# CS816 Software Production Engineering
# Project Report

**Instructor**
*Prof. B. Thangaraju*

**TA**
*Suchi*

| Team Members | Roll No. |
|---|---|
| Darshitkumar Pipariya | MT2022035 |
| Dishangkumar Patel | MT2022039 |


**International Institute of Information And Technology, Bangalore**

**May 2023**

**Project Name : My Clean Room**
**( A feedback collection and visualisation application)**

# Table of Contents

# Introduction to My Clean Room

My Clean Room is a software suite (website + mobile application) to collect and visualise feedback in pseudo real-time. Users of the system will be students, supervisors of the cleaning staff.

Students submit feedback with ratings and remarks using a mobile application for their respective room cleaning work.This feedback was recorded for individual house keeping staff personnel.

Supervisors will be able to visualise the feedback in an aggregated manner in terms of Pie Charts and Line Charts. Also will be able to see min, max and average rating for current day cleaning works, count of total no. of rooms cleaned.

Analytics dashboard facilitates supervision using values that appear in charts, and helps the supervisor to track performance of individual cleaning staff members over the time. In contrast to maintaining feedback in a physical manner using a pen and notebook.

## Flows or problems addressed using this project

1. Data tampering can be avoided
2. Evaluating the performance of individuals can be done easily without going through a list of handwritten feedback manually which is a slow, and tedious process for daily evaluation.
3. Paper wastage can be prevented.

## Project feature and functionalities

1. Digital feedback collection, management and insights
2. One click analysis and visualisation of performance of each worker
3. Preventing paper wastage
4. Ease of taking informed decision
5. Easy to use mobile application and web application ( Very intuitive User Experience )
6. Prevents falsey data submission and tampering of data

Apart from this applications used by students and supervisors comes with some default features of

1. Uses authentication and authorization ( username, password combo)

# Important Assumption

1. Student to room mapping has be done by some mechanism
2. Students will not be able to create accounts on their own. Account will be created by some admin of hostel ( using some other mechanism )
3. List of housekeeping staff with their details already available in system, ready to be used
4. Supervisors are also pre created
5. Housekeeping staff members are assigned to some supervisor by default

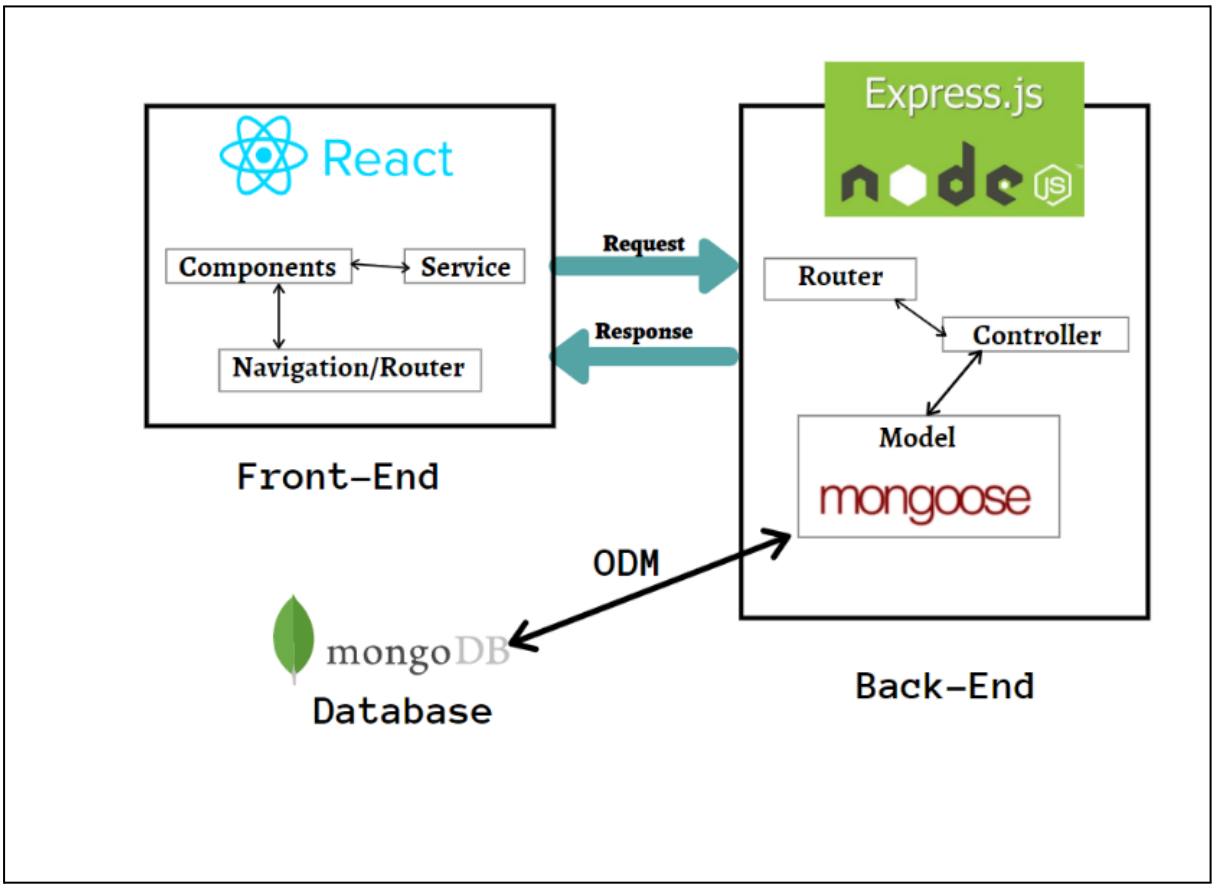
System Configuration and Infrastructure

Host System Configuration

Operating System : Ubuntu 22.04 ( LTS - Jammy Jellyfish )
CPU and RAM : intel i5 pentium processor 4 cores, 8 GB Memory
Disk Space : 1TB Hard drive

# Technology Stack

## Overview




## Backend:

- NodeJs ( javascript runtime )
- Express js ( web application framework for REST API )
- Joi ( data validation middleware)
- Winston ( Logging Library )
- Jest ( Testing Automation framework )
- Supertest ( library to test http servers )
- Mongoose ( MongoDB ODM )
- npm ( build tool )

## Database:

- MongoDB ( NoSQL Document based database )

## Frontend :

- ReactJs ( Declarative javascript library to build modular user interfaces )
- React-Native ( Declarative javascript library to build mobile application )

- Expo ( Application compiling and build Tool )

## DevOps Tools :

- Git / Github ( version control system )
- Docker ( Containerization Technology )
- Docker Hub ( Docker image registry )
- Github Actions / Jenkins ( Automation Tool for CI )
- Ansible (Suite of software tool to enable Infrastructure as Code ) used as Continuous Deployment tool
- Docker Compose ( Container orchestration tool )
- Ansible Vault ( Secret management )
- ELK Stack ( Log Monitoring, Text based search engine, visualisation tool)

# Software Development Life Cycle

## Installations

### React

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.

