**BSHC4 – Cloud Application Development Project**

**Doctor-Patient Management System**

**Surendra Dura**

**15007669**

Project submitted in partial fulfilment of the BSc in Computing at the National College of Ireland, delivered by Dr Keith Maycock

Contents

[1. Introduction (20 marks) 3](#_Toc5001530)

[2. Development Strategy 3](#_Toc5001531)

[(I) Database Design (10 marks) 3](#_Toc5001532)

[(II) Implementation (20 marks) 5](#_Toc5001533)

[**(III) Design Patterns (10 marks)** 7](#_Toc5001534)

[3. Testing (20 marks) 7](#_Toc5001535)

[4. Deployment (20 marks) 8](#_Toc5001536)

[5. Bibliography 8](#_Toc5001537)

# 1. Introduction (20 marks)

Doctor-Patient management system is a health application developed using Ruby on rails which help the doctor to manage the patient details.

The main functionalities of the system area as follows:

1. Add Patient record – allow the doctor to add patient record with information like first name, last name, dob, email, phone, gender, address, condition, level.
2. View/Edit Records – allow the login admin to view the doctor all the doctor record, doctor to view the patient details and edit their details as well as patient details.
3. Destroy Records – doctor to allow the destroy the patient details
4. Sign up /login – allow to make the account or login in the system for the doctor.
5. Shifted Decorator – which help to add the extra method called Blood test, X-ray, CT scan, Endoscopy.
6. Add doctor record – the login used can added the doctor details.

# 2. Development Strategy

## (I) Database Design (10 marks)

In the rails, when we used the rails scaffold command or when we developed the model for the table. After that rake dB: migrate command help use to developed the database table file inside the db./migration folder.

Relationship:

Doctor has many patients.

User has many doctor.

1. Doctor table:

create\_table "doctors", force:: cascade do |t|

t.string "firstname"

t.string "lastname"

t.string "email"

t.string "phone"

t.string "address"

t.string "specialist"

t.datetime "created\_at", null: false

t.datetime "updated\_at", null: false

t.integer "user\_id"

t.index ["user\_id"], name: "index\_doctors\_on\_user\_id"

end

1. Patients table:

create\_table "patients", force: :cascade do |t|

t.string "firstname"

t.string "lastname"

t.date "dob"

t.string "email"

t.string "phone"

t.string "address"

t.integer "doctor\_id"

t.datetime "created\_at", null: false

t.datetime "updated\_at", null: false

t.string "sex"

t.string "gender"

t.string "condition"

t.integer "level"

t.index ["doctor\_id"], name: "index\_patients\_on\_doctor\_id"

end

1. Shifted Patient Table

create\_table "shifteds", force: :cascade do |t|

t.string "firstname"

t.string "lastname"

t.date "dob"

t.string "email"

t.string "phone"

t.string "address"

t.string "firm"

t.decimal "cost"

t.string "description"

t.datetime "created\_at", null: false

t.datetime "updated\_at", null: false

end

1. Users Table

create\_table "users", force:: cascade do |t|

t.string "email", default: "", null: false

t.string "encrypted\_password", default: "", null: false

t.string "reset\_password\_token"

t.datetime "reset\_password\_sent\_at"

t.datetime "remember\_created\_at"

t.datetime "created\_at", null: false

t.datetime "updated\_at", null: false

t.boolean "admin", default: false

t.index ["email"], name: "index\_users\_on\_email", unique: true

t.index ["reset\_password\_token"], name: "index\_users\_on\_reset\_password\_token", unique: true

end

## (II) Implementation (20 marks)

Tis section contains details on different functionalities in the application.

**Functionality 1: Patient Record:** allow the doctor to add, edit or destroy the patient details

The add patient Record request is implemented using an MVC approach. The functionality is

implemented in the following files:

• Model: here list the name of the patient.rb file/files of your model/models used to implement this

functionality

• View: here list the name of the view views/patients folder (index.html.erb files) used to implement this functionality

• Controller: here list the name of the controller (patients\_controller .rb file) used to implement this

functionality

Path: <http://localhost:3000/doctors/1/patients>

**Functionality 2: Doctor Record:** allow the login current user to add the doctor details.

The add doctor Record request is implemented using an MVC approach. The functionality is

implemented in the following files:

• Model: here list the name of the doctor.rb file/files of your model/models used to implement this

functionality

• View: here list the name of the view views/doctors folder (index.html.erb files) used to implement this functionality

• Controller: here list the name of the controller (patients\_controller .rb file) used to implement this

functionality

Path: <http://localhost:3000/doctors>

**Functionality 3: Shifted Decorator:** allow the user to develop a shifted patient details with the extra function.

The add shifted patient Record request is implemented using an MVC approach. The functionality is

implemented in the following files:

• Model: here list the name of the shifted.rb file/files of your model/models used to implement this

functionality

• View: here list the name of the view views/shifteds folder used to implement this functionality

• Controller: here list the name of the controller (shifteds\_controller .rb file) used to implement this

Functionality

Path: <http://localhost:3000/shifteds>

**Functionality 4: User Registration and Login on the Site:** allow the user to register or login in the system.

The registration or login in the system is implemented using an MVC approach. The functionality is

implemented in the following files:

• Model: here list the name of the user.rb file/files of your model/models used to implement this

functionality

• View: here list the name of the view views/devise folder used to implement this functionality

• Controller: here list the name of the controller (sessions\_controller .rb file) used to implement this

Functionality

Path: <http://localhost:3000/doctors>

**Functionality 5: Check the fee of patient:** allow the patient to check the fee on the based of level

The check the fee is implemented using an MVC approach. The functionality is

implemented in the following files:

• Model: here list the name of the patient.rb file/files of your model/models used to implement this

functionality

• View: here list the name of the view views/Patient folder used to implement this functionality

• Controller: here list the name of the controller (patient\_controller .rb file) used to implement this Functionality

For this method, the custom gem is used inside the folder gem-feeSuggestion.

Path: <http://localhost:3000/doctors/1/patients>

## **(III) Design Patterns (10 marks)**

Decorator Pattern is implemented to allow a user to shift the patient and select the

optional extras come on in the system (Blood test,Xray,ctscan,endoscopy). The pattern for this functionality is implemented in the file shifted\_decorator.rb (located in the folder lib) and is used in the file shifteds\_controller.rb.

# 3. Testing (20 marks)

**The rails support different type of testing which are as follows:**

**Unit testing: performed on our models**

**Rails test:models to run this testing**

**The file is developed inside the test folder caller user\_test.rb.**

**Functional testing: performed on our controllers**

**Rails test:controllers to run this testing**

**Integration testing: performed for the end users**

# 4. Deployment (20 marks)

The review of 3 suitable PaaS solutions clearly highlighting the advantages

and disadvantages of each service are:

* Heroku app only can run when the code is push in the git hub but the local service can run.
* Heroku use the postgres database where as local used sqlite.

I used c9 editor for code to push in the git hub and the Heroku.

<https://healthappmg.herokuapp.com/users/sign_in>

For admin:

email:suren.dura@yahoo.com

password:dura123

The name of the database service used is Postgres database.

# Bibliography

Softcover.io. (2017). *Ruby on Rails Tutorial (Rails 5)*. [online] Available at: https://www.railstutorial.org/book/beginning#sec-deploying [Accessed 23 Jul. 2017].

YouTube. (2017). *Deploying a Ruby on Rails Application to Heroku*. [online] Available at: https://www.youtube.com/watch?v=mabGJ-vuABc [Accessed 23 Jul. 2017].