VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Internet and Web Programming

J-Component Project Report

(2020 - 2021)

MEDICARE - Online Clinic Administrator using Node JS

Under the guidance of Prof. Jayakumar K

School: School of Computer Science and Engineering

Course Code: CSE3002

Slot: B2

Team members:

1. Hrishita Chakrabarti, 18BCE0408
2. Srinivas N, 18BCE0048
3. Rahul, 18BCE0018

TABLE OF CONTENT

Abstract

In the modern medical world, doctors are aided by a wide variety of tools, machines and software. This advancement in technology has facilitated the large-scale collection, storage and processing of medical data from patients. With the increasing demands being placed on medical staff which is only getting worse amid the COVID-19 pandemic, it is difficult for medical personnel to keep track of this data. Due to the high complexity and rise of data in healthcare, Software has become a very integral part of healthcare and is increasingly being applied within the field to solve the complex problems of Hospital Management. This is being done by increasing availability of healthcare data and rapid progress of analytics techniques i.e. boosting the efficiency of patient analysis. We have made a Hospital Management Application which can be used by many hospitals to maintain their databases more efficiently. This Application mainly helps to make the lives of patients easier by collecting all the important information required for a patient into a single place and therefore saving a lot of time. One of our reasons to make this is that due to a lot of steps before a patient is admitted or treatment of a patient starts, their condition sometimes takes a fatal and non-recovering turn as we have seen in the case of COVID-19, where a lot of patients are losing their lives because of waiting outside of hospitals.

Introduction

We have made the application by keeping in mind that the hospital conditions are not ideal, that is, due to a large population visiting hospitals every day it is not possible that every patient that tries to get admitted into a Hospital will get admitted and moved into a room immediately. The patient might have to wait sometime so that a room is vacated.

We have made a system in which each disease is given a score. Each patient is given a score according to the diseases he has. This score is made according to how serious the disease is, that is, how much at risk a patient is if he/she has that particular disease. This score helps the doctors and nurses and the helping staff of the hospital to determine which patient should be admitted first. The patient who has the highest score among the waiting patients is given the highest priority and given the room even if many patients are admitted before him/her.

i. Problem Statement:

With the ongoing pandemic situation, hospitals across the world are facing huge issues in terms of administration. With all medical workers being forced to work overtime, human errors are increasing and this is posing to be a great threat for patients seeking immediate attention. One of the major issues right now is keeping track of admitted and released patients and assigning them beds according to priority. At the moment it is a first come first serve basis wherein sometimes a patient has to be taken off the treatment procedure to accomodate a more serious case leading to improper treatments which consequently are contributing to the rise in fatal complications amongst patients. There is a serious need for an automation of the hospital management process especially with a priority ordering of the patients based on their need for immediate medical attention.

ii. Technical Specification:

The minimum system requirements to get this project are as follows:

- Processor: dual core @ 2.4 GHz (i5 or i7 Intel processor or equivalent AMD) or higher.
- RAM: 2GB or more.
- Hard disk space: 10 GB
- Browser: Chrome, Mozilla Firefox, Safari or any other.
- Operating System: Windows 7/above with Service Pack 1 or Ubuntu 18.04 or Apple OS X

Existing System Problems

With the ever increasing number of cases of COVID-19, hospitals across the world are finding it difficult to keep pace with the number of admissions due to their limited resources such as number of beds available, number of masks. This process is even more painful due to the inefficiencies of the existing system in the organization bed allocations and patient detail updation. The main issues identified are:

- Can only be accessed at central workstations available on certain floors to get hold of important data.
- Most data is stored in patient charts which can be misplaced, out of date or just inefficient to fill.
- Doctors depend on nurses to monitor patients and inform them directly about any updates and regular checkups that are pending.
- The immense pressure from work when the medical sector is overburdened can lead to errors and oversights which may cost lives.

How we solve these issues:

- The system is available to all registered users (patients, nurses and hospital administrators) on their phones as well making it easier to check on the go.
- Data is stored in a secure database and can be accessed from any authorized account when needed so there is no risk of misplacement or incorrect labelling.
- Nurses are no longer burdened with the task of remembering all crucial details and thus no longer have to face the immense pressure of a life depending on their memory
- Rather than depending on human allocation which can be inefficient, patients are automatically prioritized based on the severity of their condition and can be allocated the appropriate room immediately.

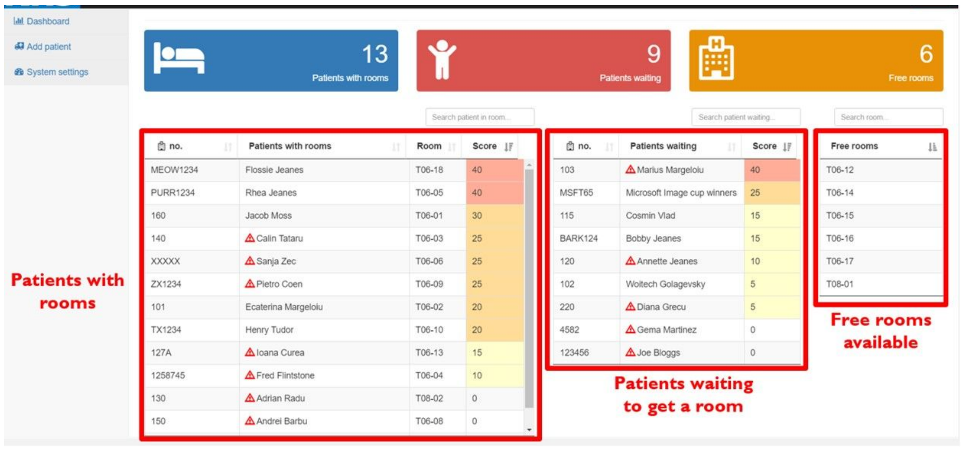
<u>Proposed System Design</u>:

The project is divided into the following modules:

1. **Dashboard Module**: Contains Vital info such as patient names, room numbers and free rooms at a glance.
2. **Login Module**: Used to differentiate between doctors and admins.
3. **Patients Module**: Used to add/remove patient names and information regarding their condition.
4. **Doctors Module**: Used by doctor to update patient details.
5. **Settings Module**: Used to add/remove new categories of diagnosis and rooms from the database.

i. <u>Dashboard Module</u>:

The main page of the website from where the hospital can monitor the patient's status in their allotted rooms and can also keep track of the list of patients kept in waiting and the free rooms available to a lot of them. A red alert is seen against the name of a patient whose status hasn't been updated in the last 24 hours.

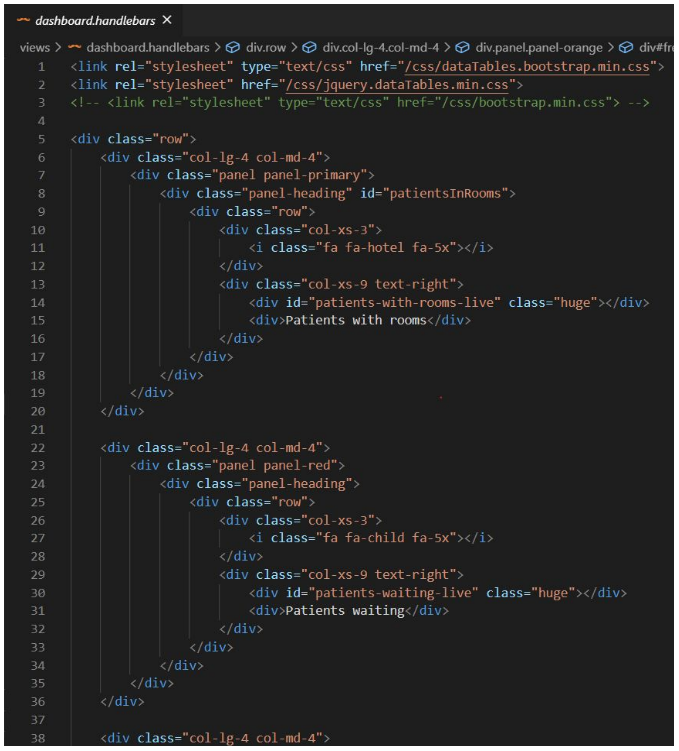


no.	Patients with rooms	Room	Score
MEOW1234	Flossie Jeanes	T06-18	40
PURR1234	Rhea Jeanes	T06-05	40
160	Jacob Moss	T06-01	30
140	⚠Calin Tataru	T06-03	25
XXXXX	⚠Sanja Zec	T06-06	25
ZX1234	⚠Pietro Coen	T06-09	25
101	Ecaterina Margeloiu	T06-02	20
TX1234	Henry Tudor	T06-10	20
127A	⚠Ioana Curea	T06-13	15
1258745	⚠Fred Flintstone	T06-04	10
130	⚠Adrian Radu	T08-02	0
150	⚠Andrei Barbu	T06-08	0

no.	Patients waiting	Score
103	⚠Marius Margeloiu	40
MSFT65	Microsoft Image cup winners	25
115	Cosmin Vlad	15
BARK124	Bobby Jeanes	15
120	⚠Annette Jeanes	10
102	Woitech Golagevsky	5
220	⚠Diana Grecu	5
4582	⚠Gema Martinez	0
123456	⚠Joe Bloggs	0

