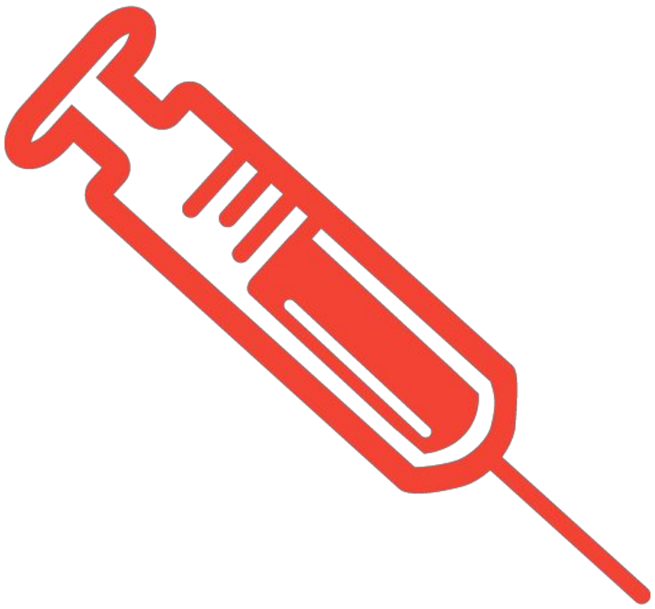
Interactive Patient Information Management System

Software Design Document



-Group 6-

-Group Leader-
Ryan Schachte

-Deputy Leader-
Katelyn Duffy

-Group Members-
Wade Kariniemi
Reman Koshaba
Kevin Duong
Minghao Wei(Ming)
Po-Kai(Jerry) Huang
Javier Peralta
Jezreel Garcia
Sang Le

September 28, 2015

TABLE OF CONTENTS

Section 1 - Introduction

Section 1.1 Purpose

The purpose of this Software Design Document will briefly explain the functionality and design of the Interactive Patient Information Management System. This document will give in detail the terms of User Interface Design, Data Design, and implementation of the Interactive Patient Information Management System. The following sections of this document will provide descriptions about Interactive Patient Information Management System.

Section 1.2 Scope

The Interactive Patient Information Management System is a web-based system that is designed to improve the current patient care system. It allows users to update and gain access to appointment schedules, health records, lab records, and patient health status information. The system allows patients to make appointments from their own home, with their doctor, through the internet. In addition, The Interactive Patient Information Management System contains many functionalities that can help healthcare providers easily manage patient information. Once the information is submitted, it will store into a database and become accessible to healthcare providers and patients for future use.

Section 1.3 Overview

The following section is an overview of the system for the Interactive Patient Information Management System. Here the system is briefly described to generalize the usage and functionality of the IPIMS. It also explains in terms of its design and the context of the system.

Following this section is the Architectural Design. Here the overall structure of the system is broken down into core components, and how they interact with each other. The system is depicted with the relationships between modules that bring about the functionality of the system. Responsibilities and roles of the system are explained as well.

The next transition will be the design of the system in terms of data, components, and human interface. This describes how the data is used. From there, the Component Design will explain how each component is used, in a systematic way. The User Interface Design describes how the system will function from the viewpoint of the User in terms of the front-end client-side portion of the system. This will include a description of how the user will use the system and the expected feedback from the system.

The final part of the document will be the requirements Matrix. In this section, we discuss the organization and relationships between the requirements and functions that are implemented within the system in the form a a 2-D matrix. With all of these sections, this document will provide guidelines on how the system will work in a more component, data, and interface based approach.

Section 1.4 Reference Material

Refer to Appendix-C for Reference Material

Section 1.5 Definitions and Acronyms

Refer to Appendix-A for Acronyms
Refer to Appendix-B for Definitions in the Glossary

Section 2 - System Overview

The IPIMS is an interactive web-based system that will benefit healthcare providers with easy access to records, patients, appointments, lab records and hospital statistics. The system also allows patients of the healthcare provider to easily manage and access their healthcare needs such as appointments and prescriptions.

The IPIMS functionality enables maximum efficiency through the use of organized data and helps facilitate the access of those who are need of healthcare services. The implementation of online storage and accessibility rids of paper-based management to increase efficiency and allow for proper management.

Our system is designed to allow access based on user-permissions determined on the server-side to allow and disallow certain functions to certain authenticated users at any given point in time on the client-side. Our development team is utilizing the powerful backend framework Django powered by the Python programming language. The server-side component that will store and authenticate user data and sessions will be ran under the SQLite database manager. The front-end will be a mixed component of Twitter Bootstrap, JQuery, Javascript, HTML and CSS.

Section 3 - Architectural Design

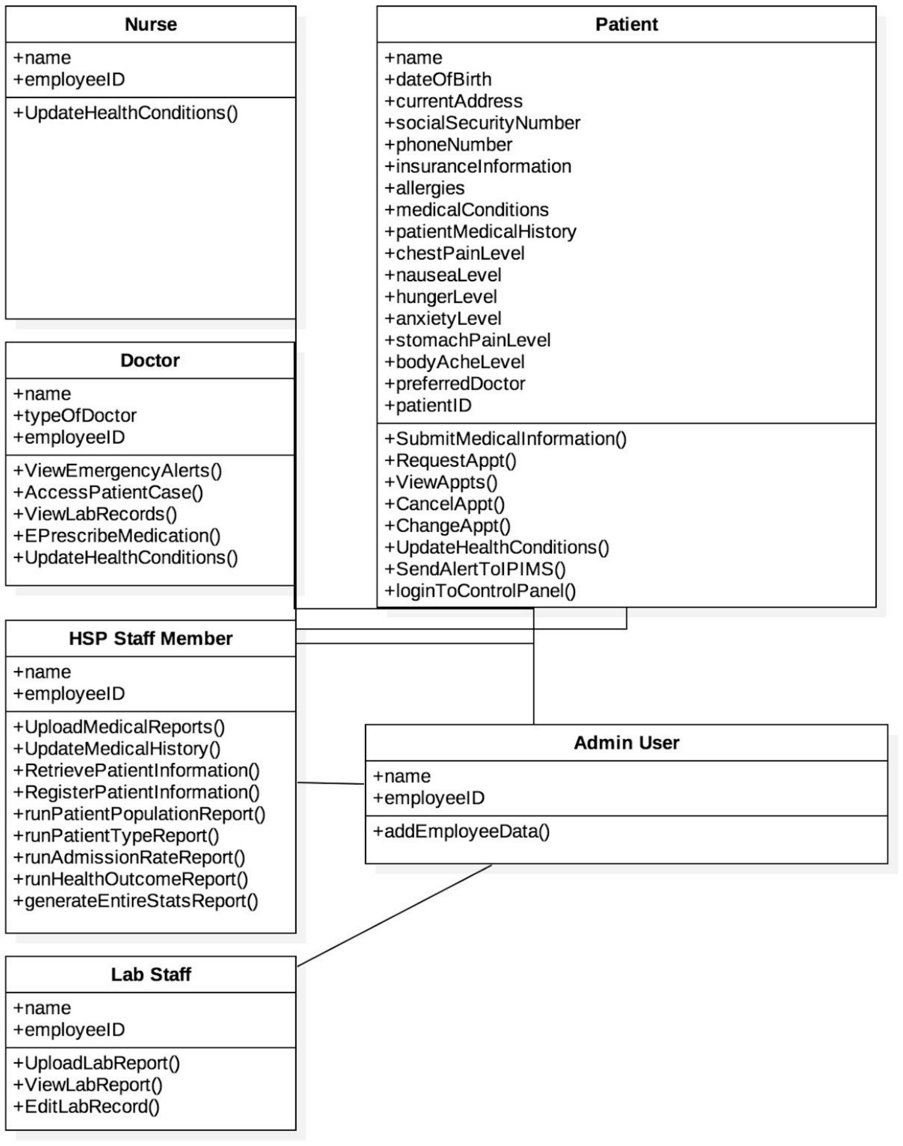


Figure 1 - Overview Class Description

Section 3.1: Architectural Description

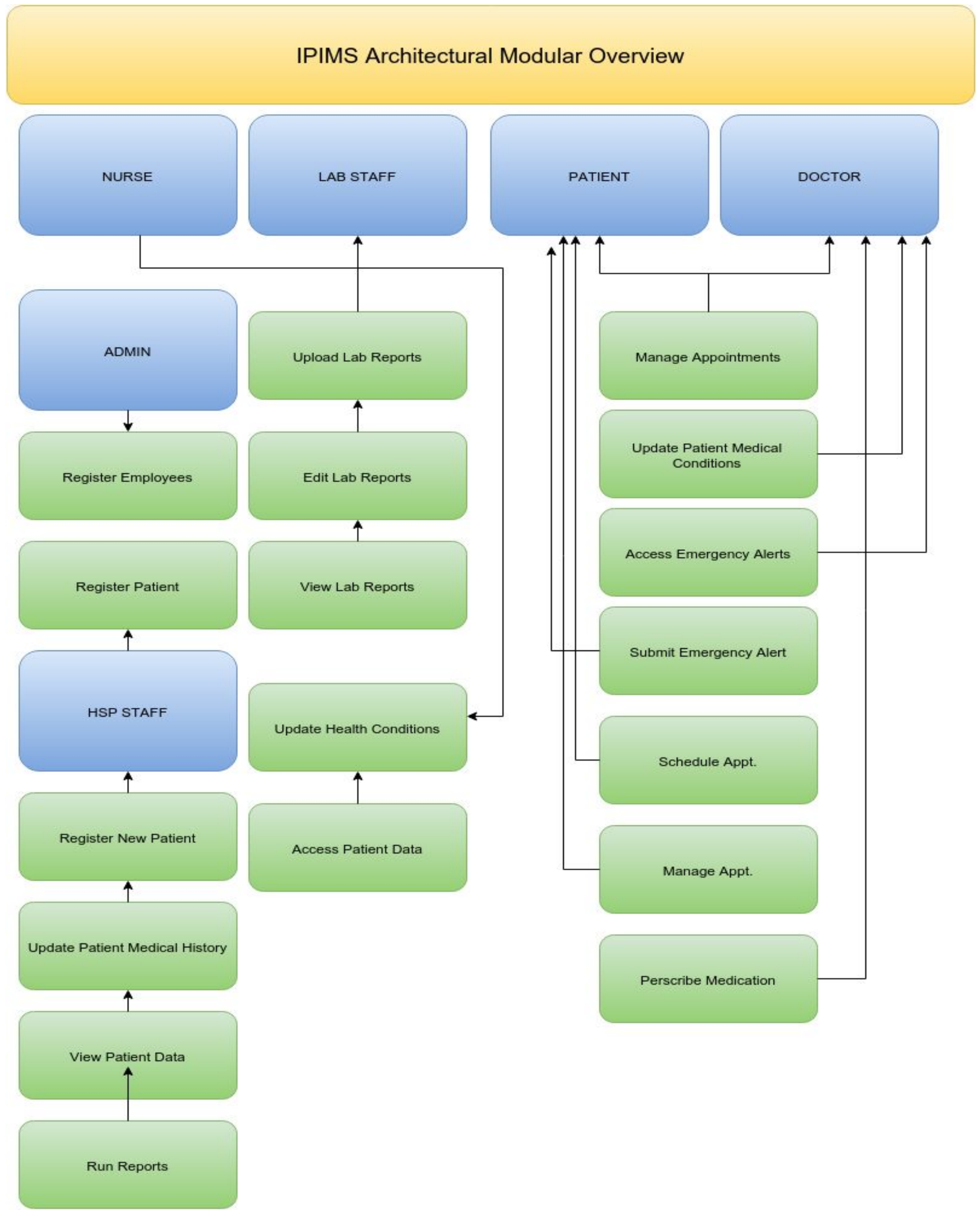


Figure 2 - Modular Architectural Overview

From a top level perspective, the IPIMS consists of two major components, a server side application and a database. The server side application inputs, processes user data and stores it in the database for later use. The coordination between server side, client side, and database will be managed by the Django web framework. The front end of the IPIMS that is served to the client will be a graphical user interface consisting of html, css, and javascript.

The server side application is then split into six subsystems: Nurse, Doctor, Patient, Admin, HSP Staff and Lab Staff. Each subsystem has certain responsibilities as well as differing levels of interactivity with other subsystems. The Nurse class is responsible for viewing and updating the health conditions of patients. The Doctor class is responsible for viewing alerts, prescribing medication, and viewing or updating records. HSP Staff Members generate, edit, and process patient information such as medical reports, medical history and statistics. Lab staff are responsible for viewing, generating and editing lab records.

Depending on what role each subsystem is assigned to, collaboration may be limited. In terms of communication between subsystems, Patient and Doctor can communicate with one another. Further collaboration would include a decentralized patient updating function which allows for subsystems of differing roles to be able to update patient information when needed and will be visible to all subsystems that can view it. This ensures patient information is up to date. To facilitate further collaboration, information generated from subsystem roles such as lab staff may be viewed by other subsystems such as Doctor.

Section 3.2 Decomposition Description

Class: Nurse

The role of the nurse in this system is to view and update health conditions. There are no other permissions given to this class which keeps certain functions within this system compartmentalized.

This subsystem has the following attributes:
- name
- employeeID

Nurse Interface Description-

UpdateHealthConditions()- updates health conditions of patients

Class: Patient

Functional Description -
The patient class is the primary user within the system. The patient class encompasses the attributes for the patient visiting the hospital. The patient has an account that contains

healthcare medical history, personal data, hospital visits, current appointments/past appointments and emergency alert features under severe cases.

This subsystem has the following attributes:
- name
- dateOfBirth
- currentAddress
- socialSecurityNumber
- phoneNumber
- insuranceinfomation
- allergies
- medicalConditions
- patientMedicalHistory
- chestPainLevel
- nauseaLevel
- hungerLevel
- anxietyLevel
- stomachPainLevel
- bodyAcheLevel
- preferredDoctor
- patientID

Patient Interface Description-

SubmitMedicalInformation()-Sends patient information to the database
RequestAppt()- patient can submit a request to the IPIMS for an appointment
ViewAppts()- Displays all of the patient's currently scheduled appointments
CancelAppt()- Cancels patient's appointments
ChangeAppt()- Changes patient's current appointment
UpdateHealthConditions()- updates health condition of patient based on user input
SendAlertToIPIMS()- sends alert to IPIMS which alerts the Doctors
loginToControlPanel()- logs patient into their home screen where the control panel is viewed

Class: HSP Staff

Functional Description -
The HSP staff will be responsible for admitting patient accounts into the system, uploading & updating records, retrieving patient information and running reports.

The HSP staff will consist of the following attributes:
- Name
- Employee ID

HSP Staff Interface Description -

```
UploadMedicalReports() – Upload the medical report history of the patient in question.
UpdateMedicalHistory() – Append medical report history to the patients current history.
RetrievePatientInfo() – Ability to query patients that live within the IPIMS system.
RegisterPatientInfo() – Admit a new patient into the IPIMS.
GenerateEntireStatsReport() – Run all the statistical reports in the IPIMS.
```

Class: Doctor

Functional Description-
The doctor is responsible for viewing emergency alerts sent from patients, prescribing medication, viewing lab reports, updating health conditions in the database for that specific patient and accessing a patient case.

Doctor will have the following attributes:

- name
- typeOfDoctor
- employeeID

Doctor Interface Description

```
ViewEmergencyAlerts() – Views the alerts that were sent by the IPIMS or patient
AccessPatientCase() – View the scheduled appointment requested by the patient.
EPerscribeMedication() – Send a prescription to the pharmacy that the patient needs.
UpdateHealthConditions() – Update the resolution of the case of the patient.
```

Class: Lab Staff

Functional Description
Lab staff are responsible for uploading lab reports, viewing lab reports and editing lab records.

Lab Staff will have the following attributes:

- name
- employeeID

Lab Staff Interface Description

UploadLabReport() – Ability to upload report data generated for each patient in question ViewLabReport() – Ability to view previously generated report data EditLabReport() – Ability to see and edit current information in the lab report

Class: Admin User

Functional Description
The responsibility of the Admin is to add employee information to the IPIMS. Information will include role, employee ID and name.

