

National College of Ireland  
BSc in Computing  
2015/2016

VO THI THUY LINH  
x11113065  
thuylinhrndm@yahoo.com

**ANONYMOUS AUTOMATED RESPONSE SYSTEM**  
Technical Report



National  
College *of*  
Ireland

## Table of Contents

Executive Summary .....	3
1 Introduction.....	4
1.1 Background.....	4
1.2 Research .....	4
1.3 Aims .....	6
1.4 Technologies .....	7
1.5 Structure .....	12
2 System .....	14
2.1 Requirements .....	14
2.1.1 Functional requirements.....	15
2.1.2 Data requirements.....	216
2.1.3 User requirements.....	26
2.1.4 Environmental requirements .....	27
2.1.5 Usability requirements.....	28
2.2 Design and Architecture.....	29
2.3 Implementation .....	42
2.4 Testing.....	429
2.5 Graphical User Interface (GUI) Layout.....	61
2.6 Users testing.....	614
2.7 Evaluation .....	66
3 Conclusions .....	67
4 Further development or research.....	68
5 References .....	69
6 Appendix.....	701
6.1 Project Proposal .....	71
6.2 Project Plan .....	77
6.3 Requirement Specification .....	80
6.4 Monthly Journal .....	107
6.5 Survey .....	120

## **Executive Summary**

The main objective of “Anonymous Automated Response System” is for Educational purposes in that it provides a service which automatically suggests possible answers to queries immediately after users (e.g. students, employees...) post their problems or their questions. It also allows other users (e.g. counselors or consultancy or teachers...) to interact with them by posting answer(s) or giving advice. The posts and the answers are saved and displayed to all users without any personal information. The users can see their own post with their information and the other users can only see the problem and the answer without any personal information.

AARS leverages the speed and processing power of information technology to help users to locate useful relevant resources and personalized expert advice, conveniently, immediately and anonymously.

Immediately after a problem is posted, AARS makes calls to the Google Custom Search API and renders a list of relevant resources. Result received are sorted and ordered and only the most relevant and useful hyperlinks are retained. By using sort algorithm, the system will automatically provide suggestions with highest rating.

An appointment can be arranged if the user wishes to chat personally with the counsellor or consultant. Moreover, the application will economize on the use of time, in that it is available at anytime and anywhere.

The application can be found at: <https://afternoon-waters-88881.herokuapp.com>

# **1 Introduction**

## **1.1 Background**

The reasons I have chosen to do this project are the following: First of all I want to use my knowledge, skills and new technologies which I have learnt to create something useful that can be of benefit to the users. The reason I chose the topic because I found that many people especially young people have problems communicating with friends, parents, families, colleagues and school. They have no one with whom to share their concerns or no one they can trust.

I was encouraged by a group of consultants who were longing to help people to discover more meaningful ways of communicating. So I decided to build the application that would allow the users freedom to share their own problem and discuss with consultants

## **1.2 Research**

I found in my research that there are some organizations that have a webpage which helps people who have problems with friends, families, school or work. Their services allow the users to make a call from a mobile or from a landline. Ideally there is someone to receive the call, listen and give advice. However problems may arise in that a caller may have to wait a considerable time for someone to answer the call or the organization may have set specific times when the calls will be answered. This arrangement may not suit the immediate need of the caller.

The following gives details of a few such organizations dealing with a cross section of society who is facing different types of problems, for example,

- "Samaritans" who provide a network of people you could 'ask' about anything;
- "ReachOut.com" helps young people get through tough times by providing quality mental health information and covering issues that can impact on mental health, "ReachOut.com" takes the mystery out of mental health;

- The website, “SpunOut.ie”, carries a range of health information for young people, including mental health, sexual health, exam stress and general lifestyle information. “SpunOut” also has an extensive online directory allowing site visitors to search for supports and services in their area;
- Childline.ie provides a free and confidential listening service to children and young people up to the age of 18. The Childline helpline is open every day, 24 hours a day and Childline Online Chat is open every day from 10am to 10pm.

In studying the above examples it seems to me that an improvement is needed, to be an improvement to facilitate the users’ access to an immediate response to their queries.

In the light of this information I decided to develop a response system which automatically suggests possible answers to queries immediately after users (e.g. students, employees...) post their problems or their questions. It also allows other users (e.g. counselors or consultancy or teachers...) to interact with them by posting answer(s) or giving advice. The posts and the answers are saved and displayed to all users without any personal information. The users can see their own post with their information and the other users only see the problem and the answer without any personal information.

An appointment can be arranged if the user wishes to speak or chat personally with the counsellor or consultant. The application will have a booking appointment system. In addition, this application will use Gmail API for chatting and speaking functionalities. Moreover, the application will economize on the use of time, in that it is available at anytime and anywhere.

I have spoken with people from different backgrounds about my idea and have asked them what they would want, if they were to use the application. Teachers in secondary school, for example agree with me that the students have problem with

studies, friends and families. Students feel they have nobody to talk to or trust, they are afraid to share their problems and as the result they keep them to themselves until perhaps it is too late to solve the problems. Teachers think the application will give students opportunities to share their problems without identifying themselves. This refers only to a group of teachers who will answer the questions or give an advice. This will also help teachers themselves understand students and be able to help them find solutions before it is too late.

Employees working in companies also have to deal with personal problems, and those evolved in relation to their colleagues or their employers find it difficult to discuss these with the people concerned and they maintain that if they could access this application it would be a great help to improving personal job satisfaction. As a result they would be able to focus clearly on the overall objectives of the organization and thus improve their productivity.

### **1.3 Aims**

Anonymous Automated Response System provides a free service to help people, especially young people to get through tough times. It also gives opportunities to share their problems without identifying themselves. It also automatically suggests possible answers to queries immediately after users post their problems or their questions. The system will economize on the use of time, in that it will be available at anytime and anywhere.

The system aims to give users the benefit of expert advice from consultants, counsellors or teachers, through direct advice, or answers. No one else will have access to their personal information, but other users of the system will be able to view the general problems and answers, and thus benefit by someone else's questions.

Another aims is to facilitate any person who wishes to speak personally with a consultant or counsellor by arranging a booking appointment.

A final aim is to modify, create and display calendar events, as well as working many other calendar related objects.

## **1.4 Technologies**

The application is built in Ruby on Rails which is an open source web application framework written in the Ruby programming language; it is a full-stack framework. Ruby on Rails uses well-known software engineering patterns and principles such as active record pattern, convention over configuration (COC), don't repeat yourself (DRY) and model-view-controller (MVC).

### **Ruby**

Ruby is a language of careful balance. Yukishiro Matsumoto designed this language in 1995 influence by Perl, Eiffel, Python, Smalltalk and others. It's a dynamically typed, fully object-oriented, general-purpose scripting language.

In Ruby, everything is an object. Ruby is used in typical scripting language applications such as text processing and "glue" or middleware programs. It's suitable for small, ad-hoc scripting tasks that, in the past, may have been solved with Perl. Writing small programs with Ruby is as easy as importing the modules you need and writing an almost BASIC-like "sequence of events" type of program.

### **Ruby on Rails**

Ruby on Rails (RoR) is open source web framework written in the Ruby programming language and all the applications in Rails are written in Ruby. Ruby on Rails is focused on productivity and enforces agile web development.

The Ruby on Rails framework was designed for database-backed web applications according to the Model-View-Controller (MVC) pattern. It was created as a response to heavy web frameworks such as J2EE and the .NET framework. Many of the common tasks for web development are built-in in the framework to work out-of-the-box. This includes email management, object-database mappers, file

structures, code generation, how the elements are named and organized and so on. All of these conventions allow developers to write less code and develop agile applications.

Ruby on Rails architecture has the following features:

- Model-View-Controller architecture.
- Representational State Transfer (REST) for web services.
- Supports the major databases (MySQL, Oracle, MS SQL Server, PostgreSQL, IBM DB2, and more).
- Open-source server side scripting language.
- Convention over configuration
- Scripts generators to automate tasks.
- Use of YAML machine, which is a human-readable data serialization format.
- The above-described features are distributed in the following Rails' components

Rails is made up of several components

- Action mailer is responsible for providing e-mail services
- Active record provides object-relational mapping to classes.
- Action pack provides the controller and view layers of the MVC patterns. These modules capture the user requests made by the browser and map these requests to actions.
- Action web services

## **Git**

For a version control system it has used Git which allows you to track the history of a collection of files and includes the functionality to revert the collection of files to another version. GitHub is a Web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

## **First-time Repository Setup**

Navigate to the root directory of the app and initialize a new repository:

```
$ git init
```

Add files in the Anonymous Automated Response application to git, then commit the results.

```
$ git add.  
$ git commit -m "Initial commit"
```

## GitHub

Push the code up to GitHub, which is a social code site, optimized for hosting and sharing Git repositories. I pushed up the application as follows:

```
$ git remote add origin  
https://github.com/thuylinhrndm/AnonymousAutomatedResponse.git  
$ git push -u origin master
```

## Cloud platforms:

### Heroku:

The application is deployed on cloud platform using Heroku which is a platform as a service (PaaS). Heroku allows deploying directly from popular tools like Git, GitHub or Continuous Integration (CI) systems. First install the Heroku Toolbelt. This provides you access to the Heroku Command Line Interface (CLI), which can be used for managing and scaling the applications and add-ons. A key part of the toolbelt is the heroku local command, which can help in running applications locally. Once installed, use the Heroku command from command shell.

