Year 2

Year Two Project

Semester 4

ITT Dublin | Old Blessington Rd, Dublin 24.

Design Specification

X00131787, X00101556.

Table of Contents

[Introduction 1](#_Toc475807190)

[Screen Design 2](#_Toc475807191)

[Input and Output Screens 2](#_Toc475807192)

[Login Screen 2](#_Toc475807193)

[Patient Screen 4](#_Toc475807194)

[Appointment Screen 6](#_Toc475807195)

[Patient Search Screen 8](#_Toc475807196)

[Change Patient Details Screen 9](#_Toc475807197)

[Prescribe Medication Screen 10](#_Toc475807198)

[Payment Screen 11](#_Toc475807199)

[View Chart Screen 12](#_Toc475807200)

[Database Design 13](#_Toc475807201)

[Entity Relationship Diagram 13](#_Toc475807202)

[Program Design 14](#_Toc475807203)

[Pseudocode 14](#_Toc475807204)

[Login 14](#_Toc475807205)

[Add Patient 15](#_Toc475807206)

[Change Patient Details 16](#_Toc475807207)

[Generate Invoice 17](#_Toc475807208)

[Make Appointment 19](#_Toc475807209)

[Admit Patient 20](#_Toc475807210)

[Discharge Patient 21](#_Toc475807211)

[Test Design 22](#_Toc475807212)

[Test Case Matrix 22](#_Toc475807213)

# Introduction

This document contains the input/output screens, the pseudocode for some of our key methods, the entity relationship diagram for the system and the test matrices for our key methods. We decided that the visual representation of the system was going to focus on the functionality of the system, as it would be only intended for use by the staff of a hospital. The pseudocode briefly outlines the logic behind the methods intended for the system. Our entity relationship diagram, or ERD, outlines the logical links between the entities intended for the system and how they relate to one another. Finally, our test matrices are the templates which we intend to use for the testing of our system upon reaching full functionality, which will be a system which correctly implements the originally mentioned concepts in the User Requirements Document.

# Screen Design

## Input and Output Screens

### Login Screen

User enters username and password supplied to them by the administrator to log into the HPMS system.



Figure – Login Screen

If details are correct the main system is loaded notifying the user that they have been successfully logged in. Main content and Side Navigation are tailored to which level of user logs into the system.

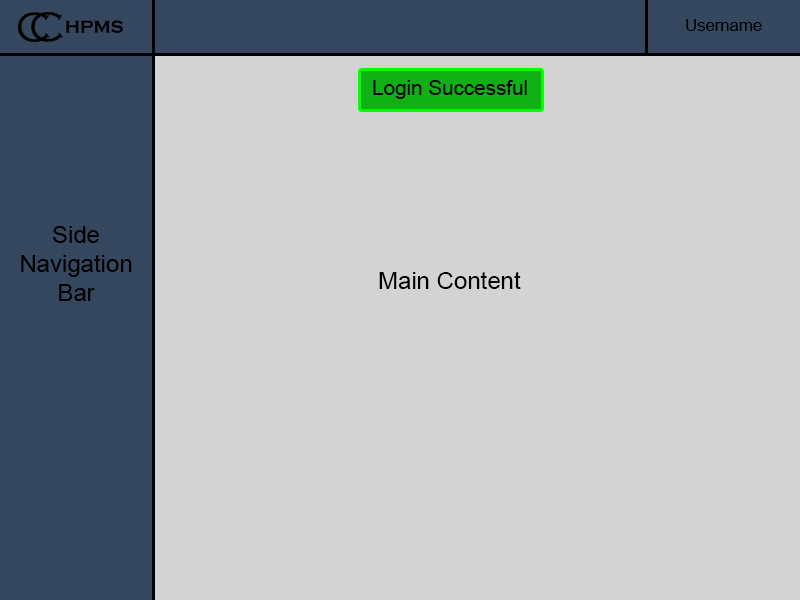


Figure – Login Screen Successful

If the login is incorrect (the username and password do not match what is in the database) an error screen is displayed notifying the user.

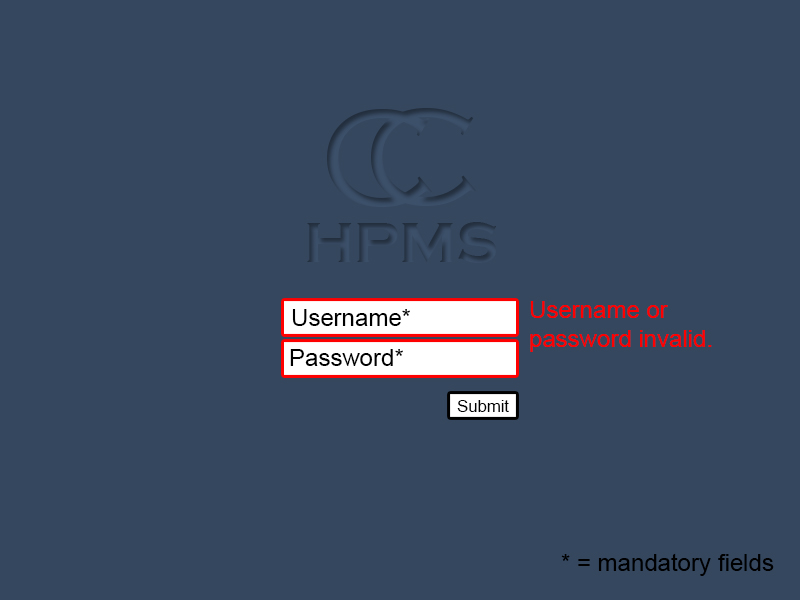


Figure – Login Screen Fail

### Patient Screen

User uses the add a patient link in the side navigation bar (If they have appropriate access to do so) A form is displayed with mandatory fields, non-mandatory fields and checkboxes that will create the patient object.