Admin User will have the following attributes:
- name
- employeeID

Admin User Interface Description

addEmployeeData()- ability to add employees to the IPIMS

Section 3.3 Design Rationale

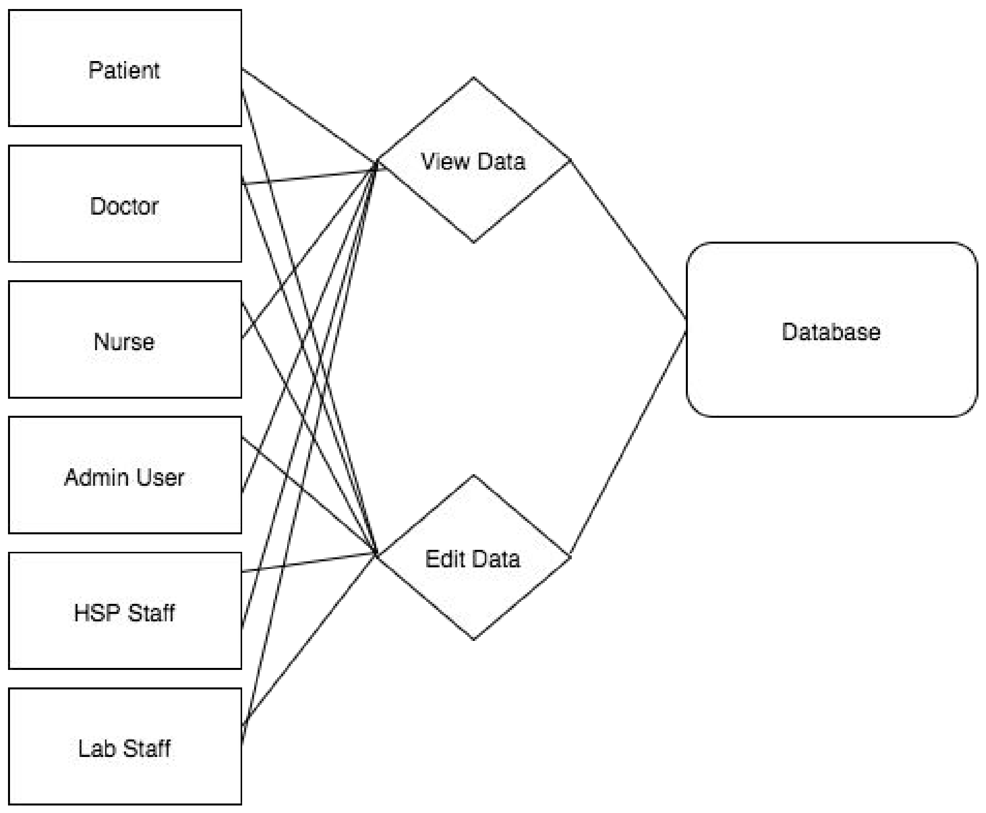
Our design rationale is based on permissions for each associated actor. The Interactive Patient Management System mainly divides into two parts -- client side and server side. On the client-side, pages with display on the front-end based on associated permissions and access-levels for functionality defined within the IPIMS. The server-side provides permissions on a per-actor basis. These permissions grant associations between the different database tables to ensure each actor can only view their respective functions and not access those they do not belong to. We designed specific functions per-actor to clearly segregate who is to do what on what pages and permissions that are defined on the server-side. As a group, we initially discussed a system in which users are registered separately with a token, provided by the hospital, to separate permissions based on employee type. This directly went against the requirements specification and that’s why we designed the functionality and permissions system with a server-side authentication system in place with the admin user to manually register employees and grant their respective permissions.

Section 4 - Data Design

Section 4.1: Data Description

Our users in the system, appointments, employees, permissions and structures are all stored in a common database. The IPIMS is utilizing SQLite to control and manage all the data entities, storage and dynamically persisted data throughout the entire system.

Each data class, such as nurse, doctor, patient, etc., is associated as a data table within the Sqlite database. The functions described below illustrate how each row within the associated tables are filled to allow persistent data to be accessible whenever and wherever, regardless of the system utilizing the data.



Section 4.2: Data Dictionary:

The system entities/major data which will be used in the implementation of the health care system are listed below alphabetically. Section 5 details the functional implementation of each associated function through the use of pseudo-code.

Admin Entity

addEmployeeData(string Name, int employeeId)
This method should be called to add new employees to the employee database. this method <u>will not run</u> unless both parameters are filled.

+int EmployeeId - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*
+string Name - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*

Doctor Entity

+ViewEmergencyAlerts(boolean alert, string patientID)
This method should be called to view emergency alerts that are sent by sendAlertToIPIMS(). This method <u>will not run</u> if patientID is empty or alert is false.

+AccessPatientCase(int patientID)
This method should be called to access patient cases individually.

+ViewLabRecords(int patientID)
This method should be called to view patient lab records individually.

+EPrescribeMedication(int patientID, string prescriptionName)
This method should be called to prescribe medication to an individual patient.

+UpdateHealthConditions(int patientID)
Shared with nurse/doctor/patient. See shared Methods

+int EmployeeId - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*
+string Name - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*
+string typeOfDoctor - *indicates specialization of doctor staff*

HSP Staff Entity

+UploadMedicalReport(int patientID, string Name)
This method should be used to upload a new patient medical report to the database

+UpdateMedicalHistory(labRecords, patientInformation,patientID)
This method should be used to update the existing medical history of a patient

+RetrievePatientInformation()
This function should be used to the get an existing patients information

+RegisterPatientInformation()
This method should be used to register a new patient's information into the database

+runPatientPopulationReport()
This method should be used to run a report which returns the population of patients in the database.

+runPatientTypeReport()
This method should be used to run a patient type report

+runAdmissionRateReport()
This method should be used to run a statistical report which will return the current admission rates

+runHealthOutcomeReport()
This method should be used to run a report on the health outcomes of a certain patient.

+generateEntireStatsReport()
This function should be used to generate a report on all applicable statistics from the database.

+int EmployeeId - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*
+string Name - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*

Nurse Entity

+UpdateHealthConditions() *shared with patient/nurse/doctor/HSP Staff. see shared Methods*

+int EmployeeId - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*

+string Name - *shared with patient/nurse/doctor/HSP Staff. see shared Methods*

Patient Entity

+SubmitMedicalInformation()
This method should be used to submit patient medical information which will return a boolean of submitted or not submitted.

+RequestAppt()
This method should be used to create an appointment request. This request is approved or denied based on doctor availability.

+ViewAppts(doctorID, patientID)
This function returns patient's applicable appointments

+CancelAppts()
this method should be used to cancel patient's applicable appointments

+ChangeAppt()
this method should be used to edit a patient's current appointment

+updateHealthConditions()
shared with nurse/doctor/patient. see shared Methods
+sendAlertToIPIMS()
This method should be used to send an emergency alert to the IPIMS when patient levels are severe

+loginToControlPanel(string email, char[] password)
This method should be used to log into the user portal. parameters email and password are required or method will not be invoked.

+int anxietyLevel - *integer which indicates the level of anxiety of the patient*
+int bodyAcheLevel - *integer which indicates the level of body ache of the patient*
+int chestPainLevel - *integer which indicates the level of chest pain of the patient*
+int hungerLevel - *integer which indicates the hunger level of the patient*
+int nauseaLevel - *integer which indicates the nausea level of the patient*

+int patientId - *sequence of integers which hold the identification number of the patient*
+ int phoneNumber-*integer sequence which holds phone number of the patient*
+ int socialSecurityNumber - *integer length 9 which holds social security number of a patient***.**
+ int stomachPainLevel - *integer which indicates the level of stomach pain of a patient*
+string [ ] allergies - *string contains list of patient allergies*
+string[ ] currentAddress - *string which holds the current address information of the user*
+string [ ] dateOfBirth - *string which holds the date of birth information of the patient*

+string[ ] insuranceInformation - *string array holds current insurance information of the patient*
+string [ ] medicalConditions - *string array holds list of current medical conditions of the patient.*
+ string [ ] patientMedicalHistory - *string array which holds medical history information of patient*
+string Name - *shared with patient/nurse/doctor/HSP Staff. see shared*
+ string preferredDoctor - *string which holds the preferred doctor of a certain patient*

Lab Staff Entity

+UploadLabReport(int patientId)
This method allows the user (lab staff) to upload a lab report for a specific patient. The patient is found by entering the patient identification number.
+ViewLabReport(int patientId)
This method allows the user (lab staff) to view a lab report for a specific patient. The patients lab report is found by entering the patient identification number.
+EditLabRecord(int patientId)
This method allows the user (lab staff) to edit a lab report for a specific patient. The patient's lab report is found by entering the patient identification number

+int employeeId
+string name

Shared Functions Amongst All Entities

+UpdateHealthConditions()

This method should be used to update patient health conditions. the only required parameter is boolean patientName and patientID match, any changes can be made or not made.

+int EmployeeId - *integer of length 9 that uniquely identifies each employee and their position*
+string Name - *string that stores current name of user*

Section 5 - Component Design

In this section, we detail the operations from a more technical perspective. Each button, each entity and each user-interaction is associated with some back-end implementation described below through the use of pseudo-code. We logically illustrate the functionality of each operation from a OO standpoint.

5.1 - View Emergency Alerts Pseudo-Code

```
if (viewEmergencyAlerts.button is clicked && patientID) {
    AccessPatientCase(patientID); //Button loads page for the individual user/patient
}
```

5.2 - Access Patient Case Pseudo-Code

```
function AccessPatientCase(int patientID) {
    for (each_associated_case for patientID) {
        display patient case data to page
    }
```

5.3 - View Lab Records Pseudo-Code

```
function ViewLabRecords(int patientID)
{
    for (each_record associated with patientID) {
        display record[x][patientID] to page
    }
}
```

5.4 - E-Prescribe Medication Pseudo-Code

```
EPrescribeMedication(int patientID, string prescriptionName) {

    string prescriptionName = $this.prescriptionName;
    if (patientID != NULL && prescriptionName != NULL)
    {
        store prescription in users patient-case
    }
}
```

5.5 - Upload Medical Report Pseudo-Code

```
if(updloadMedicalReport.button is clicked) {
    AccessPatientCase(int patientID);
    if(Name matches employeeID) {
        uploadReport();
    }
}
```

5.6 - Update Medical History Pseudo-Code

```
if(patientID != patientInformation(patientID) {
    return error('wrong patient!');'
    break;
} else {
    while(medicalHistory !=  labRecords && patientInformation) {
        for(i = 0; i<records.length) {
            replace(oldRecords, newRecords);
            replace(oldInformation, newInformaiton);
        }
        break;
    }
}
```

5.7 - Retrieve Patient Information Pseudo-Code

```
if(RetrievePatientInformation.button is clicked)
{
    patient.getName();
    patient.getId();
}
```

5.8 - Register Patient Information Pseudo-Code

```
if(registerPatient.button is clicked) {
    name = textField.get(name);
    patientId = textField.get(patientId);
    registerPatientInformation(name, patientId);
}
```

5.9 - Run Patient Population Report Pseudo-Code

```
if(patientPopulation.button is clicked) {
    currentPatientNames = [ ]
    for(i = 0; i<numberOfPatients; i++) {
        currentName = Name;
        +=currentPatientNames[ currentName];
        numberOfPatients = i ;
    }
    return currentPatientNames[] && i ;
}
```

5.10 - Run Patient Type Report Pseudo-Code

```
if(patientType.button is clicked) {
    currentPatientType = [ ]
    for(i = 0; i < numberOfPatients; i++)
        {
        currentPatientType += patient.getPatientType();
        }
    return currentPatientType[ ];
}
```

5.11 - Run Patient Type Report Pseudo-Code

```
if(admissionRate.button is clicked) {
        admissionAccepted = 0;
        size = 0;
    for(i = 0; i < numberOfPatients; i++)
        {
            if(patient.isAccepted)
            {
                admissionAccepted++;
            {

            size = i;
        }
    admissionAcceptanceRate =  admissionAccepted/size;
}
```

5.12 - Run Health Outcome Report Pseudo-Code

```
if(healthOutcome.button is clicked) {
    healthOutcomeArray = [ ];
    for(i = 0; i < numberOfPatients; i++)
    {
        healthOutcomeArray += patient.getHealthOutcome();
    {
    return healthOutcomeArray[ ];
}
```

5.13 - Generate Entire Stats Report Pseudo-Code

```
if(entireStats.button is clicked) {
    runPatientPopulationReport();
    runPatientTypeReport();
    runAdmissionRateReport();
    runHealthOutcomeReport();
}
```

5.14 -Update Health Conditions Pseudo-Code

```
currHealthCondition = [ ]
if(updateHealthConditions button is clicked)
```

```
    {
        for(i = 0; i < numberOfPatients; i++)
            {
            currHealthCondition[i]  = textfield.get(currHealthCondition[i])
            }}
```

5.15 - Submit Medical Information Pseudo-Code

```
if (new.Login) {
    updateMedicalInformation;
} elseif(updateMedicalInfo.button is clicked) {
    display patient info(!edit);
} else( cancel) {
    return false;
}
return true;
}
```

5.16 - Request Appointment Pseudo-Code

```
if(appt.available) {
    reserveAppt;
} else {
    chooseNewTime;
}
```

5.17 - View Appointment Pseudo-Code

```
if(viewAppts.Button(click)) {
    viewApptCalendar();
}
viewApptCalendar(doctor.id, patient.id) {
    doctorCalendar.display(biweekly);
}
```

5.18 - Change Appointment Pseudo-Code

```
while(appt status == !set) {
    isChange();
}
```

```
isChange(RequestAppt, CancelAppts) {
    if(RequestAppt) {
        CancelAppt(current);
        ChangeAppt(new);
    }
}
```

5.19 - Submit Medical Information Pseudo-Code

```
if(updateHeallthConditions(alert)) {
    sendAlert();
}
```

5.20 - Login To Control Panel Pseudo-Code

```
if(username && password == verified && match) {
    accept.login
} else {error.message}
```

5.21 - Update Health Conditions Pseudo-Code

```
if(patientMedicalHistory == updated &&  isDoctor | isNurse | isPatient) {
    if(healthConditions !== severe) {
        updateHealthConditions.button(isClicked);
    } else(alert)
}
```

5.22 - Cancel Appointment Pseudo-Code

```
cancelAppt(int apptId) {
    delete($this.appt(apptId));
}
```

5.23 - Upload Lab Report Pseudo-Code

```
uploadLabReport(int patientId)
{
if(uploadLabReport.button is clicked) {
        AccessPatientCase(int patientID);
        if(patientId is found) {
                uploadLabReport();
        }
}
```

5.24 - View Lab Report Pseudo-Code

```
viewLabReport(int patientId)
{
        if(viewLabReport.button is clicked) {
                        for (each_report matching patientID) {
                                display report[i][patientID]
                        }
        }
}
```

5.25 - Edit Lab Records Pseudo

```
editLabRecord(int patientId)
{
        if(editLabRecord.button is clicked){
                if(patientId is found) {
                        load lab record page(patientID)
                }
                else{
                        display "The patient is not found in the database"
                }
        }
}
```

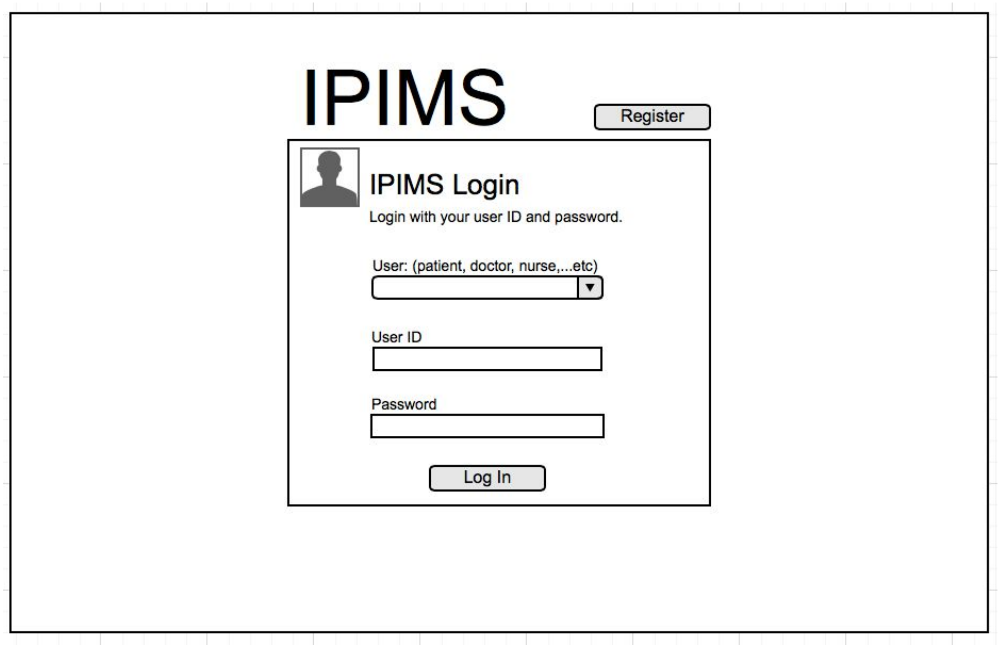
Section 6: User Interface Design

Section 6.1: Overview of User Interface

The user interface consists of menus, drop down menus, checkboxes, buttons, tables, and links that provide easy access to the IPIMS for all users. There are specific user interfaces for a patient, doctor, nurse, lab staff, HSP staff, and admin. The simplicity of each design will allow the user to navigate the site quickly and efficiently minimizing confusion. Each page has a clear purpose and is very organized outlining all functionalities.

