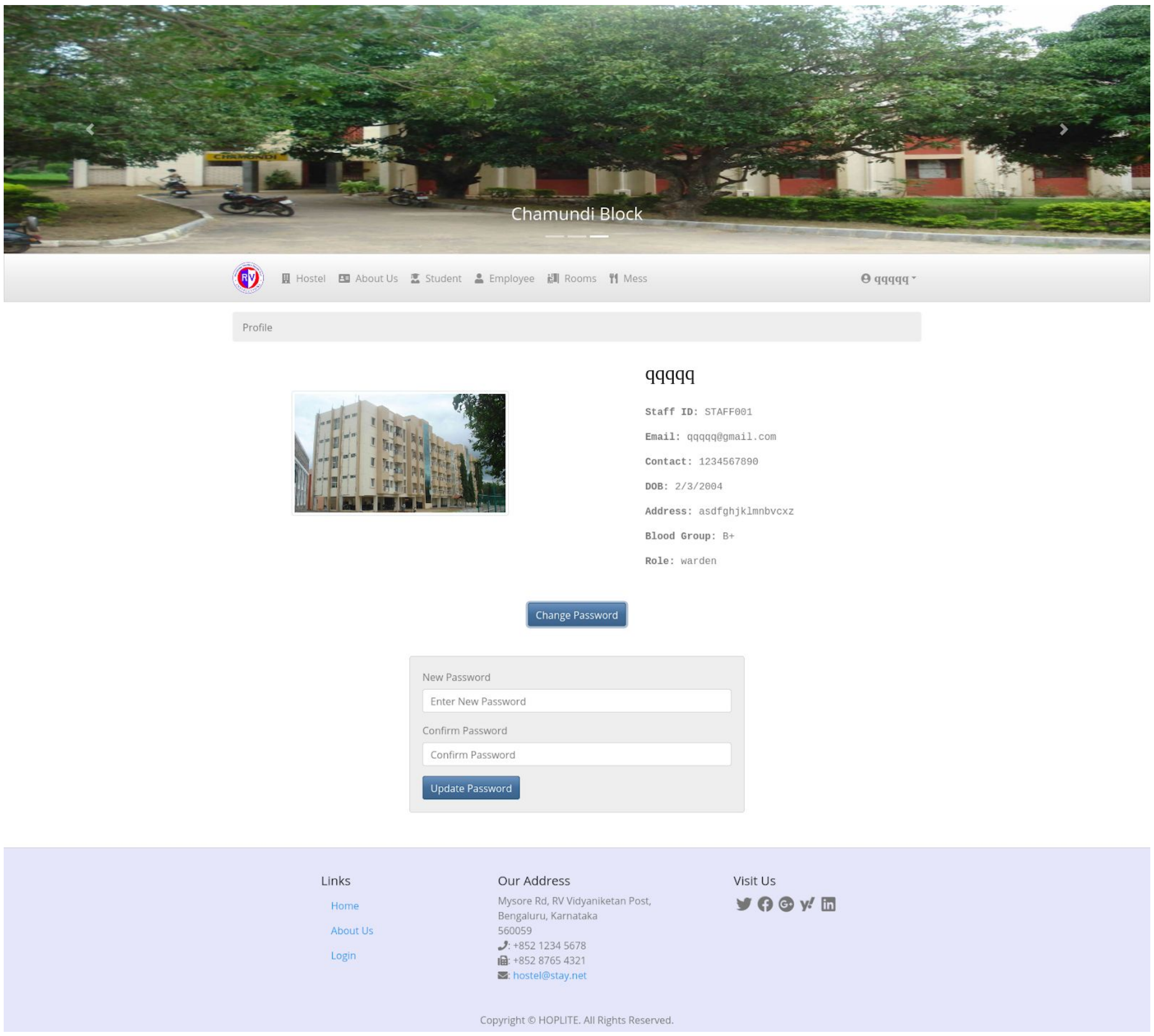
kasab
- USN: 1RV17IS026
- Room No.: 004
- Contact: 12345690
- Branch: ISE
- Year: 1

qwerty
- USN: 1RV18IS022
- Room No.: 001
- Contact: 1234567890
- Branch: ISE
- Year: 1

suresh
- USN: 1RV18ME022
- Room No.: 002
- Contact: 123456783
- Branch: ME
- Year: 1