Free rooms
T06-12
T06-14
T06-15
T06-16
T06-17
T08-01

Fig 1: Dashboard Interface

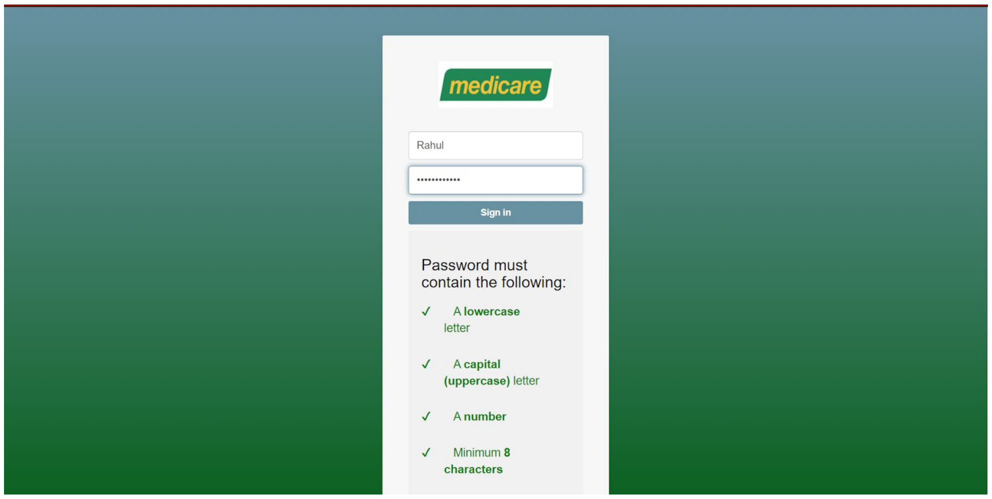

dashboard.handlebars
views > dashboard.handlebars > div.row > div.col-lg-4.col-md-4 > div.panel.panel-orange > div#fre

```
<link rel="stylesheet" type="text/css" href="/css/dataTables.bootstrap.min.css">
<link rel="stylesheet" href="/css/jquery.dataTables.min.css">
<!-- <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css"> -->

<div class="row">
    <div class="col-lg-4 col-md-4">
        <div class="panel panel-primary">
            <div class="panel-heading" id="patientsInRooms">
                <div class="row">
                    <div class="col-xs-3">
                        <i class="fa fa-hotel fa-5x"></i>
                    </div>
                    <div class="col-xs-9 text-right">
                        <div id="patients-with-rooms-live" class="huge"></div>
                        <div>Patients with rooms</div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="col-lg-4 col-md-4">
        <div class="panel panel-red">
            <div class="panel-heading">
                <div class="row">
                    <div class="col-xs-3">
                        <i class="fa fa-child fa-5x"></i>
                    </div>
                    <div class="col-xs-9 text-right">
                        <div id="patients-waiting-live" class="huge"></div>
                        <div>Patients waiting</div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="col-lg-4 col-md-4">
```

Fig 2: Dashboard Handlebar Code Snippet

ii. <u>Login Module</u>:

Hospital administrators can login to the website to add/modify the patient's status or add a new room. Doctors can log in to keep a track of a patient’s treatment.



Password must contain the following:

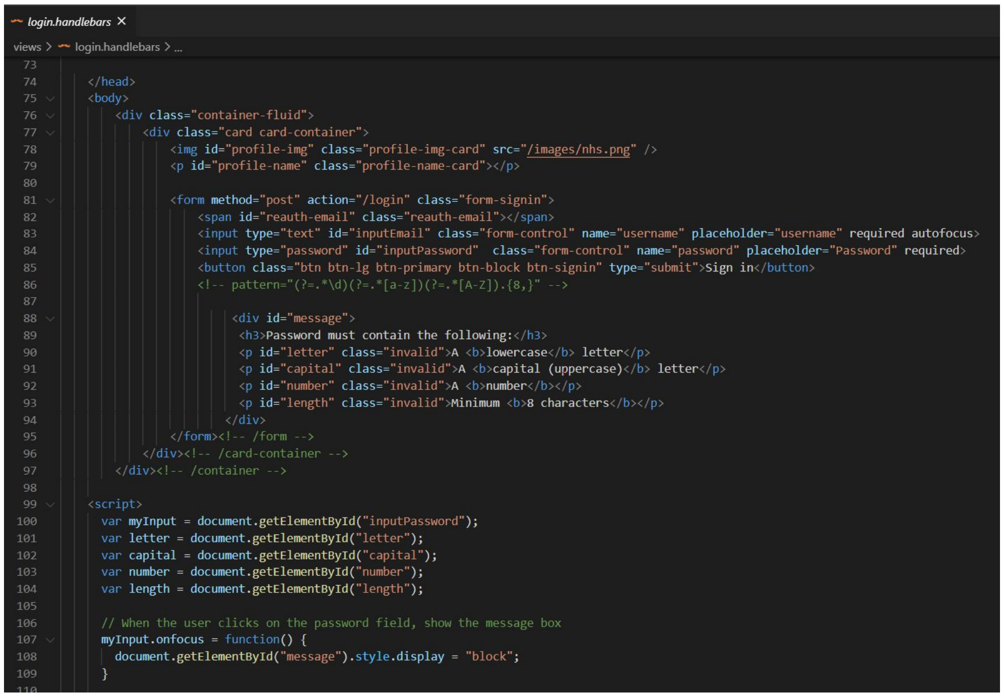
✓ A **lowercase** letter

✓ A **capital (uppercase)** letter

✓ A **number**

✓ Minimum **8 characters**

Fig 3: Login Page Interface

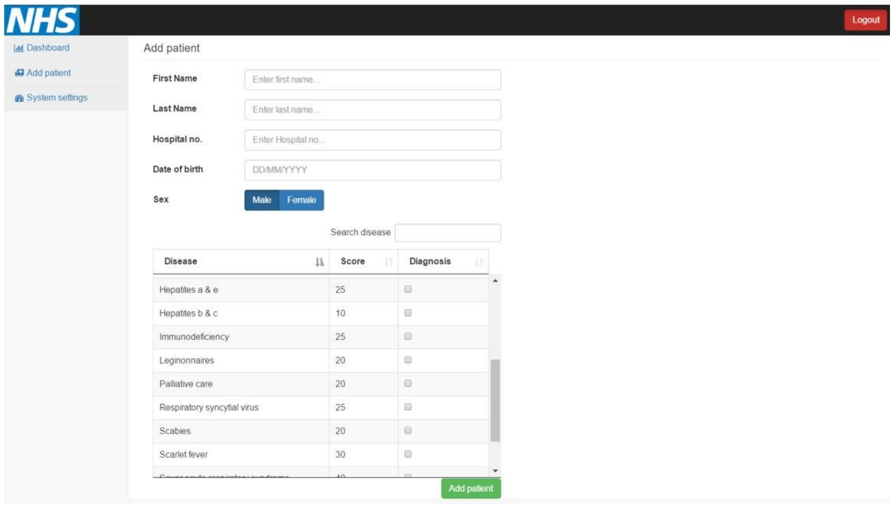


```
    </head>
    <body>
        <div class="container-fluid">
            <div class="card card-container">
                <img id="profile-img" class="profile-img-card" src="/images/nhs.png" />
                <p id="profile-name" class="profile-name-card"></p>

                <form method="post" action="/login" class="form-signin">
                    <span id="reauth-email" class="reauth-email"></span>
                    <input type="text" id="inputEmail" class="form-control" name="username" placeholder="username" required autofocus>
                    <input type="password" id="inputPassword"  class="form-control" name="password" placeholder="Password" required>
                    <button class="btn btn-lg btn-primary btn-block btn-signin" type="submit">Sign in</button>
                    <!-- pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}" -->

                        <div id="message">
                         <h3>Password must contain the following:</h3>
                         <p id="letter" class="invalid">A <b>lowercase</b> letter</p>
                         <p id="capital" class="invalid">A <b>capital (uppercase)</b> letter</p>
                         <p id="number" class="invalid">A <b>number</b></p>
                         <p id="length" class="invalid">Minimum <b>8 characters</b></p>
                        </div>
                </form><!-- /form -->
            </div><!-- /card-container -->
        </div><!-- /container -->

    <script>
      var myInput = document.getElementById("inputPassword");
      var letter = document.getElementById("letter");
      var capital = document.getElementById("capital");
      var number = document.getElementById("number");
      var length = document.getElementById("length");

      // When the user clicks on the password field, show the message box
      myInput.onfocus = function() {
        document.getElementById("message").style.display = "block";
      }
```

