Report on Global Health System Application

Name: Trupti Kulkarni	Name: Apurv Deshpande
Student No: x15012948	Student NO: x15001687
https://damp-cove-8247.herokuapp.com/	https://desolate-everglades-6265.herokuapp.com

***Abstract*— The Application is aimed at providing online medical appointment facility for users to book an appointment at a hospital with specific doctors. The application maintain the information about the patient their profiles, doctor and hospital.**

Keywords—Patient; Doctor; Appoitnment; Hospital

I. INTRODUCTION

The project is a cloud-based application to facilitate appointment system for users. This application stores the data about the user and helps them to create appointments.

II. NEED FOR CLOUD

In old System to get an appointment at a hospital was a tedious job. Also the data stored previously was in a computer in an excel format. Hence due to mismanagement there was a chance that the data can get lost. But now all the data can be stored in cloud. Also the track on all the appointments can be easily maintained, as the information will be available in the cloud.

Also the amount of the data is very massive which is very difficult to maintain on on-premise application which require much cost in maintaining the security.

To avoid all these factors, cloud based application is the best option.

III. SYSTEM REQUIREMENT SPECIFICATION

The main modules of our project are:

1) Doctor module: This deals with all the appointments made by patients. This help to generate bills to patients based on specific complications. Respective Doctors can view the appointment made by patients with them.
2) Patient module: Patients can make appointments with their respective doctors. They can be classified into three types:
    a. First time Patients
    b. Follow Up Patients
    c. Routine Check up Patients
3) Appointment module: It provides appointment of patient with doctor on specific date and time. It has a list of doctors from where a patient can chose a doctor and books an appointment
4) User module: A user is the one who create an account in the application. He has to provide email and address. He has to register a profile. Then he can make appointment of himself as a patient or he can create appointment on behalf of some other patient.

IV. SDLC METHODOLOGY

This application implements the software development life cycle. To generate this application we use "SPIRAL MODEL"

SPIRAL MODEL: Steps of Implementation

1. All the software requirements for our application were defined in details considering aspects of existing application.
2. Using the design of application we created a prototype of application.
3. Taking into consideration the strengths, weakness, risks and defining the additional requirements we created the second prototype stage.
4. Then we tested the second prototype and iterated it to get the desired application.

Based on the refined prototype, the final system was constructed, tested and evaluated

V. Ruby on Rails Framework:

To build this application we are using MVC architecture.We are using Ruby on Rails Framework to design this application.

Advantages of Ruby On Rails Frameworks:

1. Ruby of Rails works effectively as other languages by writing very small code compared to other languages.
2. By writing a very less code you can develop an application.
3. Its long list of tools can help you write a very complex application easily.
4. In ruby on rails the code basically explains itself. Its like writing in a language you are comfortable with from your childhood.
5. There is lot of external libraries that we can use using the rails.
6. Ruby on Rails coding does not provide direct access to the database through coding which in turn becomes more secure in terms of attacks like SQL injection and Code injections.

We use Ruby on Rails for our application because:

1. It with the MVC architecture easily provides us with the relationship between the tables doctor, patient and appointments.
2. External gem like devise help us with providing authentication and authorization so that only registered users can use this application.

Advantages of MVC framework:

1. It can be used to create a 3 tier system. Business logic is totally separated from the view.
2. A single piece of data can be given multiple view functionalities like index, show, new, edit.
3. Code reusing is an important feature of MVC architecture.
4. It helps us to implement the complex relationship between the tables.
5. Complex view relationship can be easily monitored as it gives us nested view.

VI. DESIGN

As the basic idea of the application was that patientshould be able to create aapointment of any doctor irrespective of the geographical location.

Hence we have considered Doctor , Appointment/Booking & patient as main entity in our application.

In order to join all these three table together we have chosen has many to many mapping between Patient and Doctor and considered appointment as Join table.

There are many possible ways to design this application. As our application has many to many relationship between doctor and patient one way was to use has belong to.

But in this we have to create a new join table between the patient and the doctor and later add the primary keys of both the table.

Instead of this it would be easy to use has_many through

In order to implement this functionality in Ruby on Rails , we have chosen has-many-through association because This association indicates

that the declaring model can be matched with zero or more instances of another model by proceeding through a third model.

VII.GEMS USED

A. Gem used by Apurv::

1. Devise: Devise gem was used to add authenticated users to register for the application. It is used to register the users email and password and save them in a database. It is also used to create profile for the users. Devise gem gives us authentication and authorization.
2. Bill: This gem is used to create bill for the treatment. It is used to generate bill for the common applications.