Student Reports

4.2.4 Security features



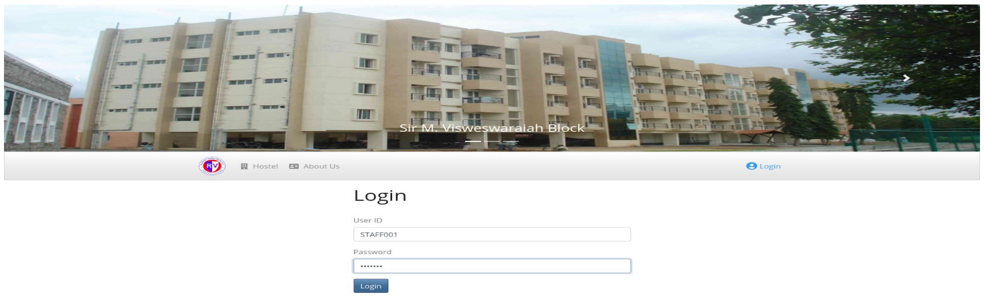
qqqqq

Staff ID: STAFF001
Email: qqqqq@gmail.com
Contact: 1234567890
DOB: 2/3/2004
Address: asdfghjklmnbvcxz
Blood Group: B+
Role: warden

Our Address
Mysore Rd, RV Vidyaniketan Post,
Bengaluru, Karnataka
560059
: +852 1234 5678
: +852 8765 4321
: hostel@stay.net

Copyright © HOPLITE. All Rights Reserved.

