NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY
BANGALORE

An Autonomous Institution With A+ Grade By NAAC UGC | Approved by UGC / AICTE / Govt. of Karnataka | Bengaluru

WEB - BIG DATA Course Project Report

Submitted By,
Praneeth MVLSSS (1NT19IS112)
Pratheek P Nayak (1NT19IS117)
Rajesh CR (1NT19IS125)

Submitted To,
Mrs. Swetha R
Ms. Akarsh DP

Abstract:

Hospitals are a very large sector in any economy. People get sick commonly and get admitted to hospitals. Some hospitals even get hundreds and thousands of admissions within a week. This makes manual management and allotment of rooms to the patients a very hard job. So a very useful system or a software is required to manage all the data of the patients and their information. This makes the job of the hospital workers easier and also saves a lot of time.

Introduction:

Hospitals have unique data requirements. They have to maintain a lot of patient data on a day to day basis. The details include patients' information, what kind of disease they have and whether or not they have been allotted rooms. So to meet the needs of the hospitals' data management problems we are creating a web application using MongoDB and NodeJS. Through the application we will be able to add, delete, update the details of the patients. Also we can add new users to the system as well.

Literature Survey:

Big Data in Healthcare Management: A Review of Literature

Link: https://www.researchgate.net/publication/326957164_Big_Data_in_Healthcare_Management_A_Review_of_Literature

A literature review of current technologies on health data integration for patient-centered health management:

Link: https://journals.sagepub.com/doi/10.1177/1460458219892387

Objectives:

1. To create a web page for hospital management
2. To create login page
3. To create a system that will be able to add, delete and update patients info

4. To create a system that will allow user to add diseases and rooms into the system

Problem Statement:

Creating a web application based on MongoDB and NodeJS for Hospital Management.

Methodology:

Using MongoDB and NodeJS.

MongoDB is used as a database.

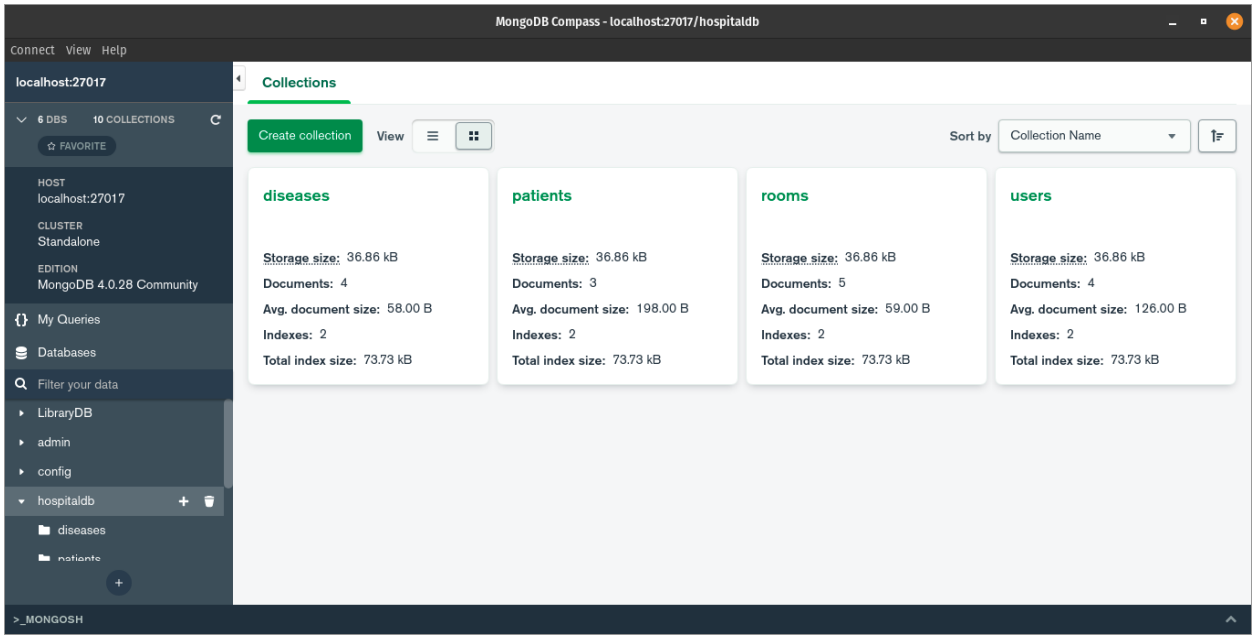
One database and we will have 4 collections namely,