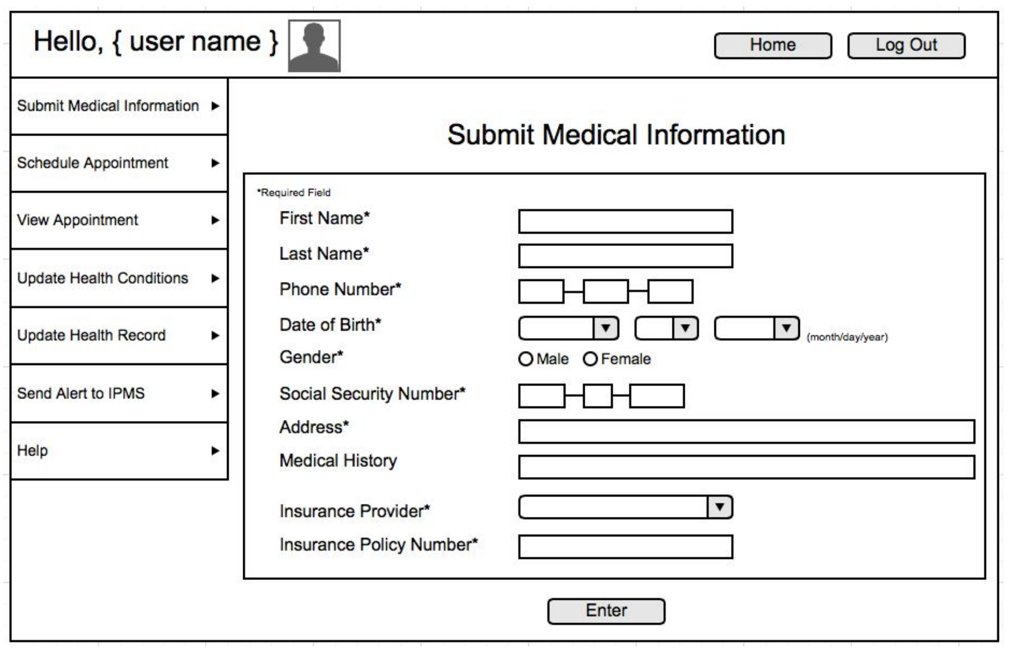
Section 6.2: Screen Images

Login:


IPIMS
Register
IPIMS Login
Login with your user ID and password.
User: (patient, doctor, nurse,...etc)
User ID
Password
Log In

- The "Login" page will appear once the user clicks on the login button from the main page. The "Login" page allows the user to declare what type of user they are, as well as enter their user ID and password.

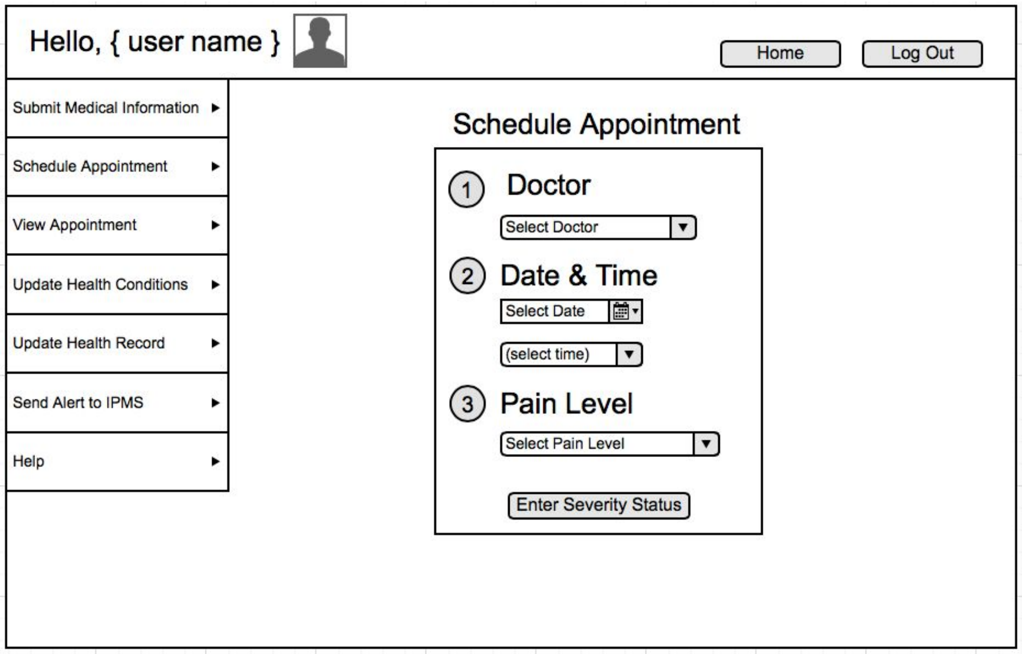
Patient: Submit Medical Information



- **Once the Patient has logged in, they have the option to submit medical information. The "Submit Medical Information" page allows the Patient to enter personal information such as name, phone number, date of birth, gender, social**

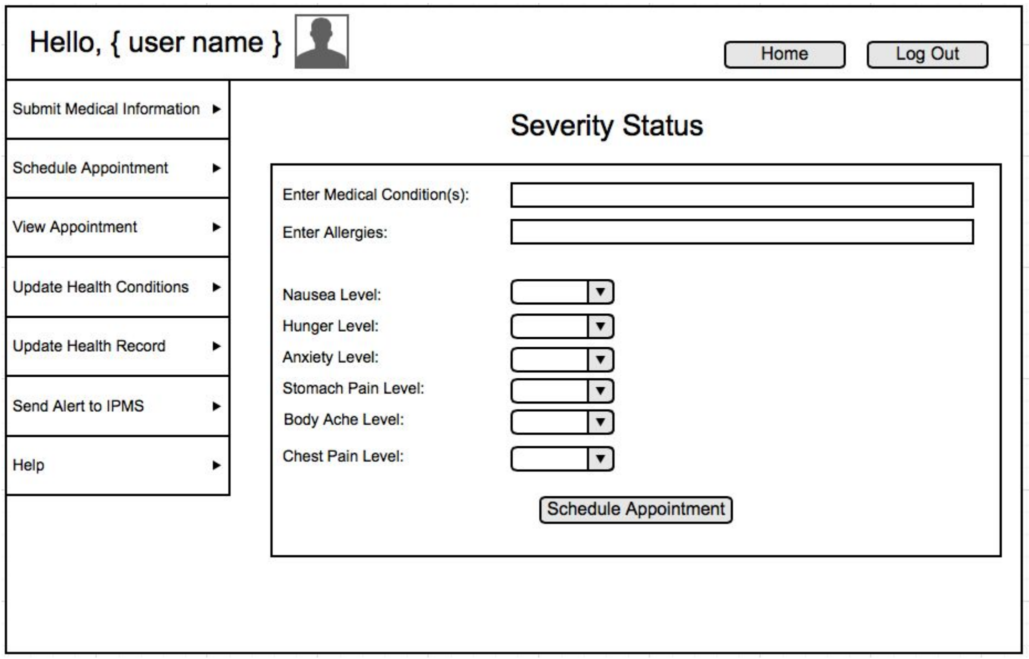
security number, address, medical history, insurance provider, and insurance policy number. After the required fields have been completed, the Patient can click "Enter" to send their information to the HSP staff for review and registration.

Patient: Schedule Appointment



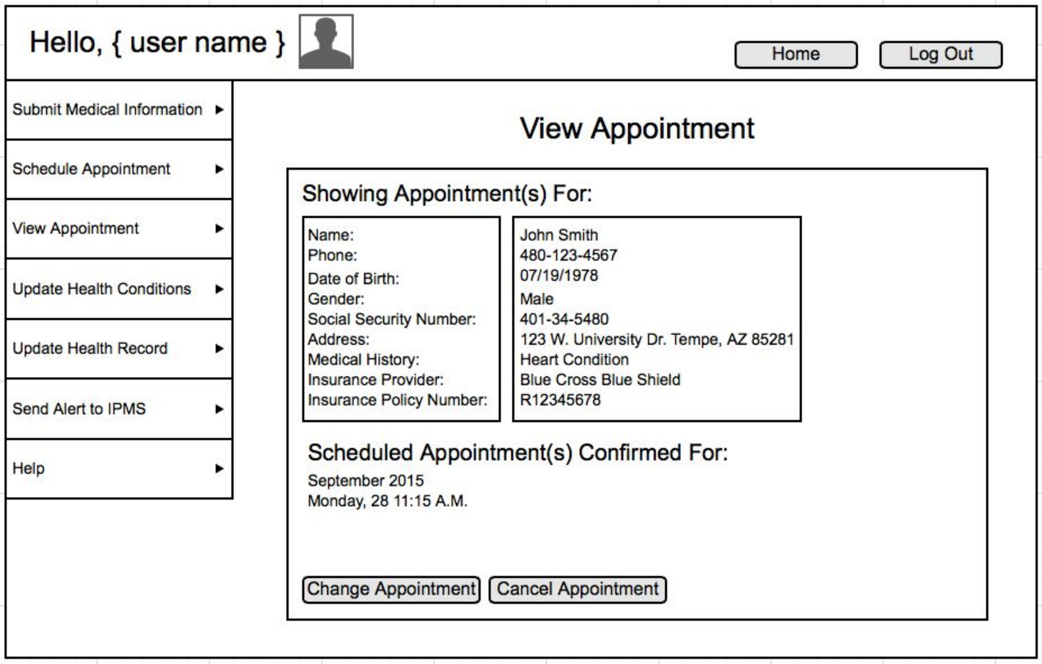
- Once the Patient has logged in, they have the option to schedule an appointment. The "Schedule Appointment" page allows the patient to select doctor, date, time, and pain level. The Patient will be able to enter their severity status once they enter answers for those questions.

Patient: Severity Status



- Once the Patient has decided on a doctor, date, time, and pain level when scheduling an appointment they will be able to enter medical conditions, allergies, nausea level, hunger level, anxiety level, stomach pain level, body ache level, and chest pain level on the “Severity Status” page.

Patient: View Appointment



View Appointment

Showing Appointment(s) For:

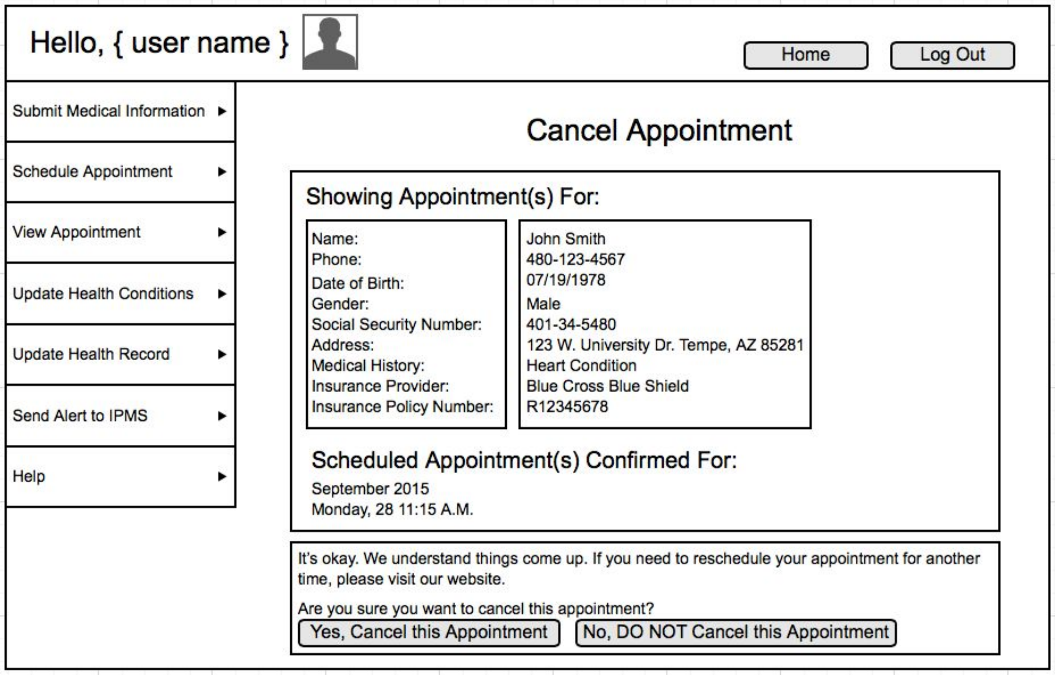
Name:	John Smith
Phone:	480-123-4567
Date of Birth:	07/19/1978
Gender:	Male
Social Security Number:	401-34-5480
Address:	123 W. University Dr. Tempe, AZ 85281
Medical History:	Heart Condition
Insurance Provider:	Blue Cross Blue Shield
Insurance Policy Number:	R12345678

Scheduled Appointment(s) Confirmed For:

September 2015
Monday, 28 11:15 A.M.

- Once the Patient has logged in, and previously scheduled an appointment they can select view appointment on the side bar to access the “View Appointment Page.” This page will allow the Patient to access all information for any scheduled appointment as well as the option to change or cancel appointments.

Patient: Cancel Appointment



Cancel Appointment

Showing Appointment(s) For:

Name:	John Smith
Phone:	480-123-4567
Date of Birth:	07/19/1978
Gender:	Male
Social Security Number:	401-34-5480
Address:	123 W. University Dr. Tempe, AZ 85281
Medical History:	Heart Condition
Insurance Provider:	Blue Cross Blue Shield
Insurance Policy Number:	R12345678

Scheduled Appointment(s) Confirmed For:

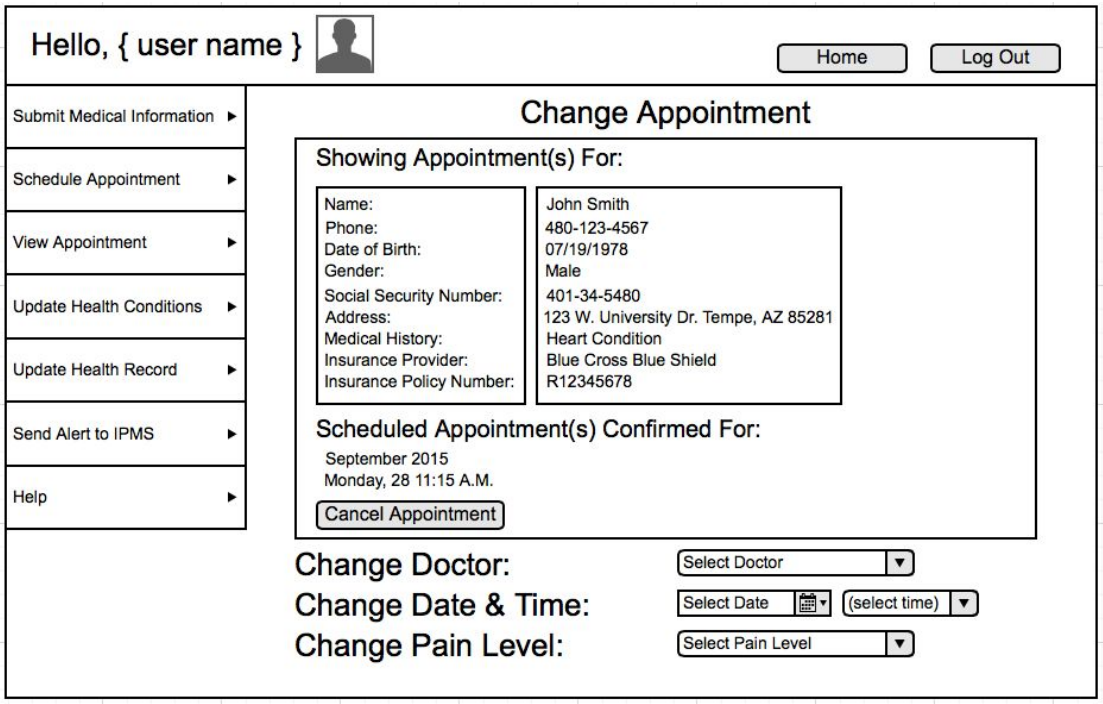
September 2015
Monday, 28 11:15 A.M.

It's okay. We understand things come up. If you need to reschedule your appointment for another time, please visit our website.

Are you sure you want to cancel this appointment?

- After the Patient has accessed the “View Appointment” page they have the option to cancel their previously scheduled appointment. If they decide to cancel it will take them to the “Cancel Appointment” page. The page will display the appointment the Patient wishes to cancel and provide the Patient with the option to cancel or not to cancel their appointment.