**Update local package indexes before installation** :

Keep the local packages and softwares updated.

```
dishang@dell:~$ sudo apt-get update_
```

### Installing NodeJS and NPM

Inorder to run React, Node environment shall be installed before starting,

```
sudo apt install  nodejs
sudo apt install npm
```

```
dishang@dell:~$ node -v
v16.18.0
dishang@dell:~$ npm -v
8.19.2
dishang@dell:~$ _
```

### Initialise react project using

Following command will install tool that will be used to initialise React Project

```
npm install create-react-app

#  following command will create and initialise react project

npx create-react-app mycleanroom
```

### Express

Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

**Express App**

Initializing node project with the default config:

```
npm init -y
```

-y automates the config with node default values i.e without going through the interactive process.

**Running the Node Application locally:**

# Testing

Supertest provides a high-level abstraction for testing HTTP, while still allowing you to drop down to the lower-level API provided by superagent.

```
ckend > __test__ > auth.test.js > supervisor > name
const app = require("../app");
const server = require("../index");
const supertest = require("supertest");
const mongoose = require("mongoose");
const { Supervisor, Student } = require("../api/user/user.model");
const httpStatus = require("http-status");

const student = {
  rollNo: "MT2022039",
  roomNo: "B576",
  name: "Dishang Patel",
  dob: "09-02-2000",
  gender: "M",
  password: "mypassword",
};

const supervisor = {
  supId: "SUP111",
  password: "superpassword",
  name: "Hanumanth BSK",
  dob: "01-05-1985",
  gender: "M",
};

beforeEach(() => {
  // timeout of 1 min added so that jest
  // don't exits before DB connection get established
  jest.setTimeout(60000);
});

afterAll(async () => {
  // delete users that we created for testing purpose
  await Student.deleteOne({ rollNo: { $eq: student.rollNo } });
  // await mongoose.connection.close();
  server.close();
});

describe("AUTHENTICATION ENDPOINT TESTING", () => {
  // creating users for testing

  test("POST /api/auth/login/[role:student]", async () => {
    await new Student(student).save();

    supertest(app)
      .post("/api/auth/login/student")
      .send({
        username: student.rollNo,
        password: student.password,
      })
      .expect(httpStatus.BAD_GATEWAY)
      .end((err) => {});
  }, 60000);
});
```

backend > __test__ > reachability.test.js > ...

```
const app = require("../app");
const server = require("../index");
const supertest = require("supertest");
const mongoose = require("mongoose");

beforeEach(() => {
  // timeout of 1 min added so that jest don't exits before DB connection get established
  jest.setTimeout(60000);
});

afterAll(async () => {
  // await mongoose.connection.close();
  server.close();
});

describe("SERVER REACHABILITY TEST", (done) => {
  test("GET /api", (done) => {
    supertest(app).get("/api").send().expect(200).then(done()).catch(done);
  }, 60000);

  test("GET /wrongendpoint", (done) => {
    supertest(app)
      .get("/wrongendpoint")
      .send()
      .expect(404)
      .then(done())
      .catch(done);
    // .then((res) => {
    //   done();
    // })
  }, 60000);
});
```

OBJ

```
const app = require("../app");
const server = require("../index");
const supertest = require("supertest");
const mongoose = require("mongoose");

beforeEach(() => {
  // timeout of 1 min added so that jest don't exits before DB connection get established
  jest.setTimeout(60000);
});

afterAll(() => {
  mongoose.connection.close();
  server.close();
});

describe("SERVER REACHABILITY TEST", () => {
  test("GET /api", (done) => {
    supertest(app)
      .get("/api")
      .send()
      .expect(200)
      .then((res) => {
        done();
      })
      .catch(done);
  });

  test("GET /wrongendpoint", (done) => {
    supertest(app)
      .get("/wrongendpoint")
      .send()
      .expect(404)
      .then((res) => {
        done();
      })
      .catch(done);
  });


});
```

## Source Control Management(SCM)

Source Control Management is used for tracking the file change history, source code, etc. It helps us in many ways in keeping the running project in a structured and organised way.

Repository Link

- Web Application and Backend Server https://github.com/dishanG09/MyCleanRoom
- Mobile Application https://github.com/Darshitpipariya/SPE_MAJOR

Frontend is created in MyCleanRoom repo inside client/ directory and backend is created in the backend/ directory.

# Containerization with Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker builds images automatically by reading the instructions from a Dockerfile -- a text file that contains all commands, in order, needed to build a given image. A Dockerfile adheres to a specific format and set of instructions which you can find at Dockerfile reference.

## Backend Dockerfile

```
FROM node

WORKDIR /mycleanroom/backend

COPY package*.json ./

RUN npm install

COPY . ./

CMD ["npm","start"]
```

## Frontend Dockerfile

```
FROM node as build-stage

WORKDIR /mycleanroom/frontend

COPY . ./

RUN npm install

RUN npm run build

# --- adding server ---

FROM nginx:alpine

WORKDIR /usr/share/nginx/html

RUN rm -rf ./*

COPY --from=build-stage /mycleanroom/frontend/build .

ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

Running docker build using this Dockerfile as the source creates the required Docker image that is ready to run our application.
We need to build the Docker image and push it to Docker Hub which requires logging in to the DockerHub account as well.
This will be covered in the Continuous Integration section.

## Docker Compose

Docker Compose is a tool that was developed to help define and share multi-container applications. With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

The big advantage of using Compose is you can define your application stack in a file, keep it at the root of your project repo (it's now version controlled), and easily enable someone else to contribute to your project. Someone would only need to clone your repo and start the Compose app.

On the host machine, follow the instruction of this link
[https://docs.docker.com/compose/install/](https://docs.docker.com/compose/install/)

After installation, you should be able to run the following and see version information.