Change password



Login

User ID

STAFF001

Password

•••••••

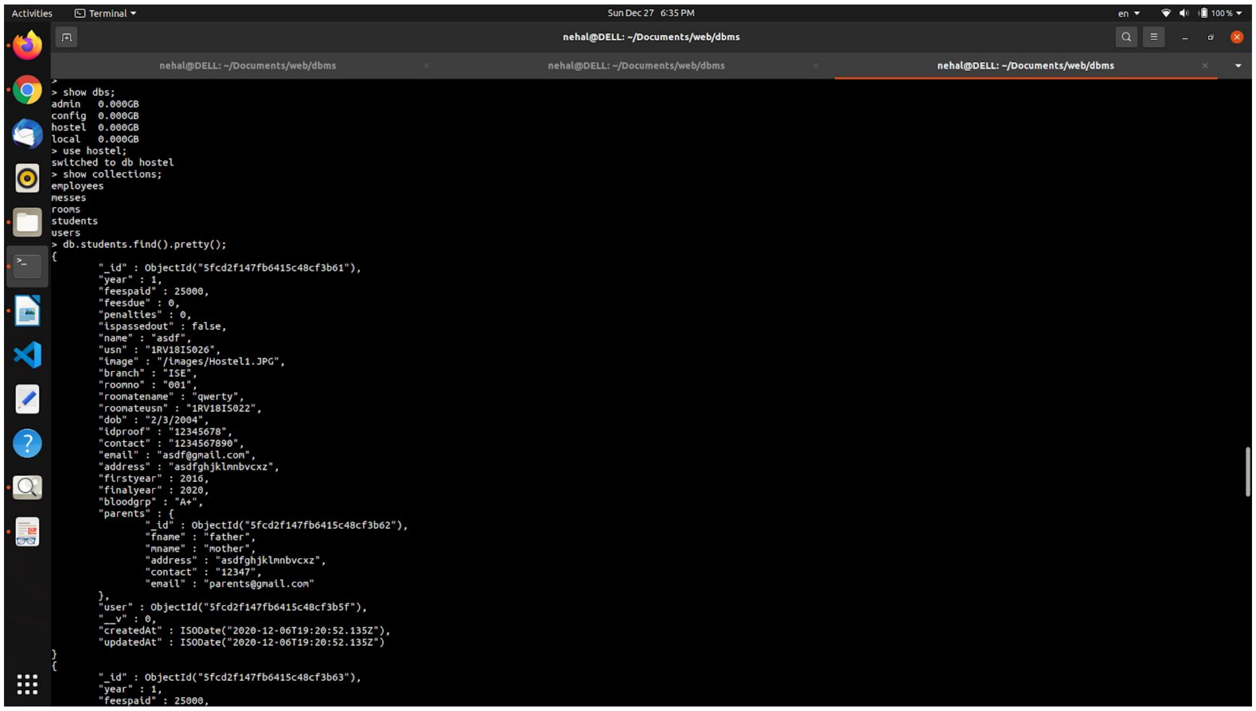
Login

Login page

TESTING AND RESULTS

5.1 Database Testing

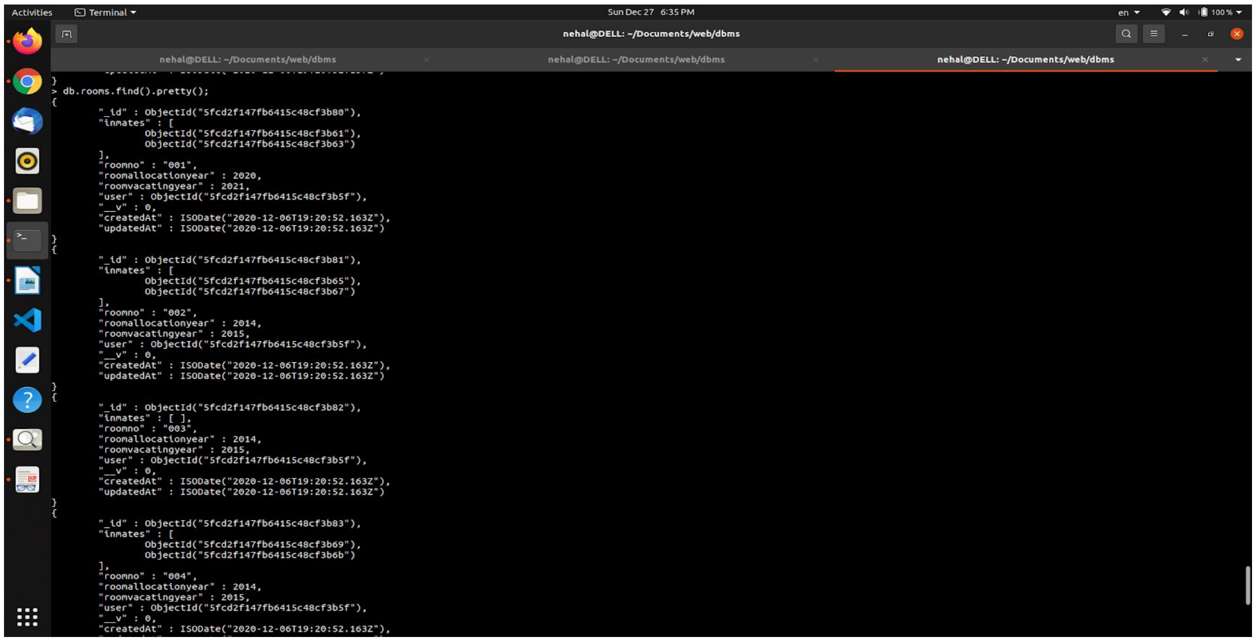
5.1.1 Test cases



Activities
Terminal
Sun Dec 27 6:35 PM
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms

```
>
> show dbs;
admin   0.000GB
config  0.000GB
hostel  0.000GB
local   0.000GB
> use hostel;
switched to db hostel
> show collections;
employees
messes
rooms
students
users
> db.students.find().pretty();
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b61"),
        "year" : 1,
        "feespaid" : 25000,
        "feesdue" : 0,
        "penalties" : 0,
        "ispassedout" : false,
        "name" : "asdf",
        "usn" : "1RV18IS026",
        "image" : "/images/Hostel1.JPG",
        "branch" : "ISE",
        "roomno" : "001",
        "roomatename" : "qwerty",
        "roomateusn" : "1RV18IS022",
        "dob" : "2/3/2004",
        "idproof" : "12345678",
        "contact" : "1234567890",
        "email" : "asdf@gmail.com",
        "address" : "asdfghjklmnbvcxz",
        "firstyear" : 2016,
        "finalyear" : 2020,
        "bloodgrp" : "A+",
        "parents" : {
                "_id" : ObjectId("5fcd2f147fb6415c48cf3b62"),
                "fname" : "father",
                "mname" : "mother",
                "address" : "asdfghjklmnbvcxz",
                "contact" : "12347",
                "email" : "parents@gmail.com"
        },
        "user" : ObjectId("5fcd2f147fb6415c48cf3b5f"),
        "__v" : 0,
        "createdAt" : ISODate("2020-12-06T19:20:52.135Z"),
        "updatedAt" : ISODate("2020-12-06T19:20:52.135Z")
}
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b63"),
        "year" : 1,
        "feespaid" : 25000,
```