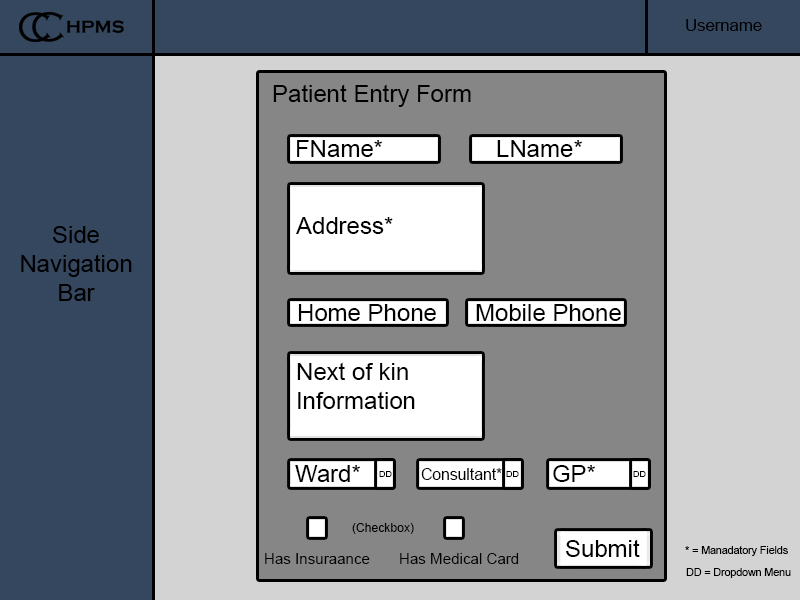


Figure – Patient Screen

If the ward is not currently full and information has been entered correctly the user will be returned to the home page notifying them that the patient has been added successfully.

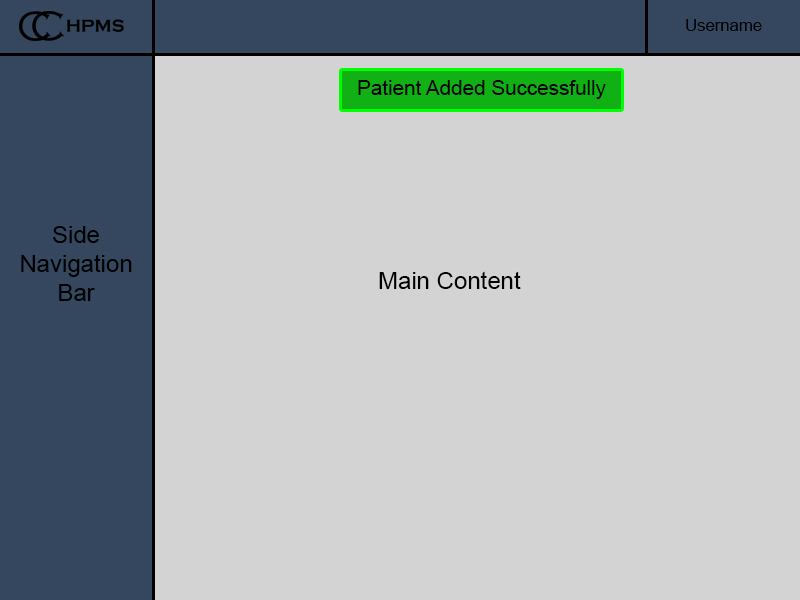


Figure – Patient Screen Successful

If there are errors in the form the field is indicated where the error is.

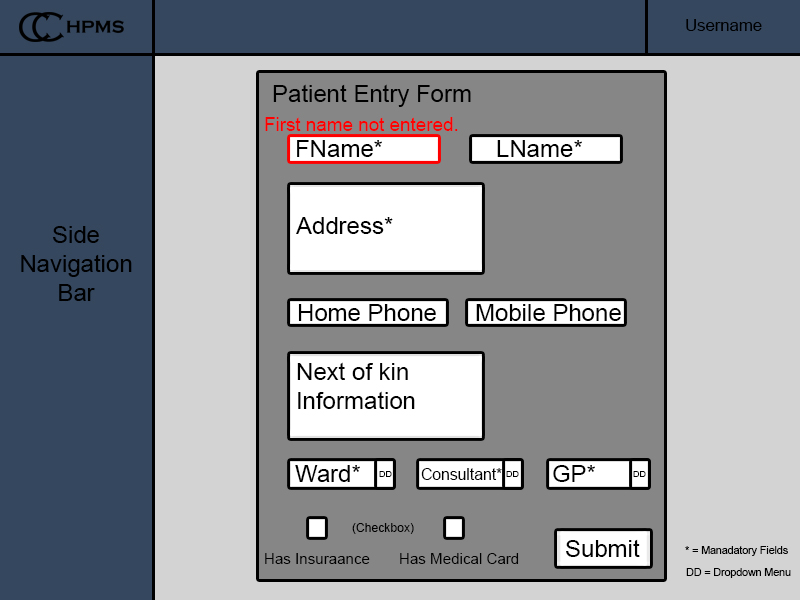


Figure – Patient Screen Error

If the ward is full the patient will be added to the system and a notification is displayed notifying the user that the patient has been placed in the standby list.