Fig 4: Login Handlebar Code Snippet

iii. <u>Patients Module</u>:

Hospital administrators can add/remove patients from a given room. The medical history, diagnosis and treatment are also stored for the assigned doctor's reference and can be edited only by the assigned doctor.



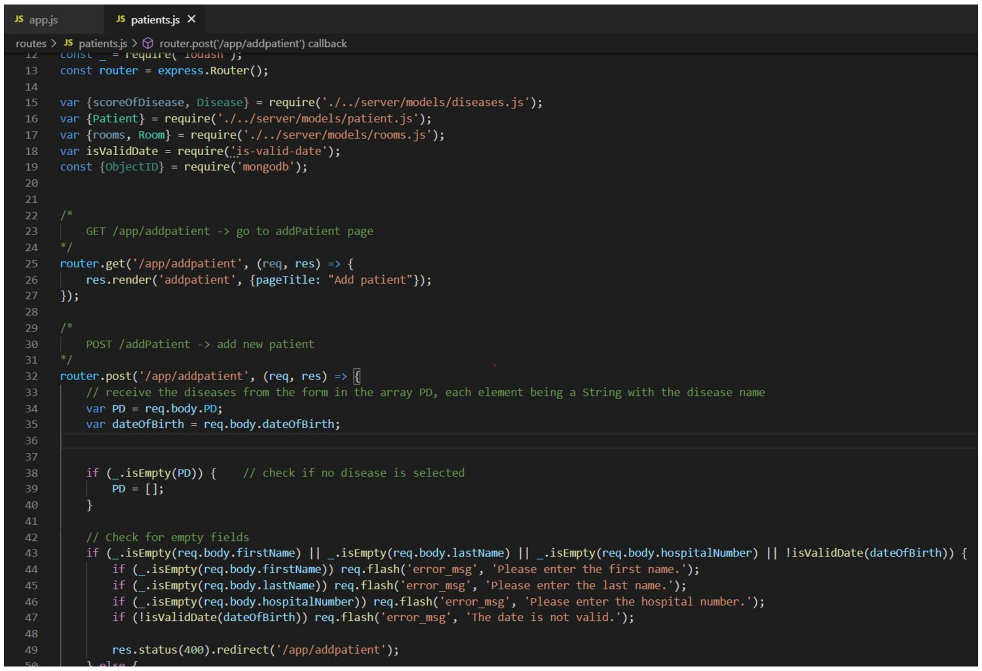
Add patient

First Name	Enter first name...
Last Name	Enter last name...
Hospital no.	Enter Hospital no...
Date of birth	DD/MM/YYYY
Sex	Male Female

Search disease

Disease	Score	Diagnosis
Hepatites a & e	25	☐
Hepatites b & c	10	☐
Immunodeficiency	25	☐
Leginonnaires	20	☐
Palliative care	20	☐
Respiratory syncytial virus	25	☐
Scabies	20	☐
Scarlet fever	30	☐
Severe acute respiratory syndrome	40	☐

Fig 5: Add Patient Details Interface

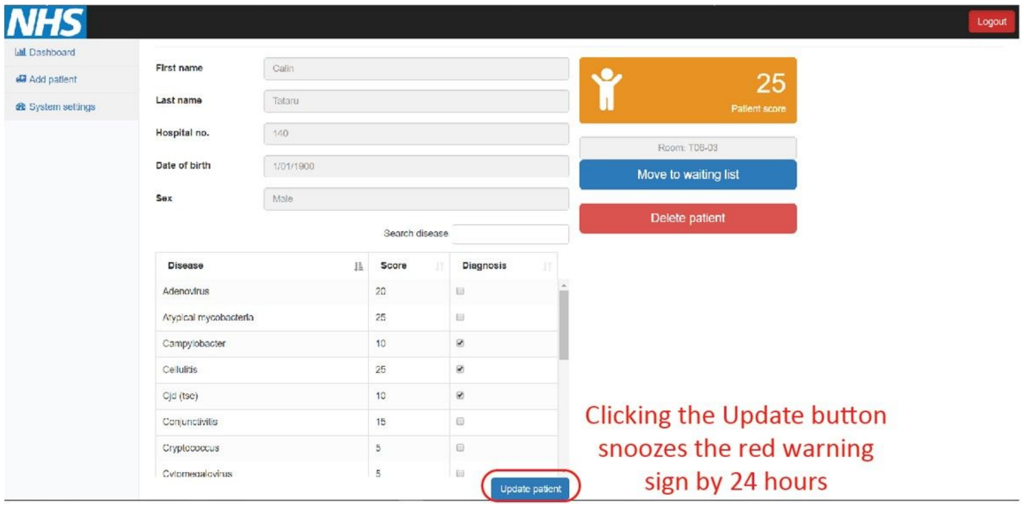

app.js
patients.js
routes > patients.js > router.post('/app/addpatient') callback

```javascript
const _ = require('lodash');
const router = express.Router();

var {scoreOfDisease, Disease} = require('./../server/models/diseases.js');
var {Patient} = require('./../server/models/patient.js');
var {rooms, Room} = require('./../server/models/rooms.js');
var isValidDate = require('is-valid-date');
const {ObjectID} = require('mongodb');


/*
    GET /app/addpatient -> go to addPatient page
*/
router.get('/app/addpatient', (req, res) => {
    res.render('addpatient', {pageTitle: "Add patient"});
});

/*
    POST /addPatient -> add new patient
*/
router.post('/app/addpatient', (req, res) => {
    // receive the diseases from the form in the array PD, each element being a String with the disease name
    var PD = req.body.PD;
    var dateOfBirth = req.body.dateOfBirth;


    if (_.isEmpty(PD)) {    // check if no disease is selected
        PD = [];
    }

    // Check for empty fields
    if (_.isEmpty(req.body.firstName) || _.isEmpty(req.body.lastName) || _.isEmpty(req.body.hospitalNumber) || !isValidDate(dateOfBirth)) {
        if (_.isEmpty(req.body.firstName)) req.flash('error_msg', 'Please enter the first name.');
        if (_.isEmpty(req.body.lastName)) req.flash('error_msg', 'Please enter the last name.');
        if (_.isEmpty(req.body.hospitalNumber)) req.flash('error_msg', 'Please enter the hospital number.');
        if (!isValidDate(dateOfBirth)) req.flash('error_msg', 'The date is not valid.');

        res.status(400).redirect('/app/addpatient');
    } else {
```

Fig 6: Add Patient Details Node JS Snippet

iv. <u>Doctors Module</u>:

A doctor gets his own personal view for the patients under his care. If the patient's medical status hasn't been updated in 24 hours, a red alert will appear next to the patient's name.