```
$ docker-compose version
```

At the root of the app project, create a file named docker-compose.yml

Next, we'll define the list of services (or containers) we want to run as part of our application.

docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications. (compose-spec.json)

```yaml
version: "3"
services:
  mcr_backend:
    image: dishang09/mcr_backend
    container_name: mcr_backend
    working_dir: /mycleanroom/backend
    stdin_open: true
    ports:
      - 13131:12345
    volumes:
      - appvol:/mycleanroom/backend/logs
    networks:
      mcr_net:
        ipv4_address: 172.30.1.9
    entrypoint: ["npm", "start"]

  mcr_frontend:
    image: dishang09/mcr_frontend
    container_name: mcr_frontend
    expose:
      - 9090
    ports:
      - "9090:80"
    depends_on:
      - mcr_backend
    stdin_open: true
    networks:
      mcr_net:
        ipv4_address: 172.30.1.11

volumes:
  appvol:

networks:
  mcr_net:
    ipam:
      driver: default
      config:
        - subnet: 172.30.1.0/24
```

# Ansible

Ansible is an open-source automation tool, or platform, used for IT tasks such as configuration management, application deployment, intraservice orchestration and provisioning.
Ansible is mainly used to perform a lot of tasks that otherwise are time-consuming, complex, repetitive, and can make a lot of errors or issues.
Note: We are going to be pulling the Docker Hub image to the host system for Ansible deployment.
Creating Inventory file
The inventory file is used to specify the list of managed hosts/server machines.

**Inventory File**

```
[prod]
vm1 ansible_host=192.168.0.110 ansible_connection=ssh ansible_user=dishang ansible_sudo_pass=9200
```

**Ansible Vault**

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control.

```
backend > enc-env
$ANSIBLE_VAULT;1.1;AES256
65393864663033353333393335306330616338393763623732393066373062313032353735343431
65373331313939363936353039383734616232306532313 40a663331623733363934666531306236
37616263353162363764643863323939633939393962333532383936346661346535633831353637 61
33633730356163636 40a65633163383535623137396536373330303235303737323931306231333 4
35373566383764353563386139303334633033393138633935373263383762653331396331623636
38623664373534333734646130363263343265653765336538346661383134373939643266333731
66316437396531616339333631663365313838353862366437663361613437623161643736646266
39363536343237313236356138623539353638623563613534653139643763373465353537356266
61343938646134376466353362623738363037366632613836396461393939386632343434363565
39313162316162343137623630316339313765346138636261316434306638343933303130306665
63386332373739353333333735633763333233323636373131666535336362323130373134646334
65306133353431323466
```

We are using yaml files for storing environment configuration instead of a .env file because Ansible Vault requires a yaml or JSON type file for encrypting and decrypting. This will become more clear when the playbook is explained.

```
backend > .env
DB_URI='mongodb+srv://dishang09:5kAI2GcRAQYpWSX8@mcrcluster.ygwhkup.mongodb.net/?retryWrites=true&w=majority
PORT=13131
NODE_ENV='dev'
JWT_KEY=alca@fd9332rd
```

# Continuous Integration:

## Jenkins

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

Here is Jenkins Pipeline Script

JenkinsFile

```
pipeline {

    environment{
        DOCKERHUB = credentials('hub_credentials')
        JWT_KEY = credentials('JET_KEY')
        PORT = credentials('SERVER_PORT')
        MCR_DB_URI = credentials('MCR_DB_URI')
        MCR_VAULT_PASSWORD = credentials('MCR_VAULT_PASSWORD')
    }

    agent any

    stages {


        stage('building project images'){
            steps{
                sh '''
                    docker build -t dishang09/mcr_backend ./backend
                    docker build -t dishang09/mcr_frontend ./client
                '''
            }
        }
        stage('stopping containers'){

            steps{

                sh '''
                    docker stop mcr_backend mcr_frontend
                '''
            }

        }
        stage('testing application'){

            steps{

                sh '''
                    cd ./backend
                    export NODE_ENV=TEST
                    export DB_URI=$MCR_DB_URI
                    export PORT=$PORT
                    export JWT_KEY=$JWT_KEY
                    npm ci
                    npm test
                    cd ..
                '''

            }

        }


        stage('uploading image to registry'){
            steps{
                sh '''
                    docker login -u $DOCKERHUB_USR -p $DOCKERHUB_PSW
                    docker push dishang09/mcr_backend
                    docker push dishang09/mcr_frontend
                '''
            }
        }

        stage('cleaning local images'){
            steps{
                sh 'docker rmi dishang09/mcr_backend dishang09/mcr_frontend'
            }
        }

        stage('deploy application'){
            steps{

                ansiblePlaybook(
                inventory: 'inventory',
                playbook: 'deployment_playbook.yaml',
                vaultCredentialsId: 'MCR_VAULT_PASSWORD')
            }
        }
    }
}
```

### Github Action for Mobile Application