Diseases

Patients

Rooms

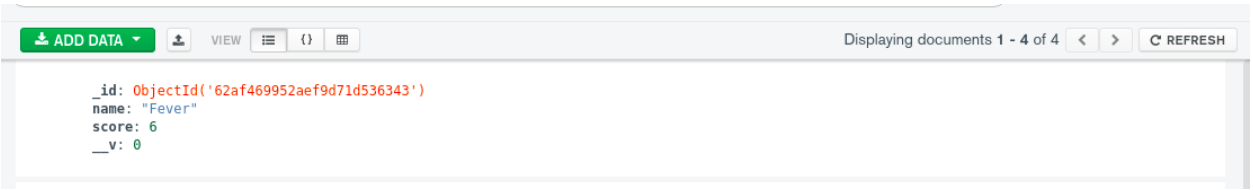
Users



As we can see in hospital db we have 4 collections

Collection 1: diseases

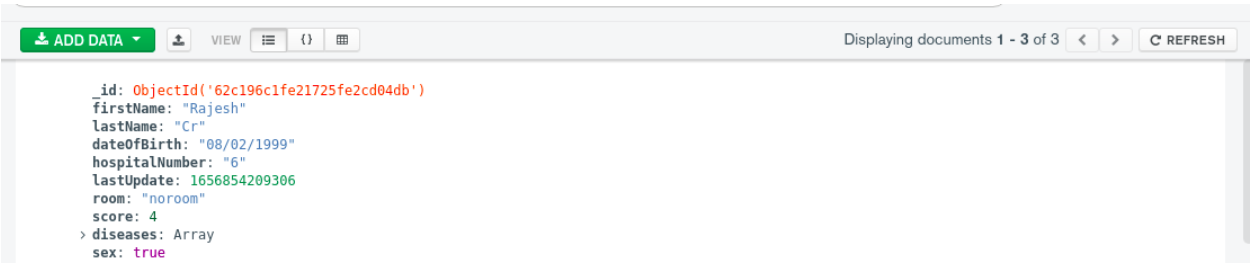
Here we will have a name and score. Let's look at how one document looks like in this collection



Collection 2: patients

Here we will have patient's firstName, lastName, dateOfBirth, hospitalNumber, room, score, disease, sex.

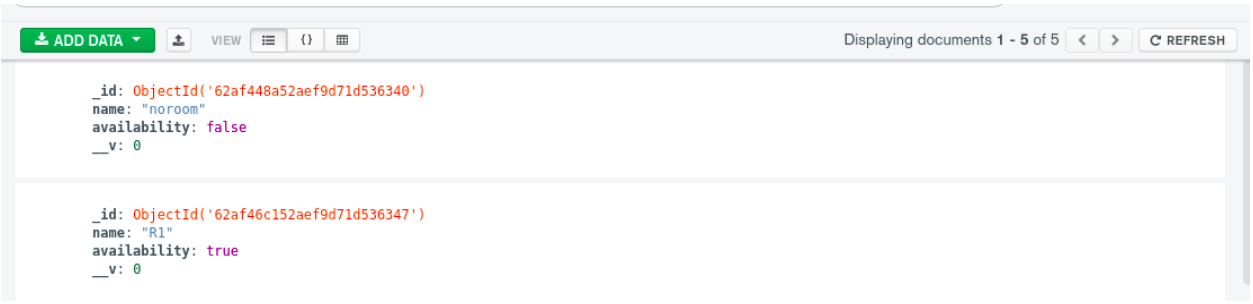
Let's look at a document:



Collection 3: rooms

Here we will have room number and its availability

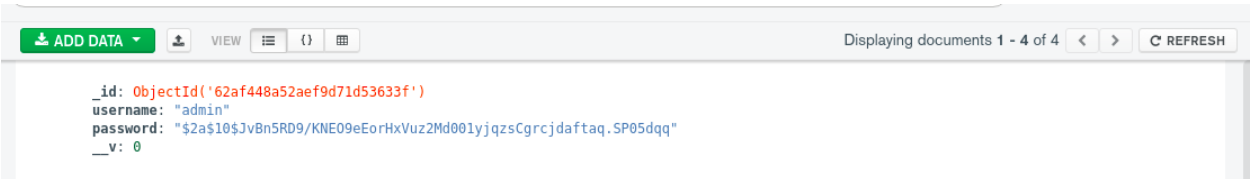
Let's look at one of the documents



Collection 4: users

In this collection we will have user's username and password stored

Let's look at one of the documents



As we can see that the password is hashed and stored.

Algorithm and Code Snippets:

Adding diseases, patients, rooms, users
Deleting diseases, patients, rooms, users
Updating diseases, patients, rooms, users

Code GitHub Link:
https://github.com/nayakpratheek17/HospitalManagement-WebBD/tree/main/routes
Adding and deleting diseases

```
/
*
        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');
                res.status(200).redirect('/app/systemsettings');
            }).catch((err) => {
                console.log(err);
                res.status(400).redirect('/app/systemsettings');
            });
        } else {
            res.status(400).redirect('/app/systemsettings',{messages:
    req.flash('success_msg', 'Succesful test') });
        }
    });


    /*
        POST /app/deletedisease -> delete a disease from the system
    */
    router.post('/app/deletediseases', (req, res) => {
        var diseasesToDelete = req.body.DD;


        if (_.isArray(diseasesToDelete)) {
            for (var i = 0; i < diseasesToDelete.length; ++i) {
                // 1. Delete the disease from the system
                var disease = diseasesToDelete[i];
                Disease.find({
                    name: diseasesToDelete[i]
                }).remove().catch((err) => {
                    console.log(err);
                });
```

```
                var promise = new Promise ((resolve, reject) => {
                     resolve(disease);
                     reject(disease);
                });
```