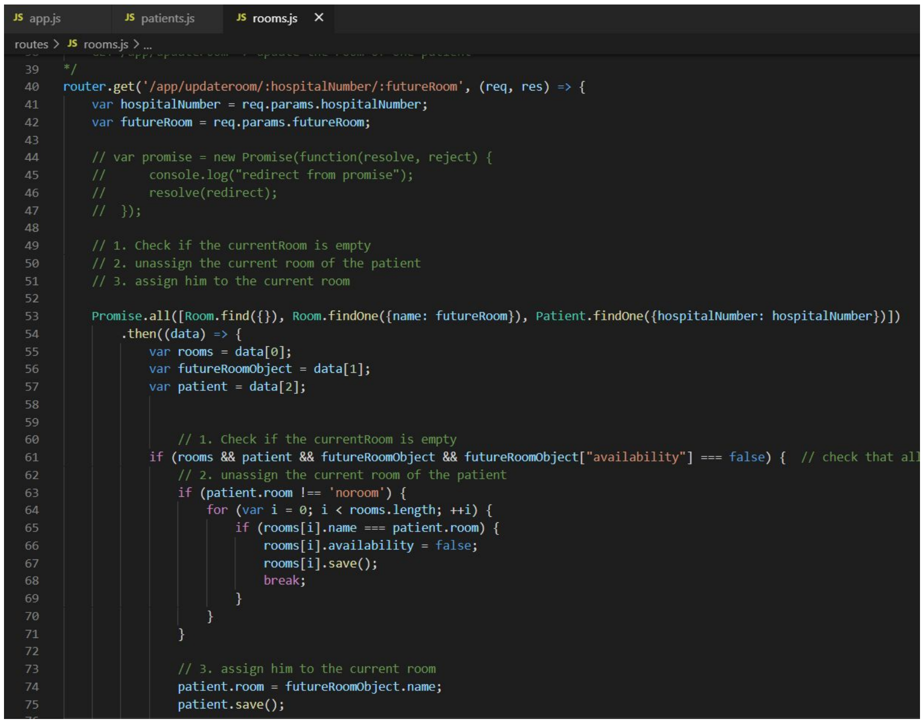


First name	Calin
Last name	Tataru
Hospital no.	140
Date of birth	1/01/1900
Sex	Male

Search disease

Disease	Score	Diagnosis
Adenovirus	20	☐
Atypical mycobacteria	25	☐
Campylobacter	10	✓
Cellulitis	25	✓
Cjd (tsc)	10	✓
Conjunctivitis	15	☐
Cryptococcus	5	☐
Cytomegalovirus	5	☐

Fig 7: Patient Details View interface

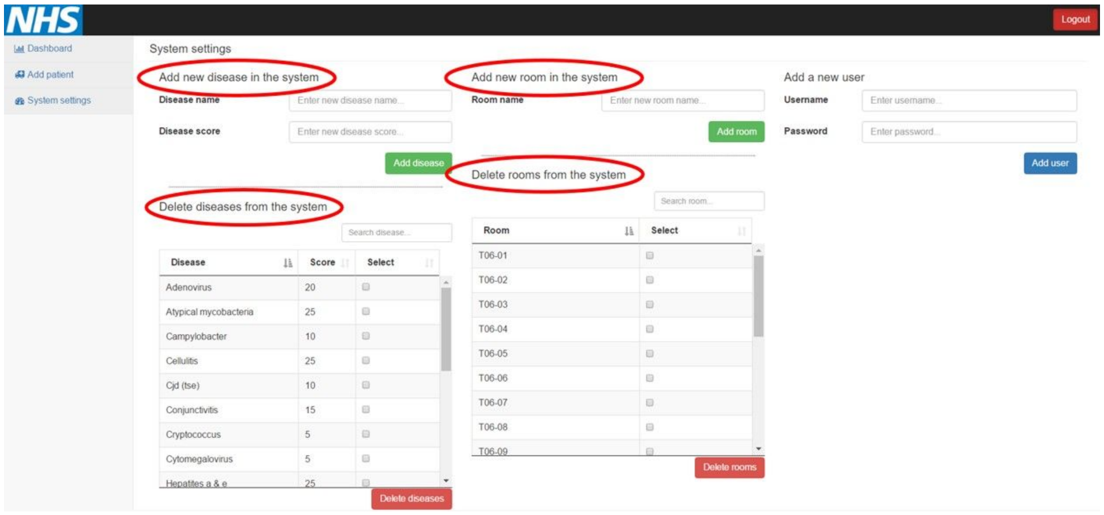


```javascript
*/
router.get('/app/updateroom/:hospitalNumber/:futureRoom', (req, res) => {
    var hospitalNumber = req.params.hospitalNumber;
    var futureRoom = req.params.futureRoom;

    // var promise = new Promise(function(resolve, reject) {
    //      console.log("redirect from promise");
    //      resolve(redirect);
    //  });

    // 1. Check if the currentRoom is empty
    // 2. unassign the current room of the patient
    // 3. assign him to the current room

    Promise.all([Room.find({}), Room.findOne({name: futureRoom}), Patient.findOne({hospitalNumber: hospitalNumber})])
        .then((data) => {
            var rooms = data[0];
            var futureRoomObject = data[1];
            var patient = data[2];


                // 1. Check if the currentRoom is empty
            if (rooms && patient && futureRoomObject && futureRoomObject["availability"] === false) {  // check that all
                // 2. unassign the current room of the patient
                if (patient.room !== 'noroom') {
                    for (var i = 0; i < rooms.length; ++i) {
                        if (rooms[i].name === patient.room) {
                            rooms[i].availability = false;
                            rooms[i].save();
                            break;
                        }
                    }
                }

                // 3. assign him to the current room
                patient.room = futureRoomObject.name;
                patient.save();
```

Fig 8: Patient Details Node JS Snippet

v. <u>**Settings Module**</u>:

Add a new category of diagnosis, add/remove a room from the database (only accessible by the hospital administrators).



System settings

Add new disease in the system

Disease name	Enter new disease name...
Disease score	Enter new disease score...

Add new room in the system

Room name	Enter new room name...

Add a new user

Username	Enter username...
Password	Enter password...

Delete diseases from the system

Search disease...

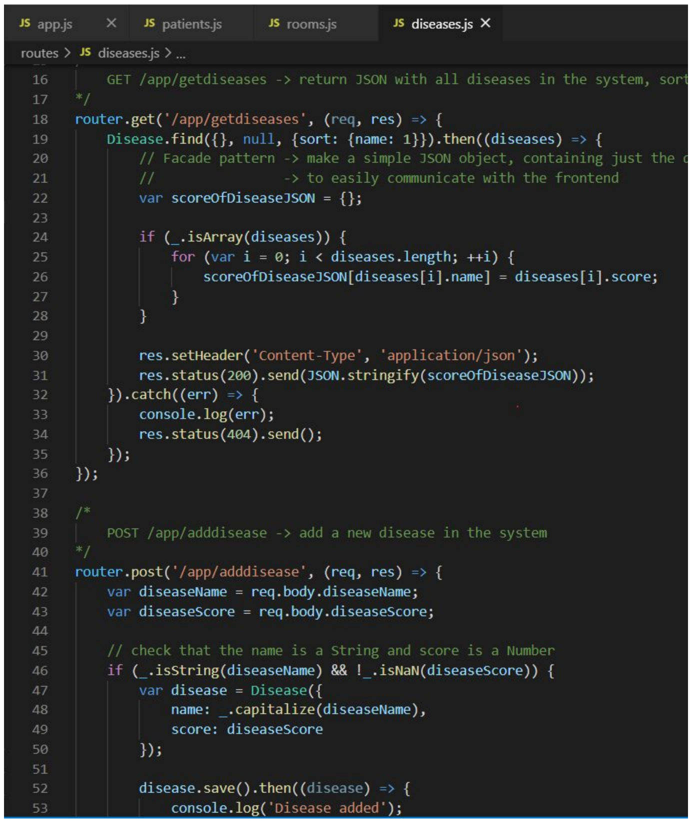
Disease	Score	Select
Adenovirus	20	☐
Atypical mycobacteria	25	☐
Campylobacter	10	☐
Cellulitis	25	☐
Cjd (tse)	10	☐
Conjunctivitis	15	☐
Cryptococcus	5	☐
Cytomegalovirus	5	☐
Hepatites a & e	25	☐

Delete rooms from the system

Search room...