Find all students



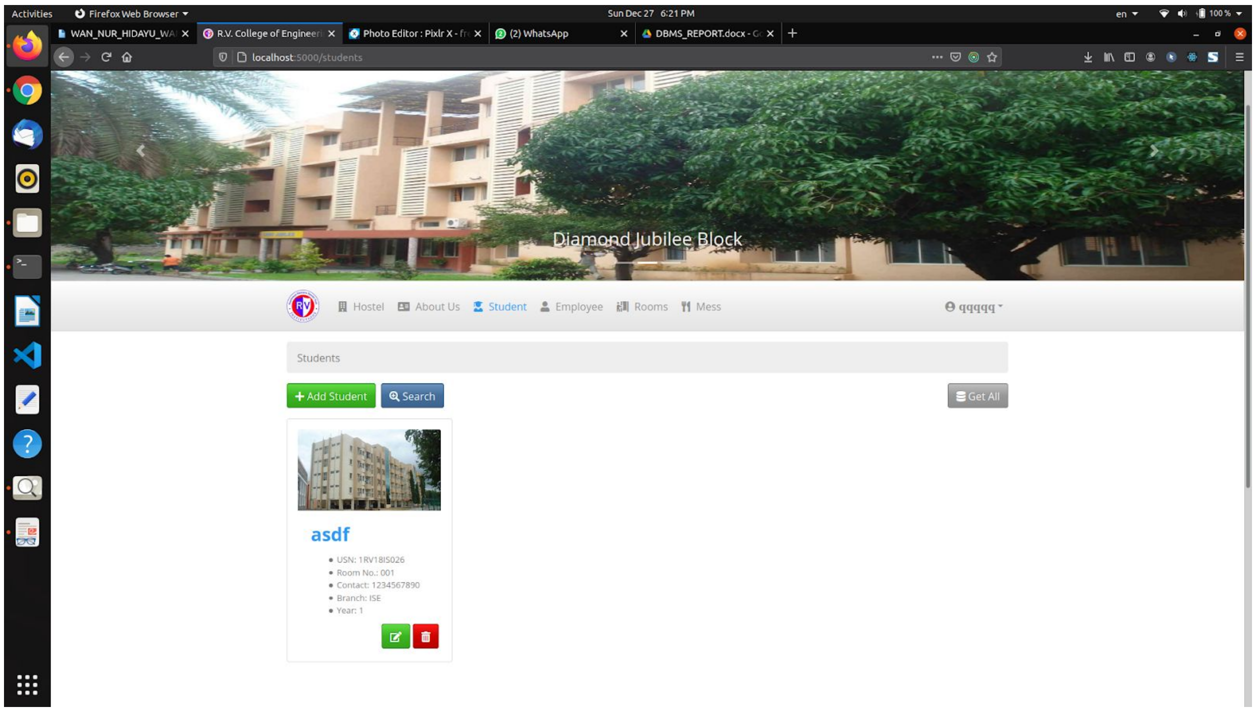
Activities
Terminal
Sun Dec 27 6:35 PM
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms
nehal@DELL: ~/Documents/web/dbms

```
}
> db.rooms.find().pretty();
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b80"),
        "inmates" : [
                ObjectId("5fcd2f147fb6415c48cf3b61"),
                ObjectId("5fcd2f147fb6415c48cf3b63")
        ],
        "roomno" : "001",
        "roomallocationyear" : 2020,
        "roomvacatingyear" : 2021,
        "user" : ObjectId("5fcd2f147fb6415c48cf3b5f"),
        "__v" : 0,
        "createdAt" : ISODate("2020-12-06T19:20:52.163Z"),
        "updatedAt" : ISODate("2020-12-06T19:20:52.163Z")
}
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b81"),
        "inmates" : [
                ObjectId("5fcd2f147fb6415c48cf3b65"),
                ObjectId("5fcd2f147fb6415c48cf3b67")
        ],
        "roomno" : "002",
        "roomallocationyear" : 2014,
        "roomvacatingyear" : 2015,
        "user" : ObjectId("5fcd2f147fb6415c48cf3b5f"),
        "__v" : 0,
        "createdAt" : ISODate("2020-12-06T19:20:52.163Z"),
        "updatedAt" : ISODate("2020-12-06T19:20:52.163Z")
}
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b82"),
        "inmates" : [ ],
        "roomno" : "003",
        "roomallocationyear" : 2014,
        "roomvacatingyear" : 2015,
        "user" : ObjectId("5fcd2f147fb6415c48cf3b5f"),
        "__v" : 0,
        "createdAt" : ISODate("2020-12-06T19:20:52.163Z"),
        "updatedAt" : ISODate("2020-12-06T19:20:52.163Z")
}
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b83"),
        "inmates" : [
                ObjectId("5fcd2f147fb6415c48cf3b69"),
                ObjectId("5fcd2f147fb6415c48cf3b6b")
        ],
        "roomno" : "004",
        "roomallocationyear" : 2014,
        "roomvacatingyear" : 2015,
        "user" : ObjectId("5fcd2f147fb6415c48cf3b5f"),
        "__v" : 0,
        "createdAt" : ISODate("2020-12-06T19:20:52.163Z"),
```