Patient: Change Appointment



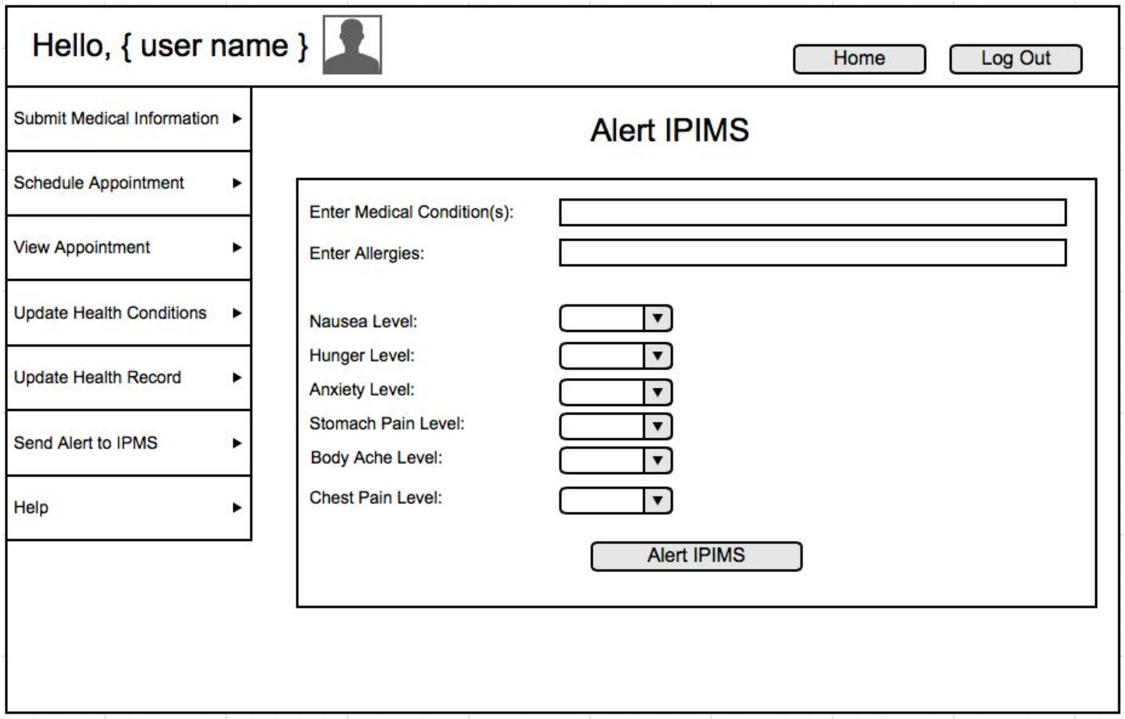
Name:	John Smith
Phone:	480-123-4567
Date of Birth:	07/19/1978
Gender:	Male
Social Security Number:	401-34-5480
Address:	123 W. University Dr. Tempe, AZ 85281
Medical History:	Heart Condition
Insurance Provider:	Blue Cross Blue Shield
Insurance Policy Number:	R12345678

Scheduled Appointment(s) Confirmed For:
September 2015
Monday, 28 11:15 A.M.

Change Doctor:
Change Date & Time:
Change Pain Level:

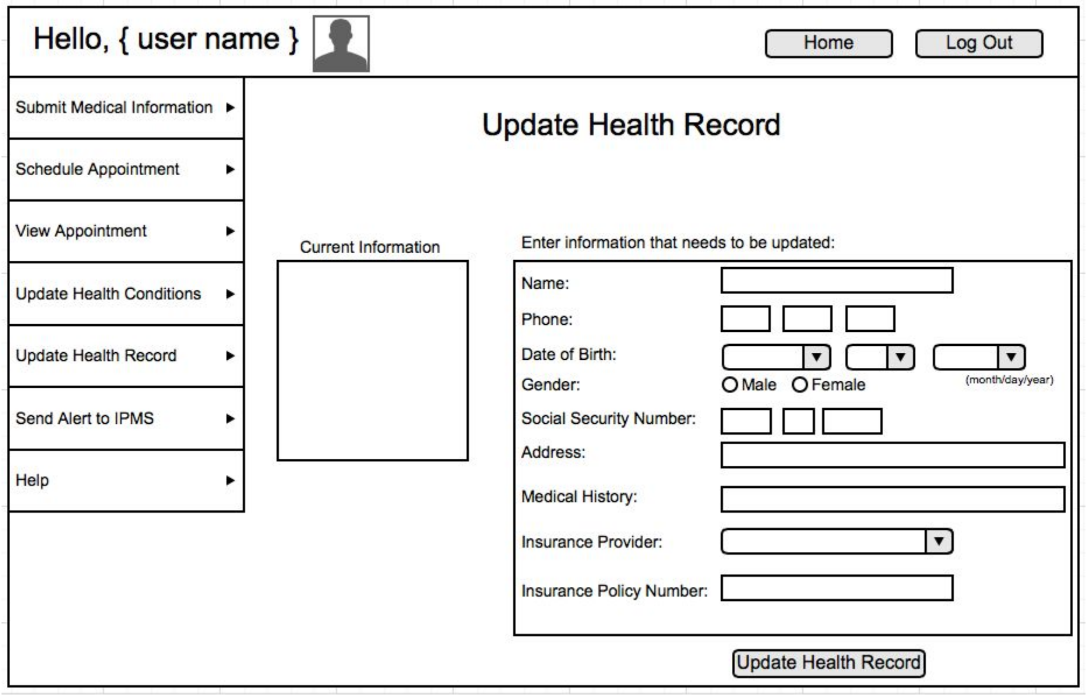
- After the Patient has accessed the “View Appointment” page they have the option to change their previously scheduled appointment. If they decide to change, it will take them to the “Change Appointment” page. The page will show the previous scheduled appointment, and the option to select a new doctor, date, time, and pain level.

Patient: Alert IPIMS



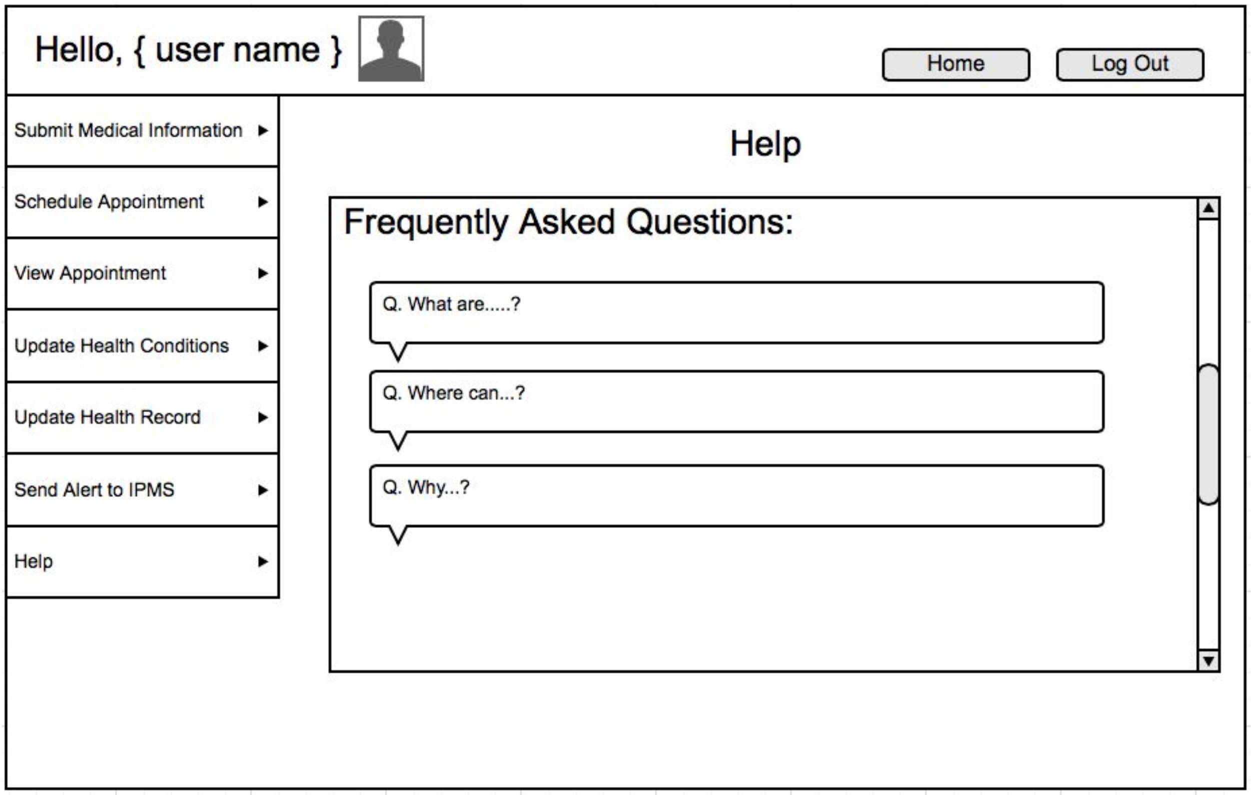
- Once the Patient has logged in, the can select Send Alert to IPIMS from the side bar. The “Alert IPIMS” page will allow the Patient to enter current medical conditions and when they click on the “Alert IPIMS” button it will immediately notify a doctor.

Patient: Update Health Record



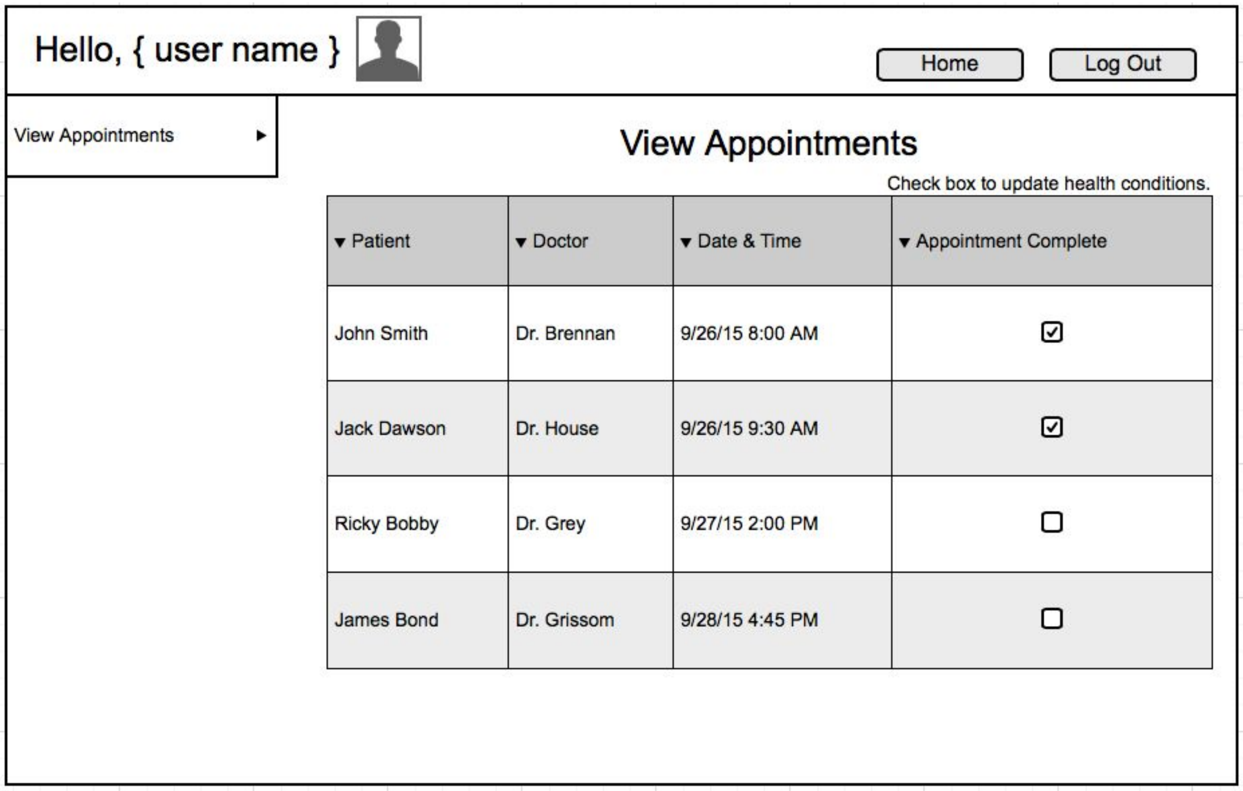
- Once the Patient has logged in they are able to select Update Health Record from the side bar. The “Update Health Record” page will show the Patient their previously entered information, and provide the Patient with a form they can fill out if updates are necessary.

Patient: Help



- Once the Patient has logged in they can select Help from the side bar. The “Help” page will provide the Patient with frequently asked questions and guides on navigating the website.

Nurse: Update Health Conditions



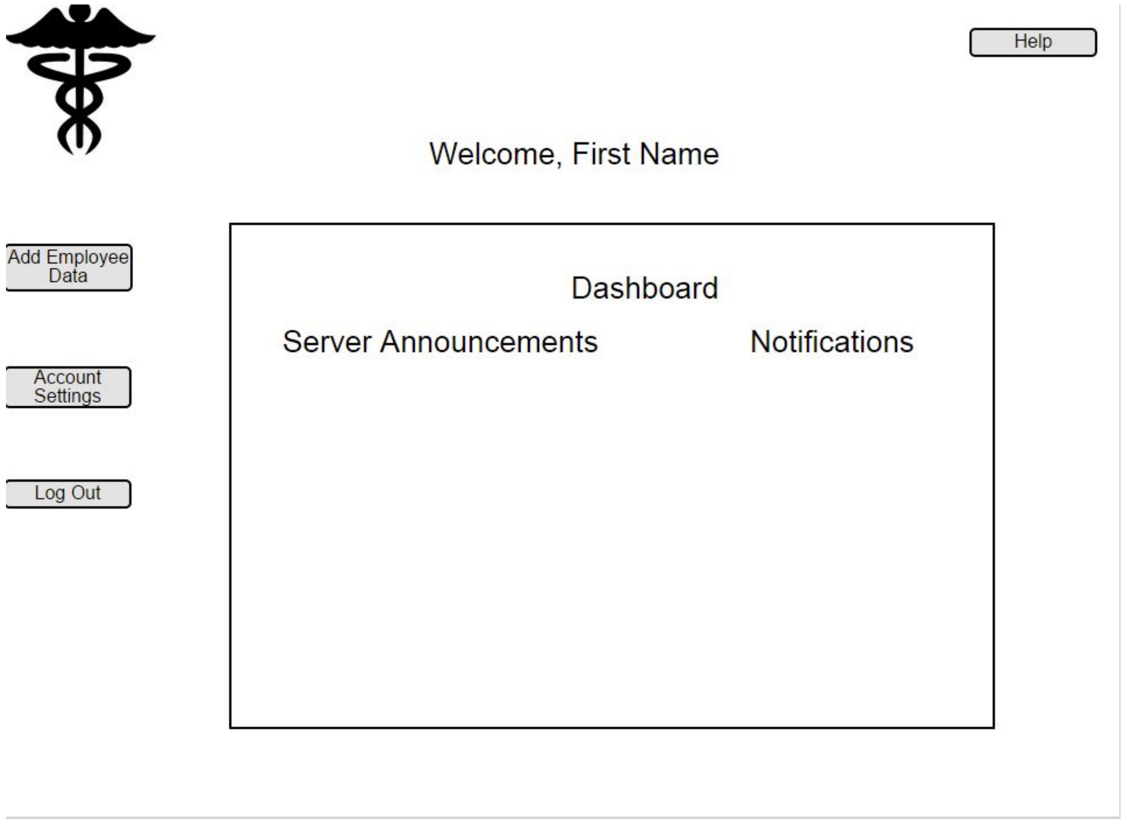
View Appointments

Check box to update health conditions.