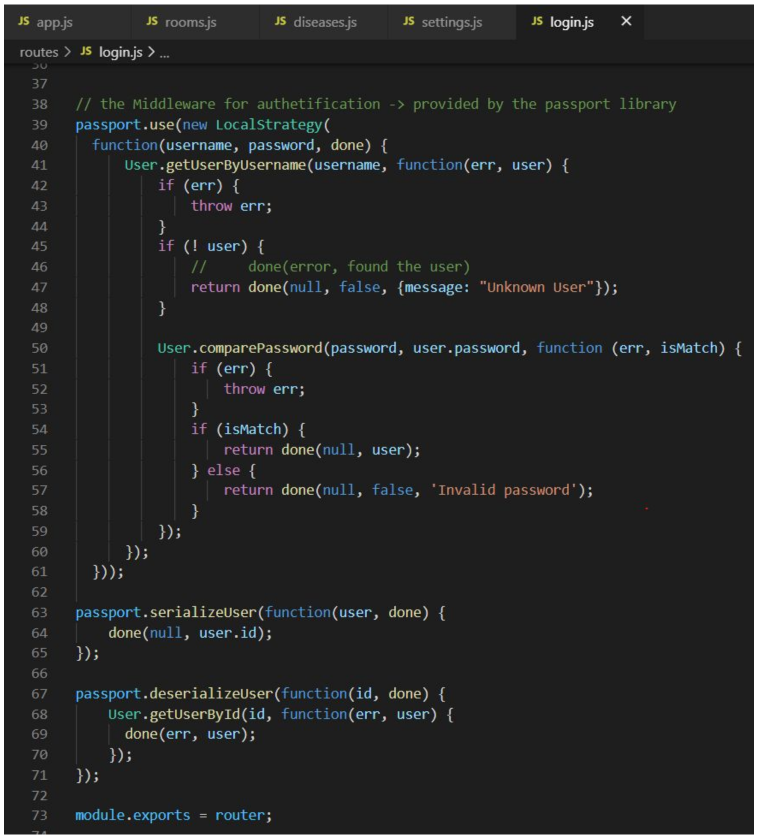
Room	Select
T06-01	☐
T06-02	☐
T06-03	☐
T06-04	☐
T06-05	☐
T06-06	☐
T06-07	☐
T06-08	☐
T06-09	☐

Fig 9: Settings Interface



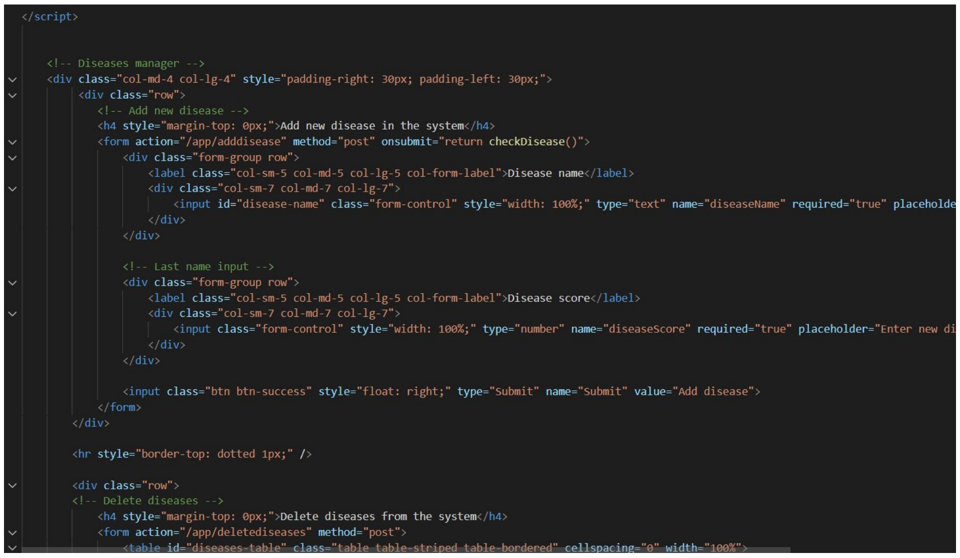
```
    GET /app/getdiseases -> return JSON with all diseases in the system, sort
*/
router.get('/app/getdiseases', (req, res) => {
    Disease.find({}, null, {sort: {name: 1}}).then((diseases) => {
        // Facade pattern -> make a simple JSON object, containing just the
        //                -> to easily communicate with the frontend
        var scoreOfDiseaseJSON = {};

        if (_.isArray(diseases)) {
            for (var i = 0; i < diseases.length; ++i) {
                scoreOfDiseaseJSON[diseases[i].name] = diseases[i].score;
            }
        }

        res.setHeader('Content-Type', 'application/json');
        res.status(200).send(JSON.stringify(scoreOfDiseaseJSON));
    }).catch((err) => {
        console.log(err);
        res.status(404).send();
    });
});

/*
    POST /app/adddisease -> add a new disease in the system
*/
router.post('/app/adddisease', (req, res) => {
    var diseaseName = req.body.diseaseName;
    var diseaseScore = req.body.diseaseScore;

    // check that the name is a String and score is a Number
    if (_.isString(diseaseName) && !_.isNaN(diseaseScore)) {
        var disease = Disease({
            name: _.capitalize(diseaseName),
            score: diseaseScore
        });

        disease.save().then((disease) => {
            console.log('Disease added');
```

Fig 10: Disease Interface Node JS Snippet



```
// the Middleware for authetification -> provided by the passport library
passport.use(new LocalStrategy(
  function(username, password, done) {
      User.getUserByUsername(username, function(err, user) {
          if (err) {
              throw err;
          }
          if (! user) {
              //     done(error, found the user)
              return done(null, false, {message: "Unknown User"});
          }

          User.comparePassword(password, user.password, function (err, isMatch) {
              if (err) {
                  throw err;
              }
              if (isMatch) {
                  return done(null, user);
              } else {
                  return done(null, false, 'Invalid password');
              }
          });
      });
  }));

passport.serializeUser(function(user, done) {
    done(null, user.id);
});

passport.deserializeUser(function(id, done) {
    User.getUserById(id, function(err, user) {
        done(err, user);
    });
});

module.exports = router;
```

Fig 11: User Login Node JS Snippet



```
</script>


    <!-- Diseases manager -->
    <div class="col-md-4 col-lg-4" style="padding-right: 30px; padding-left: 30px;">
        <div class="row">
            <!-- Add new disease -->
            <h4 style="margin-top: 0px;">Add new disease in the system</h4>
            <form action="/app/adddisease" method="post" onsubmit="return checkDisease()">
                <div class="form-group row">
                    <label class="col-sm-5 col-md-5 col-lg-5 col-form-label">Disease name</label>
                    <div class="col-sm-7 col-md-7 col-lg-7">
                        <input id="disease-name" class="form-control" style="width: 100%;" type="text" name="diseaseName" required="true" placeholde
                    </div>
                </div>

                <!-- Last name input -->
                <div class="form-group row">
                    <label class="col-sm-5 col-md-5 col-lg-5 col-form-label">Disease score</label>
                    <div class="col-sm-7 col-md-7 col-lg-7">
                        <input class="form-control" style="width: 100%;" type="number" name="diseaseScore" required="true" placeholder="Enter new di
                    </div>
                </div>

                <input class="btn btn-success" style="float: right;" type="Submit" name="Submit" value="Add disease">
            </form>
        </div>

        <hr style="border-top: dotted 1px;" />

        <div class="row">
        <!-- Delete diseases -->
            <h4 style="margin-top: 0px;">Delete diseases from the system</h4>
            <form action="/app/deletediseases" method="post">
                <table id="diseases-table" class="table table-striped table-bordered" cellspacing="0" width="100%">
```