B. Gem used by Trupti::

1. Devise : Devise is the gem that has added following functionality to the Global Health Application ::
   a. authentication functionality to the application.
   b. Devise's controllers has been customized in order to save the custom role of "patient" and "doctor".
   c. Also Devise's views has been customized in order to integrate email engine to send welcome mail to the patient .
   d. Integrating a Devise extension to estimate password strength
   e. Customizing Home Page of the Devise
2. generate_password

generate_password is the custom gem created with the purpose to create auto generated password with the help of provided name and contact number.

Basic Code :

@num1=dname[0..3]

@num2=contact % 4

@result = @num1 + @num2.to_str

C. Gems used by both (Apurv & Trupti):

1. Brakeman::

Brakeman is the gem that checks the ruby application and scans for the common security weaknesses and vulnerabilities that your application is holding.

Brakeman is used as a static code analyzer in this application. Prior to every deployment iteration of code to heroku , brakeman is used to test for the vulnerabilities present in the code .

Test Results ::

Scanned/Reported	**Total**
Controllers	11
Models	5
Templates	41
Errors	0
Security Warnings	8 (5)

Warning Type	**Total**
Attribute Restriction	5
Mass Assignment	3

+SECURITY WARNINGS+

Confidence	Class	Method	Warning Type	Message
Weak	AppointmentsController	Create	Mass Assignment	Unprotected mass assignment near line 46: Appointment.new(appointment_params)
Weak	DoctorsController	Create	Mass Assignment	Unprotected mass assignment near line 33: Doctor.new(doctor_params)
Weak	PatientsController	Create	Mass Assignment	Unprotected mass assignment near line 28: Patient.new(patient_params)

As we can see the confidence on the create() of each of the controller is in the Weak part due to Mass Assignment as indicated by the brakeman.

After removing the Mass assignment , from the controllers and re-running the test following is the summary ::

Scanned/Reported	**Total**
Controllers	11
Models	5
Templates	41
Errors	0
Security Warnings	0

VIII.IMPLEMENTATION

Database Schema:

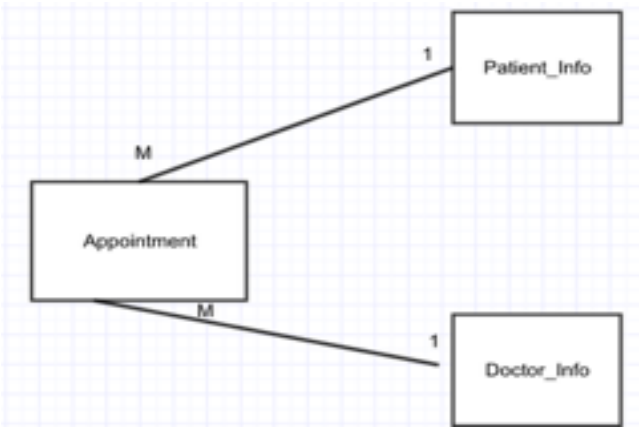
Total no of models :: 5

List of Models ::

- User
- Appointments
- Doctors
- Patients
- Admin

Relationships embedded in the models :

Appointments , patients and doctors are associated with each other using has many through relationships.



ER Diagram of Patient , Doctor and Appointment

Implementation::

The whole application is divided in 3 parts :

- Authentication and Registration with email Integration.
- Dynamic Appointment System
- Appointment Cost Calculator

I. Authentication and Registration with email Integration.

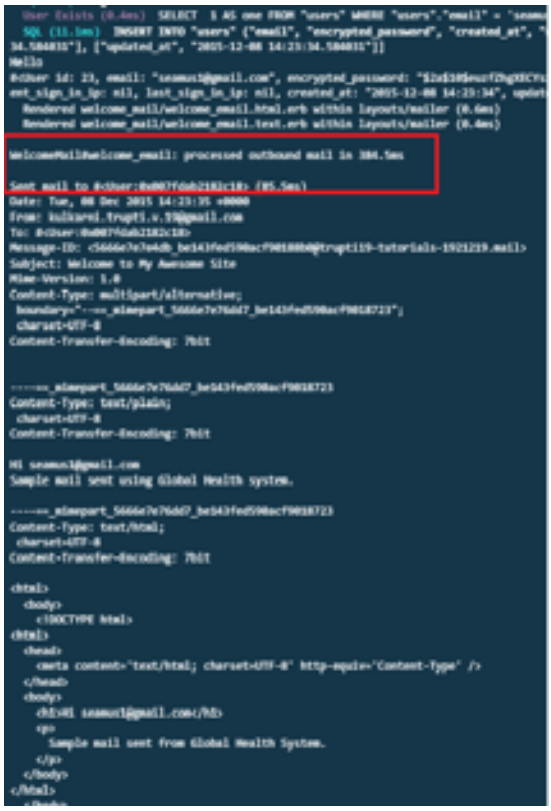
For authentication and registration of user , devise gem has been used .