▾ Patient	▾ Doctor	▾ Date & Time	▾ Appointment Complete
John Smith	Dr. Brennan	9/26/15 8:00 AM	☑
Jack Dawson	Dr. House	9/26/15 9:30 AM	☑
Ricky Bobby	Dr. Grey	9/27/15 2:00 PM	☐
James Bond	Dr. Grissom	9/28/15 4:45 PM	☐

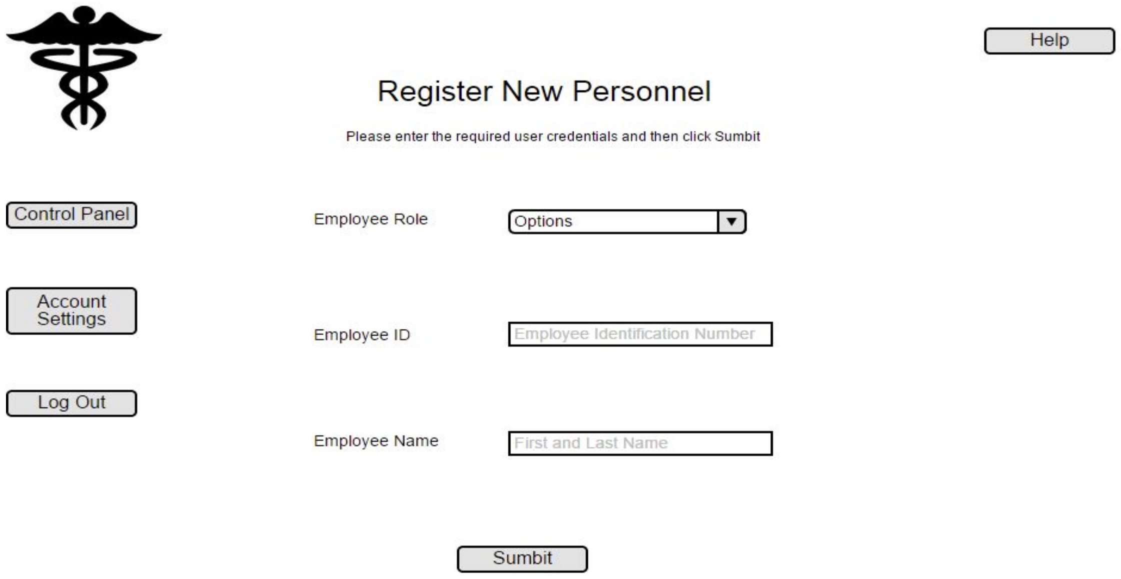
- Once the Nurse is logged in they are able to select View Appointment from the side bar. This is where they will be able to view all appointments and update the patient's health condition by checking the box in the appointment complete section.

Admin: Control Panel



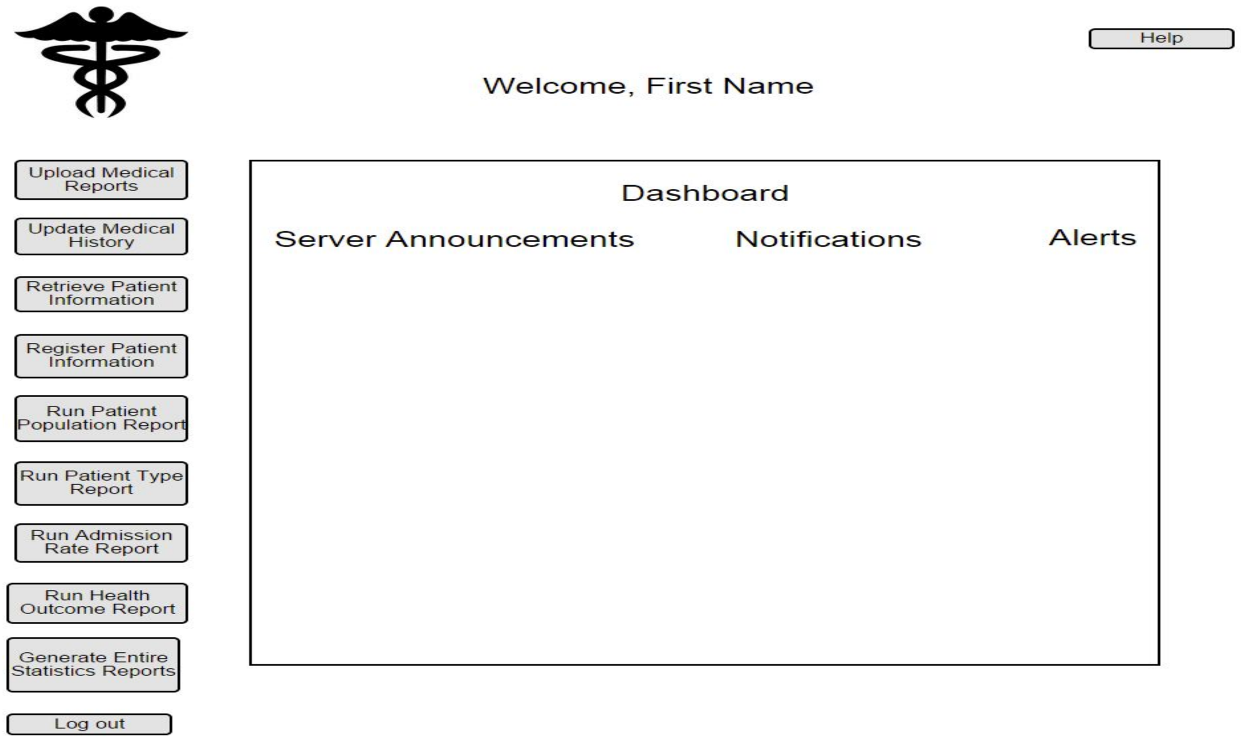
- This is the main control panel, which the Administrator can use to select main functions.

Admin: Add New Employee



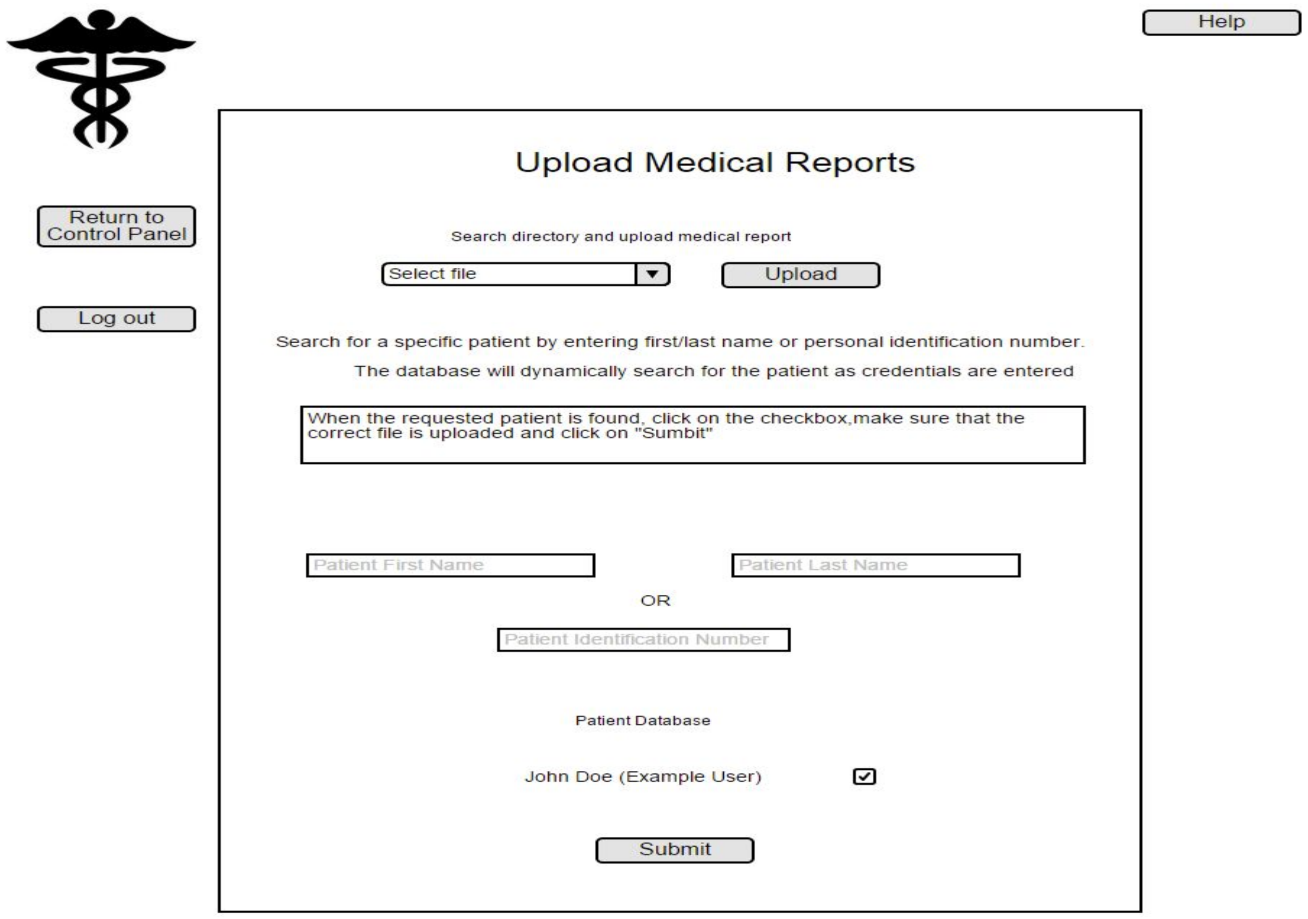
- The Administrator has the add new employee functionality attached to his/her account. The admin will click “Add New Employee” and will be redirected to the page, which will then prompt them to enter the employee role, employee ID, and employee first and last name, which will all be added into the database when the “Submit” is clicked.

HSP Staff: Control Panel


Help
Welcome, First Name
Upload Medical Reports
Update Medical History
Retrieve Patient Information
Register Patient Information
Run Patient Population Report
Run Patient Type Report
Run Admission Rate Report
Run Health Outcome Report
Generate Entire Statistics Reports
Log out
Dashboard
Server Announcements
Notifications
Alerts

- The main control panel for the HSP Staff. From this page, staff can view announcements, notification, and view IPIMS alerts.

HSP Staff: Upload Medical Reports

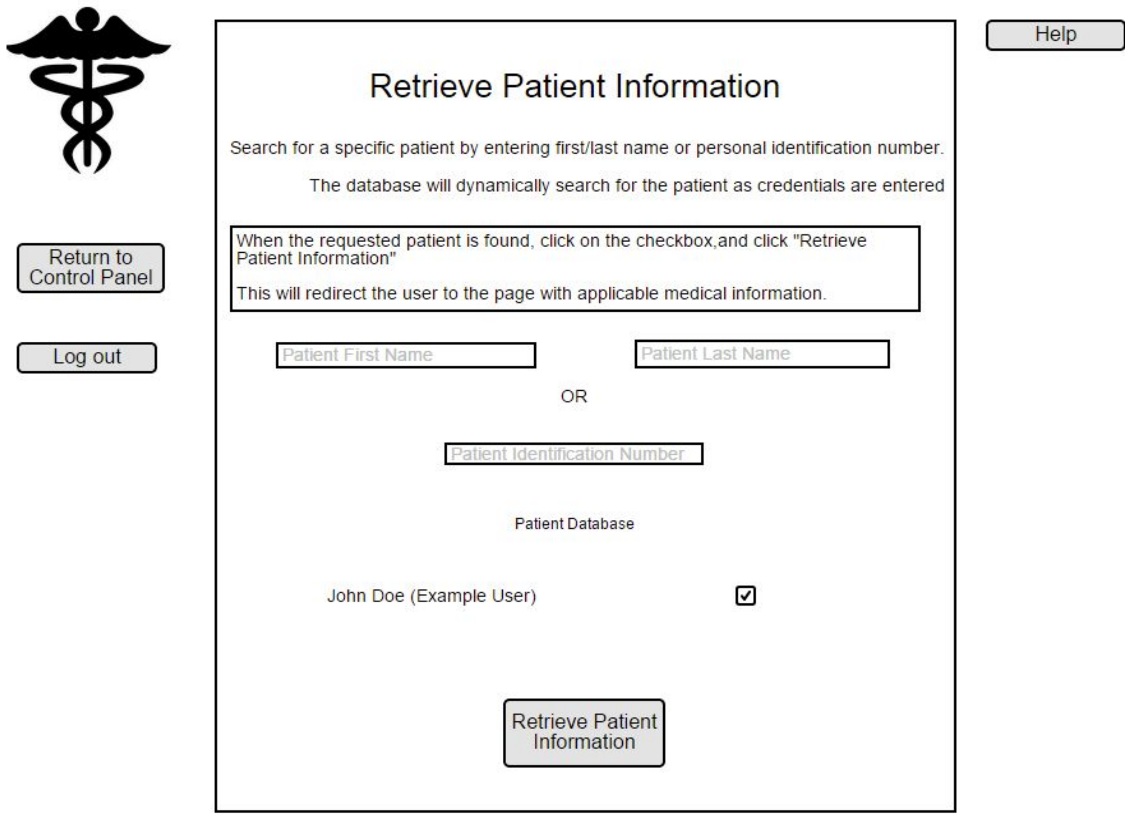


Search for a specific patient by entering first/last name or personal identification number.
The database will dynamically search for the patient as credentials are entered

When the requested patient is found, click on the checkbox,make sure that the correct file is uploaded and click on "Sumbit"

- HSP Staff have the “Upload Medical Reports” which will allow them to, as the name suggests, upload new patient medical reports to the database. They will do this first clicking “Upload Medical Reports” from the control panel, and then searching their file directory for the correct document, entering a patient first/last name or ID, check the applicable user found in the database, and clicking “Submit"

HSP Staff: Retrieve Patient Information



Retrieve Patient Information

Search for a specific patient by entering first/last name or personal identification number.

The database will dynamically search for the patient as credentials are entered

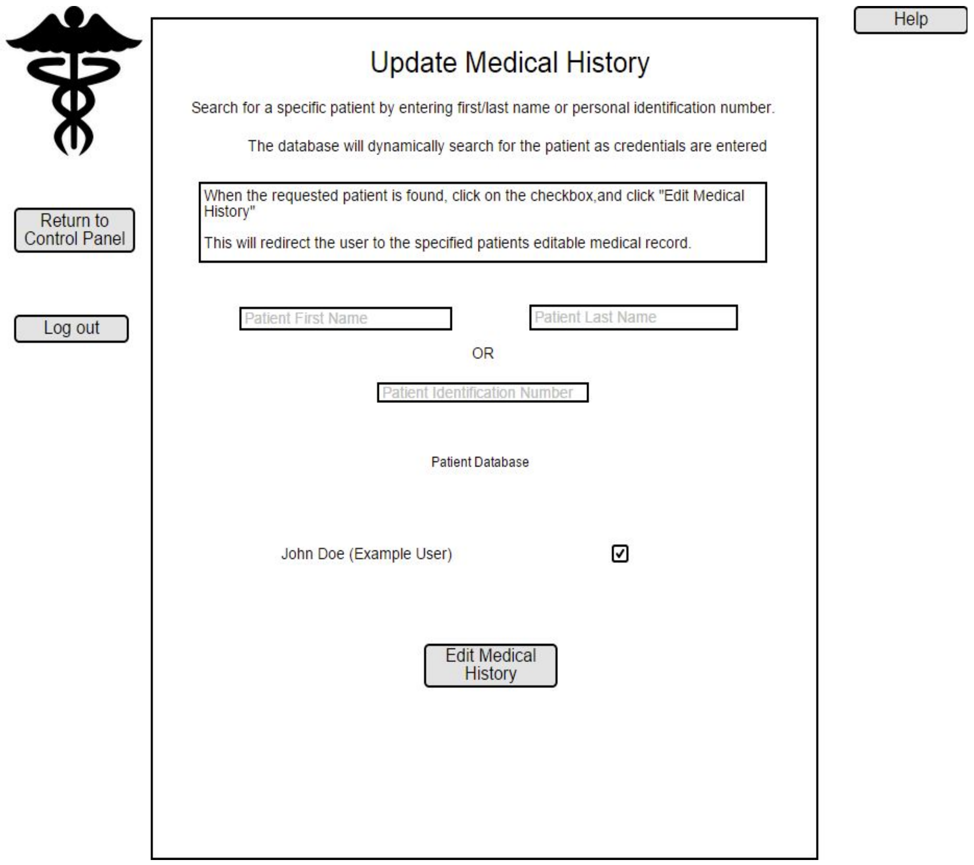
When the requested patient is found, click on the checkbox,and click "Retrieve Patient Information"

This will redirect the user to the page with applicable medical information.

John Doe (Example User) ✓

- Staff can retrieve patient information by clicking “Retrieve Patient Information” from their control panel. They will be redirected to the patient information retrieval page, where they find the patient in the database (through first and last name or ID), check the applicable patient, and click on “Retrieve Patient Information”

HSP Staff: Update Medical History



Update Medical History

Search for a specific patient by entering first/last name or personal identification number.