Adding Patients:

```
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;

                    // console.log(dateOfBirth);
                    // console.log(isValidDate(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 {
        // set the sex of the new patient
        var sex = req.body.sex;
        if (sex === "male") {
            sex = true;
        } else {
            sex = false;
        }


        // make a new patient and add it in the database
        var patient = Patient({
            firstName: _.capitalize(req.body.firstName),
            lastName: _.capitalize(req.body.lastName),
            sex: sex,
            dateOfBirth: dateOfBirth,
            hospitalNumber:
_.toUpper(req.body.hospitalNumber),
            diseases: PD,
            lastUpdate: (new Date().getTime())
        });


        patient.save().then((patient) => {
            patient.updateScore();
            res.status(200).redirect('/app');
        }).catch((err) => {
            console.log(err);
```

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

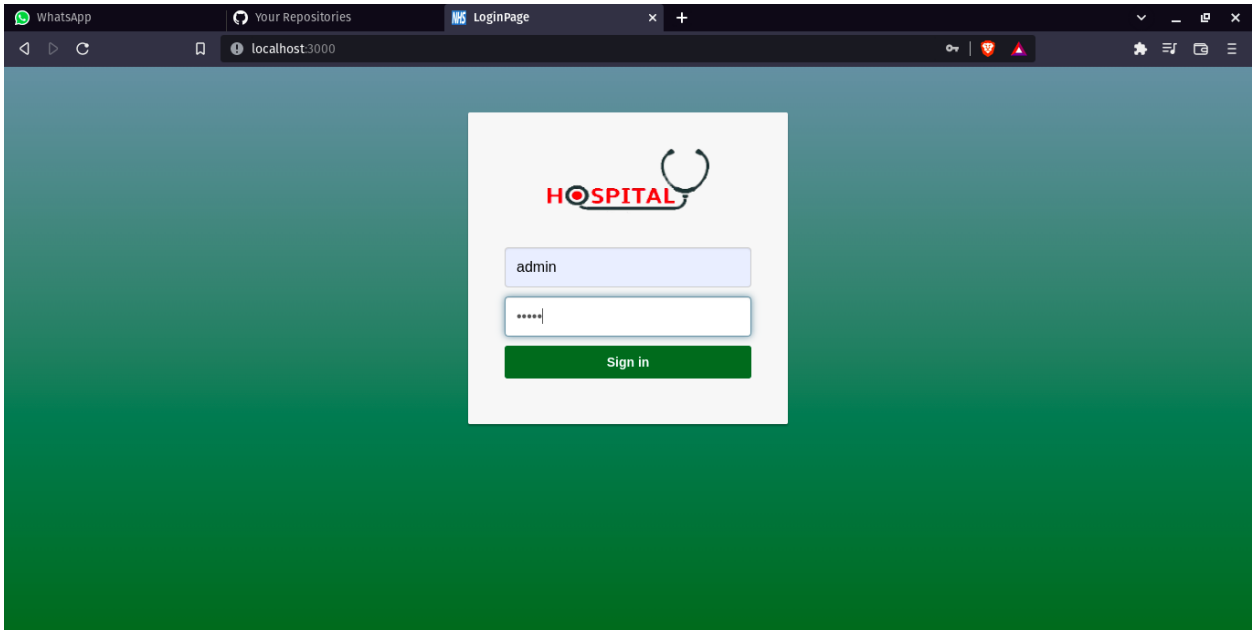
Adding New User:

```
router.post('/app/adduser',
(req, res) => {
                                    var username = req.body.username;
                                    var password = req.body.password;


                                // validation
                                    req.checkBody('username', 'Username is
                                required').notEmpty();
                                    req.checkBody('password', 'Password is
                                required').notEmpty();


                                // if there are errors, flash messages on the
                                screen
                                var errors = req.validationErrors();
                                if(errors) {


                                res.status(400).redirect('systemsettings');
                                } else {
                                    // if everything is OK, create a new user
                                in the database
                                    var newUser = new User({
                                        username,
                                        password
                                    });
```