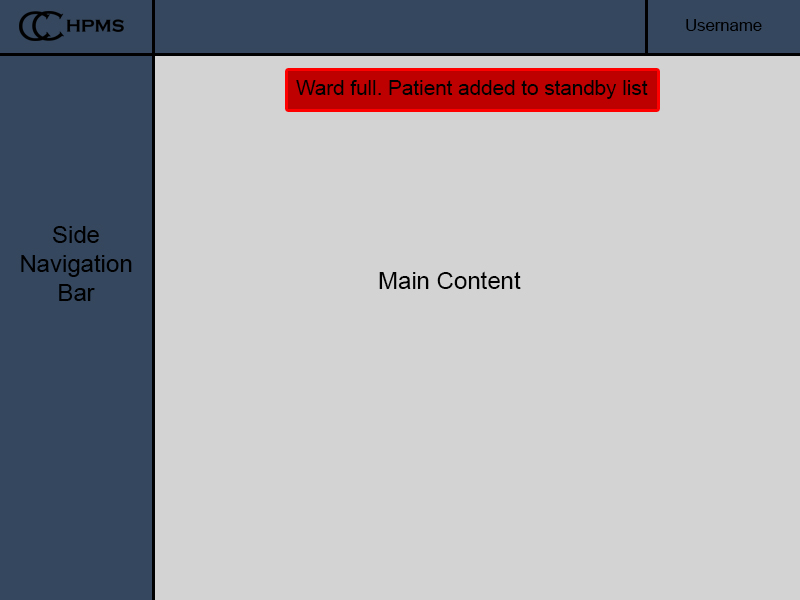


Figure – Patient Screen Ward Full

### Appointment Screen

When manually creating an appointment, all fields will be left blank, if this page is accessed through a patient object (clicking make an appointment on the patient information page) Patient and Consultant will automatically be added and the only fields that will need to be filled will be date and time.

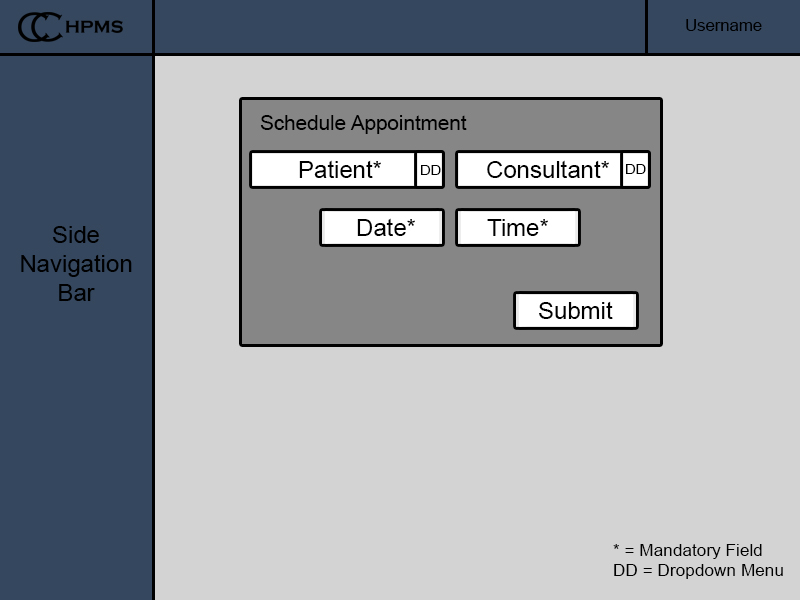


Figure – Appointment Screen

Successful creation of an appointment will return the user to the home screen with a notification that it has been created. Unsuccessful will return the user to the Schedule Appointment page notifying them what they did wrong (No patient/consultant, conflict with date and time etc.)

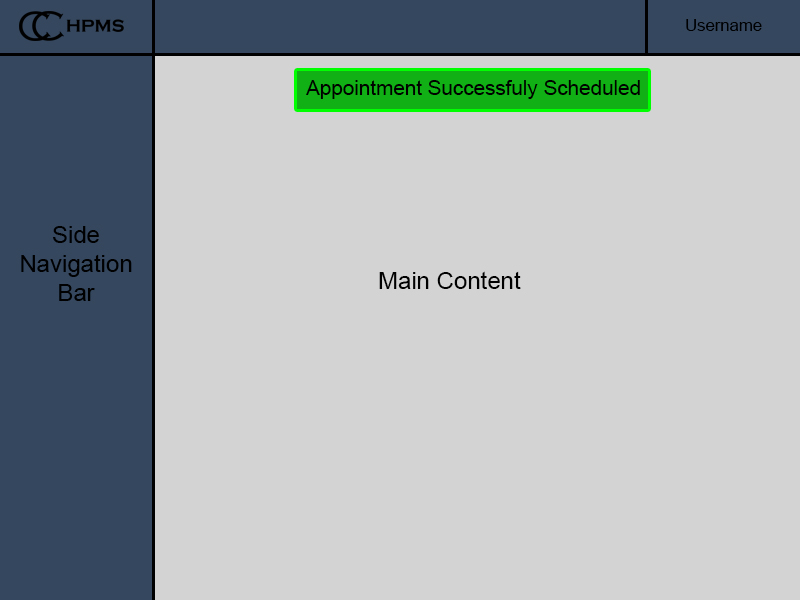


Figure – Appointment Successful

If there is an error the field that has an error is indicated.