Fig 12: Room Details Handlebar Snippet

UML Diagrams

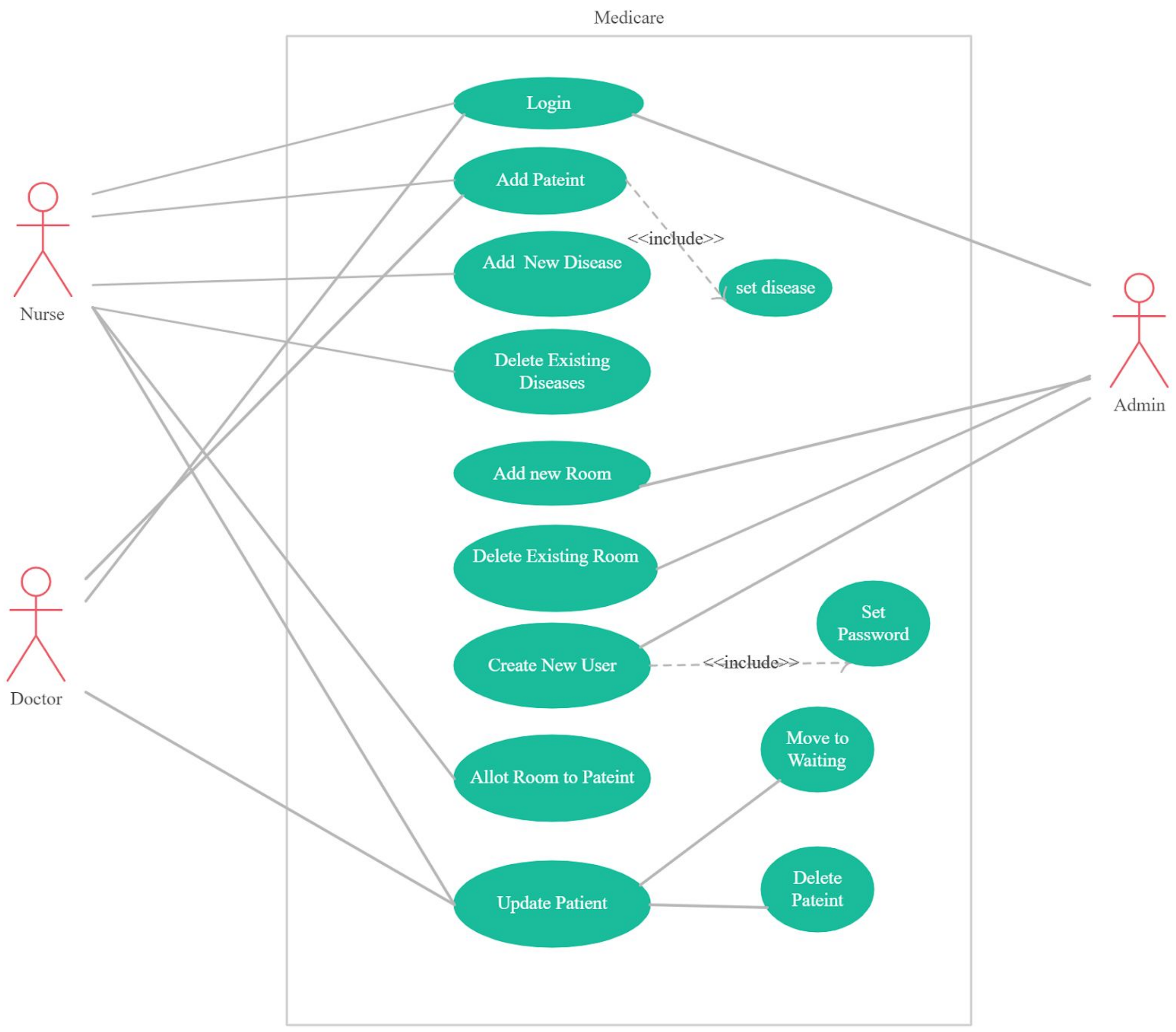


Fig 13.1 Use Case Diagram

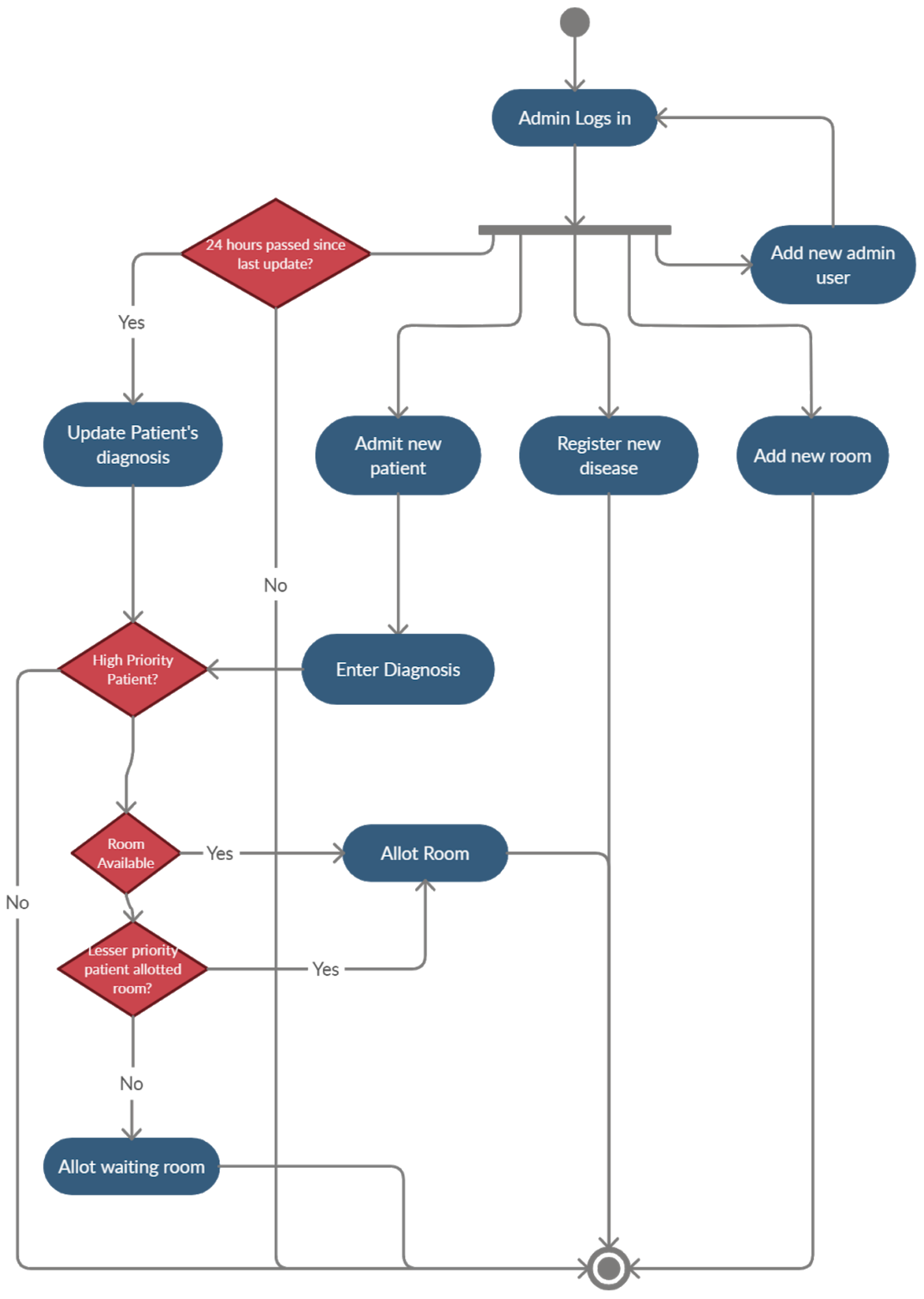


Fig 13.2: Activity Diagram

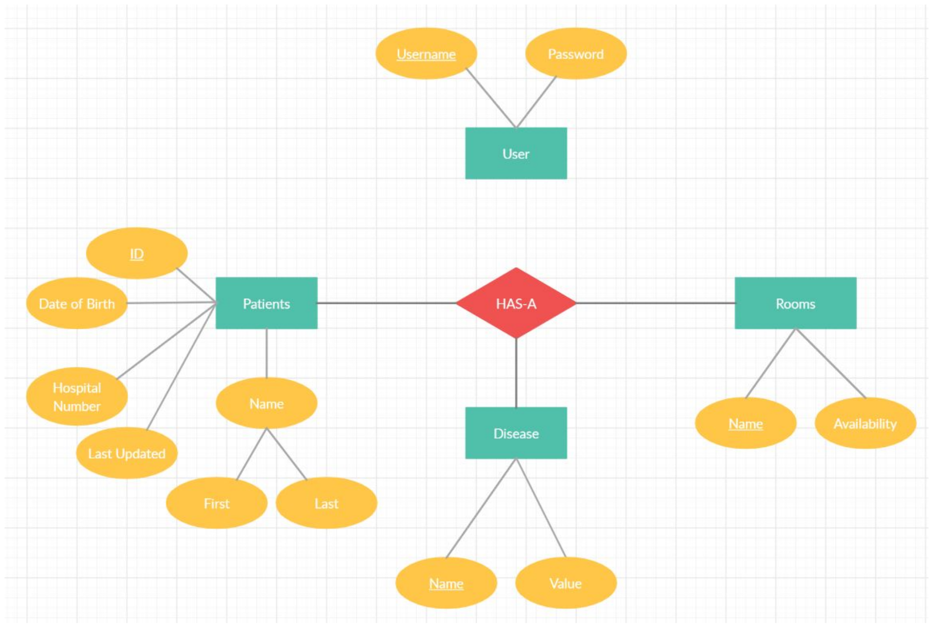
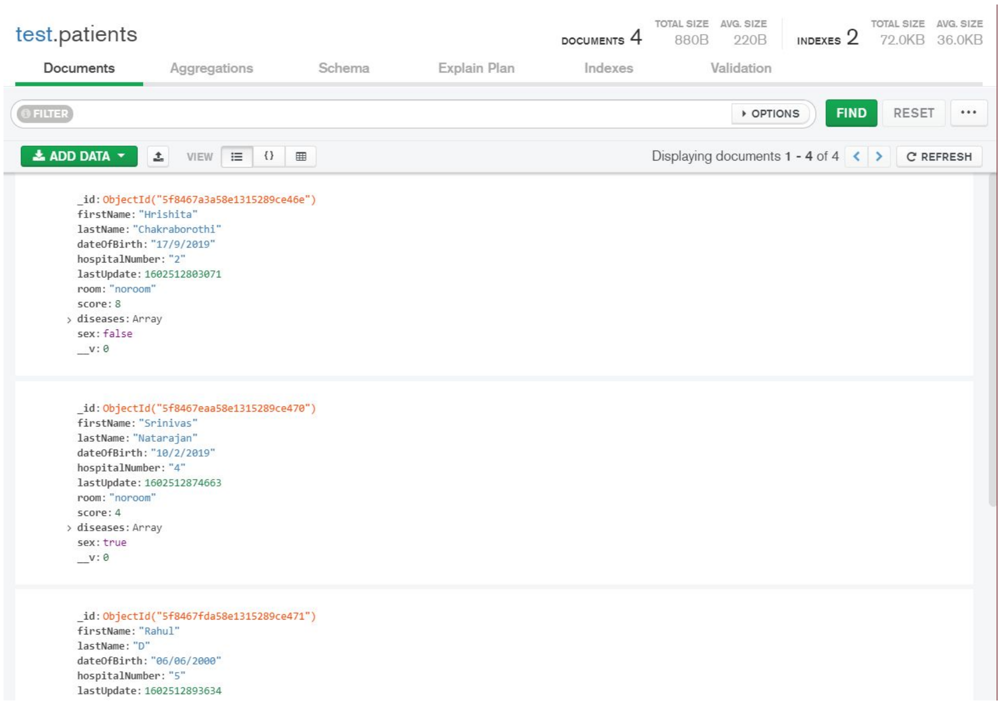


Fig 13.3: ER diagram of DB

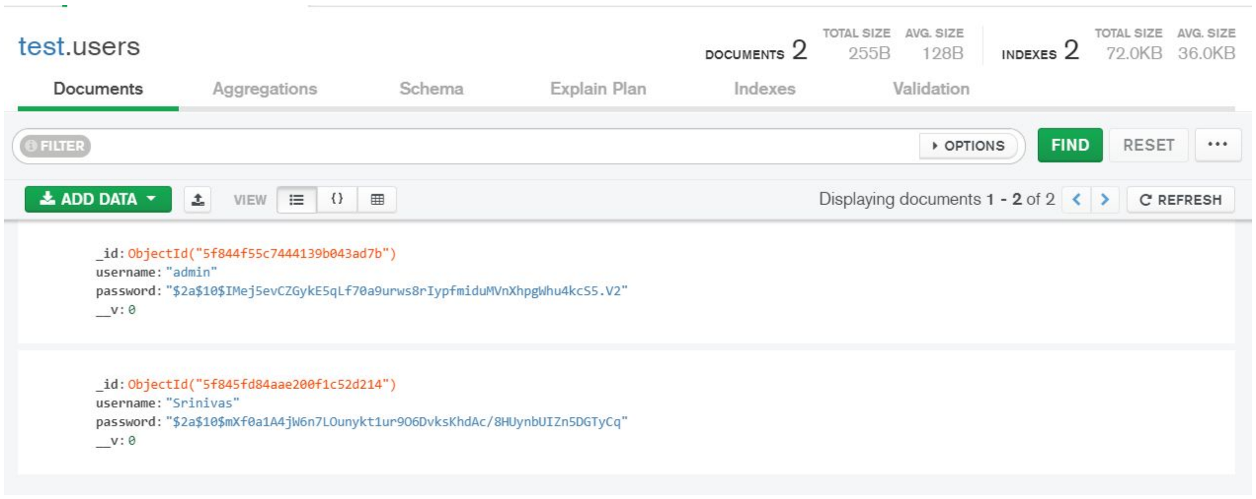
<u>Results</u>

We have used MongoDB as our Database Management System. Using MongoDB Compass we are able to view and manipulate our website's data using a user-friendly GUI interface. We have 4 tables (here, collections) for our project namely Admin Users, Patients, Diseases and Hospital Rooms. The data used in this prototype are all dummy data for a fictional hospital (as seen in figs 13-16) to simulate a real-life