```yaml
name: Expo CI CD

on:
  push :
    branches : [main]
jobs:
  Build-for-android:
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v2.7.0
      - uses: actions/setup-node@v2.5.2
        with:
          node-version: 16.19.1
      - uses: actions/setup-java@v1.4.4
        with:
          java-version: '17.0.6' # The JDK version to make available on the path.
          java-package: jdk # (jre, jdk, or jdk+fx) - defaults to jdk
          architecture: x64 # (x64 or x86) - defaults to x64
      - uses: expo/expo-github-action@8.0.0
        with:
          expo-version: 6.3.0
          eas-version: 3.8.1
          expo-username: ${{ secrets.EXPO_CLI_USERNAME }}
          expo-password: ${{ secrets.EXPO_CLI_PASSWORD }}
          token: ${{ secrets.EXPO_TOKEN }}
      - name: Install deps
        run: npm install
      - name: build Android Application
        run:  eas build --profile preview --platform android --non-interactive

```

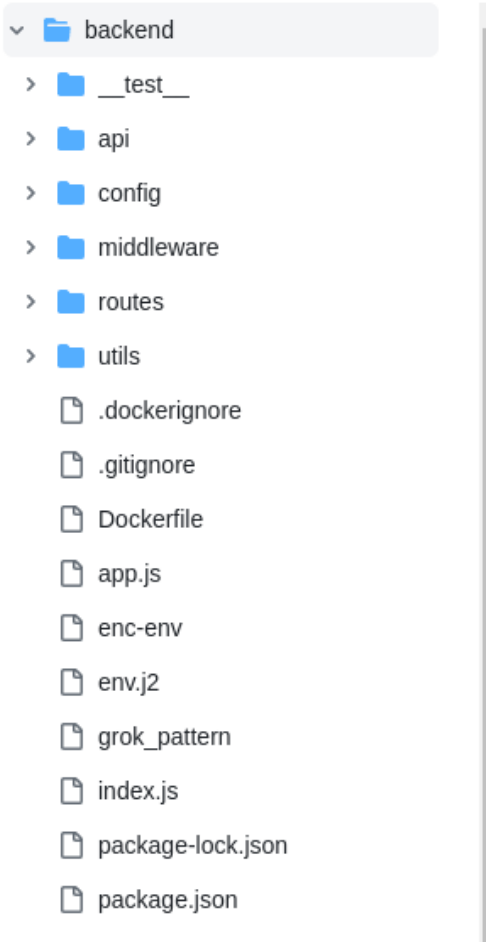
## Ansible Playbook

The main purpose of creating playbooks is that it encapsulates all the tasks under one playbook file.
In this playbook, the toughest task is to use the encrypted env-enc.yaml file and decrypt it somehow to send to the managed nodes and somehow place it inside the running container.
That is why we use the vars_files module with the Ansible playbook. We specify the env-enc.yaml as the file to look for and because Jenkins invoked the Ansible playbook with the Vault credentials, the file will now be decrypted for direct use but we cannot copy this file into the Docker container, which is why we use templating and this is where that env.j2 template comes into the picture.
The backend/env.j2 file which contains Jinja2 template variables will now be replaced by the decrypted environment variables in env-enc.yaml and we use the template module to store the created file under the name, env.yaml under the root directory of the managed nodes.
Now all we have to do is copy this env.yaml file into the backend container.
Everything else is explained in the comments of the playbook whose image is attached Below.

```yaml
- name: Deploying 'MyCleanRoom' Project
  hosts: prod
  vars_files:
    - ./backend/enc-env
  become: yes
  tasks:
    - name: copy compose file to deployment host
      copy:
        src: ./docker-compose.yaml
        dest: ./

    - name: generate env variable file out of jinja2 template
      template:
        src: backend/env.j2
        dest: .env

    - name: stop all the services
      command: docker-compose down

    - name: create all the service, without starting containers
      command: docker-compose up --no-start

    - name: copy env variable file inside backend container
      command: docker cp .env mcr_backend:/mycleanroom/backend

    - name: start the services
      command: docker-compose up -d
```

# Code WalkThrough

**Backend**



### Web Application

- client
  - public
  - src
  - .dockerignore
  - .gitignore
  - Dockerfile
  - README.md
  - package-lock.json
  - package.json
  - yarn.lock
- Jenkinsfile
- README.md
- deployment_playbook.yaml
- docker-compose.yaml
- inventory

### Mobile Application

- .github/workflows
- assets
- components
- context
- screens
- util
- .gitignore
- App.js
- README.md
- app.json
- babel.config.js
- eas.json
- index.js
- package-lock.json
- package.json

# Models

● It is known as the lowest level which means it is responsible for maintaining data.
● The Model is actually connected to the database so anything you do with data - adding or retrieving data is done in the Model component.
● It responds to the controller requests because the controller never talks to the database by itself. The Model talks to the database back and forth and then it gives the needed data to the controller.

### feedback

```
const feedbackSchema = new Schema(
  {
    hkId: { type: Types.String, required: true },
    student_roll_no: { type: Types.String, required: true },
    student_room_no: { type: Types.String, required: true },
    rating: { type: Types.Number, required: true, min: 0, max: 5 },
    remarks: { type: Types.String },
  },
  { timestamps: true }
);
```

### Users

```
const userSchema = new Schema({
  name: { type: Types.String, required: true, min: 5 },
  dob: { type: Types.Date, required: true },
  gender: { type: Types.String, required: true },
  username: { type: Types.String, required: true },
  password: { type: Types.String, required: true },
});

const supervisorSchema = new Schema(
  {
    supId: { type: Types.String, required: true, unique: true },
    password: { type: Types.String, required: true },
    name: { type: Types.String, required: true },
    gender: { type: Types.String },
    dob: { type: Types.String, required: true },
  },
  { timestamps: true }
);
```

```
const studentSchema = new Schema(
  {
    rollNo: { type: Types.String, required: true, unique: true },
    password: { type: Types.String, required: true },
    name: { type: Types.String, required: true },
    gender: { type: Types.String },
    dob: { type: Types.String, required: true },
    roomNo: { type: Types.String, required: true, unique: true },
    reset_password_flag: { type: Types.Boolean, default: false },
  },
  { timestamps: true }
);

const hkStaffSchema = new Schema(
  {
    hkId: { type: Types.String, required: true, unique: true },
    name: { type: Types.String, required: true },
    gender: { type: Types.String },
    dob: { type: Types.String, required: true },
    supId: { type: Types.String, required: true },
    password: { type: Types.String, default: "null" },
  },
  { timestamps: true }
);
```

# Dependencies:

### Backend

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "jest --detectOpenHandles"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "http-status": "^1.6.2",
    "joi": "^17.9.2",
    "jsonwebtoken": "^9.0.0",
    "mongoose": "^7.0.5",
    "winston": "^3.8.2"
  },
  "devDependencies": {
    "jest": "^29.5.0",
    "supertest": "^6.3.3"
  }
}
```

## Frontend

```
1   {
2     "name": "client",
3     "version": "0.1.0",
4     "private": true,
5     "dependencies": {
6       "@emotion/react": "^11.11.0",
7       "@emotion/styled": "^11.11.0",
8       "@mui/material": "^5.12.3",
9       "@testing-library/jest-dom": "^5.14.1",
10      "@testing-library/react": "^13.0.0",
11      "@testing-library/user-event": "^13.2.1",
12      "chart.js": "^4.3.0",
13      "react": "^18.2.0",
14      "react-chartjs-2": "^5.2.0",
15      "react-dom": "^18.2.0",
16      "react-router-dom": "^6.11.1",
17      "react-scripts": "5.0.1",
18      "web-vitals": "^2.1.0"
19    },
20    "scripts": {
21      "start": "react-scripts start",
22      "build": "react-scripts build",
23      "test": "react-scripts test",
24      "eject": "react-scripts eject"
25    },
26    "eslintConfig": {
27      "extends": [
28        "react-app",
29        "react-app/jest"
30      ]
31    },
32    "browserslist": {
33      "production": [
34        ">0.2%",
35        "not dead",
36        "not op_mini all"
37      ],
38      "development": [
39        "last 1 chrome version",
40        "last 1 firefox version",
41        "last 1 safari version"
42      ]
43    }
44  }
```