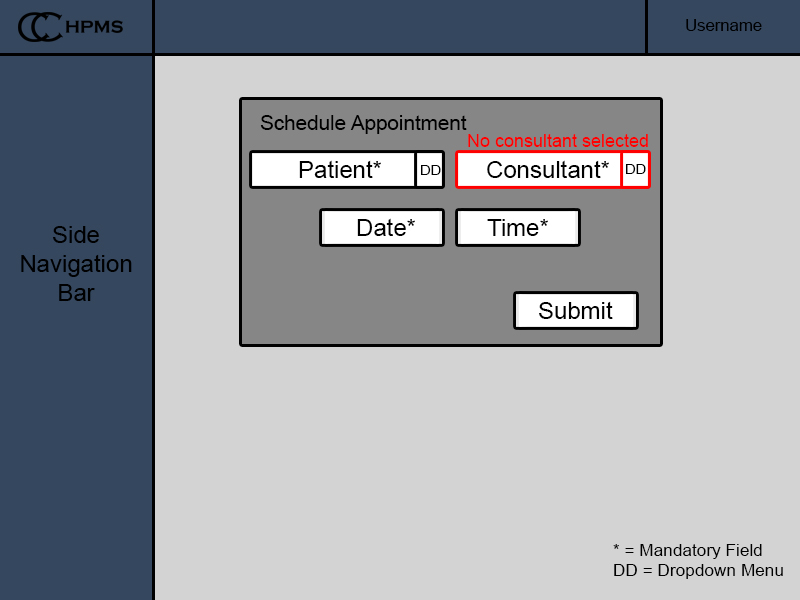


Figure – Appointment Screen Error

### Patient Search Screen

To allow access to every patient in the hospital a robust way of searching will be implemented. Allowing the user to filter by such things as; name, ward, consultant and illness, they can also search for individual patients by name or MRN. It will also provide a link next to that person bringing the user to the patient’s full information and further functionality for that individual patient.

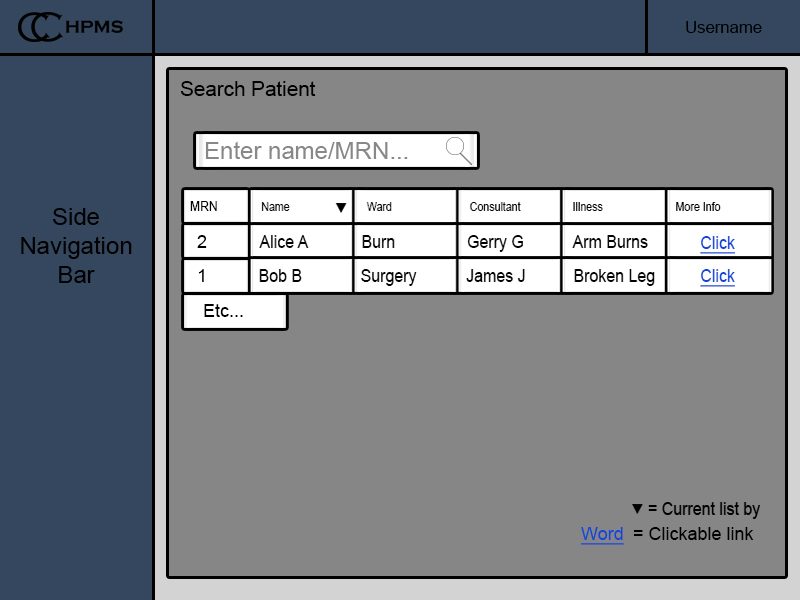


Figure – Patient Search Screen

When the link is clicked next to a patient name this page will appear with all the functionality needed to manage the patient.

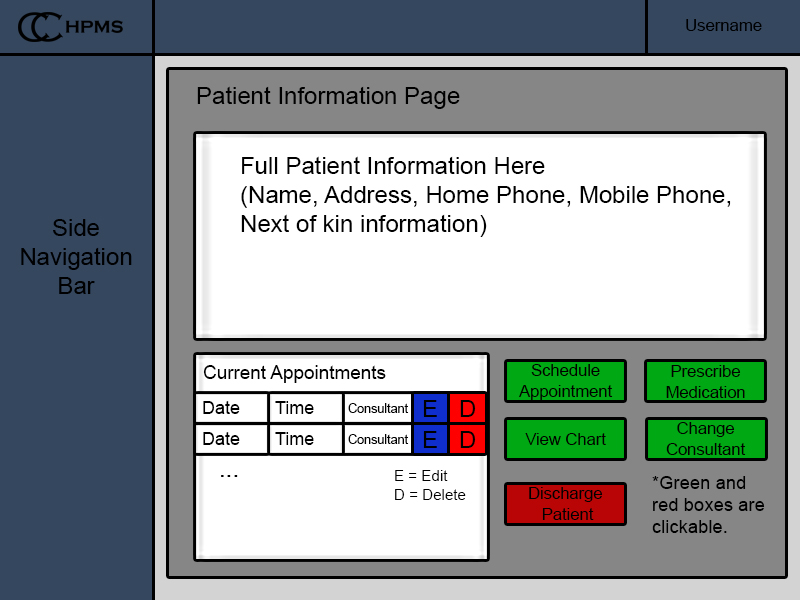


Figure – Patient Information Screen

### Change Patient Details Screen

If basic information for a patient is not correctly entered the change details page can be accessed to fix this.