hospital management situation. The relations between the 4 collections has been illustrated in the ER diagram (fig 13.3)



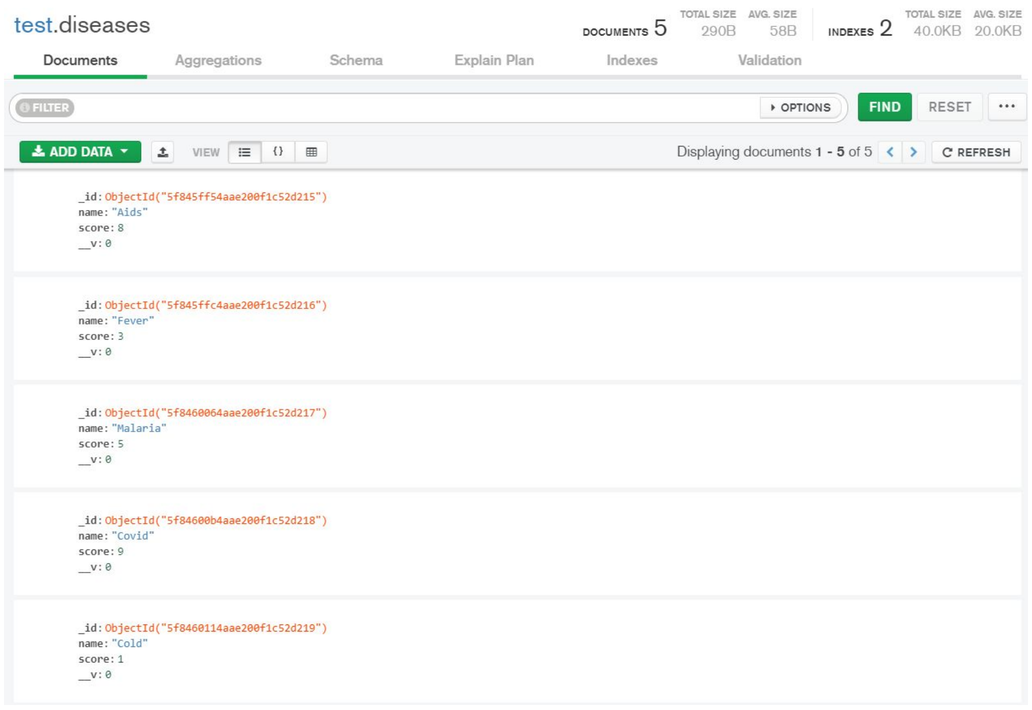
```
_id: ObjectId("5f8467a3a58e1315289ce46e")
firstName: "Hrishita"
lastName: "Chakraborothi"
dateOfBirth: "17/9/2019"
hospitalNumber: "2"
lastUpdate: 1602512803071
room: "noroom"
score: 8
> diseases: Array
sex: false
__v: 0

_id: ObjectId("5f8467eaa58e1315289ce470")
firstName: "Srinivas"
lastName: "Natarajan"
dateOfBirth: "10/2/2019"
hospitalNumber: "4"
lastUpdate: 1602512874663
room: "noroom"
score: 4
> diseases: Array
sex: true
__v: 0

_id: ObjectId("5f8467fda58e1315289ce471")
firstName: "Rahul"
lastName: "D"
dateOfBirth: "06/06/2000"
hospitalNumber: "5"
lastUpdate: 1602512893634
```