### Heroku Setup

After creating a Heroku account, I will have to install Heroku gem:

```
$ gem install heroku
```

Once installed, you can use the heroku command from your command shell.

Log in using the email address and password you used when creating your Heroku account:

```
$ heroku login
```

Authenticating is required to allow both the heroku and git commands to operate. Execute the following commands to clone the application:

```
git clone https://github.com/heroku/thuylinhrndm/ AnonymousAutomatedResponse.git
```

In this step, deploy the app to Heroku. Create an app on Heroku, which prepares Heroku to receive the source code.

```
$ heroku create
```

When the application is created, a git remote (called heroku) is also created and associated with the local git repository.

Heroku generates a random name (in this case afternoon-waters-88881) for the application, or you can pass a parameter to specify your own name.

Now deploy the code:

```
git push heroku master
```

The application is now deployed.

Now visit the app at the URL generated by its application name. As a handy shortcut, can use command line open the website as follows:

```
$ heroku open
```

The link of the application run in Heroku: <https://afternoon-waters-88881.herokuapp.com>

## **Cloud 9:**

Cloud9 IDE is an open source, online integrated development environment. It supports hundreds of programming languages such as PHP, Ruby, Perl, Python, JavaScript with Node.js, and Go. It enables developers to get started with coding immediately with pre-setup workspaces, collaborate with their peers with collaborative coding features, and web development features like live preview and browser compatibility testing.

It is written almost entirely in JavaScript, and uses Node.js on the back-end. The editor component uses Ace. As of July 2014, it uses Docker containers for its workspaces, and is hosted on Google Compute Engine

## **SignIn with OmniAuth**

The Rails community provides a wealth of plugins as Ruby Gems that simply add to your project Gemfile and install. For example, the application used OmniAuth for the sign in with Google. It simply added gem "*omniauth-google-oauth2*" in the Gemfile, then bundle install.

The application "sign in" will use simple OmniAuth. It is a library that standardizes multi-provider authentication for web applications. It was created to be powerful, secure, and flexible. Any developer can create strategies for OmniAuth that can authenticate users via disparate systems. For example: Google, Facebook, Twitter,... The reason for using OmniAuth is because most users don't like to sign up for websites. They've already signed up for so many, using different usernames and password and trying to remember them is sometimes impossible.

## **Database**

The application will connect to database using SQLite3 and PostgreSQL. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. PostgreSQL is an object-

relational database management system (ORDBMS) with an emphasis on extensibility and on standards-compliance.

## **Font end**

Foundation is used for the font end design. Foundation is a framework for building the front-end, or client-facing part, of a website or web application. It lets me quickly prototype and create sites and applications that work on any device. HTML front end will be responsible for displaying the information on multiple device types. This service will be written in JavaScript, JQuery, SCSS3 and HTML.

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites. Bootstrap makes front-end web development faster and easier. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes.

## **APIs**

Google Custom search API for automated response and suggestion. When users click on the post button, the post will save in the database, display to user and send to Google Custom search API. Google search takes information and searches for the web sites which related to that information, then automated display possible suggestion with title and links.

Simple Calendar API is used to display booking appointment's time of the days, week and months which have been made. The appointment automatically removes from the booking list when the time has passed.

### **1.5 Structure**

- Introduction gives a summary of the project
- System describes the project requirements, Design, the engineering of the software involved, testing plans, GUI layout, customer testing and evaluation.

- Conclusion describes the outcome of the project and a summary of what I learned during the progress of the project and the current status of the system.
- Further Development or research describes what I feel the future direction and development of the system will be, such as developing a mobile and/or tablet friendly version of the application in the future to adapt to the currently expanding mobile device market. It can be run with android and ios.
- References are a section where I list all my resources of learning, particularly outside of the college course that I needed in order to develop the application.
- Appendix is used for all other information such as market research.

## 2 System

### 2.1 Requirements

#### Use Case Diagram



### 2.1.1 Functional requirements

- **Web Based Interface:** This system will have a web based interface so that it can be viewable on multiple devices: PC and tablets or phones
- **User Login:** This system will have a user login to ensure only the right users get to see the content. The user should be able to use an OAuth login to then login to all the available social media feeds. This system will have users' role that allow the collection of users into a single unit against which they can apply permissions in a database.
- **Posting system:** This system will allow users post problems or questions.
- **Automated Response:** This system will display automatically the result from outside resources which relate to users problem and give user some suggestions. It also allows user to rate for the links and display the suggestion according to the highest average rate, highest number rate and most number of time users' click on the links.
- **Responding system:** This system will allow consultants or counsellors to answer the problem which users have posted.
- **Booking system:** This system will have a booking system so that user can make an appointment with consultant and a calendar which displays users' appointment. It will automatically remove the appointment when it is over
- **Chat System:** This system will have a chatting system that allow user to chat with consultants
- **Reminder system:** This system will send email to remind the users about their appointments.

#### Requirement 1 <Sign In>

##### ***Description & Priority***

This function is to allow a user to securely sign in to Anonymous Automated Response System.

## Use Case

### Scope

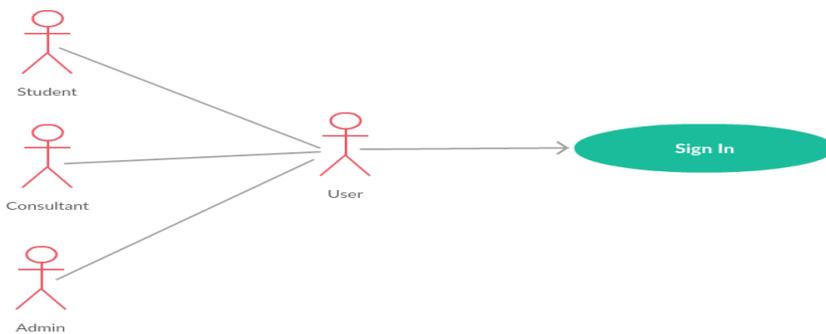
The scope of this use case is a sign in system for user

### Description

This use case describes the user secure when they sign in to Anonymous Automated Response System

### Use Case Diagram

Diagram should highlight actors and uses cases



### Flow Description

#### Precondition

The system is in initialisation mode. User must have Google account.

#### Activation

This use case starts when a user wishes to sign in to Anonymous Automated Response system.

#### Main flow

1. The user enters email and password
2. The system connects to Google account and check if the entered email and password invalid (See A1)

#### Alternate flow

A1 : <Invalid email or password>

1. The system displays error message

2. The user enters email and password again
3. The use case continues at position 3 of the main flow

### **Termination**

The system stores all the login information

### **Post condition**

The system goes into a wait state. If the use case was successful, the actor is now logged into the system. If not, the system state is unchanged

## **Requirement 2 < Posting Problem >**

### ***Description & Priority***

This function is to post the problem which the user wants to ask about or needs an advice on. It will be stored in the database.

### ***Use Case***

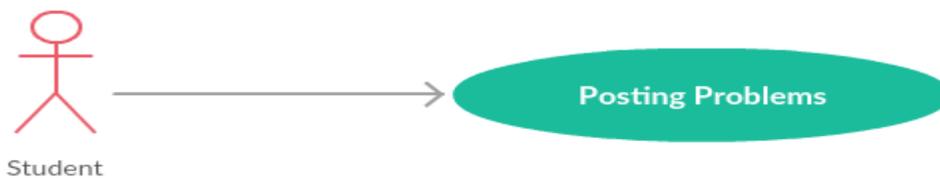
### **Scope**

The scope of this use case is to post a problem and store it in the database

### **Description**

This use case describes what users do when they have a problem and need help.

### **Use Case Diagram**



### **Flow Description**

### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system.

### **Activation**

This use case starts, when a user presses the button to post his/her own problem

## Main flow

1. The system identifies the user role and displays the post page
2. The user clicks on the button “post your own problem”
3. The system reloads the page and opens the new post page
4. The user enters problem and clicks the button “post your problem”
5. The system displays the message and says that the post is created successfully

## Termination

The system stores all the post into database. The use case terminates when the user exits

## Post condition

The system goes into a wait state

## Requirement 3 < Automated Response >

### *Description & Priority*

This function is automated it displays to user some suggestions which are already given from the other web site, while user is waiting for the answer from a consultant.

### *Use Case*

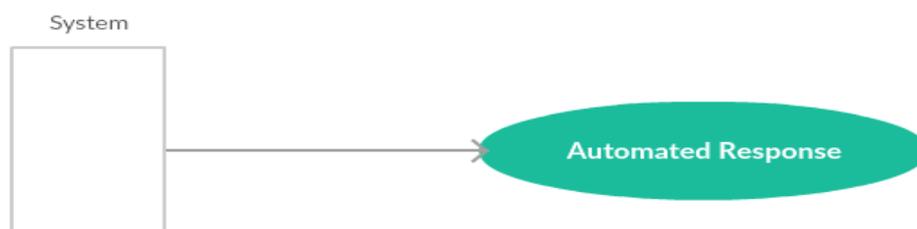
#### Scope

The scope of this use case is to automatically display some recommendations which are related to user’s problem.