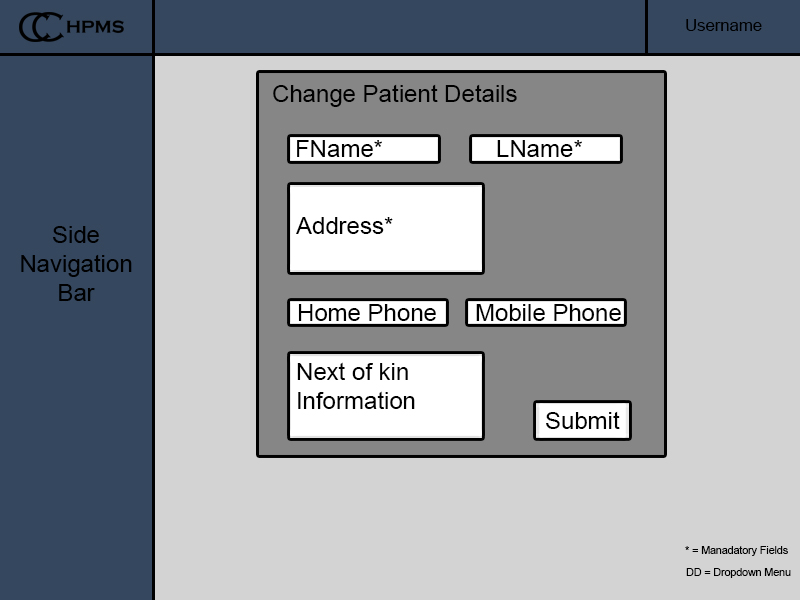


Figure - Change Details Screen

When the submit-button is clicked the patient details are changed and the user is returned to that patient’s information screen.

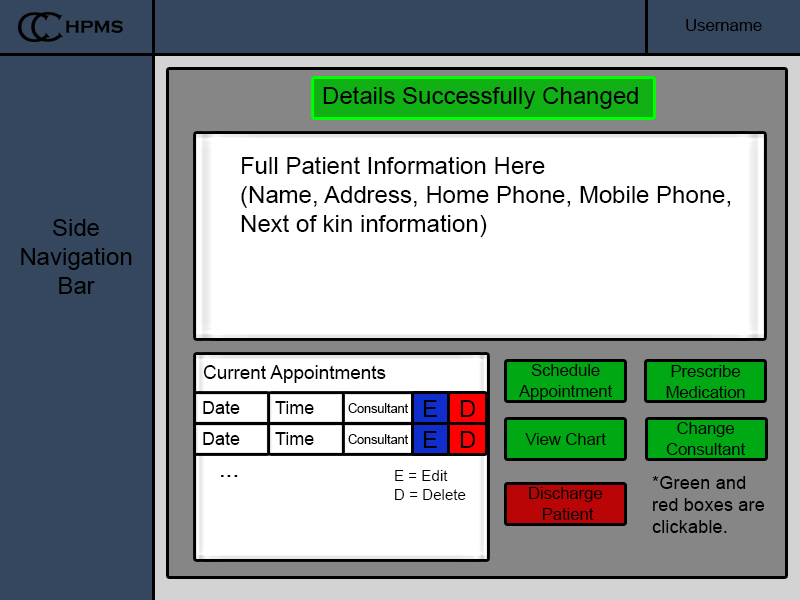


Figure - Change Details Screen Successful

### Prescribe Medication Screen

This page can be accessed manually through the navigation bar. Or through the specific patient’s pages. It is used for prescribing medication that is stored in the database to the patient. The dosage is manually entered.

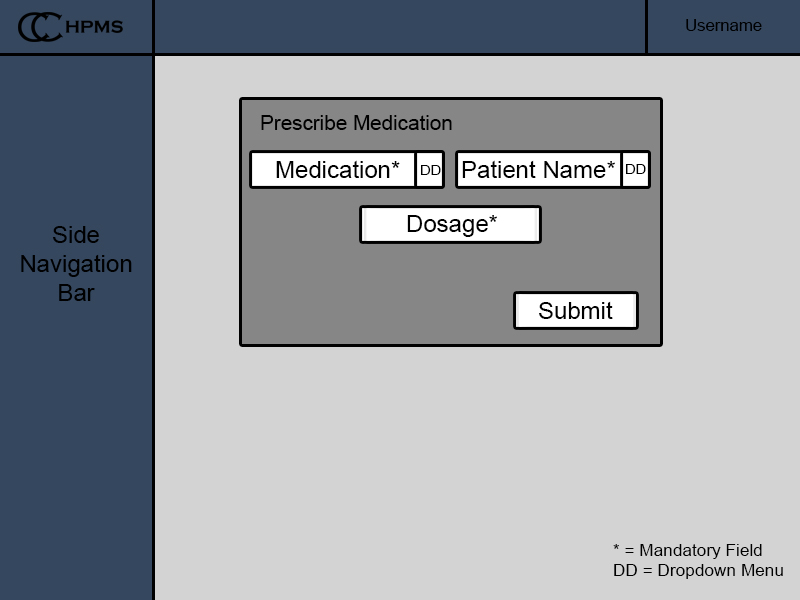


Figure - Prescribe Medication Screen

When the submit-button is hit the user is redirected back to the patient’s information screen.