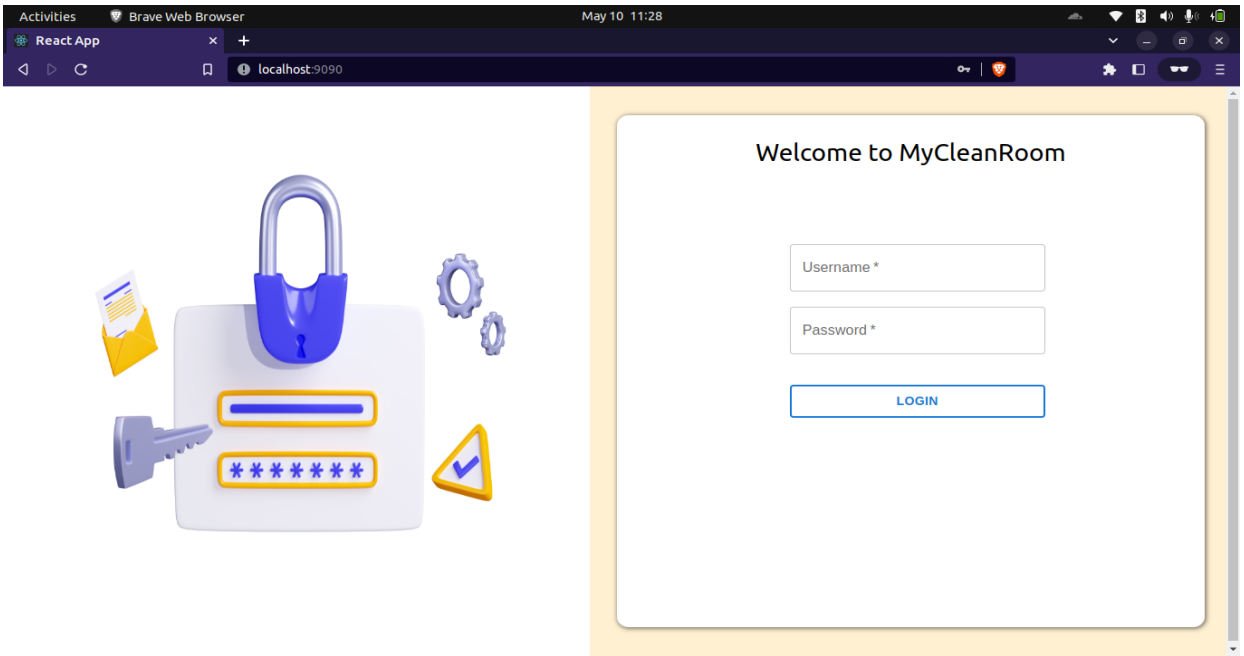
### React Native Mobile Application

```
1   {
2     "name": "spe_major_1",
3     "version": "1.0.0",
4     "scripts": {
5       "start": "expo start",
6       "android": "expo start --android",
7       "ios": "expo start --ios",
8       "web": "expo start --web"
9     },
10    "dependencies": {
11      "@expo/vector-icons": "^13.0.0",
12      "@react-native-community/netinfo": "9.3.7",
13      "@react-navigation/drawer": "^6.6.2",
14      "@react-navigation/native": "^6.1.6",
15      "@react-navigation/stack": "^6.3.16",
16      "axios": "^1.3.5",
17      "expo": "~48.0.9",
18      "expo-barcode-scanner": "~12.3.2",
19      "expo-camera": "~13.2.1",
20      "expo-secure-store": "~12.1.1",
21      "expo-status-bar": "~1.4.4",
22      "react": "18.2.0",
23      "react-native": "0.71.6",
24      "react-native-gesture-handler": "~2.9.0",
25      "react-native-reanimated": "~2.14.4",
26      "react-native-safe-area-context": "4.5.0",
27      "react-native-screens": "~3.20.0"
28    },
29    "devDependencies": {
30      "@babel/core": "^7.20.0"
31    },
32    "private": true
33  }
```

