BSHC4 – Cloud Computing

Cloud Application Development

Project

Smart Health Care-Patient Management System

John Cahill

X1378581

Contents

[Introduction 2](#_Toc512120606)

[Development Strategy 2](#_Toc512120607)

[Database Design 2](#_Toc512120608)

[Implementation 3](#_Toc512120609)

[Design Patterns 6](#_Toc512120610)

[User Interface 7](#_Toc512120611)

[Visibility of system status 7](#_Toc512120612)

[Match between system and the real world 7](#_Toc512120613)

[User control and freedom 7](#_Toc512120614)

[Consistency and standards 7](#_Toc512120615)

[Error prevention 7](#_Toc512120616)

[Recognition rather than recall 7](#_Toc512120617)

[Flexibility and efficiency of use 7](#_Toc512120618)

[Aesthetic and minimalist design 7](#_Toc512120619)

[Help users recognize, diagnose, and recover from errors 7](#_Toc512120620)

[Deployment 7](#_Toc512120621)

[Reference’s 8](#_Toc512120622)

# Introduction

I have developed a SHC-PMS (Smart Health Care-Patient Management System). SHC-PMS is an information system that is intended for use in various clinics / centres in Ireland. The patient record can be accessed in any clinic / centre all over the Ireland. Doctor’s will have their own login allowing them to access their own patients to hold any illnesses patients may have to a better degree. When the Doctor is on the home page, they will also have access to all the patients held by all the different Doctor’s. They can edit or view individual patients and they also have the authority to destroy the record if the patient has been seen too. When creating/editing the patients, there is a status that is entered if the patient has been “seen to” or is still “unseen”. At the top of the page above the patient’s information, will be information to notify the current Doctor how many patients are left to be seen to.

# Development Strategy

## Database Design

The two databases used in this project are for the patients, being the created records by the doctors and the users which is for the doctor’s to sign up. They are as following,

Users:

1. user\_id
2. email
3. password

devise\_parameter\_sanitizer.permit(:sign\_up, keys: [:firstname, :lastname, :email, :password, :remember\_me])

Patients:

1. firstname
2. lastname
3. dateofbirth
4. address
5. phonenumber
6. infection
7. injury
8. status

rails g scaffold\_controller Patient firstname:string lastname:string dateofbirth:integer address:string phonenumber:integer infection:string injury:string status:Boolean

rails g migration add\_user\_id\_to\_patient user\_id:integer

The relationship between the two databases is the user\_id is a foreign key used in every patient created. How the relationship works is for every user there is one Doctor/user and for every Doctor/user, there is many patients.