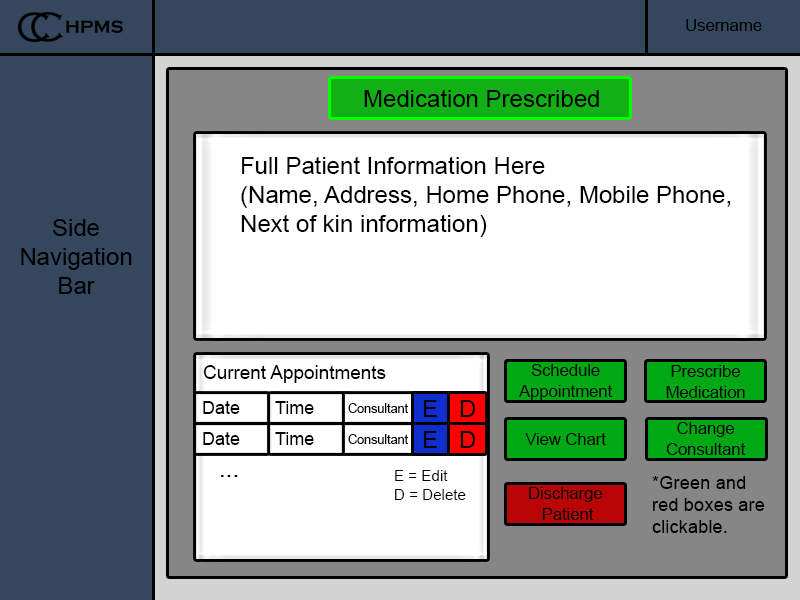


Figure – Prescribe Medication Screen Successful

### Payment Screen

The payment screen can be accessed after the patient has been discharged it has two options one for printing out of the bill to be physically sent to the patient and another option to email the patient their bill.

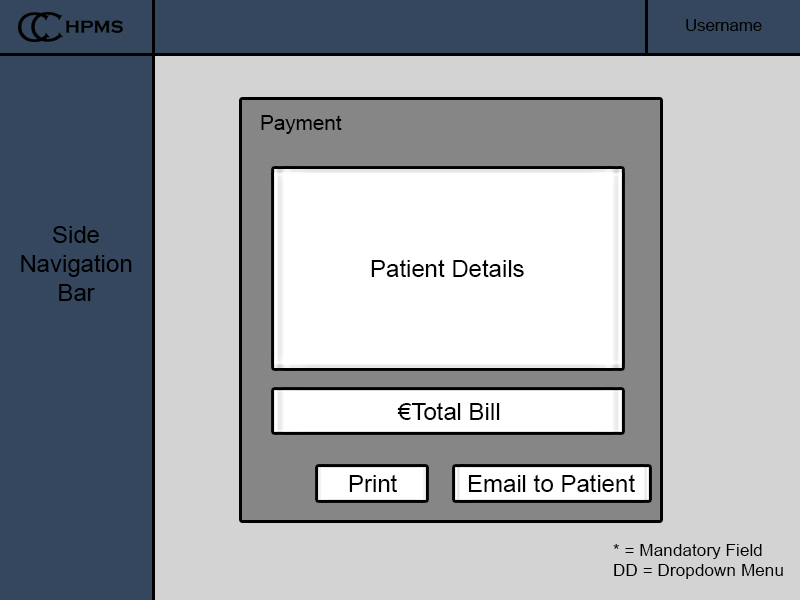


Figure - Payment Screen

The output screen lets the user know that the bill has been sent successfully.

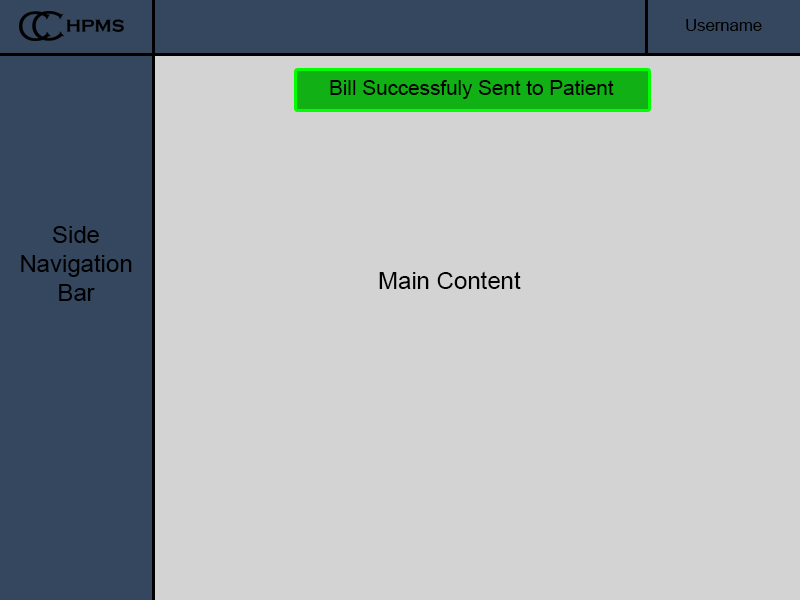


Figure - Payment Screen Successful

### View Chart Screen

This screen will be an overview of the patient’s chart information.