Get all rooms

5.2 Front End Testing

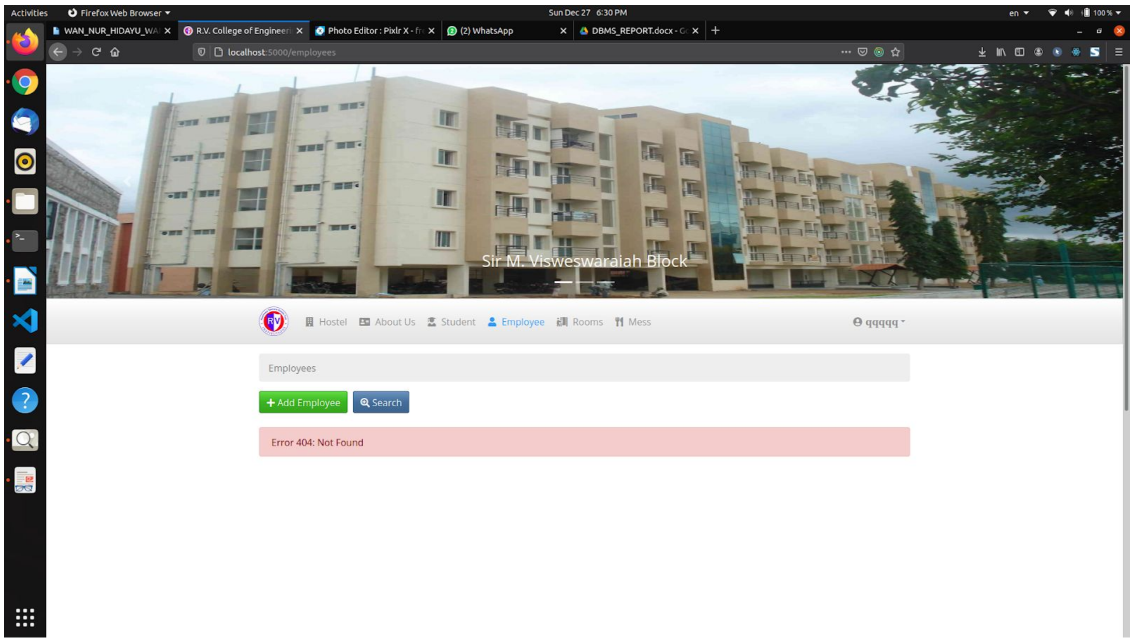
5.2.1 Test cases



asdf

- USN: 1RV18IS026
- Room No.: 001
- Contact: 1234567890
- Branch: ISE
- Year: 1

Find Student given his USN

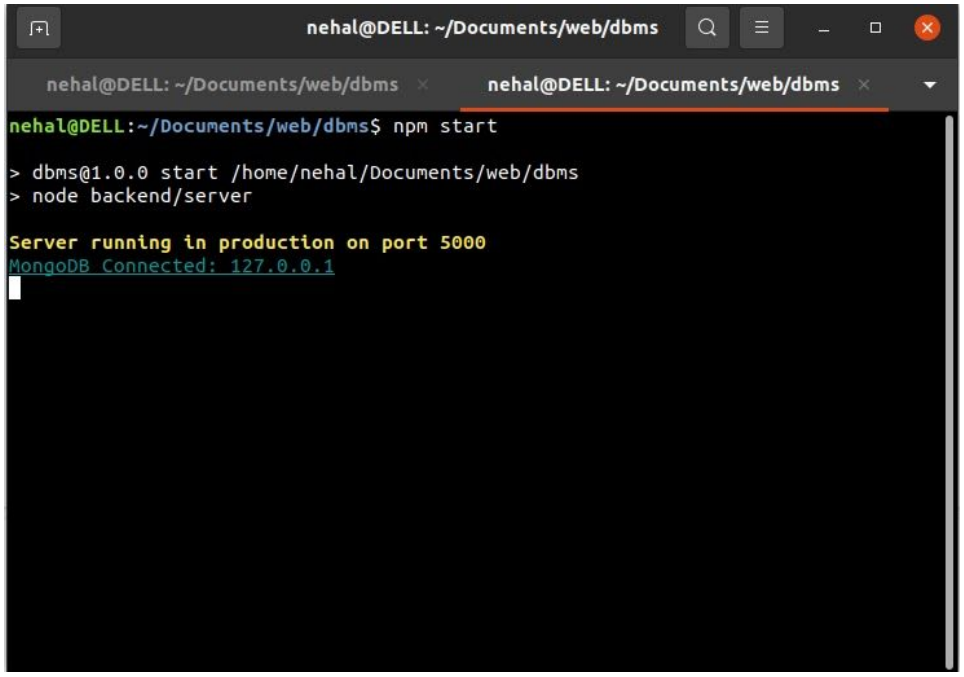


Error 404: Not Found