#### Description

This use case describes what the system does when users are waiting for the answer to the problem.

#### Use Case Diagram



## **Flow Description**

### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system.

### **Activation**

This use case starts when a user presses the “created post” button.

### **Main flow**

- 1- The system makes calls to the Google Custom Search API and renders a list of relevant resources which are given from other website.
- 2- The system automated displays the web pages which are related to the post.
- 3- The users can rate for the links
- 4- The system save the rating and that links
- 5- The system views the results which are sorted and ordered and only the most relevant and useful hyperlinks are retained sort
- 6- The system gives the suggestion according to the highest average rate, highest number rate and most number of time users' click on the links.

### **Termination**

The system stores the rating and links into database The use case terminates when the user exits

### **Post condition**

The system goes into a wait state

## **Requirement 4 < Response Problem >**

### ***Description & Priority***

This function is to let the consultant give an advice or an answer to the problem which the user has posted.

### ***Use Case***

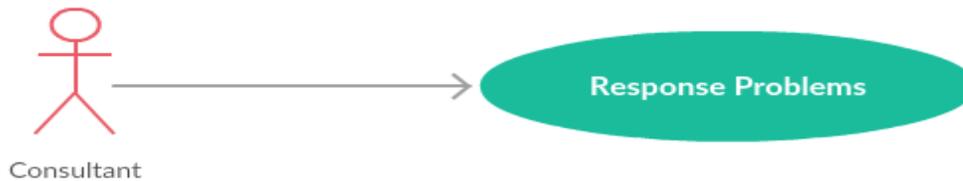
### **Scope**

The scope of this use case is to give an answer to the problem

## Description

This use case describes what the consultant does when he/she gives the answer for the problem.

## Use Case Diagram



## Precondition

The system is in initialisation mode. User must login to Anonymous Automated Response system.

## Activation

This use case starts when a user presses the answer button

## Main flow

1. The system identifies the user role and displays the post page.
2. The consultant clicks on the answer button.
3. The system reloads the page and displays the answer page.
4. The consultant enters the answer and clicks to post the answer button.
5. The system displays the message saying that the post has been created successfully
6. The system sends the alert to the user, saying that user's question has been answered.

## Termination

The system stores all the post into a database. The use case terminates when the user exits

## Post condition

The system goes into a wait state

## Requirement 5 < Booking Appointment >

### **Description & Priority**

This function allows user to book an appointment with consultant.

### **Use Case**

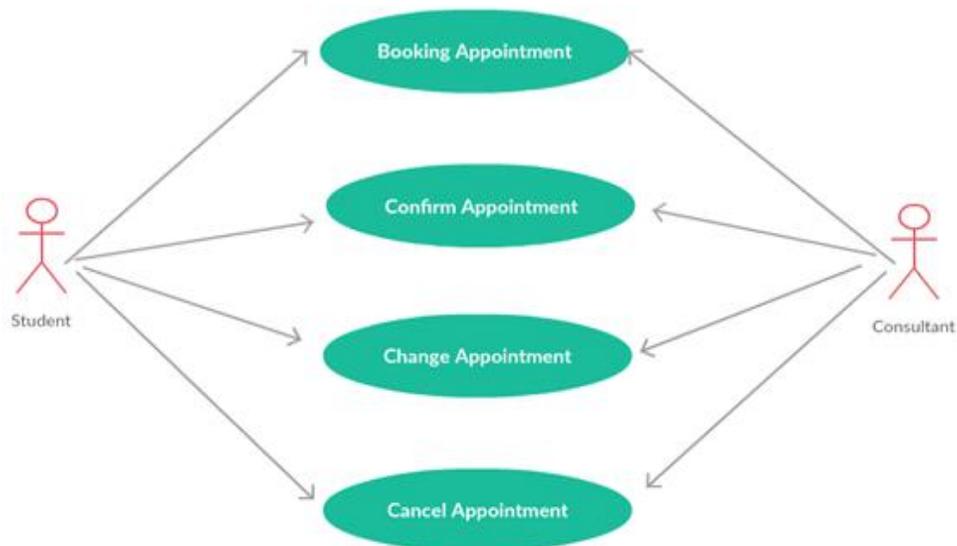
#### **Scope**

The scope of this use case is to book an appointment with consultant.

### **Description**

This use case describes how to book an appointment with consultant for more help.

### **Use Case Diagram**



### **Flow Description**

#### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system.

## **Activation**

This use case starts, when a user presses the booking appointment button

## **Main flow**

1. The system displays the booking window.
2. The user wants to make a new appointment (See A1)
3. The system displays new booking appointment window
4. The user wants to cancel an existing appointment (See A2)
5. The system displays cancel booking appointment window
6. The user wants to change an existing appointment (See A3)
7. The system displays result window

## **Alternate flow**

1. A1 Make a new appointment:
  - The system displays possible appointment times, dates and email of consultants
  - The user chooses date and time available
  - The use case continues until the user submits booking button or want to release it.
2. A2 Cancel an appointment:
  - The system displays the old booking appointment with time and date.
  - The use case continues until the user click cancel button or the user wants to release it.
3. A3 Change an appointment
  - The System shows the change appointment window where the users can change the date and time for theirs appointment.
  - The use case continues until the user click make change button or the user wants to release it.

## **Termination**

The system stores all the post into database. The use case terminates when the user exits

**Post condition**

The system goes into a wait state

**Requirement 6 < Chatting System >**

**Description & Priority**

This function allows user to chat or talk with consultant.

**Use Case**

**Scope**

The scope of this use case is to chat or talk with consultant for more help.

**Description**

This use case describes how user operates with the system.

**Use Case Diagram**



**Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system and book the appointment with consultant.

**Activation**

This use case starts when a user presses the chatting button

**Main flow**

- The system displays the chat room
- The users send messages
- The system update chat room
- The system reloads the chat room.

### **Termination**

The use case terminates when the user exits

### **Post condition**

The system goes into a wait state

## **Requirement 7 < Appointment Reminder System >**

### ***Description & Priority***

This function is to remind users about their appointments.

### ***Use Case***

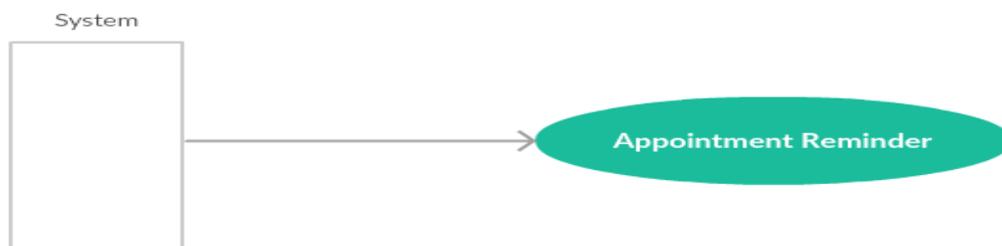
#### **Scope**

The scope of this use case is to remind users about their appointments. Appointment reminders allow system to automate the process of reaching out to users in advance of an upcoming appointment.

#### **Description**

This use case describes how the System Appointment Reminders work in the Anonymous Automated Response System.

### **Use Case Diagram**



**Flow Description****Precondition**

The system is in initialisation mode. User must book appointment in Anonymous Automated Response system

**Activation**

The use case starts when a user submits a booking appointment.

**Main flow**

1. The system stores booking information in the database.
2. The system checks the appointment list and configured time, in advance of the appointment.
3. The system sends out a reminder