## Implementation

Functionality – Register

* **Model**

/project/app/models/user.rb

* **View**

/project/app/views/devise/registrations/new.html.erb

/project/app/views/devise/menu/\_registration\_items.html.erb

/project/app/views/users/show.html.erb

* **Controller**

/project/app/controllers/users\_controller.rb

/project/app/controllers/application\_controller.rb

Functionality – Log In

* **Model** –

/project/app/models/user.rb

* **View**

/project/app/views/devise/menu/\_login\_items.html.erb

/project/app/views/users/show.html.erb

* **Controller**

/project/app/controllers/users\_controller.rb

/project/app/controllers/application\_controller.rb

Functionality – Edit Registration

* **Model**

/project/app/models/user.rb

* **View**

/project/app/views/devise/registrations/edit.html.erb

/project/app/views/devise/menu/\_registration\_items.html.erb

/project/app/views/users/show.html.erb

/project/app/controllers/application\_controller.rb

* **Controller**

/project/app/controllers/users\_controller.rb

/project/app/controllers/application\_controller.rb

Functionality – New Patient

* **Model**

/project/app/models/patient.rb

* **View**

/project/app/views/patients/new.html.erb

/project/app/views/patients/\_form.html.erb

* **Controller**

/project/app/controllers/patients\_controller.rb

Functionality – Edit Patient

* **Model**

/project/app/models/patient.rb

* **View**

/project/app/views/patients/edit.html.erb

/project/app/views/patients/\_form.html.erb

/project/app/controllers/application\_controller.rb

* **Controller**

/project/app/controllers/patients\_controller.rb

Functionality – Search Patient

* **Model**

/project/app/models/patient.rb

* **View**

/project/app/views/patients/index.html.erb

/project/app/controllers/application\_controller.rb

* **Controller**

/project/app/controllers/patients\_controller.rb

Functionality – View Profile

* **Model**

/project/app/models/user.rb

* **View**

/project/app/views/users/show.html.erb

/project/app/controllers/application\_controller.rb

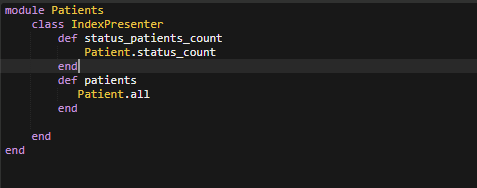
* **Controller**

/project/app/controllers/users\_controller.rb

## Design Patterns

The pattern I have chosen is Presenter’s, it is designed to move template logic to dedicated classes so that templates will have less code making them appear cleaner and more object orientated. A presenter is a class which will work with both the model and the view, it can be used on complex logic in templates and controllers to help clean the code by extracting the code out into the created class which will hold one or more methods. This makes it simpler to test controller logic when changing its assigned instance variable in a controller action into its own method within a presenter class.

In my project you can view an example from “/project/app/presenters/patients/index\_presenter.rb”. The methods are held in index\_presenter.rb.



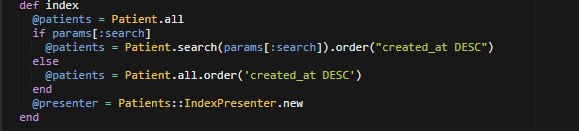
In “/project/app/controllers/patients\_controller.rb” the controller where the methods where taken from and replaced with a caller which summons “/project/app/presenters/patients/index\_presenter.rb”. This is how the controller’s logic is more clean while having its methods contained in the presenter class.

Then finally in the view found at “/project/app/views/patients/index.html.erb”, this shows how the methods are included or called in the view to run the project properly.



The figure above shows how the presenter is summoned to use the required method.

You can also view how it is being used in the original controller where it now calls on “/project/app/presenters/patients/index\_presenter.rb”. This can be found here “/project/app/controllers/patients\_controller.rb” and has the following inside.



The last tag being the code which connects the controller to index\_presenter.rb

## User Interface

### Visibility of system status

When the user is using the system, the system will notify them on what they need to do, and users are always up to date. When using the logs from Heroku, shows the feedback being under a second for its response to the user.

### Match between system and the real world

The order the user uses the system will feel intuitive and it is also in English.

### User control and freedom

User has full control wherever they are on the system, allowing them to go back to previous, or even go to where they want to easily.

### Consistency and standards

The consistency holds through for the entire system, giving any information needed to avoid confusion.

### Error prevention

The system is set up carefully, so the user has no access to parts of the system that may cause errors. Such as creating a patient before logging in. The user had to enter login or register before being allowed to make any necessary changes.

### Recognition rather than recall

When patients are made or edited, they can be viewed easily by user including their own information. Having accessibility to all the data at the click of a button depending on what they are looking for.

### Flexibility and efficiency of use

For any forms or parts information is inserted to the system, it will remember the data input, allowing the user to re-use any needed information making the system more efficient for the user and easier to navigate.

### Aesthetic and minimalist design

Any information provided for the user is precise and to the point allowing them to navigate through the system quicker.

### Help users recognize, diagnose, and recover from errors

Error messages are all in English in plain text, allowing the user to realise their mistake and make appropriate changes.

# Deployment

The cloud platform used to host this project was Heroku, I also used PostgreSQL database for Heroku to get my project online and available on the cloud. The link is the following (“https://immense-woodland-15045.herokuapp.com/”).

# Reference’s

Nielsen Norman Group. (2018). *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. [online] Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ [Accessed 21 Apr. 2018].