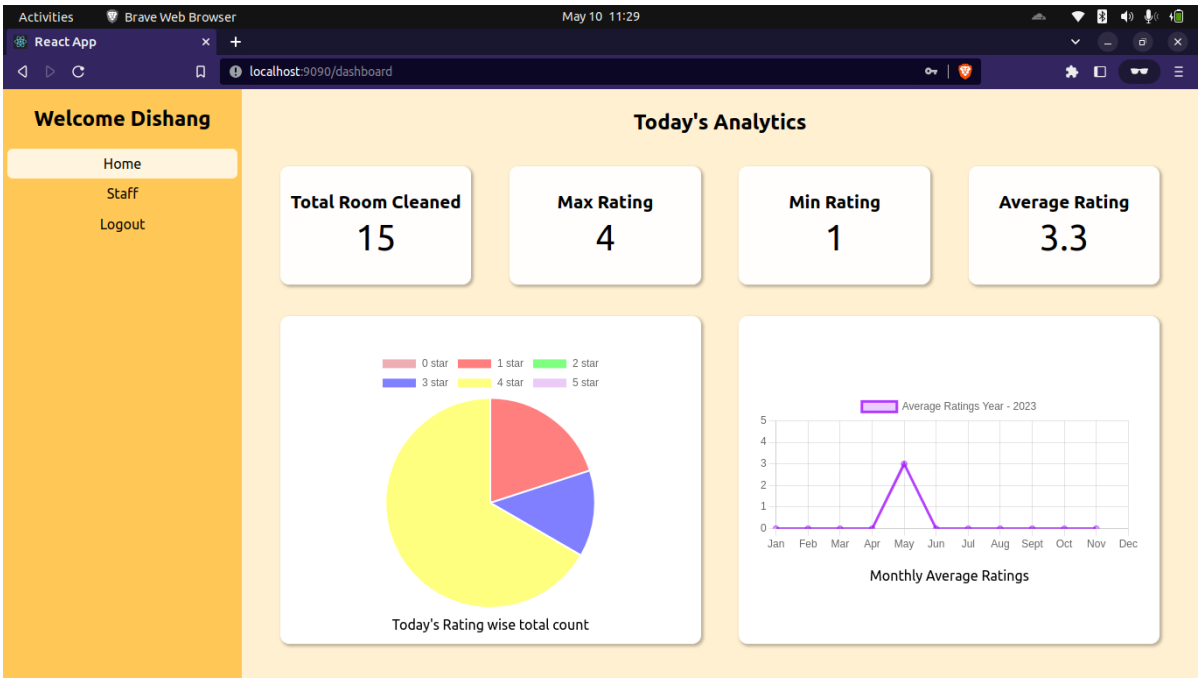
# Project / application screenshots

## Web Application

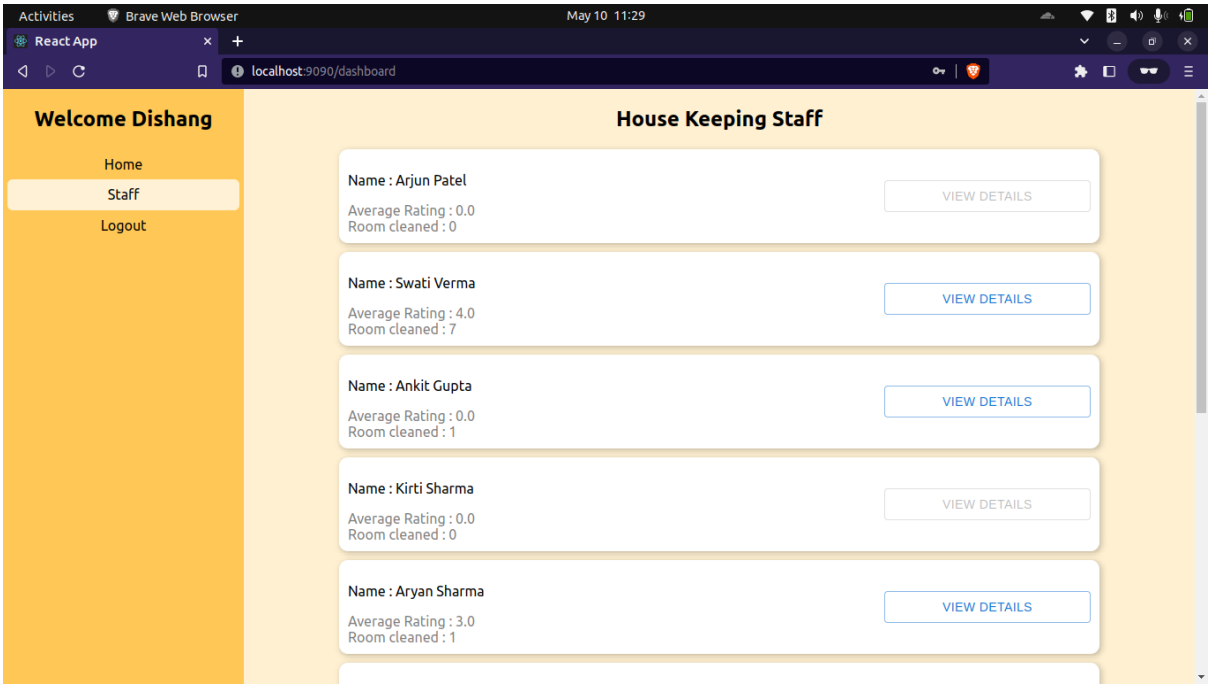
1. Login Page



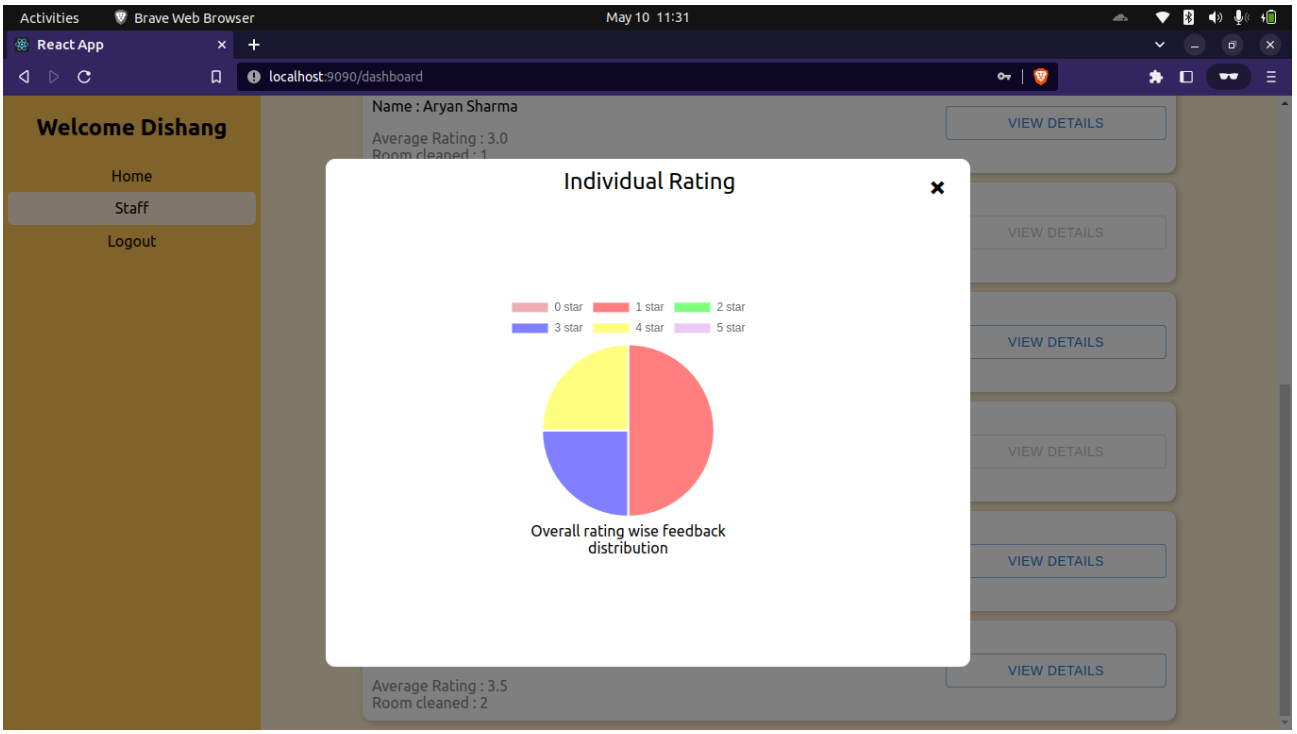

2. Home screen



3. Staff member list

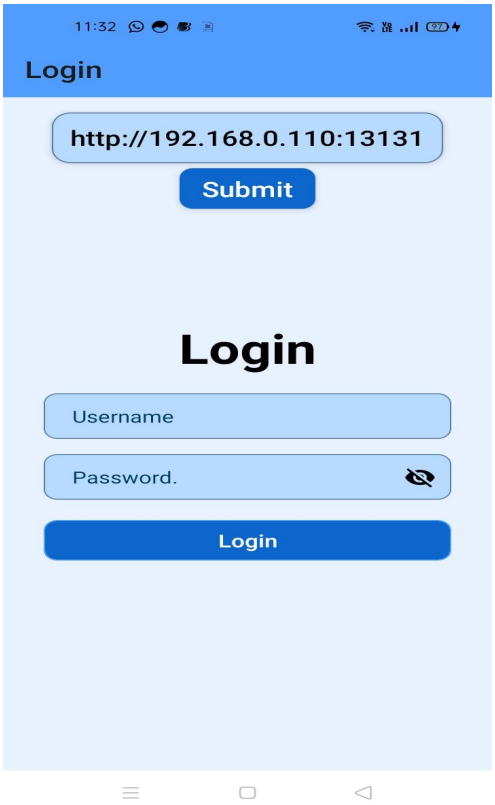



4. Individual performance rating



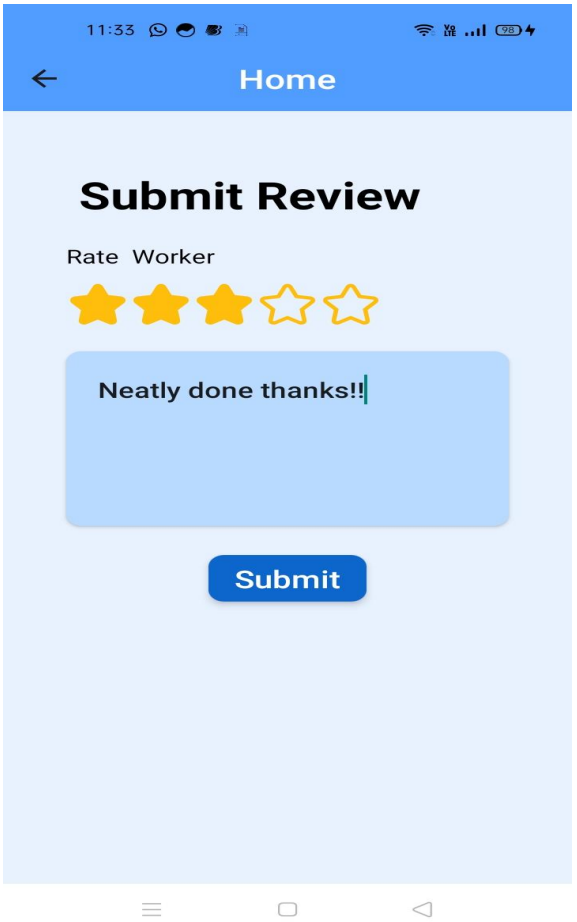