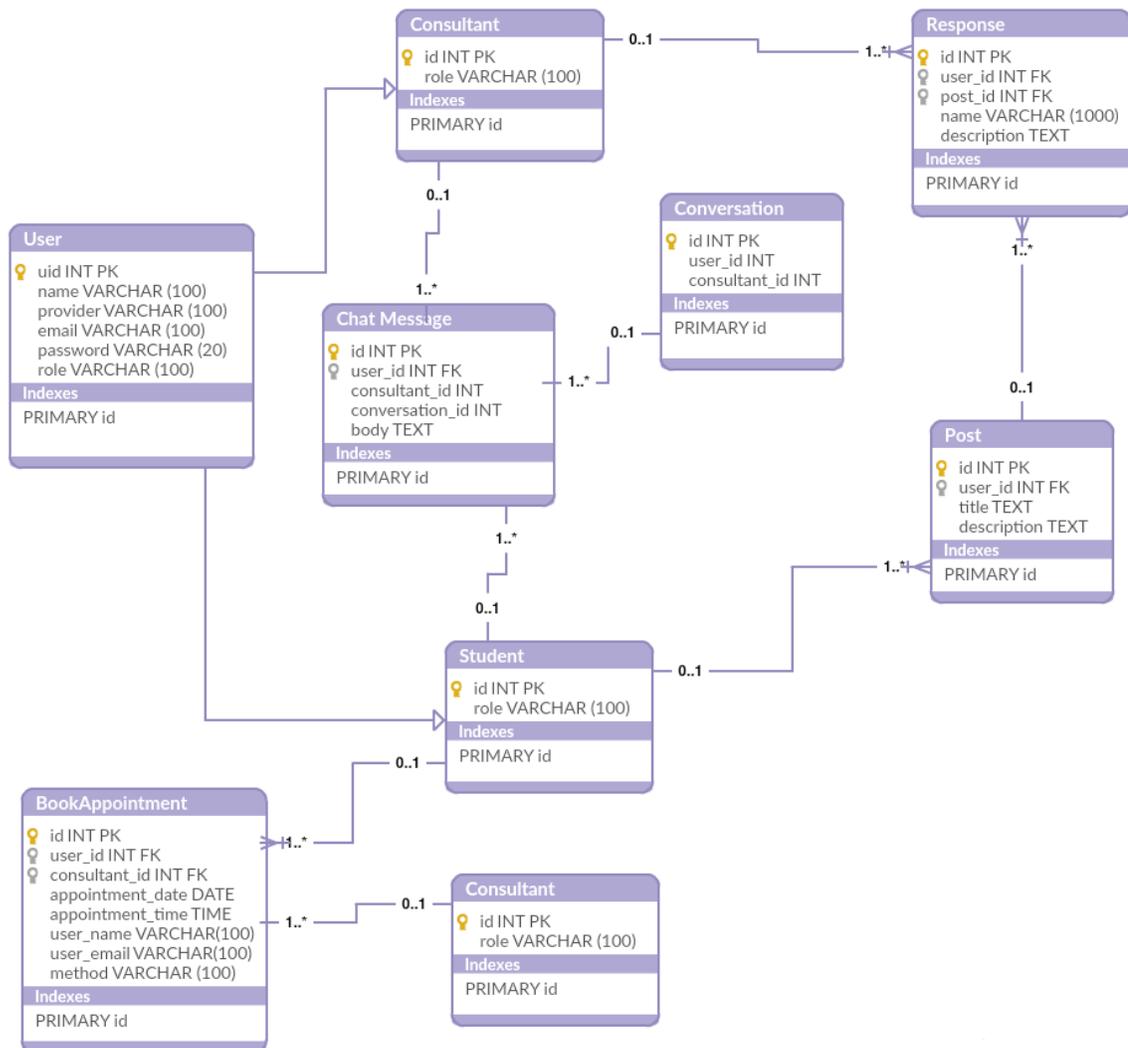
**Termination**

The use case terminates when the user exits

**Post condition**

The system goes into a wait state

## 2.1.2 Data requirements



## 2.1.3 User requirements

The application was requested by a group of consultants who are longing to help people to discover a more meaningful ways of communicating. This application allows users freedom to share their own problems and discuss with consultants.

The application will be free to consultants. Users and consultants will not know each other. There is no personal information shown. It is a secret room. Moreover, the application will economize on the use of time and in that it is available at anytime and anywhere.

This system will require the following:

- A system that will run on multiple devices computer, iPad and mobile.
- System that has a secure login system.
- A system that has a web interface.
- A system that allows user to input and display information.
- A system that can store information about user problem.
- A system that user login can operate with different roles: user, consultant and admin.
- A system that can use automated display suggestion after user has posted the problem
- A system that has a booking appointment.
- A system that has a chatting and talking system.

The application should be user friendly and simplistic with its user interface, so that the different sections are easy to use.

The system will allow for different users as follows:

**User 1:** Log in with Google account as user's role. Then users should be able to post their problems, view their own questions and list of all problems. The user should then be able to book an appointment and chat or talk with consultants or counselors as they wish.

**User 2:** Consultants log in with Google account. Consultants should be able to see all problems and manage to give the advice.

**User 3:** Admin log in with Google account. Admin should be able to see all the users, posts and have full permission on the post. This user will be a System administrator who will have user privileges to start and shut the system and monitor any issues with the system.

#### **2.1.4 Environmental requirements**

In order to develop the system the following environmental requirement needed to be adhered to and present during the development stage:

- Ruby on rails is required to develop the application.
- Git is used to track the history of a collection of files as they change over time and includes the functionality to revert the collection of files to another version
- GitHub is a Web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.
- The system will require a server on which to operate .The server will be hosted on a free cloud service called Heroku. Heroku is a platform as a service (PaaS) that enables developers to build and run applications entirely in the cloud.
- Cloud9 IDE is a freeware online integrated development environment. It combines a powerful online code editor with a full Ubuntu workspace in the cloud. It supports more than 40 languages, with class A support for PHP, Ruby, Python, JavaScript, Go, and more. It enables developers to get started with coding immediately with pre-setup workspaces, collaborate with their peers with collaborative coding features, and web development features like live preview and browser compatibility testing.
- A database is used both SQLite3 and PostgreSQL.

### 2.1.5 Usability requirements

**Simple Interface:** this solution has an easy to use interface that uses common elements found in websites and web-browsers.

**User Feedback:** this solution uses warning messages and popups to inform the user what is happening in the application.

**Good Looking Interface:** this solution has an appealing interface that uses web safe colours to ensure the user is not put off by the looks.

## 2.2 Design and Architecture

### Software Architecture

Here is a high-level overview of the Anonymous Automated Response application. The application follows the model-view-controller (MVC) architectural pattern, which enforces a separation between “domain logic” from the input and presentation logic associated with a GUI.