Employee not found

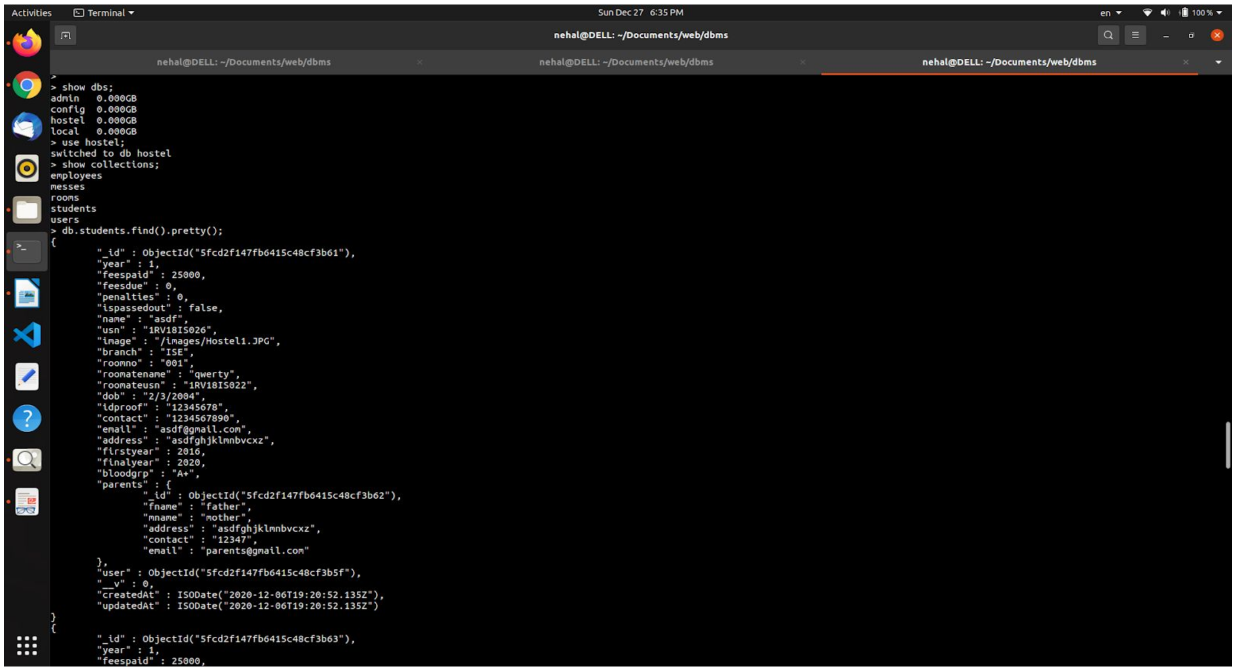
5.3 System Testing

5.3.1 Test cases



```
nehal@DELL:~/Documents/web/dbms$ npm start

> dbms@1.0.0 start /home/nehal/Documents/web/dbms
> node backend/server

Server running in production on port 5000
MongoDB Connected: 127.0.0.1
```