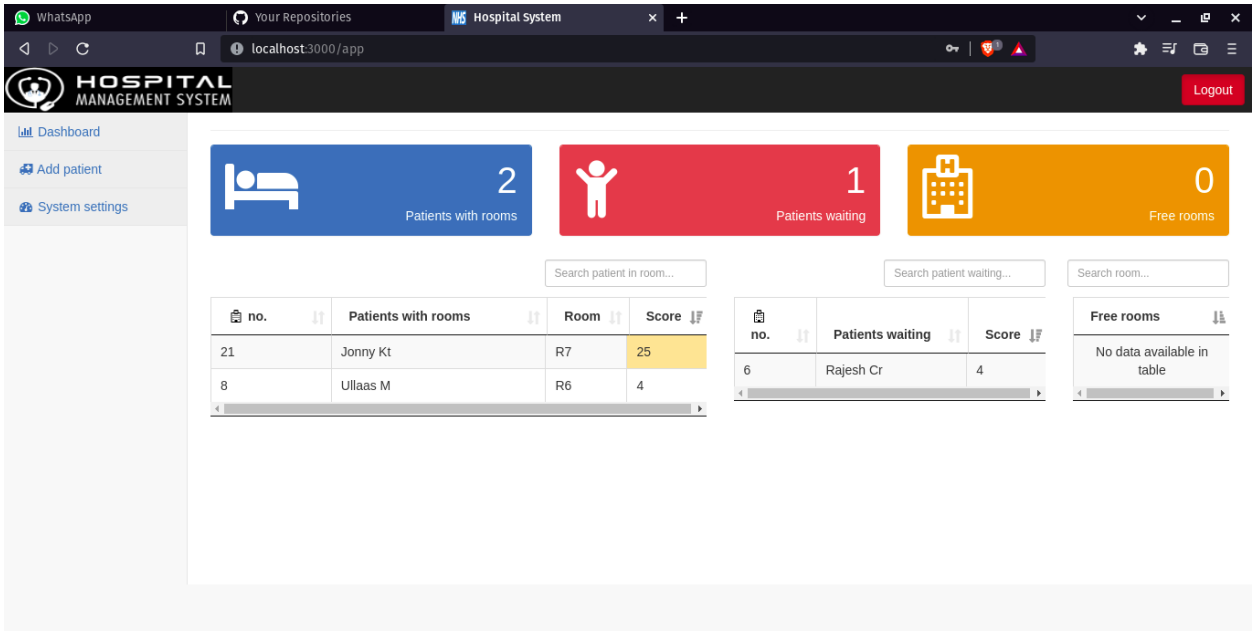
```
        User.createUser(newUser, function(err,
user) {
            if (err) {
                                console.log(err);
                return ;
            }
        });


        req.flash('success_msg', 'User succesfully
created');

res.status(200).redirect('/app/systemsettings');
    }
});
```

Models:
Database schema for all collections:
https://github.com/nayakpratheek17/HospitalManagement-WebBD/tree/main/models

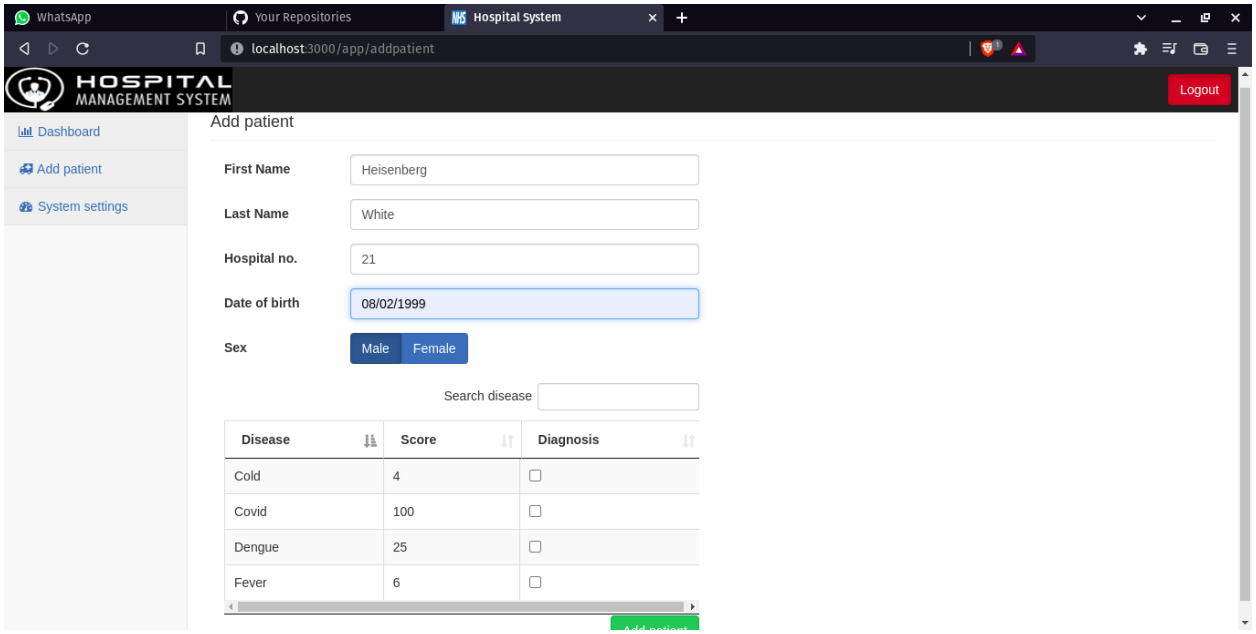
Output:
Login Page:



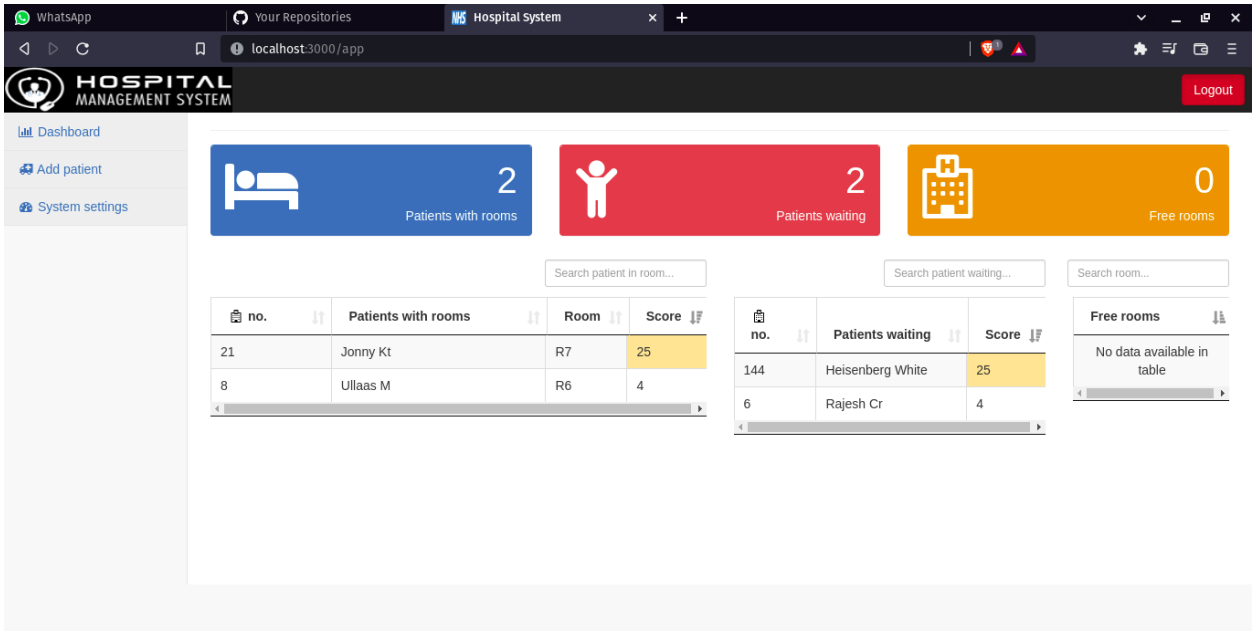
Main page:



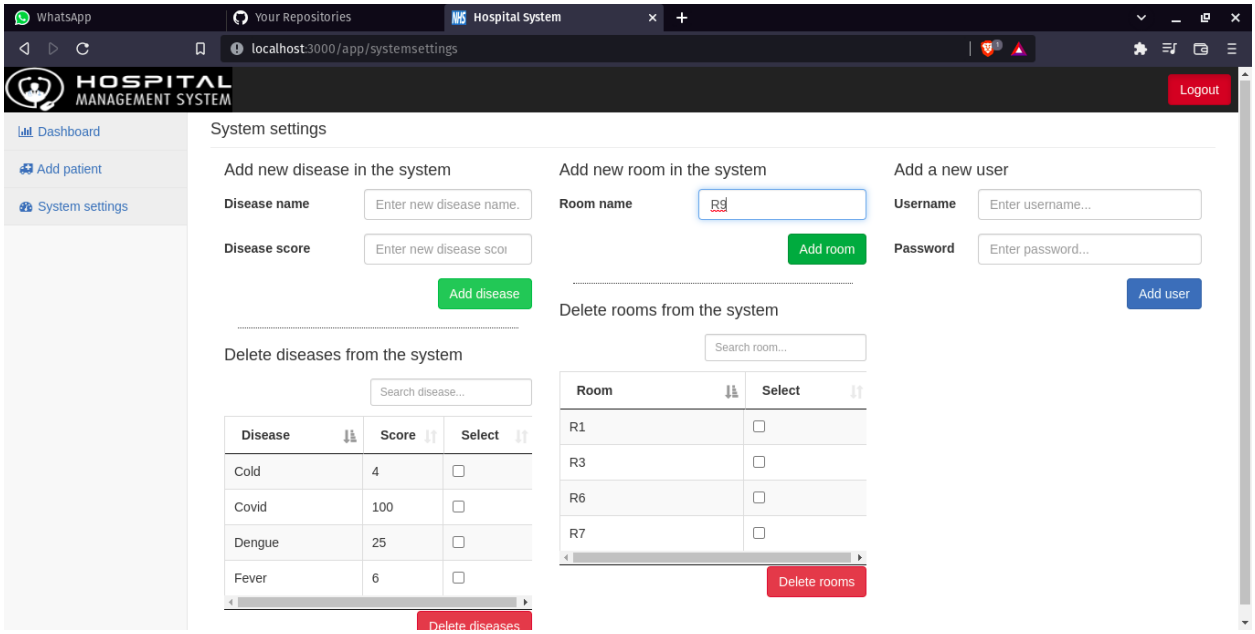
Adding Patient:



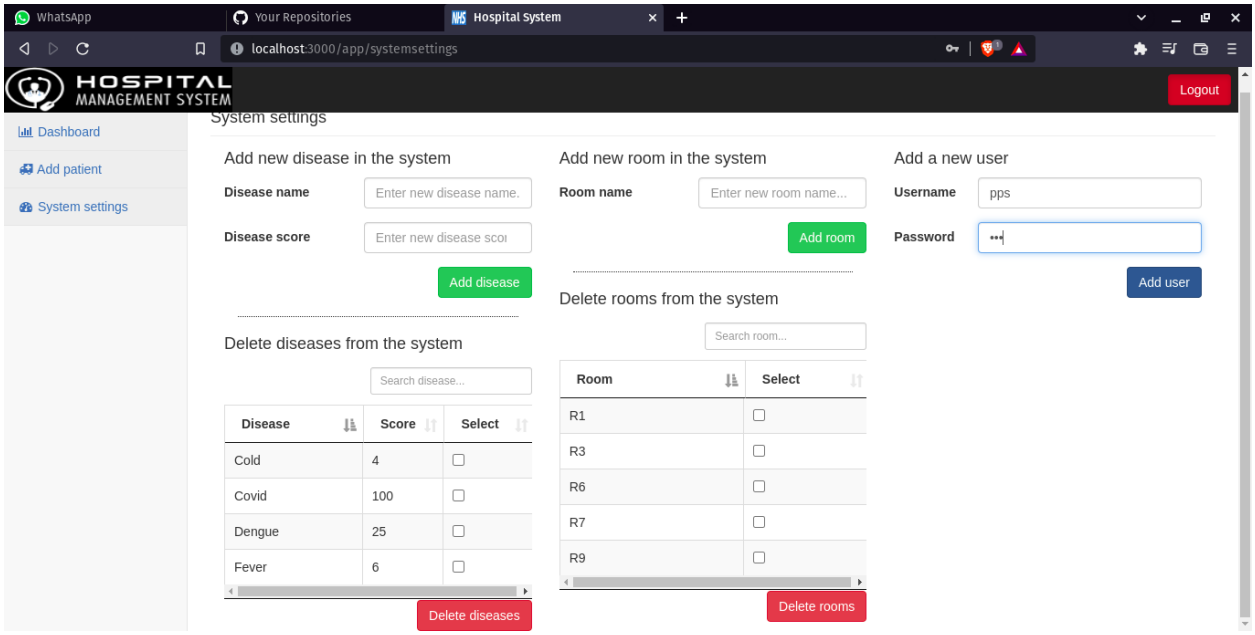
Patient reflected in main page:



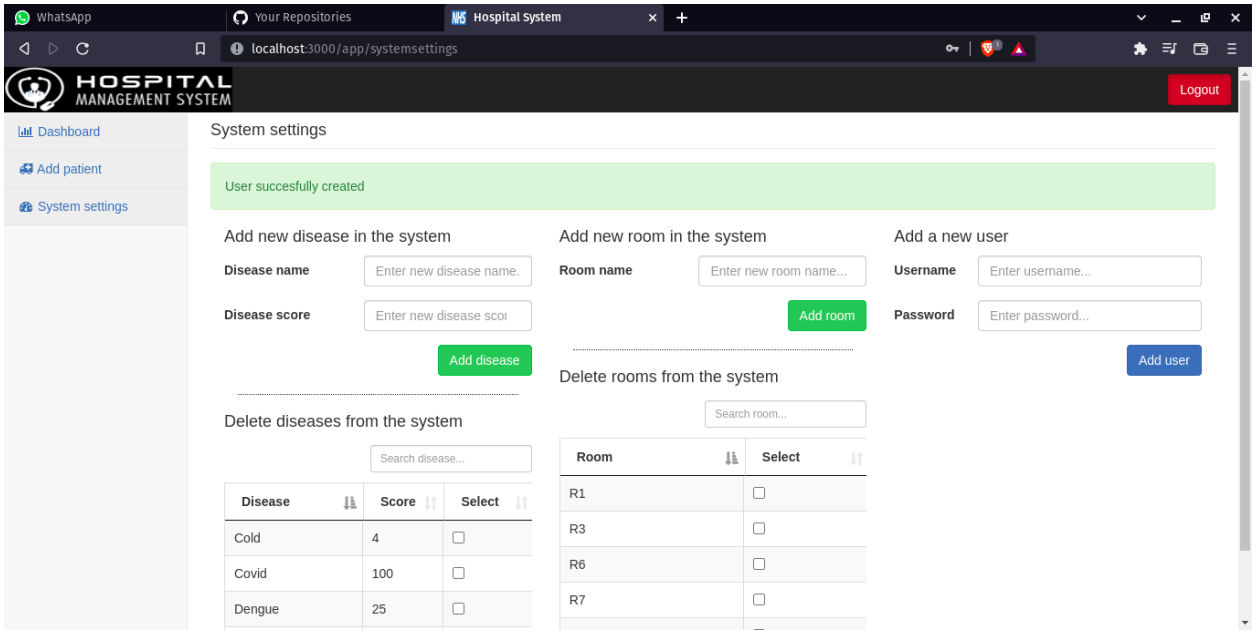
Adding New Room:



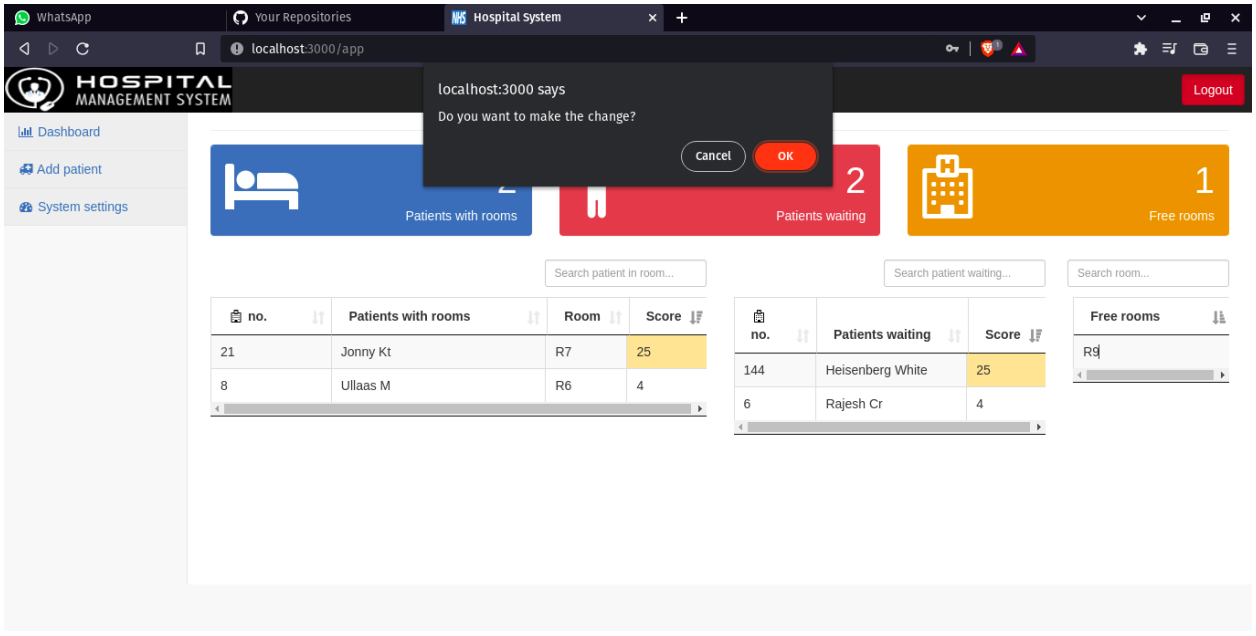
Adding new User:



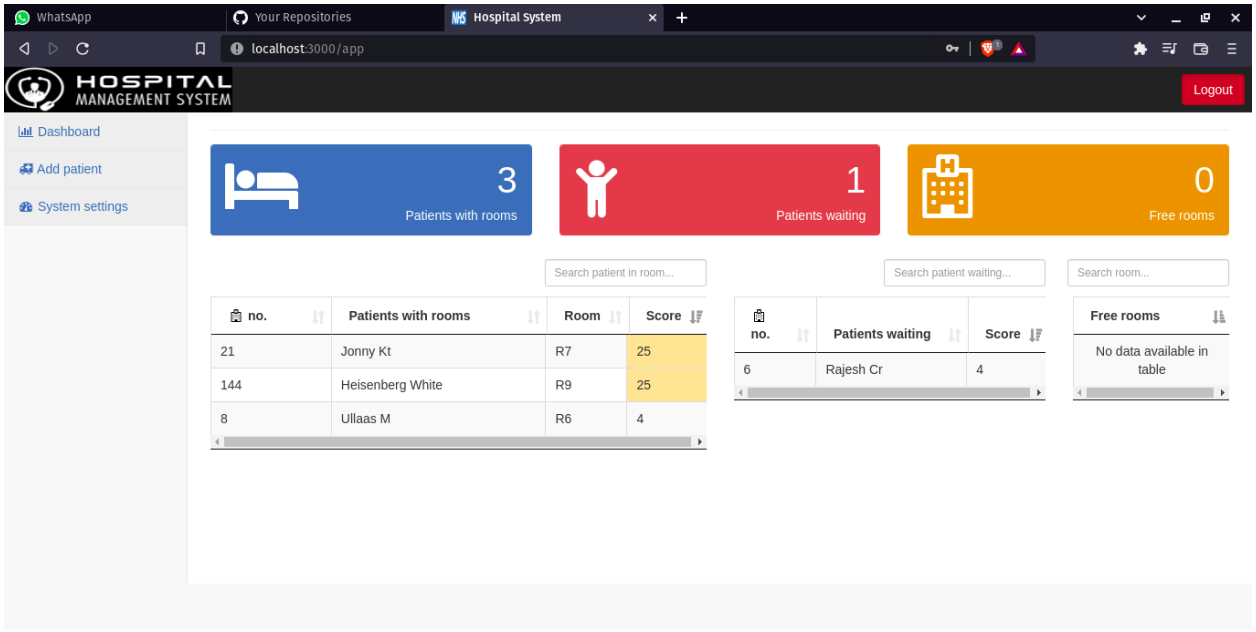
User successfully created:



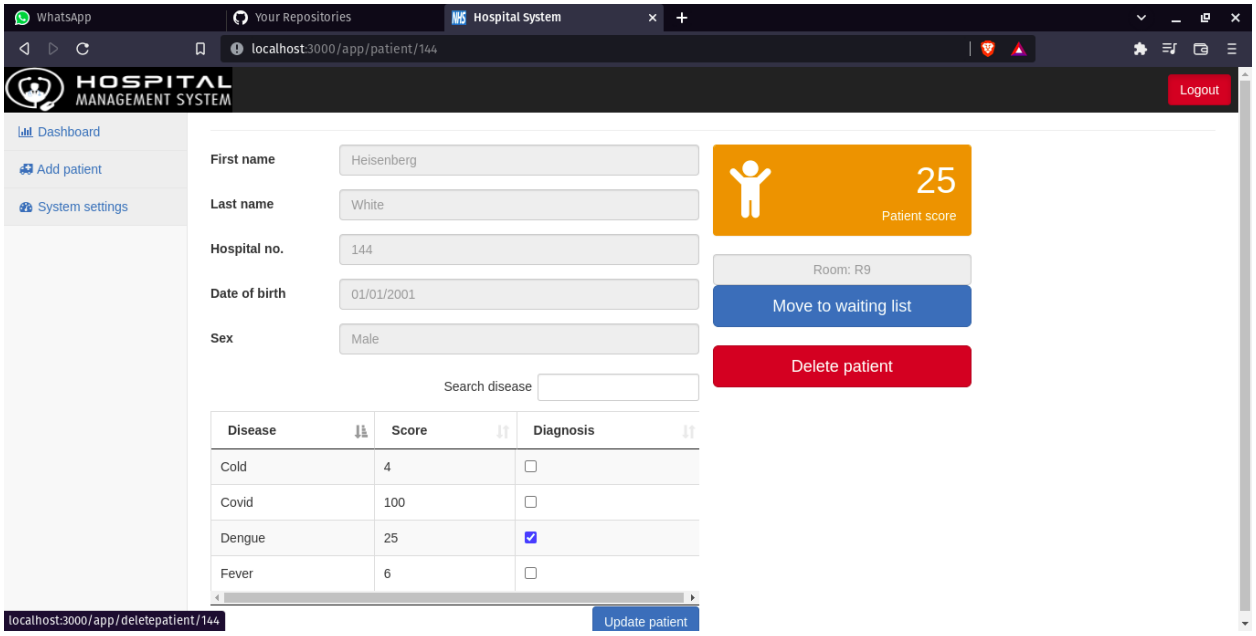
Allotting Patient Heisenberg with room R9



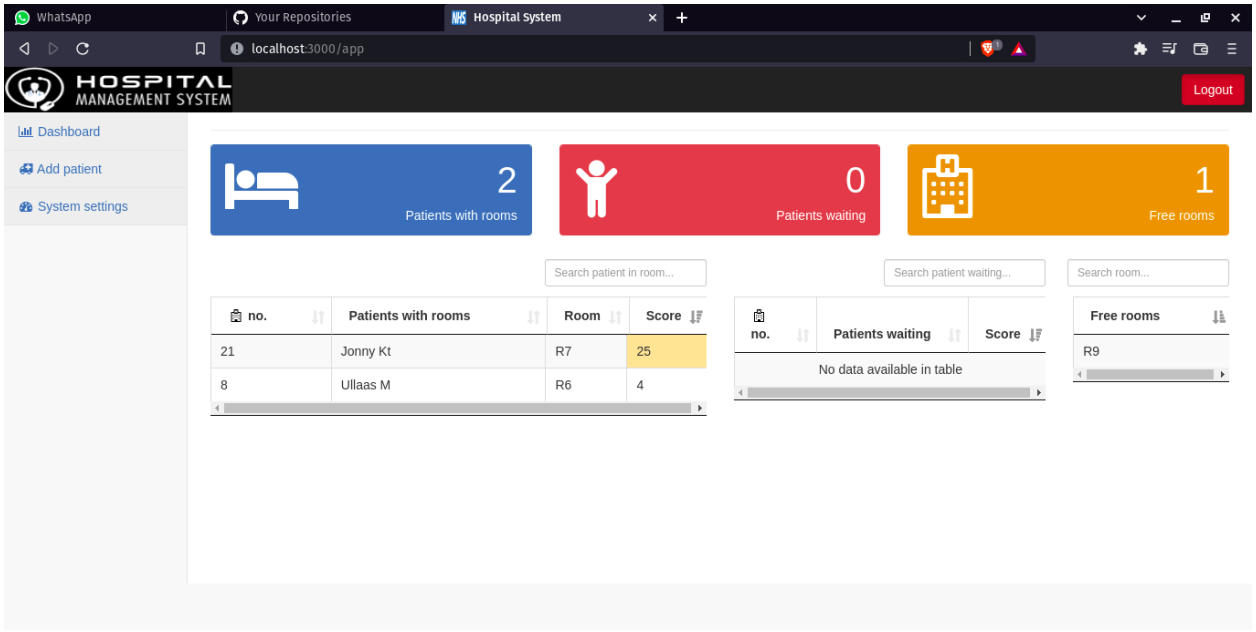
After allotting:



Deleting A Patient



Heisenberg deleted



In the settings page itself we can delete rooms, diseases all at a single place