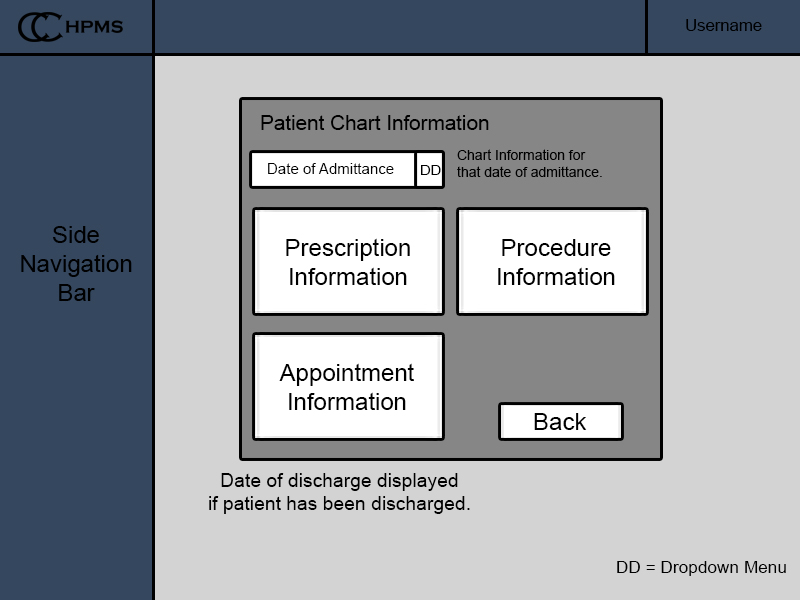


Figure - Chart Information Screen

Clicking the back button will return the user to the patient’s information page.

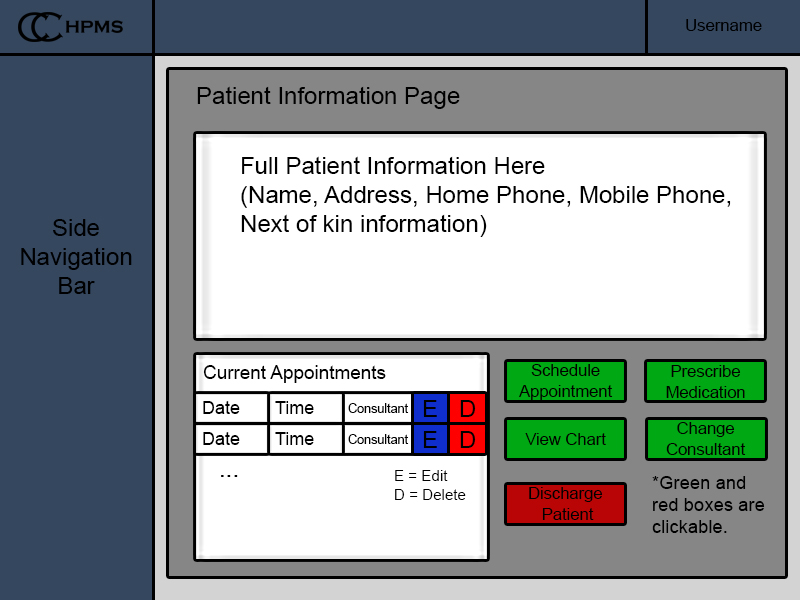


Figure - Chart Information Screen Output

# Database Design

Entity Relationship Diagram

Figure – Entity Relationship Diagram

We designed the Entity Relationship diagram with the focus being on the system holding all information in relation to the patient. Pricing of prescriptions, appointments, procedures and visits were set as static rates until we have achieved full functionality. We will then consider altering the relational database to accommodate this. For functionality purposes, we decided to plan on patients having multiple charts, each dictating the bill for that stay. All charts will be accessible through an array list in the patient class, allowing for us to compile all charts for a given patient if need be. We also considered the concept of one patient having many consultants, but felt that for the first stage it would be more practical to limit that to a one-to-many relationship until we have achieved full functionality.

# Program Design

## Pseudocode

### Login

Login(String username, String password)

**Inputs:**

username: String

password: String

**Variables:**

None

**Outputs:**

consultantProfile: View

searchPatient: View

**Pseudocode**

if(username and password are authenticated)

if(username exists in Consultant table)

Display(consultantProfile.view)

else

Display (searchPatient.view)

EndIfElse

return badRequest(render loginForm.view)

### Add Patient

addPatient(Form patientForm)

**Inputs:**

patientForm: Form

**Variables:**

patient: Patient

**Outputs:**

None

**Pseudocode**

if (patientForm has errors(input mismatch validations))

Display(render(addPatient.view)

EndIf

patient = patientForm.makePatient()

save patient -> database;

flash("Patient has been added")

Display(searchPatient.view)

### Change Patient Details

changePatientDetails(String mrn)

**Inputs**

mrn: String

**Variables:**

patient: Patient

patientForm: Form

**Outputs:**

updatePatient: view

**Pseudocode**

patient = find patient by mrn

patientForm = formFill(Patient.class).fill(patient)

if(BadRequest)

Display(“Error”)

endIf

Display(updatePatient.view))

### Generate Invoice

generateInvoice(Chart chart)

**Inputs**

chart: Chart

**variables:**

double costOfAppointment

double costOfProcedure

double CostOfMedication

double medCalc

doulbe durationOfStay;

double overNightCost;

double invoiceAmount

**Outputs:**

None

**Pseudocode**

if (medicalCard is false)

amount += (noOfAppointments \* costOfAppointment)

EndIf

if (medicalCard is false)

for(number of prescriptions)

medCalc += (costOfMedication \* dosage)

amount += medCalc

amount += costOfProcedure

amount += (overNightCost \* durationOfStay)

EndIf

if (insurance type IS NOT "self")

switch(insuranceType)

case "basic":

amount -= BASIC

break;

case "standard":

amount -= STANDARD

break

case "DayToDay":

amount -= DAY\_TO\_DAY

break

case "enhanced":

amount -= ENHANCED

break

case default:

break

if (amount <= 0)

return 0

else

return amount

EndIfElse

### Make Appointment

makeAppointment(Patient p, Consultant c, Date d, Time t)

**Inputs:**

Patient: p

Consultant: c

Date: d

Time: t

**Variables:**

Appointments[] consultantAppointments

**Outputs:**

addAppointment: view

**Pseudocode:**

Appointments[] consultantAppointments = c.getAppointments()

For(Appointment a : consultantAppointments)

If(a.getTime() == t && a.getDate() == d)

Display(“Scheduling clash”, addAppointment.view)

EndIf //If statement checking if Consultant has appointment already for that date/time.