If the patient is not registered to the system, he can register himself through the sign up link by providing email and password.

Trupti::

On registration to the system user will get the Welcome mail from the application.

Please find the below screenshot of the log file depicting that the mail is sent to the patient on registration ::



II. Dynamic Appointment System ::

Apurv & Trupti::

After logging to the system as patient user can see the list of doctors. For whom he can book the appointment.

III. Appointment Cost Calculator

Apurv & Trupti::

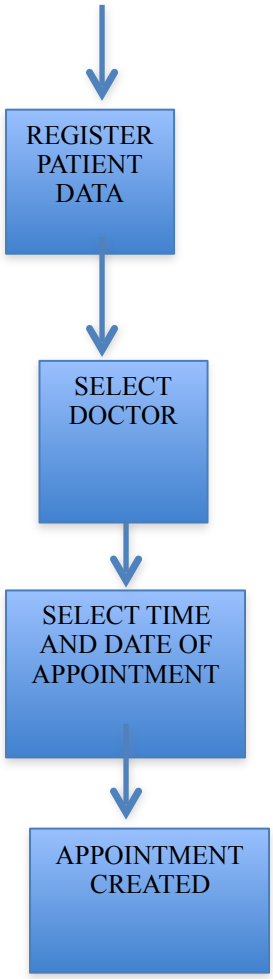
While booking the appointment patient needs to select the type of the appointment (Follow-up, First Sitting, RoutineCheck Up) . By selecting which , patient can calculate the estimate cost for the particular appointment.

Eg :: First –Sitting :: 100

Follow-up:: 150

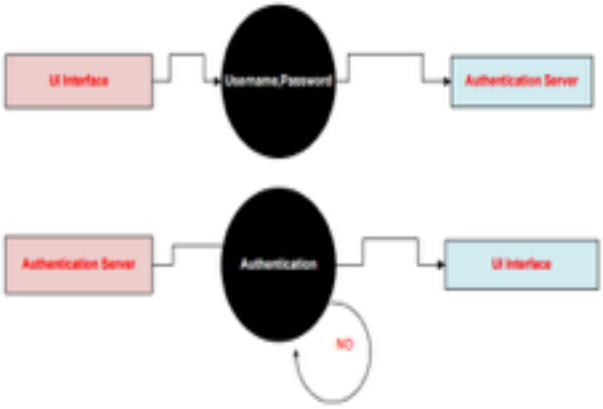
Routine Check Up ::100

DFD of Apurv's Implementation::



DFD of Trupti's Implementation ::

AUTHENTICATION DFD:



IX. Design Patterns

Design pattern is a new revolution in the coding world. Design pattern proposes solution to the repetitive problem in OOP. Design pattern imprives code reusability. Also helps in developer communication and it capture expert knowledge and design trade-offs & make
expertise widely available.

1. Decorator Pattern:

   Decorator Pattern is used to enclose one object within the other without changing the structure of the base object

   The enclosing object is called Decorator. We used the decorator pattern to decide what type of booking/appointment patient needs with the doctor. The three types we used are first sitting or follow up or the routine checkup. This will give doctor the general idea about patient. Depending on which patient can get the estimate about the cost of the appointment.

X. Testing Approach

1. Unit Testing: Test the truth:

   This test was used on all the models patients, doctor, bookings, profile and users.

2. Unique Booking Test: This testing was done so that no two patients share the same appointment time. If it arises then this test gives the error. The test was done on bookings controller and databases. If same booking by two patients on the same time with the doctor this test would raise an error.

3. Unit Testing :: To check whether email and name for the doctor are provided

   Following code snippet is included to test whether email and name for the doctor are provided::

   test "must not save a doctor when doctor name and email are not provided" do

   doctor = Doctor.new(dname: "John",email: "john.davidson@email.ie")

   assert_equal(false, doctor.save, "saved the doctor even though at least one of the name, and/or email were not provided!")

   assert

   end

4. Testing in Mailer Module ::

   Before sending mail to any patient we can preview the content of the mail in test -> mailers -> previews ->welcome_mail_preview.rb

XI. Conclusion:

The application has been successfully implemented and deployed on the cloud. The application is user friendly. It is developed in ruby on rails and deployed on heroku.

The achievements from this application are:

- Full utilization of resources that are available.
- All the records are managed and deployed on the cloud platform.
- As it is object oriented programming so further enhancements can be easily be done.

XII. Findings:

It is not possible to develop an application that has all the functionalities. The changes from the user keep on changing as they use this application. An admin module can be utilized more appropriately to achieve the functionality.