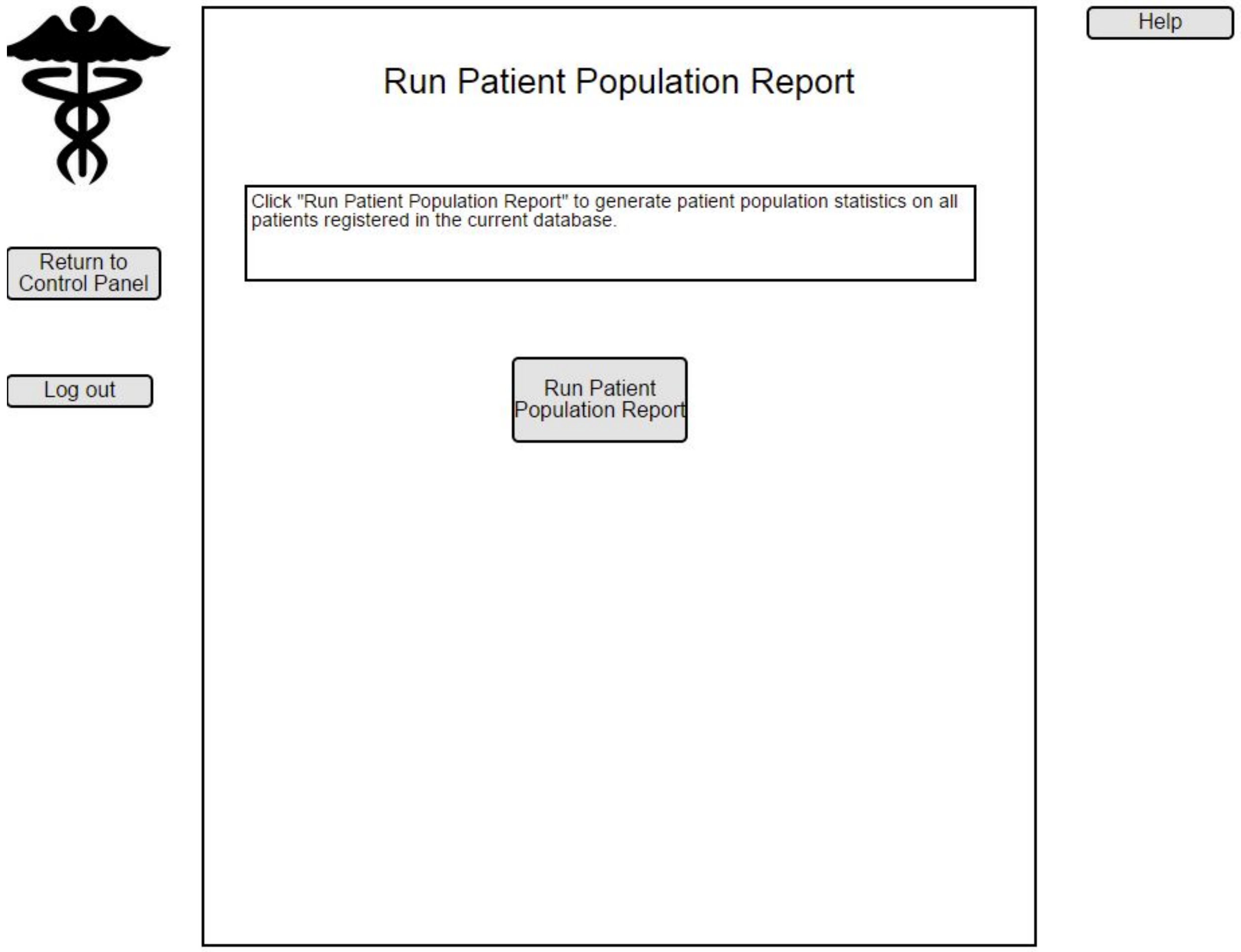
The database will dynamically search for the patient as credentials are entered

When the requested patient is found, click on the checkbox,and click "Edit Medical History"

This will redirect the user to the specified patients editable medical record.

- Staff can update the current medical history of a patient by clicking on “Update Medical History” from their control panel, finding the patient from the database, and clicking “Update Medical History”. This will redirect the user to the patient's editable medical file.

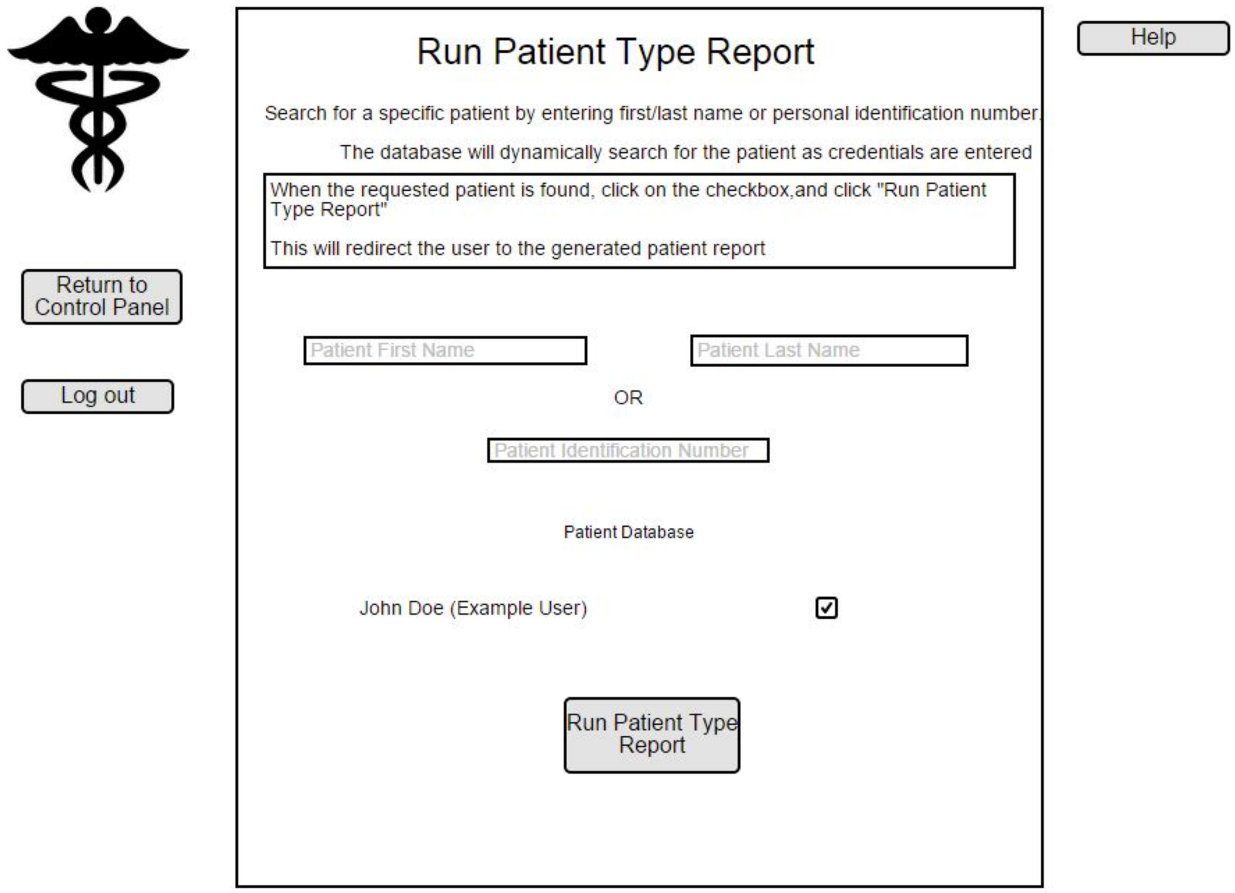
HSP Staff: Run Patient Population Report



Click "Run Patient Population Report" to generate patient population statistics on all patients registered in the current database.

- Staff can generate a patient population report by clicking “Run Patient Population Report” from their control panel. They will redirected to a separate confirmation page which will prompt them to click “Run Patient Population Report” again, which will generate patient population statistics.

HSP Staff: Run Patient Type Report



Run Patient Type Report

Search for a specific patient by entering first/last name or personal identification number.

The database will dynamically search for the patient as credentials are entered

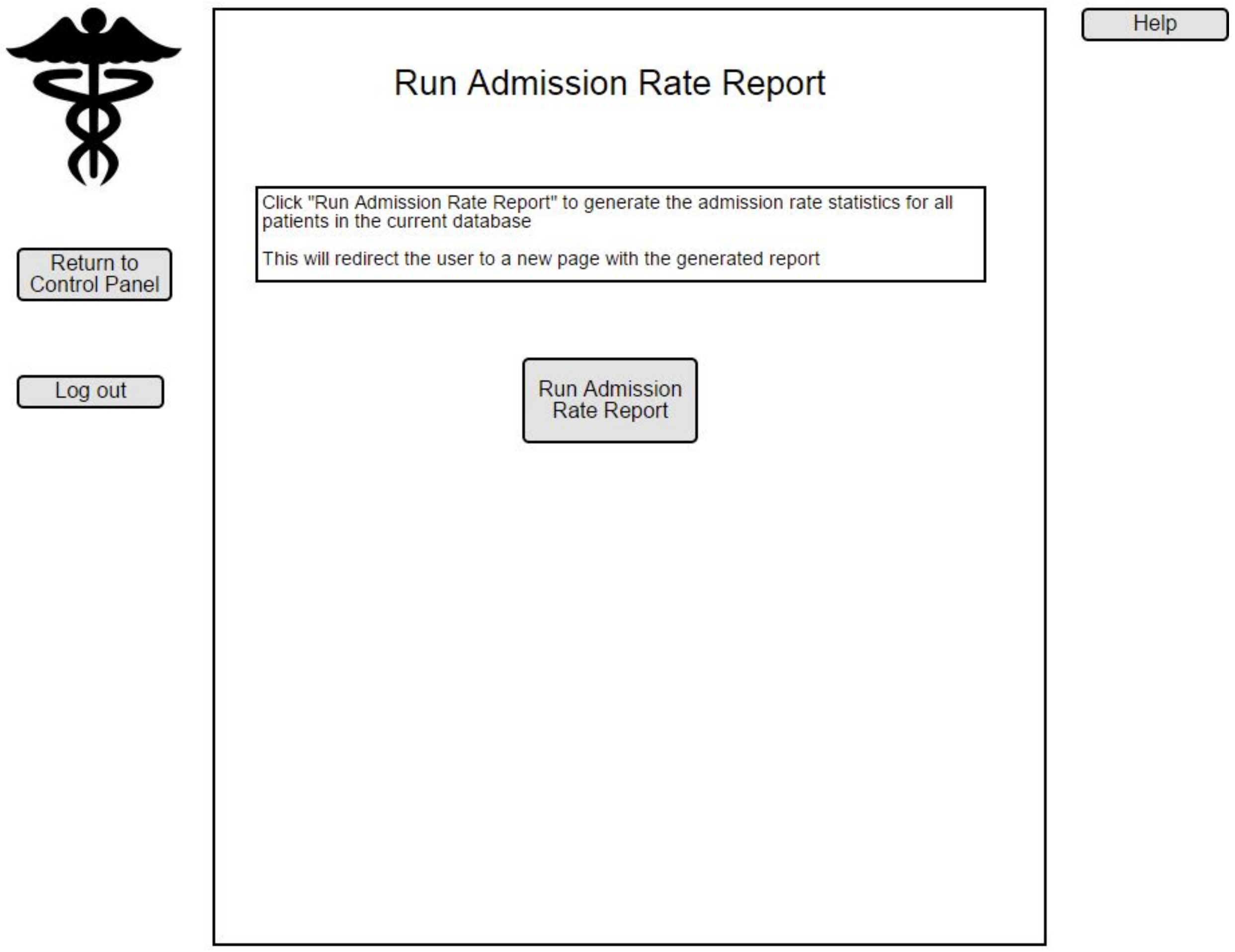
When the requested patient is found, click on the checkbox,and click "Run Patient Type Report"

This will redirect the user to the generated patient report

John Doe (Example User) ✓

- Staff can choose to “Run Patient Type Report” which will redirect to a separate page. The user (staff) will then have to find the patient in the database, and click “Run Patient Type Report” which will generate a report for the patient selected.

HSP Staff: Run Admission Rate Report

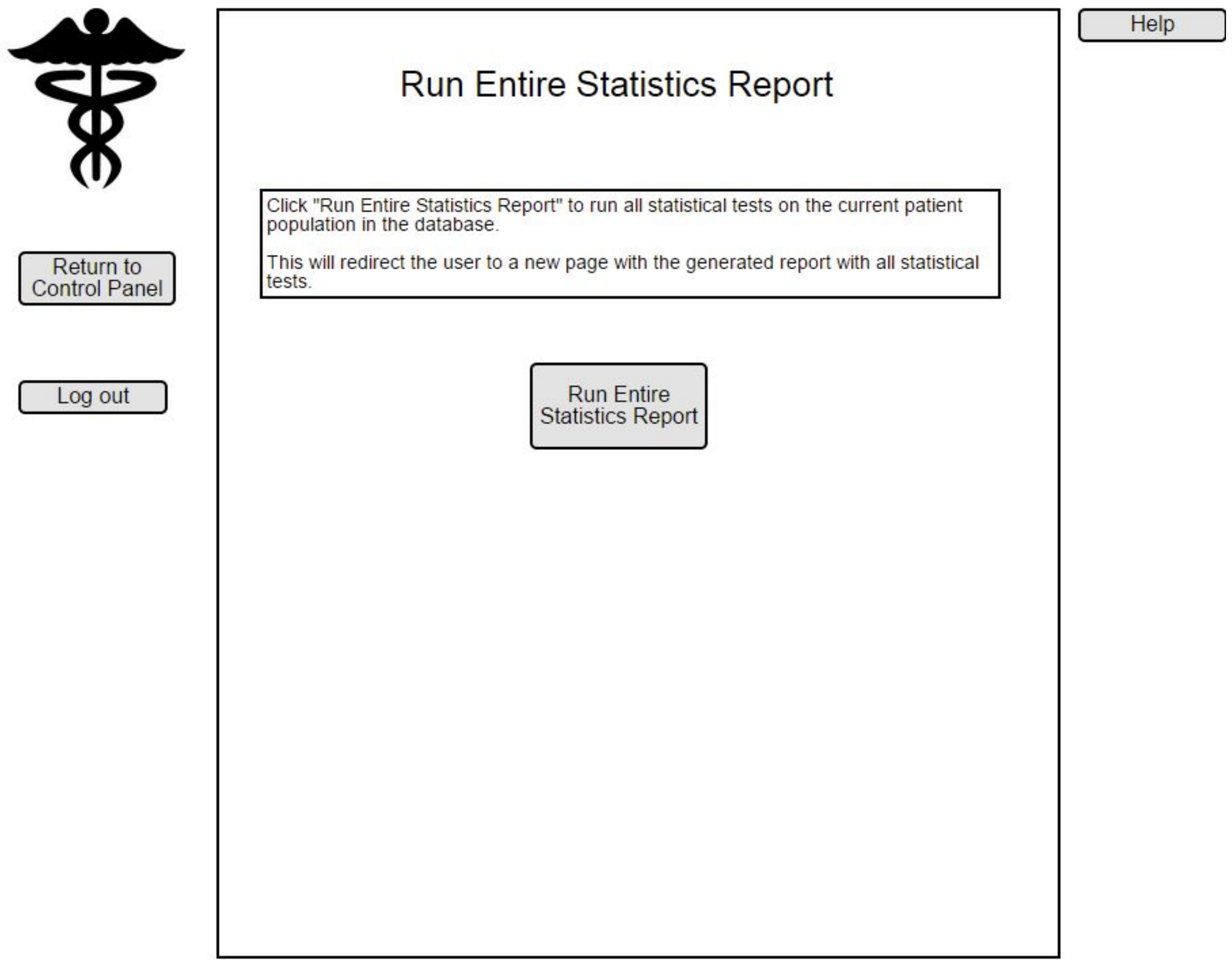


Click "Run Admission Rate Report" to generate the admission rate statistics for all patients in the current database

This will redirect the user to a new page with the generated report

- Staff can run a statistical admission rate report based on the admissions of all patients in the current database. They can do this by clicking “Run Admission Rate Report” from the control panel, which will then redirect them to a new page. From here, the user (staff) can click “Run Admission Rate Report” again which will automatically generate a report on the current standing admission rate of the database.