EndFor

Appointment a = new Appointment(p, c, d, t)

Save Appointment a -> database

### Admit Patient

admitPatient(Patient p)

**Inputs:**

Patient: p

**Variables:**

String: Illness

**Outputs:**

Index: view

**Pseudocode:**

Ward w = Ward.find.byCategory(p.getIllness)

//Ward contains stand by list for that ward. (sl) and an array of patients currently in that ward (wardPatients[])

If(w.wardpatients.size < w.getCapacity)

w.wardPatients.add(p) //Adding to array of patients in ward.

flash(“Patient successfully added to ward ” + w.getCategory)

Display (index.view)

else

w.sl.addToStandbyList(p) //Adding patient to standby list of that ward.

flash(“Ward full, patient added to standby list of ward ” + w.getCategory)

Display(index.view)

EndIfElse

### Discharge Patient

dischargePatient(Patient p)

**Inputs:**

Patient p

**Variables:**

None

**Outputs:**

View: Index

**Pseudocode:**

If(p.hasAppointments)

Print(“Cannot discharge patient while appointments scheduled.”)

Display(index.view)

Else

(Patient p -> Discharged Patients table)

(Patient p removed from Patient table)

Flash(“Patient successfully discharged. Patient added to discharged list”)

Display(index.view)

EndIfElse

# Test Design

## Test Case Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Test Case Matrix** | | | | | | | **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** | | 1 | Login | Correct username and password (Consultant) | Render(medicalProfile.view) |  |  | | 2 | Login | Correct username and incorrect Password | Error incorrect username |  |  | | 3 | Login | Incorrect Username and Incorrect Password | Error incorrect password |  |  | | 4 | Login | Username and Password (staff) | Render(searchPatient.view) |  |  | |  |  |  |  |  |  | | 1 | Change Details | Fname, lname, address, homephone, mobilephone, nextOfKin info, ward, GP, insurance, medicalCard | Render(searchPatient.view)/ Patient information updated |  |  | | 2 | Change Details | Value left blank | Error all values must be filled |  |  | | 3 | Change Details | Invalid characters entered into field | Error that is not a valid number/address, email, name |  |  | | 4 | Change Details | GP left unselected | Error please choose a GP |  |  | | 5 | Change Details | Consultant left unselected | Error please choose a Consultant |  |  | |  |  |  |  |  |  | | 1 | Search Patient | Mrn(String), fname and lname (String), phone number (String) | List of Patients with correct information |  |  | | 2 | Search Patient | No fields filled | Error please enter a search parameter |  |  | | 3 | Search Patient | Unregistered MRN | “There are no patients with that MRN please try again” |  |  | | 4 | Search Patient | Generic Name | Error too many results returned please narrow your search |  |  | | 5 | Search Patient | Phone number with to many or to few digits | Error that is not a valid number please try again |  |  | |  |  |  |  |  |  | | 1 | Make Appointment | Consultant (Object<Consultant>) | Flash(“Appointment Confirmed (Consultant Name, time and date)”) |  |  | | 2 | Make Appointment | Time and date of appointment (Time and Date) | Flash(“Appointment Confirmed (Consultant Name, time and date)”) |  |  | | 3 | Make Appointment | No data | Error you must choose a time and date or a Consultant |  |  | | 4 | Make Appointment | Time and date (Time and Date) | “Sorry there are no available consultants at this time” |  |  | | 5 | Make Appointment | Consultant (Object<Consultant>) and Time and date of appointment (Time and Date) | “Sorry there are no available consultants at this time” |  |  | | 6 | Make Appointment | Consultant (Object<Consultant>) and Time and date of appointment (Time and Date) | Flash(“Appointment Confirmed (Consultant Name, time and date)”) |  |  | |  |  |  |  |  |  | | 1 | Generate Invoice | Chart (Object <Chart>) | Amount owed (Double) and breakdown of costs (Procedure cost, appointment cost, prescription cost and cost of stay) |  |  | | 2 | Generate Invoice | Chart (Object <Chart>) | Flash(“Patient Invoice is fully covered”) and breakdown of costs (Procedure cost, appointment cost, prescription cost and cost of stay) |  |  | |  |  |  |  |  |  | | 1 | Admit Patient | Patient (Object <Patient>) | Flash(Patient Admitted)/ |  |  | | 2 | Admit Patient | Patient (Object <Patient>) | Flash(No available beds, Patient added to Standby List) |  |  | | 3 | Admit Patient | Patient (Object <Patient>) | Error patient has already been admitted |  |  | | 4 | Admit Patient | Patient (Object <Patient>) | “Patient was discharged today, are you sure?” |  |  | |  |  |  |  |  |  | | 1 | Prescribe Medication | Medication (Object <Medication>) | Flash(Medication prescribed) |  |  | | 2 | Prescribe Medication | Medication (Object <Medication>) | “This medication will conflict with current medication. Are you sure?” |  |  | |