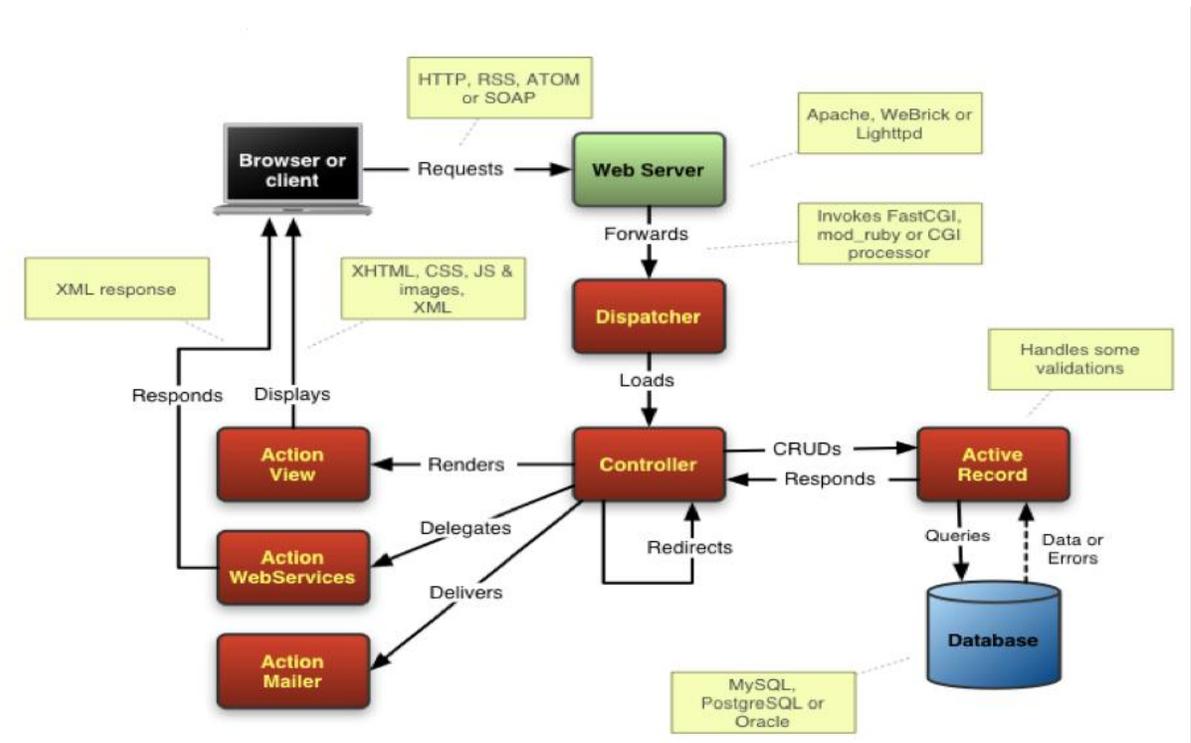


Figure: Ruby on Rails Web Application framework Architecture<sup>1</sup>

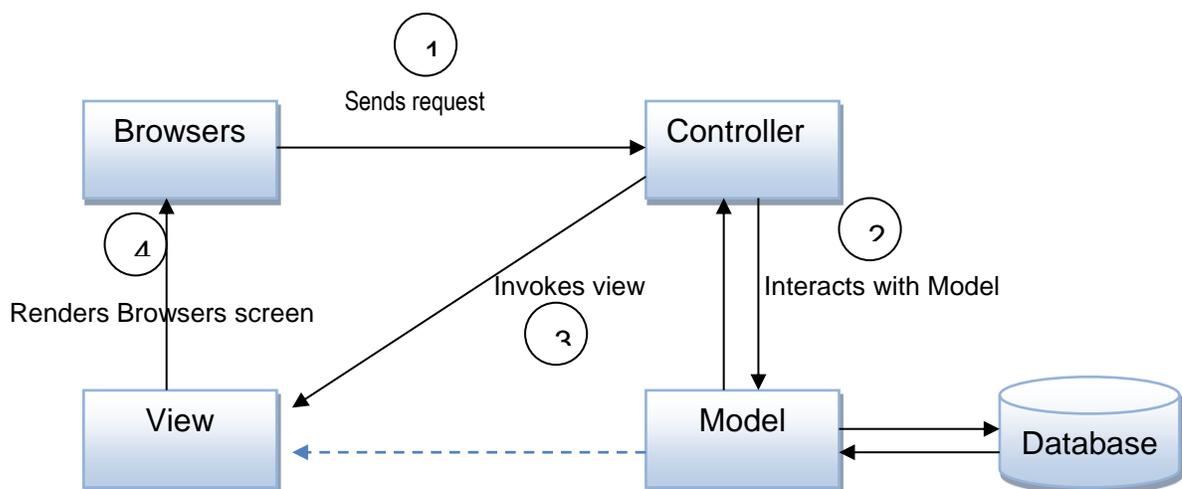
Ruby on Rails uses the Model-View-Controller (MVC) architectural pattern in order to improve the maintainability of the application. The Model centralizes the business logic; the View manages the display logic, while the Controller deals with the application flow. The MVC allows a clean separation of concerns, in the way that it keeps the business logic separated from HTML views. Additionally, it improves decoupling and testing.

<sup>1</sup> Mejia, A. (2011). *Ruby on Rails Architectural Design - Adrian Mejia's Blog*. [online] Adrianmejia.com. Available at: <http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/> [Accessed 4 Dec. 2015].

The Model layer carries the business logic of the application and the rules to manipulate the data. The Models represent the information in the database and do the appropriate validations. In Rails, database-backed model classes are derived from ActiveRecord::Base. Active Record allows presenting the data from database rows as objects and embellishing these data objects with business logic methods. Although most Rails models are backed by a database, models can also be ordinary Ruby classes, or Ruby classes that implement a set of interfaces as provided by the ActiveSupport module.

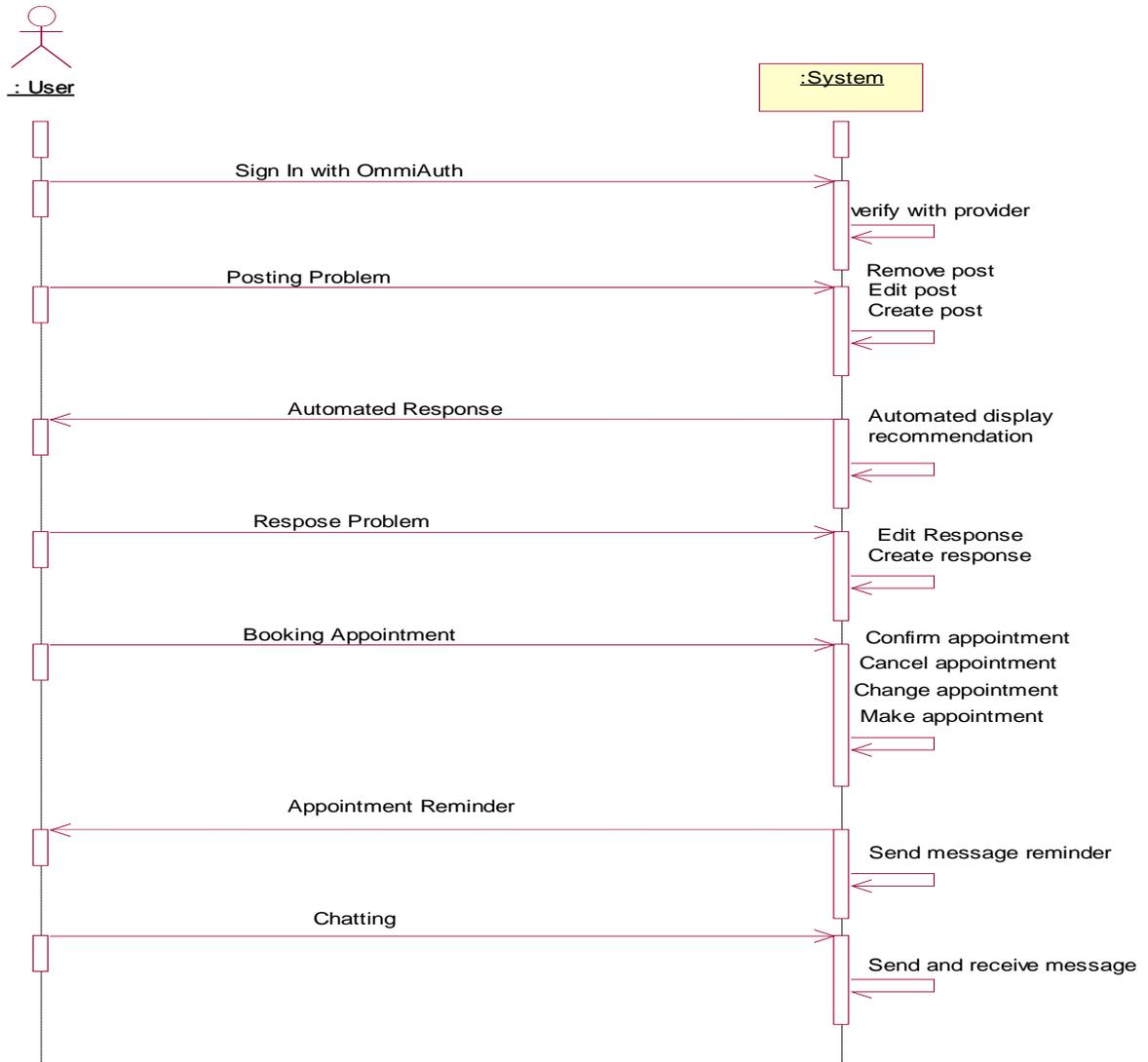
The view is the front-end of the application, representing the user interface. In Ruby on Rails, views are HTML files with embedded Ruby code. The embedded Ruby code in the HTMLs is fairly simple (loops and conditionals). It is only used to display data to the user in the form of views. Views are used to provide the data to the browsers that requested the web pages. Views can server content in several formats, such as HTML, JSON, XML, RSS and more.

Controllers interact with models and views. The incoming requests from the browsers are processed by the controllers, which process the data from the models and pass it to the views for presentation. Controllers manipulate models and render view templates in order to generate the appropriate HTTP response. In Rails, the Controller and View layers are handled together by Action Pack. These two layers are bundled in a single package due to their heavy interdependence. Each of these packages can be used independently outside of Rails.