# Mobile Application Screen shots

1. Login Screen

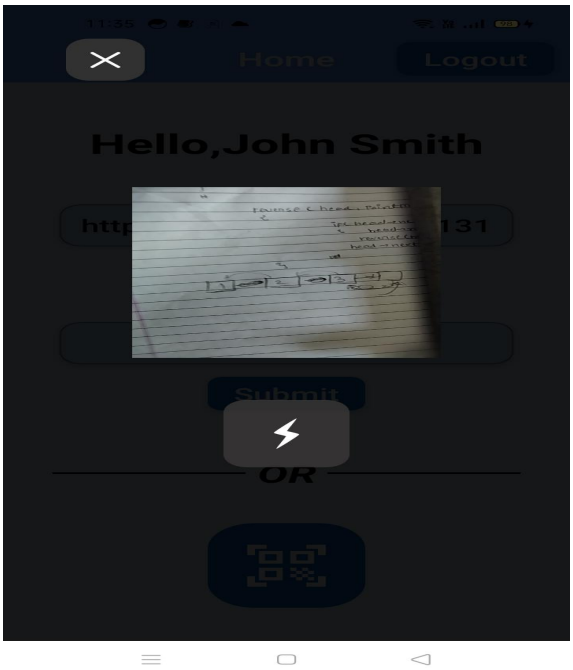



2. Home Screen

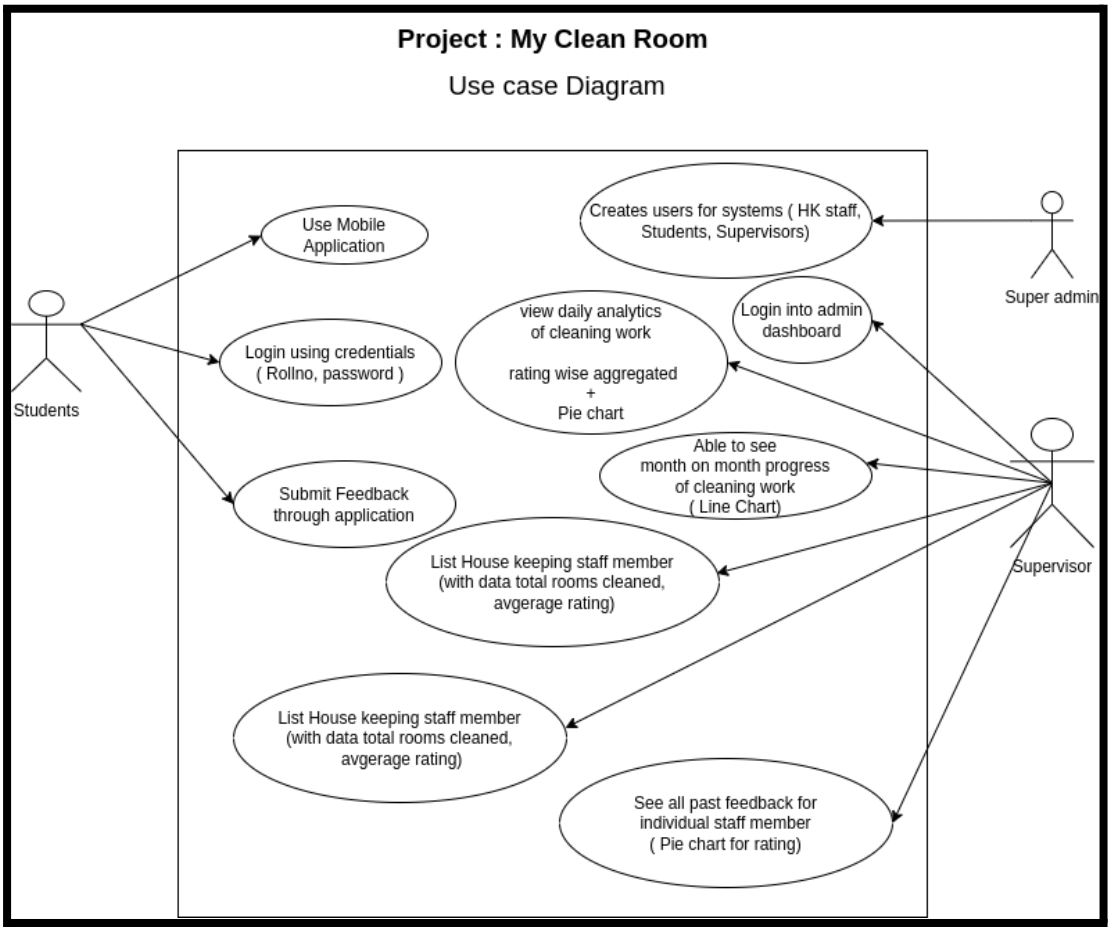

3. Feedback Form


11:33
Home
Submit Review
Rate Worker
Neatly done thanks!!
Submit

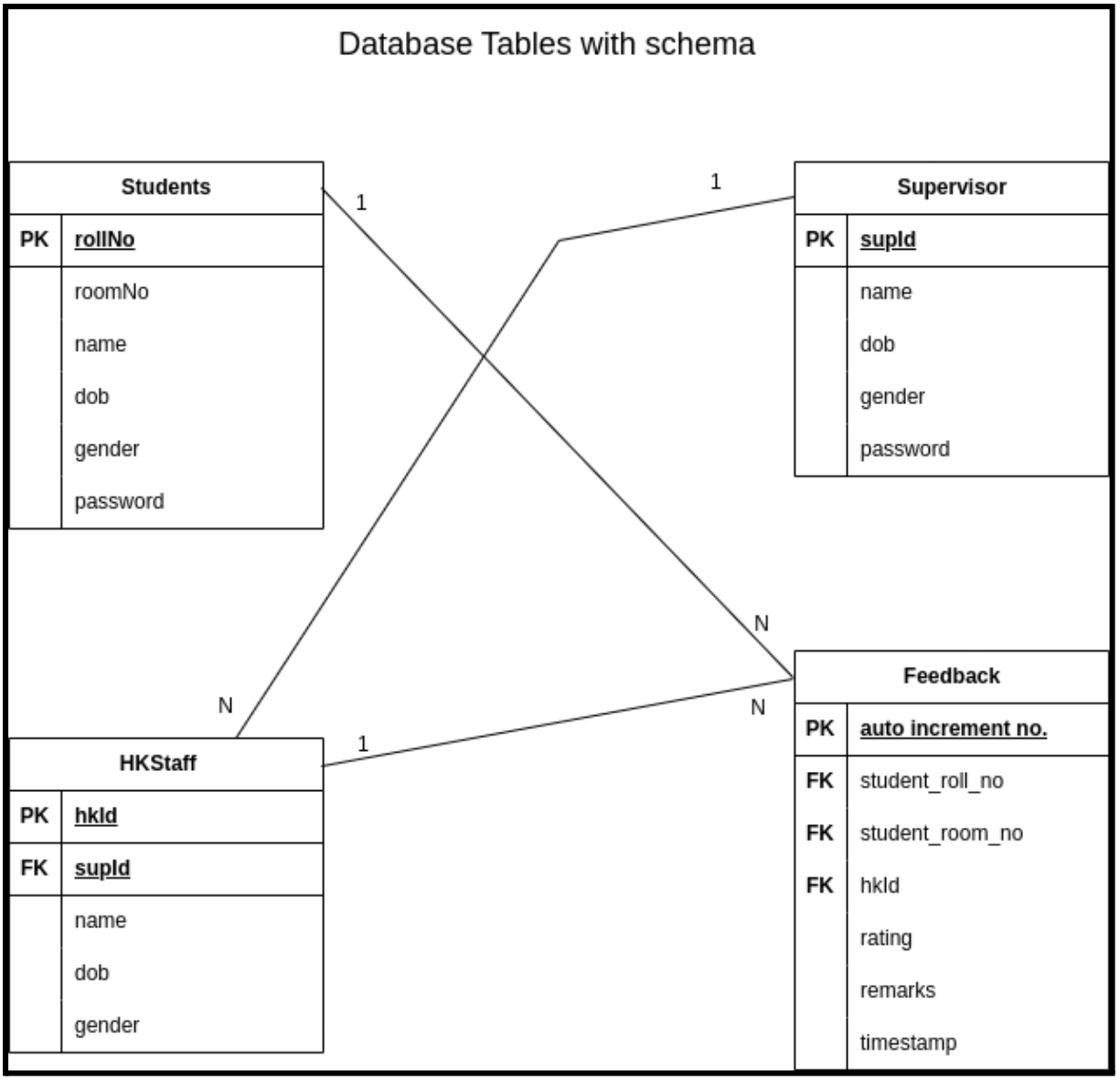
4. QR Scanner


Home
Logout
Hello,John Smith
Submit
OR

# Use Case Diagram and Database Schema Design



DB Design


Database Tables with schema
Students
PK
rollNo
roomNo
name
dob
gender
password
Supervisor
PK
supId
name
dob
gender
password
HKStaff
PK
hkId
FK
supId
name
dob
gender
Feedback
PK
auto increment no.
FK
student_roll_no
FK
student_room_no
FK
hkId
rating
remarks
timestamp
1
1
N
N
N
1