Database connectivity



```
>
> show dbs;
admin   0.000GB
config  0.000GB
hostel  0.000GB
local   0.000GB
> use hostel;
switched to db hostel
> show collections;
employees
messes
rooms
students
users
> db.students.find().pretty();
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b61"),
        "year" : 1,
        "feespaid" : 25000,
        "feesdue" : 0,
        "penalties" : 0,
        "ispassedout" : false,
        "name" : "asdf",
        "usn" : "1RV18IS026",
        "image" : "/images/Hostel1.JPG",
        "branch" : "ISE",
        "roomno" : "001",
        "roomatename" : "qwerty",
        "roomateusn" : "1RV18IS022",
        "dob" : "2/3/2004",
        "idproof" : "12345678",
        "contact" : "1234567890",
        "email" : "asdf@gmail.com",
        "address" : "asdfghjklmnbvcxz",
        "firstyear" : 2016,
        "finalyear" : 2020,
        "bloodgrp" : "A+",
        "parents" : {
                "_id" : ObjectId("5fcd2f147fb6415c48cf3b62"),
                "fname" : "father",
                "mname" : "mother",
                "address" : "asdfghjklmnbvcxz",
                "contact" : "12347",
                "email" : "parents@gmail.com"
        },
        "user" : ObjectId("5fcd2f147fb6415c48cf3b5f"),
        "__v" : 0,
        "createdAt" : ISODate("2020-12-06T19:20:52.135Z"),
        "updatedAt" : ISODate("2020-12-06T19:20:52.135Z")
}
{
        "_id" : ObjectId("5fcd2f147fb6415c48cf3b63"),
        "year" : 1,
        "feespaid" : 25000,
```

Database query testing

CONCLUSION

The proposed system for “Hostel and Mess Database Management System” is computerized. Today is the era of computers. This software project solves all the problems discussed above in the present system. The main objective of developing this project is to save time and efforts. The proposed system provides features on different tasks like, All the details related to hostelites can be found at one place like the admission details, fees details, room details , parent details ,mess details etc. All the details related to a mess employees and mess can be found here and managed by the wardens. The product will be useful to - Warden – To be able to alter and view student, employee, mess details. Mess Employee – To be able to view student details and view, alter mess details.

6.1 Limitations

ONLINE HOSTEL MANAGEMENT SYSTEM is very useful for hostel allotment and mess management . This hostel management software is designed for people who want to manage various activities in the hostel. For the past few years the numbers of educational institutions are increasing rapidly. Thereby the numbers of hostels are 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 computerized systems that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

6.2 Future Enhancements

Hostel Management Report. The “Hostel Management System" project is divided into two Schools that are growing in India to match the pace of population growth. It has outsmarted even the population growth rate of India. There are at least 2-5 good schools in every city of India (no matters how small is this) which are rushed by students. A big school, like one dealt in this project, faces a large number of problems involving data maintenance, storage and mining. The manual system of data processing and record keeping is error prone, labour intensive and thus costly. It requires a lot of paper handling which further requires proper storage facilities. Moreover, this modus operandi adversely affects the smooth functioning of the organization.

This system also figures out the human engineering considerations (ergonomics) which, in turn, has resulted in a user friendly, menu-driven Graphical User Interface. This system simplifies the upholding of large amounts of data and the speed of processing is off the capacity of any manual system. It minimizes the time and efforts of the user with efficiency. It is possible at any point of time to extend the software to stand at ceremony. This project is developed in such a way that any implementation or extension can be done easily.With so many outstanding features, it is expected that the proposed system would find its use in all modern schools.

REFERENCES

- https://nodejs.org/en/docs/
  Node.js Documentation

- https://reactjs.org/docs/getting-started.html
  ReactJS Documentation

- https://stackoverflow.com/
  StackOverflow