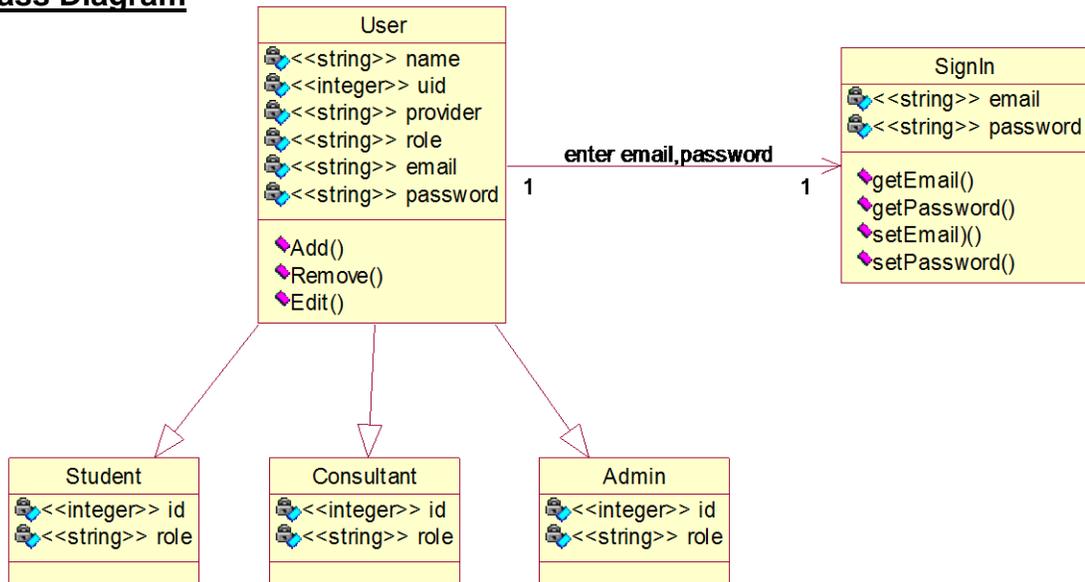


# Sequence System Diagram

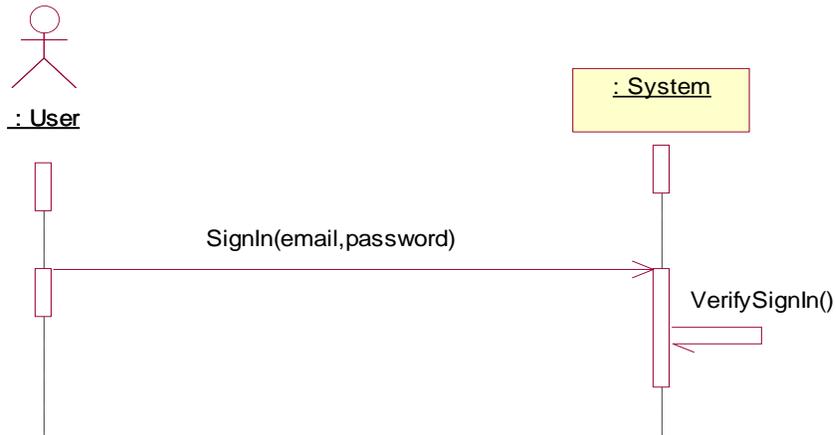


## Requirement 1: Sign In

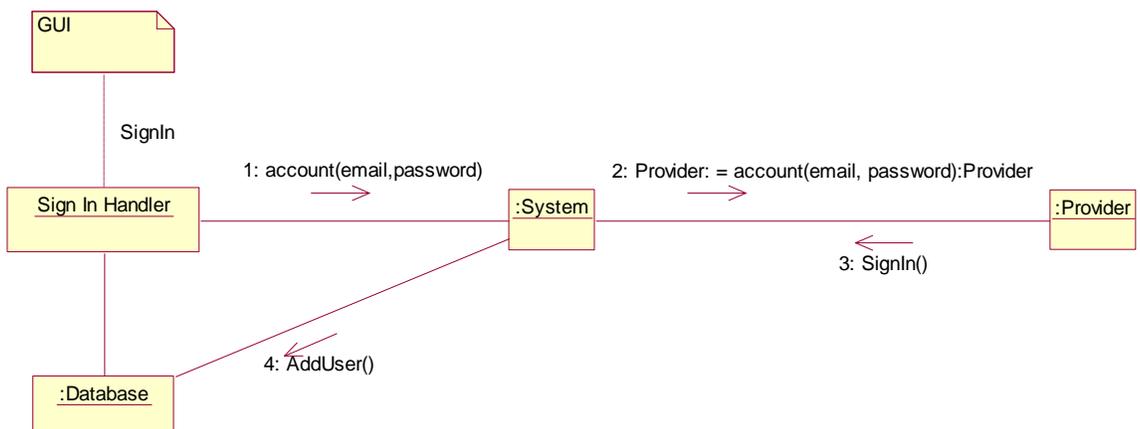
### Class Diagram



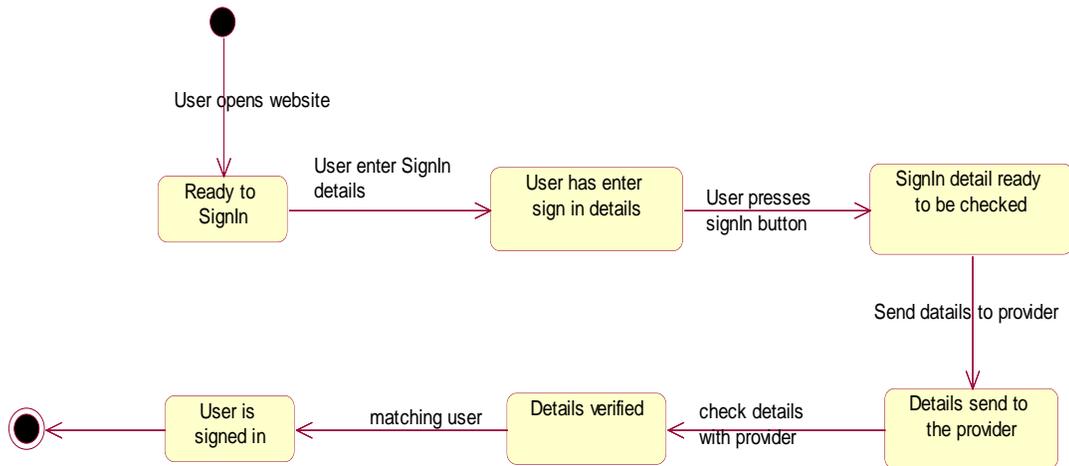
### Sequence System Diagram



### Collaboration and Patten Diagram

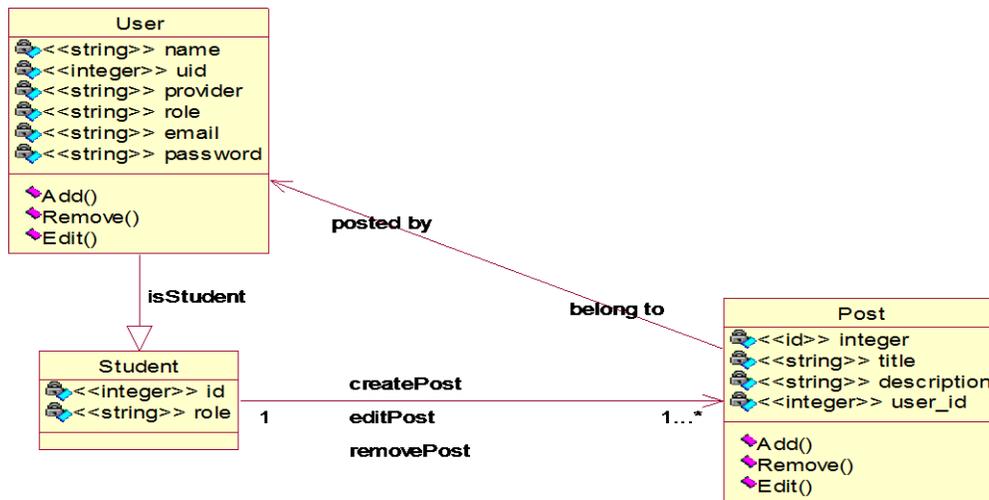


## State chart Diagram



## Requirement 2: Posting Problem

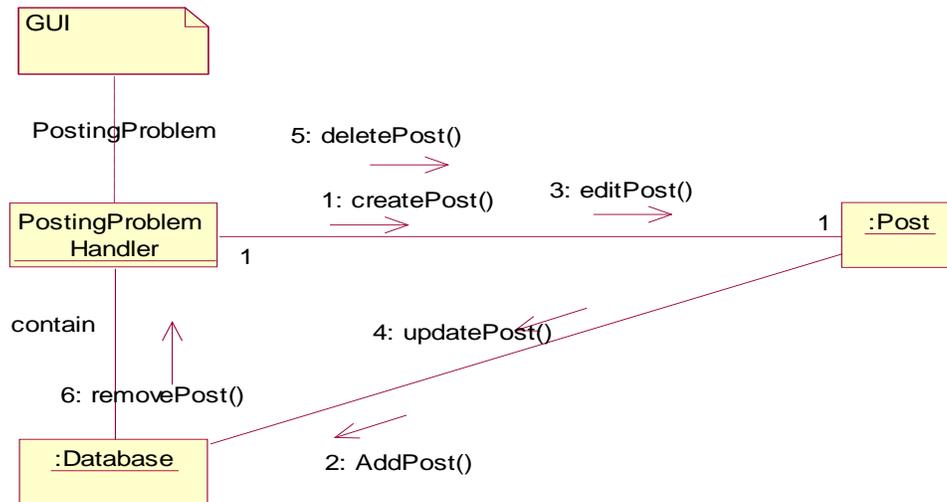
### Class Diagram



### Sequence System Diagram



## Collaboration and Patten Diagram

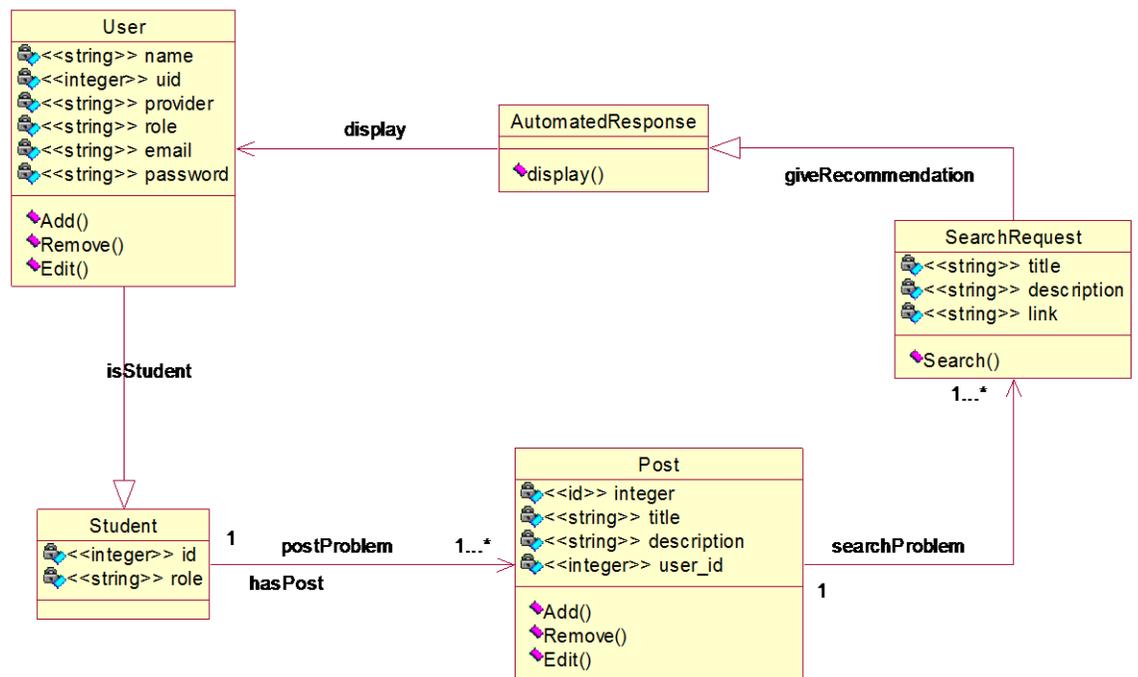