HSP Staff: Run Entire Statistics Report

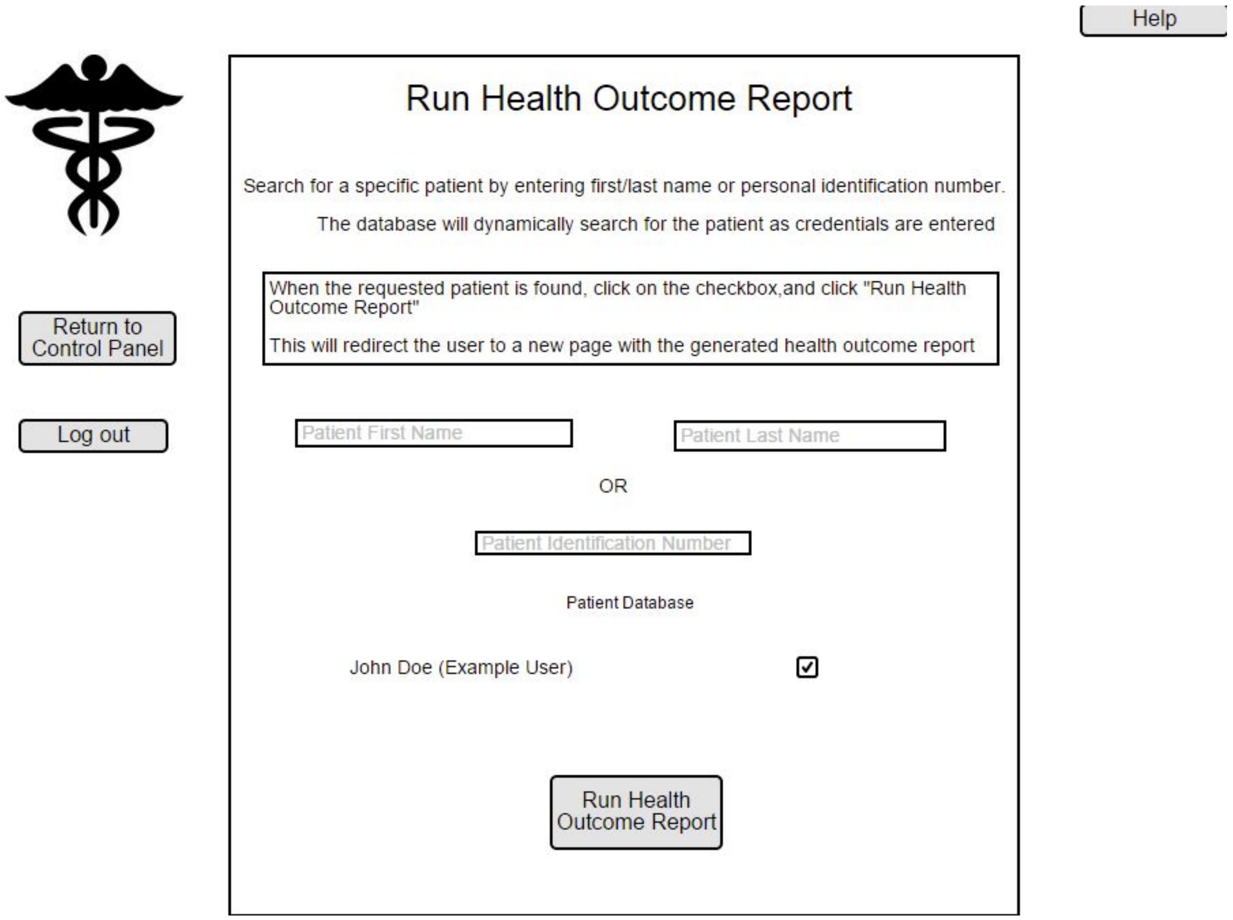


Click "Run Entire Statistics Report" to run all statistical tests on the current patient population in the database.

This will redirect the user to a new page with the generated report with all statistical tests.

- Staff can run an report which will conduct a comprehensive report on statistics (patient population, admission rate,etc) on the database. Staff can do this by clicking “Run Entire Statistics Report from the control panel, and then clicking it again on the separate page which they will redirected to from the control panel.

HSP Staff: Run Health Outcome Report


Help

Return to Control Panel

Log out

Run Health Outcome Report

Search for a specific patient by entering first/last name or personal identification number.

The database will dynamically search for the patient as credentials are entered

When the requested patient is found, click on the checkbox,and click "Run Health Outcome Report"

This will redirect the user to a new page with the generated health outcome report

Patient First Name

Patient Last Name

OR

Patient Identification Number

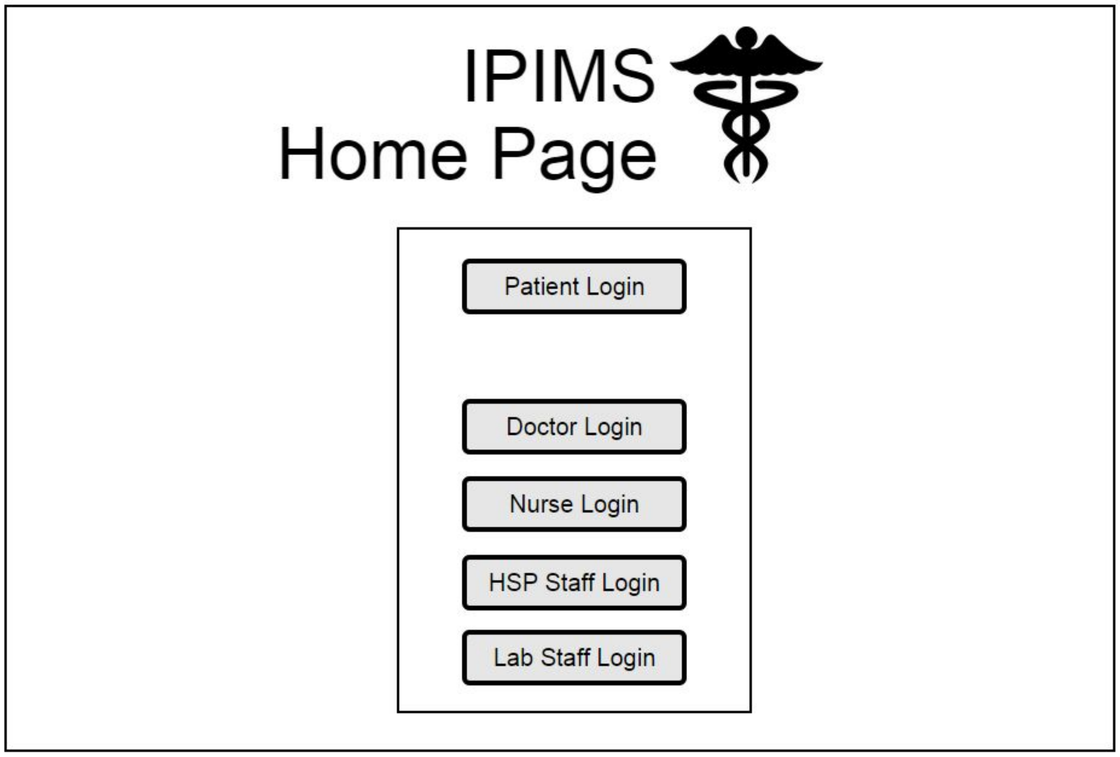
Patient Database

John Doe (Example User) ✓

Run Health Outcome Report

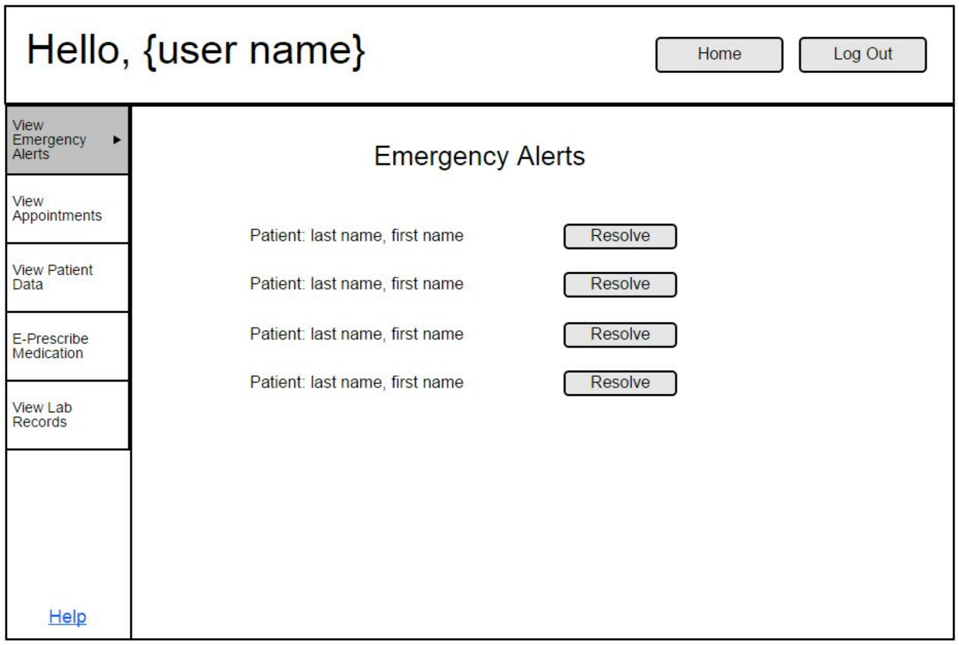
- Staff can run an report which will conduct reports on patients within the database. Staff can do this by clicking “Run Health Outcome Report” from the control panel.

IPIMS Home Page



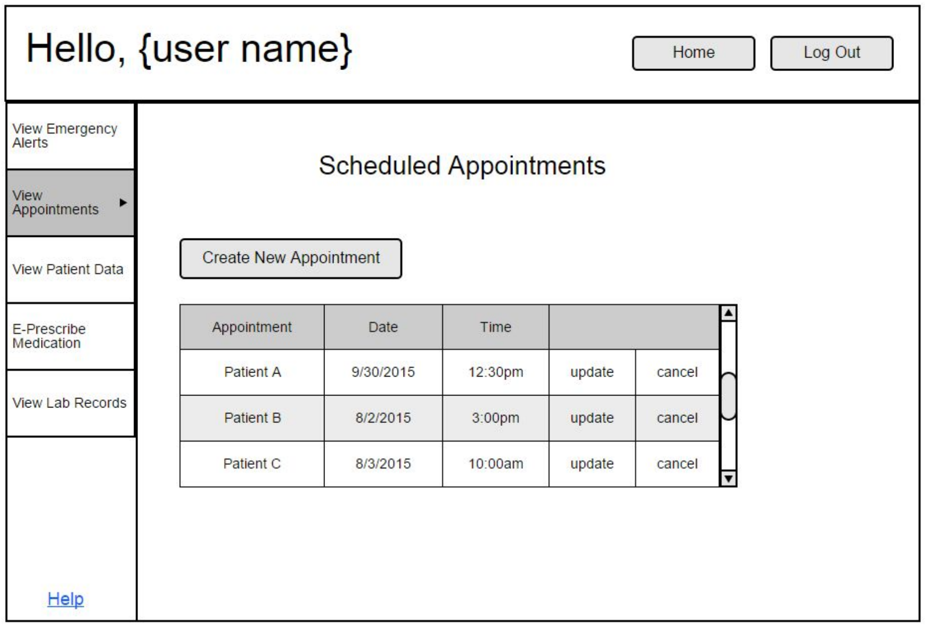
- The main portal page of the IPIMS system. Once the user has accessed this page, they may then access the login page for their role using the respective button.

Doctor: View Emergency Alerts



- When a doctor has logged in, they may select to *View Emergency Alerts*. The page will then display all current emergency alerts that are relevant to the doctor. Patient names will be displayed and a button to *Resolve* the case appears next to the names. The doctor may click this button to remove any cases from this page.

Doctor : View Scheduled Appointments



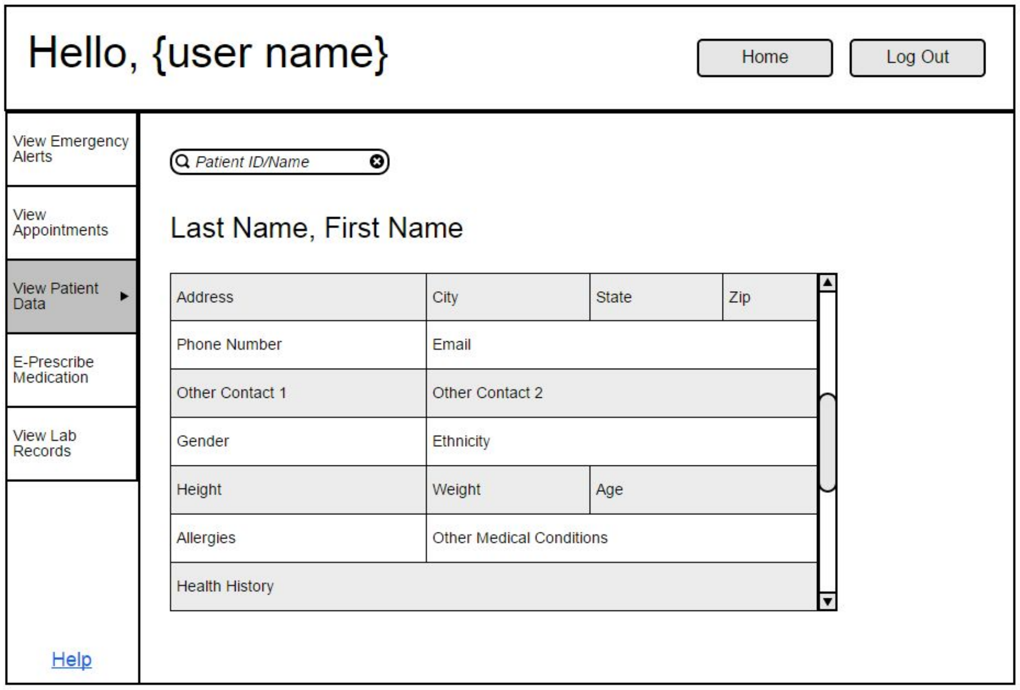
Hello, {user name}

Scheduled Appointments

Appointment	Date	Time		
Patient A	9/30/2015	12:30pm	update	cancel
Patient B	8/2/2015	3:00pm	update	cancel
Patient C	8/3/2015	10:00am	update	cancel

- When a doctor has logged in, they may select to *View Appointments*. The page will then display a list of all currently scheduled appointments with the doctor. The list is in order by date and time, and the doctor has the option to update and/or cancel any scheduled appointments from the chart. There is also a *Create New Appointment* button near the top that will allow the doctor to enter a new appointment entry into their schedule.

Doctor: View Patient Data



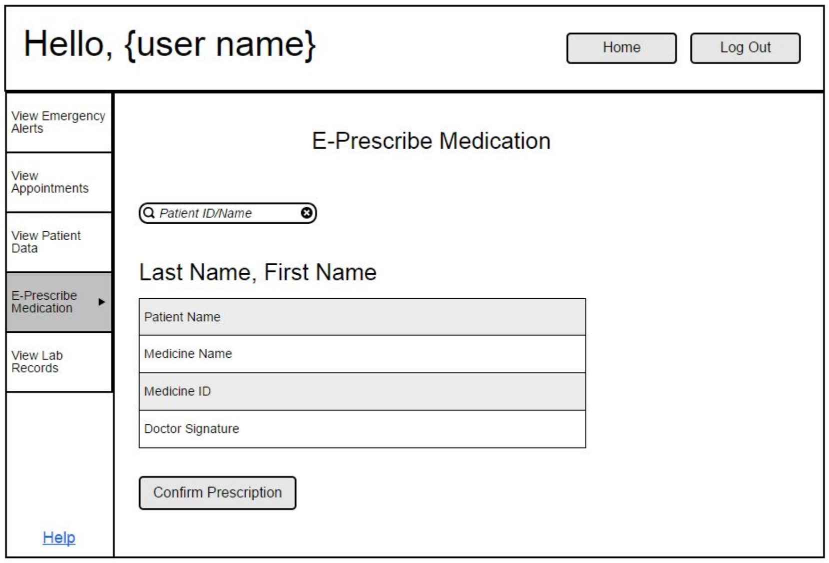
Hello, {user name}

Last Name, First Name

Address	City	State	Zip
Phone Number	Email		
Other Contact 1	Other Contact 2		
Gender	Ethnicity		
Height	Weight	Age	
Allergies	Other Medical Conditions		
Health History			

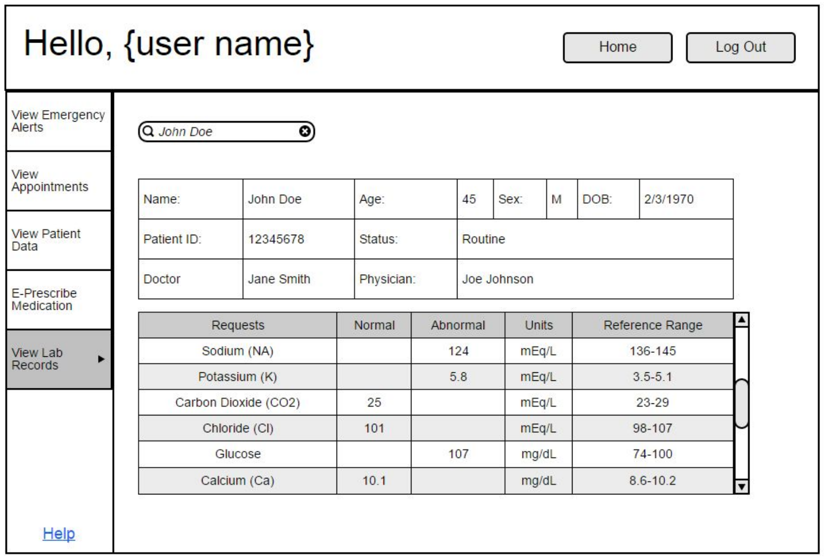
- When a doctor has logged in, they may select to *View Patient Data*. The page will then display a search bar where the doctor may enter a Patient's name or ID number. If the Patient's form exists within the database, it will be displayed on the screen. The form consists of the patient's personal information as well as medical conditions. If the Patient's form does not exist, the doctor will be notified that it could not be found. The doctor may choose to continue searching for more data forms by using the search bar again.