Fig 14: Patients Database



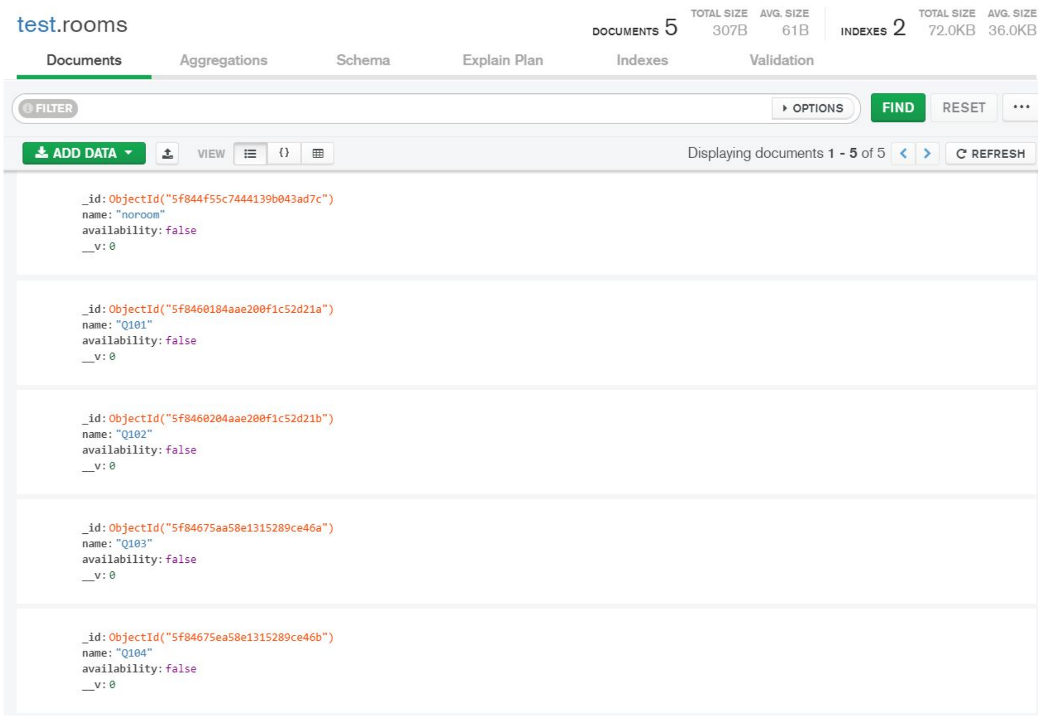
```
_id: ObjectId("5f844f55c7444139b043ad7b")
username: "admin"
password: "$2a$10$IMej5evCZGykE5qLf70a9urws8rIypfmiduMVnXhpgWhu4kcS5.V2"
__v: 0

_id: ObjectId("5f845fd84aae200f1c52d214")
username: "Srinivas"
password: "$2a$10$mXf0a1A4jW6n7LOunykt1ur9O6DvksKhdAc/8HUynbUIZn5DGTyCq"
__v: 0
```

Fig 15: User Database



```
_id: ObjectId("5f845ff54aae200f1c52d215")
name: "Aids"
score: 8
__v: 0

_id: ObjectId("5f845ffc4aae200f1c52d216")
name: "Fever"
score: 3
__v: 0

_id: ObjectId("5f8460064aae200f1c52d217")
name: "Malaria"
score: 5
__v: 0

_id: ObjectId("5f84600b4aae200f1c52d218")
name: "Covid"
score: 9
__v: 0

_id: ObjectId("5f8460114aae200f1c52d219")
name: "Cold"
score: 1
__v: 0
```

Fig 16: Diseases Database



```
_id: ObjectId("5f844f55c7444139b043ad7c")
name: "noroom"
availability: false
__v: 0

_id: ObjectId("5f8460184aae200f1c52d21a")
name: "Q101"
availability: false
__v: 0

_id: ObjectId("5f8460204aae200f1c52d21b")
name: "Q102"
availability: false
__v: 0

_id: ObjectId("5f84675aa58e1315289ce46a")
name: "Q103"
availability: false
__v: 0

_id: ObjectId("5f84675ea58e1315289ce46b")
name: "Q104"
availability: false
__v: 0
```

Fig 17: Rooms Database

Conclusion

The web application which we have developed can be used by hospitals to improve its management facilities. This application also keeps a close track on the number of beds and rooms in the hospital that are increased or decreased and also on the patients. It makes use of a priority scheduling algorithm based on the disease score that reduces the time for the allotment of bed and beginning of the treatment thereby reducing the delay in treatment.

We believe this application will prove to be of great use during the ongoing pandemic and also in the future. In the rural areas there are still many hospitals which maintain their records in registers which are difficult to maintain and are also not secure, our application provides not only security but also makes the management easier and more effective. The application is only a prototype and many features can be added to it in future to make it more effective.

References

[1] B. Chowdhury and R. Khosla, "RFID-based Hospital Real-time Patient Management System," 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), Melbourne, Qld., 2007, pp. 363-368, doi: 10.1109/ICIS.2007.159.

[2] M. Thangaraj, P. P. Ponmalar and S. Anuradha, "Internet Of Things (IOT) enabled smart autonomous hospital management system - A real world health care use case with the technology drivers," 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, 2015, pp. 1-8, doi: 10.1109/ICCIC.2015.7435678.

[3] Cao, Yubin, Li, Qin, Chen, Jing, "Hospital Emergency Management Plan During the COVID-19 Epidemic" Academic Emergency Medicine, Acad Emerg Med, Volume 27 Issue 4 2016, doi:10.1111/acem.13951

[4] Adebisi O.A, Oladosu D.A, Busari O.A and Oyewola Y.V Department of Computer Engineering Technology, The Polytechnic, Ibadan."Design and

Implementation of Hospital Management System", International Journal of Engineering and Innovative Technology (IJEIT) Volume 5, Issue 1, July 2015

[5] Si, S.-L.; You, X.-Y.; Liu, H.-C.; Huang, J. Identifying Key Performance Indicators for Holistic Hospital Management with a Modified DEMATEL Approach. Int. J. Environ. Res. Public Health 2017, 14, 934.

[6] R. M. Liaqat, A. Athar and N. A. Saqib, "Intelligent Agent Based System for Monitoring and Control of Hospital Management System," 2015 2nd International Conference on Information Science and Security (ICISS), Seoul, 2015, pp. 1-5, doi: 10.1109/ICISSEC.2015.7371009.

[7] P.W. Handayani, A.N. Hidayanto, A.A. Pinem, I.C. Hapsari, P.I. Sandhyaduhita, I. Budi, "Acceptance model of a Hospital Information System", International Journal of Medical Informatics, Volume 99, 2017, Pages 11-28, ISSN 1386-5056, https://doi.org/10.1016/j.ijmedinf.2016.12.004.

[8] Thomas C. Tsai, Ashish K. Jha, Atul A. Gawande, Robert S. Huckman, Nicholas Bloom, and Raffaella Sadun, "Hospital Board And Management Practices Are Strongly Related To Hospital Performance On Clinical Quality Metrics", Health Affairs,Volume 34, Number 8, https://doi.org/10.1377/hlthaff.2014.1282