## State chart Diagram

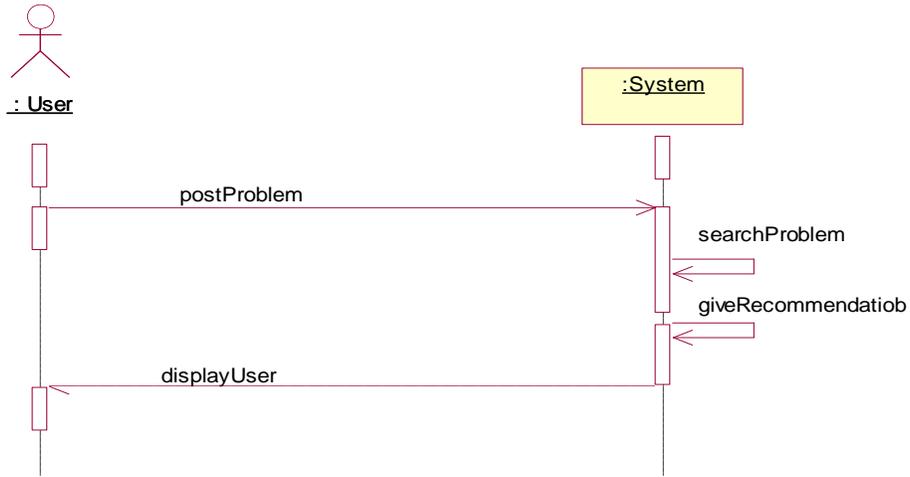


## Requirement 3: Automated Response

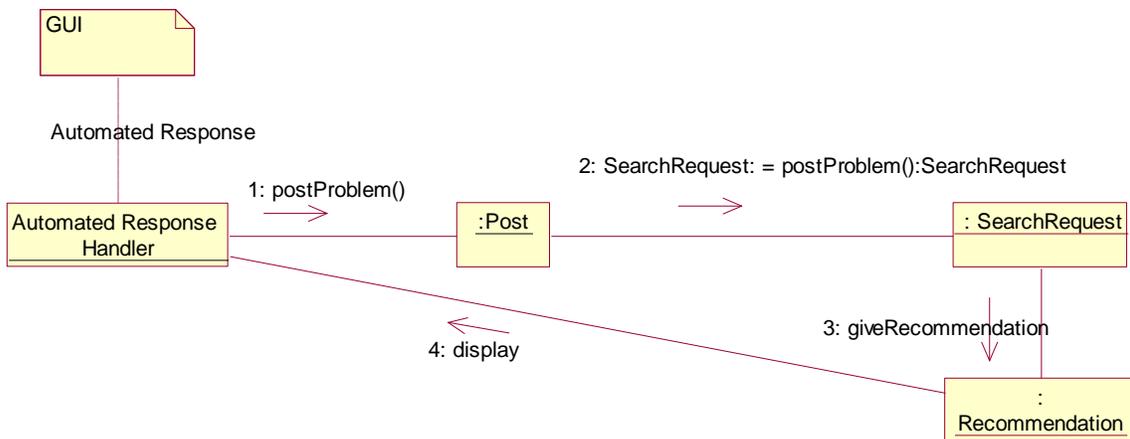
### Class Diagram



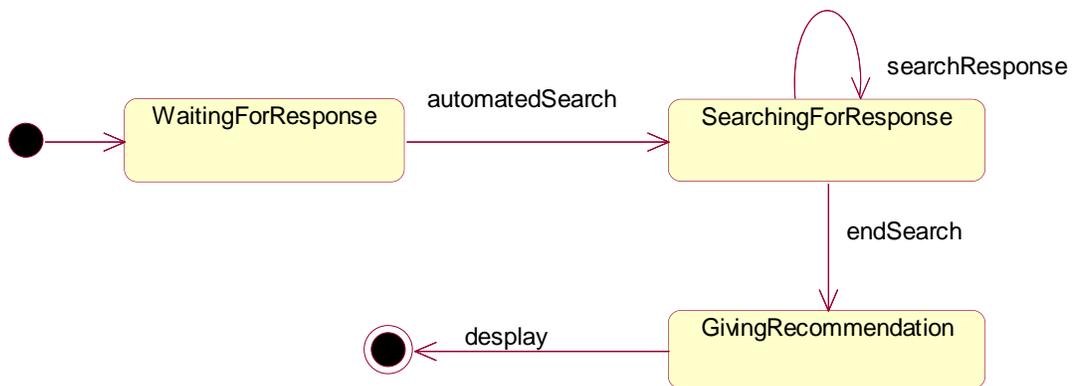
## Sequence System Diagram



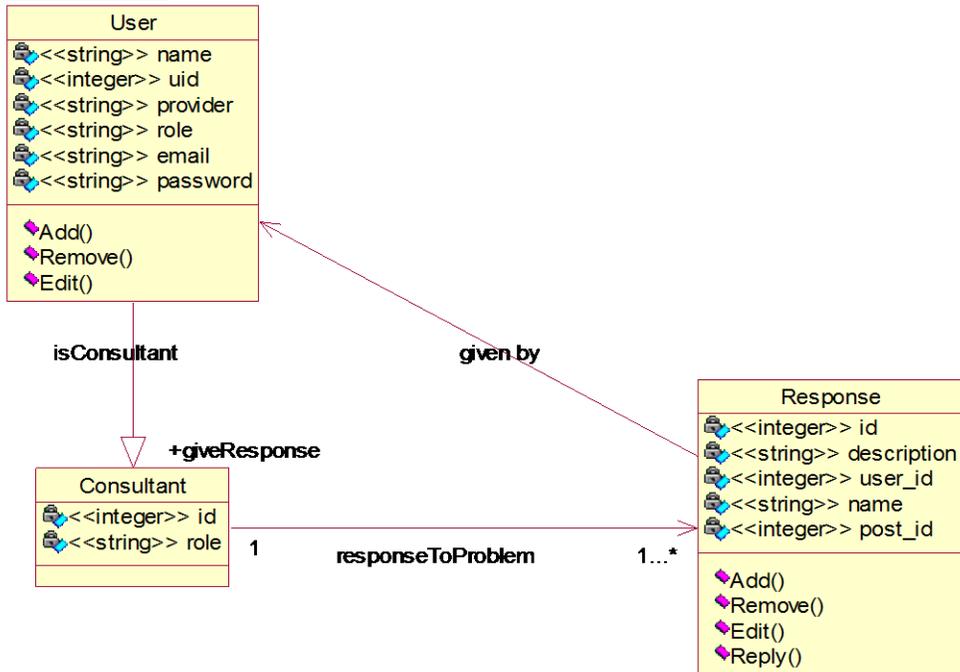
## Collaboration and Patten Diagram



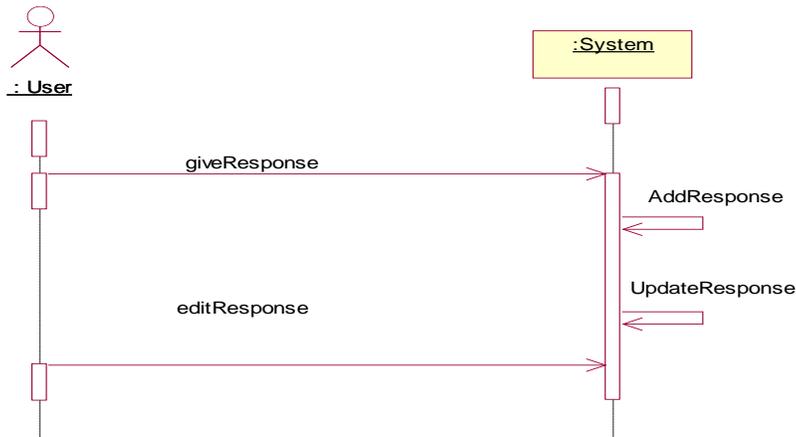
## State chart Diagram



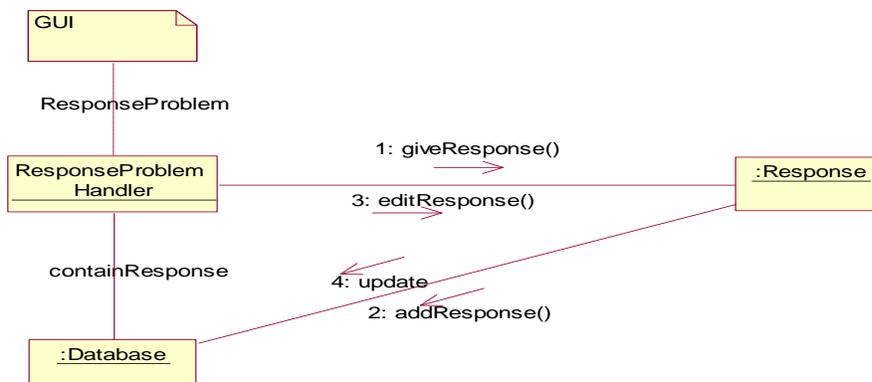
## Requirement 4: Response Problem Class Diagram