Doctor: E-Prescribe Medication



- When a doctor has logged in, they may select to *E-Prescribe Medication*. The page will then display a search bar. After the doctor has typed in a Patient's ID or name, they may press enter and if the Patient is found within the database, a prescription form will be displayed. In order to prescribe a medication, the doctor must fill out the form with the patient's name, medicine name, medicine ID, and the doctor's signature. After ensuring the information is correct, they then will press the confirm button at the bottom and the prescription will be finalized.

Doctor: View Lab Reports

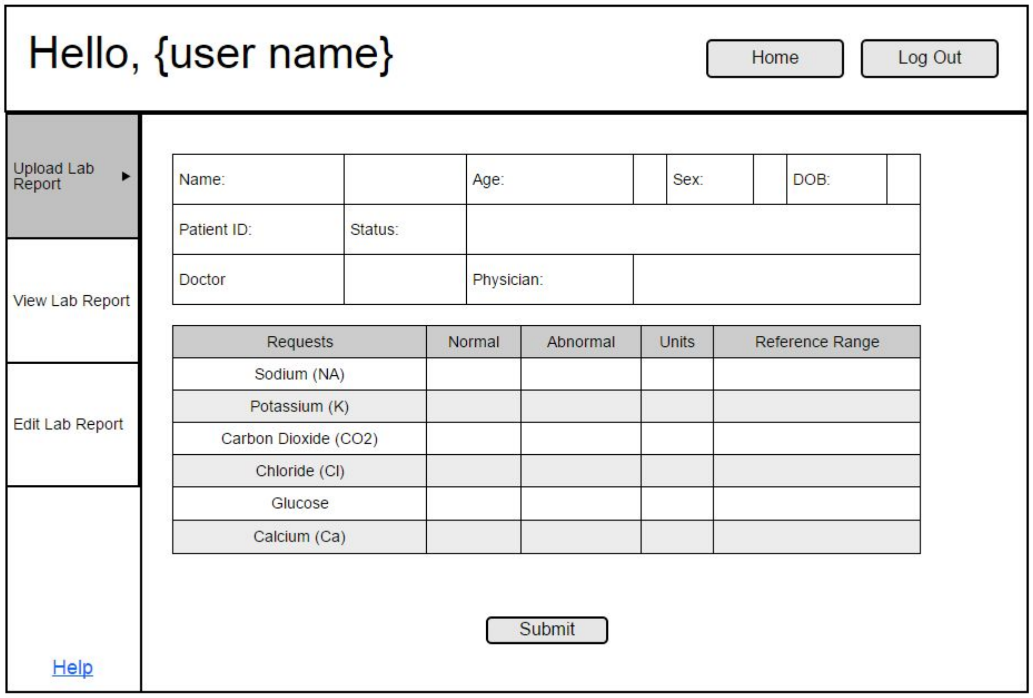


Name:	John Doe	Age:	45	Sex:	M	DOB:	2/3/1970
Patient ID:	12345678	Status:	Routine				
Doctor	Jane Smith	Physician:	Joe Johnson				

Requests	Normal	Abnormal	Units	Reference Range
Sodium (NA)		124	mEq/L	136-145
Potassium (K)		5.8	mEq/L	3.5-5.1
Carbon Dioxide (CO2)	25		mEq/L	23-29
Chloride (Cl)	101		mEq/L	98-107
Glucose		107	mg/dL	74-100
Calcium (Ca)	10.1		mg/dL	8.6-10.2

- When a doctor has logged in, they may select to *View Lab Reports*. The page will then display a search bar. After the doctor has typed in a Patient's ID or name, they may press enter and if the Patient is found within the database, their lab record will be displayed. The lab report can not be edited from this page.

Lab Staff: Upload Lab Report

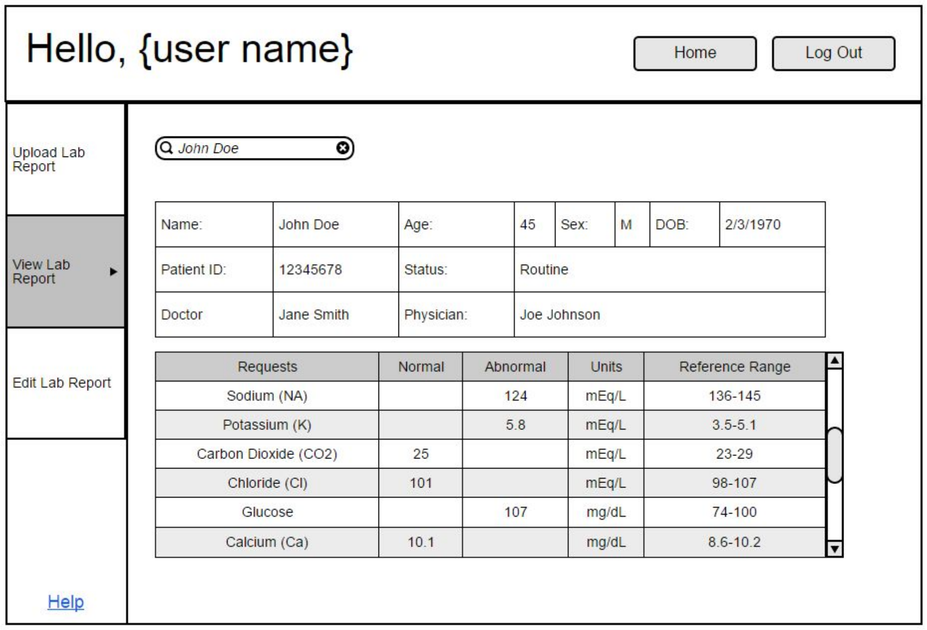


Name:		Age:		Sex:		DOB:	
Patient ID:	Status:						
Doctor		Physician:					

Requests	Normal	Abnormal	Units	Reference Range
Sodium (NA)				
Potassium (K)				
Carbon Dioxide (CO2)				
Chloride (Cl)				
Glucose				
Calcium (Ca)				

- When a lab staff employee has logged in, they may select to *Upload Lab Report*. The page will then display an empty form for a lab report. The lab staff employee is to complete the form and then submit it using the button near the bottom. The lab report will then be uploaded into the database.

Lab Staff: View Lab Report

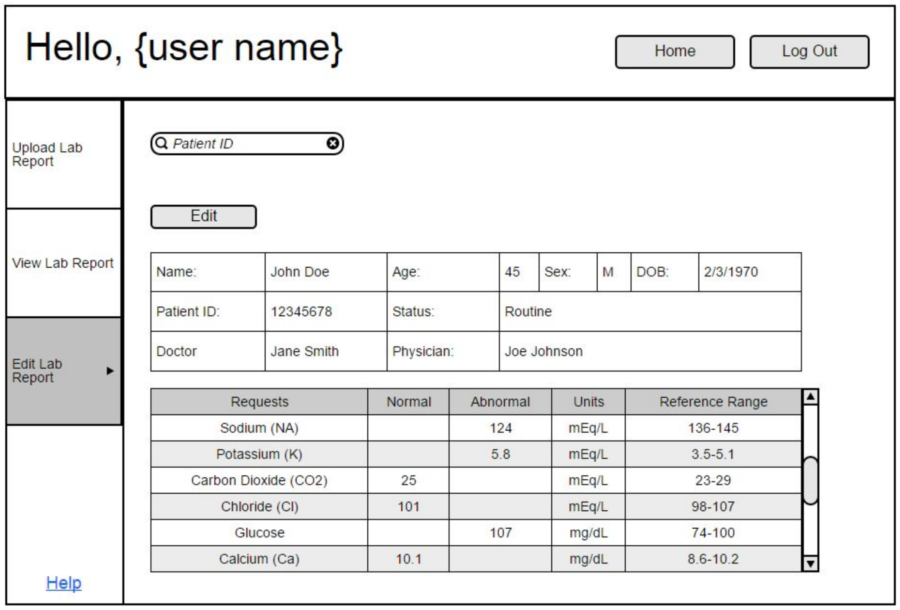


Name:	John Doe	Age:	45	Sex:	M	DOB:	2/3/1970
Patient ID:	12345678	Status:	Routine				
Doctor	Jane Smith	Physician:	Joe Johnson				

Requests	Normal	Abnormal	Units	Reference Range
Sodium (NA)		124	mEq/L	136-145
Potassium (K)		5.8	mEq/L	3.5-5.1
Carbon Dioxide (CO2)	25		mEq/L	23-29
Chloride (Cl)	101		mEq/L	98-107
Glucose		107	mg/dL	74-100
Calcium (Ca)	10.1		mg/dL	8.6-10.2

- When a lab staff employee has logged in, they may select to *View Lab Report*. The page will then display a search bar. After the lab staff has typed in a Patient's ID or name, they may press enter and if the Patient is found within the database, their lab record will be displayed. The lab report can not be edited from this page.

Lab Staff: Edit Lab Report

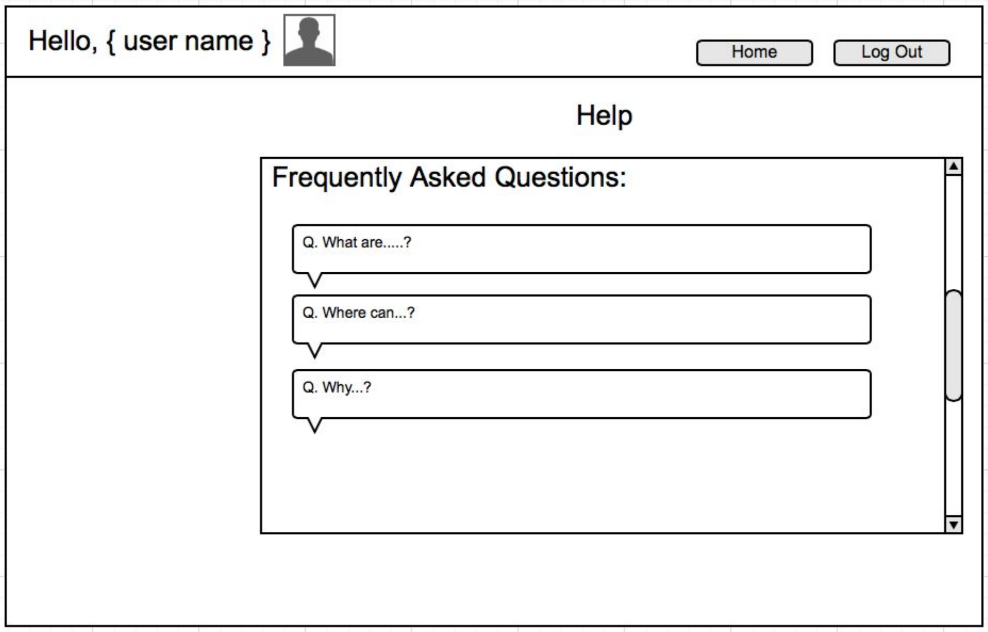


Name:	John Doe	Age:	45	Sex:	M	DOB:	2/3/1970
Patient ID:	12345678	Status:	Routine				
Doctor	Jane Smith	Physician:	Joe Johnson				

Requests	Normal	Abnormal	Units	Reference Range
Sodium (NA)		124	mEq/L	136-145
Potassium (K)		5.8	mEq/L	3.5-5.1
Carbon Dioxide (CO2)	25		mEq/L	23-29
Chloride (Cl)	101		mEq/L	98-107
Glucose		107	mg/dL	74-100
Calcium (Ca)	10.1		mg/dL	8.6-10.2

- When a lab staff employee has logged in, they may select to *Edit Lab Report*. The page will then display a search bar. After the lab staff has typed in a Patient’s ID or name, they may press enter and if the Patient is found within the database, their lab record will be displayed. From here, the lab staff can select the *Edit* button which will allow the information within the charts to be changed. Once all changes have been made, the lab staff should press the *Submit* button and the updated lab report will be uploaded to the database.

IPIMS Help



- Once the user has logged in they can select Help from the side bar. The “Help” page will provide the user with frequently asked questions and guides on navigating the website.

Section 7: Requirements Matrix

Testing ID from SRS	Title	Requirement Discription	Desired Feature	Existing Feature	Priority	Required Feature
3.2.1	Login	User logs into and is redirected to proper patient portal	X	X	High	X
3.2.2	Register	Patient can submit their information to the system for the HSP staff to verify and add into the system	X	X	High	X
3.2.3	Submit Medical Information/Registration	User medical information is properly submitted when added into the system	X	X	High	X
3.2.4	Update Health Record	User will be able to submit new health record data into the system to be added into their account	X	X	High	X
3.2.5	Alert IPIMS	Necessary alerts will alert the doctor from the patient in emergency situtations	X	X	High	X
3.2.6	Manage Appointments	Appointment manager to view and edit exisiting appts.	X	X	High	X
3.2.7	Schedule Appointment	Scheduling system to select the doctor and edit health conditions for review under the doctors care	X	X	High	X
3.2.8	View Appointment (Patient)	Ability to view each individual appt. patient has scheduled	X	X	High	X
3.2.9	Change Appointment	Ability to change the initial date for the appt.	X	X	High	X
3.2.10	Update Health Conditions (Nurse & Doctor)	Ability to update the health conditions after being evaluated	X	X	High	X
3.2.11	Help\Information	Help page is viewable by all users in the IPIMS	X	X	High	X
3.2.12	View Emergency Alerts	Doctors ability to view the emergency alerts sent by the patient	X	X	High	X
3.2.13	View Lab Reports (Doctor)	Ability for the doctor to view associated lab reports for the patient in question	X	X	High	X
3.2.14	Prescribe Medicine	Online medication perscription service available for the doctor	X	X	High	X
3.2.15	View Patient Data	Ability for the staff to search and view associated content for the user	X	X	High	X
3.2.16	Upload Medical Reports	Ability for the doctor and staff to access important medical documents associated with the patient in question	X	X	High	X
3.2.17	Update Medical History	Ability for staff to add new medical history	X	X	High	X
3.2.18	Retrieve Patient Medical Information	Ability to gather all the patient data in the ipims	X	X	High	X
3.2.19	Register Patient Information	Ability for the HSP to insert a new patient into the IPIMS system	X	X	High	X
3.2.20	Generate Statistics Report	Statistical report ran to analyze patient data	X	X	High	X
3.2.21	Upload Lab Report	Ability to upload lab report information after tests are done for the patient	X	X	High	X
3.2.22	View Lab Report (Lab Staff)	Ability for the lab staff to revert, change, view and analyze test results	X	X	High	X
3.2.23	Add Employee Data Into System	Ability for HSP to finalize the insertion of a patient into the IPIMS	X	X	High	X
3.2.24	Update Health Conditions/Severity Status	Allowing the doctor/nurse/user to change their health conditions or resolve a ca	X	X	High	X
3.2.25	View Appointment (Doctor)	Ability for doctor to view their currently scheduled queue of appts.	X	X	High	X

The matrix grid laid out above details on the different items specified in the SRS documentation that will be build, test, desire and prioritize throughout the development process. We have described each associated item under review listed with an associated ID to cross-reference with our documentation.

Section 8: Appendices

Appendix A

Acronyms

CSS	Cascading Style Sheets
HSP	Health Service Provider
HTML	HyperText Markup Language
IPIMS	Interactive Patient Information Management System

Appendix B

Glossary

CSS	Programming language for styling HTML.
Django	A free and open source web application framework.
HTML	Programming language used for creating webpages.
Javascript	A high level programming language supported by most web browsers.
JQuery	A free and open source software that uses a javascript library.
Python	A high level programming language used in Django.
SQLite	An in-Process library that implements a transactional SQL database engine.
Twitter Bootstrap	A front end web application framework.

Appendix C

References

Schachte, R., Duffy, K., & Huang, J., Peralta, J., Koshaba, R., Wei, M., Garcia, J., Duong, K., Le, S., Kariniemi, W. (2015). Interactive Patient Information Management System.