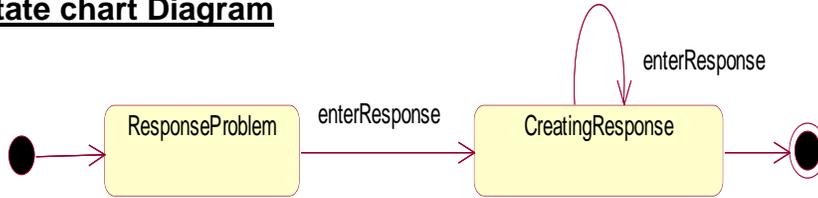
## Sequence System Diagram



## Collaboration and Patten Diagram

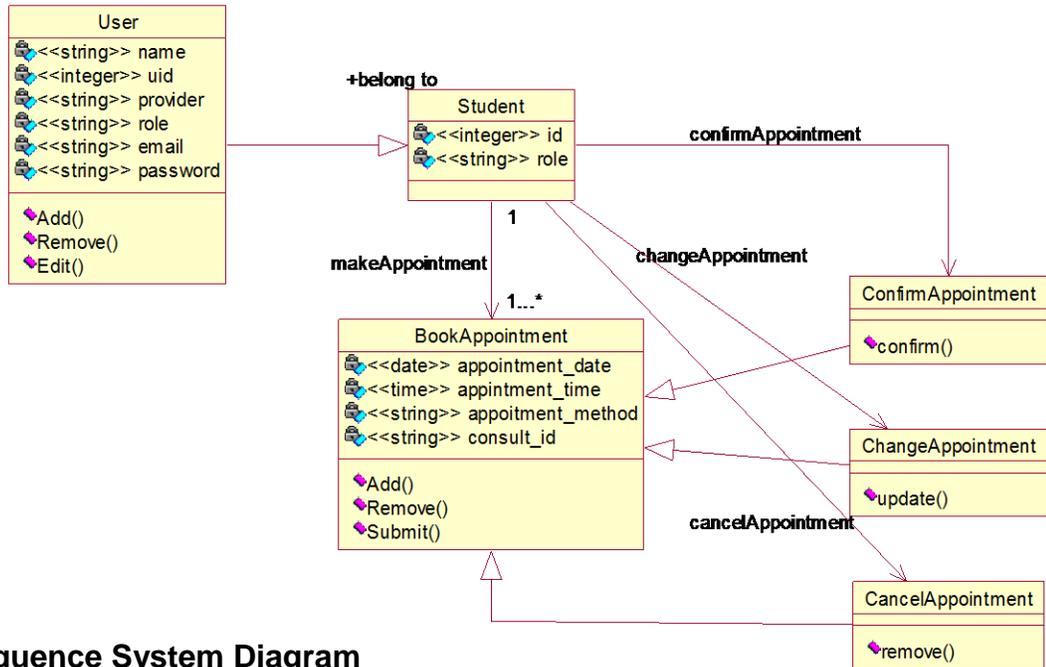


## State chart Diagram

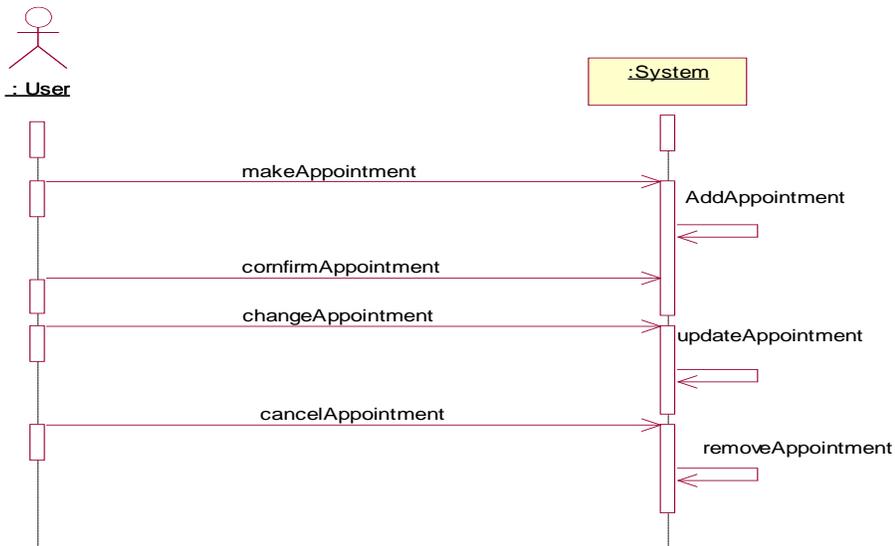


## Requirement 5: Booking Appointment

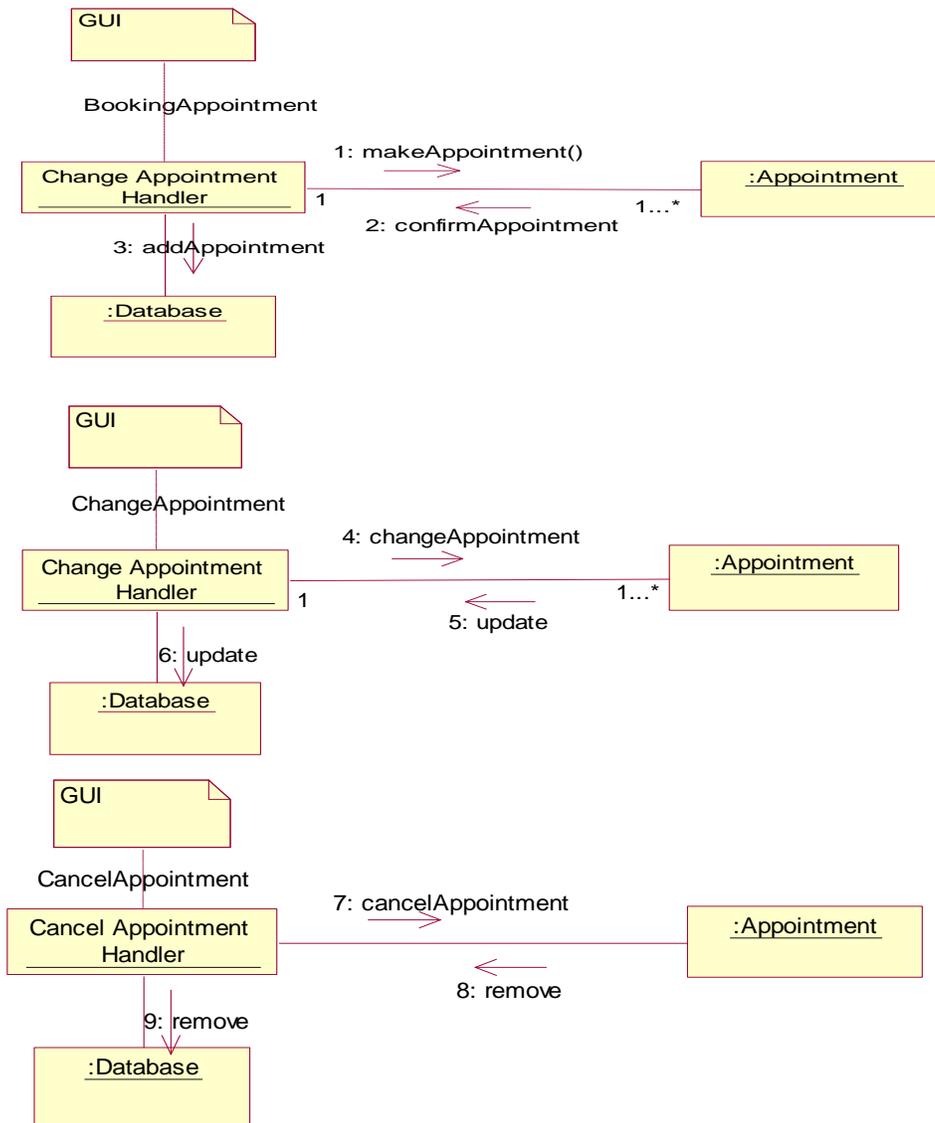
### Class Diagram



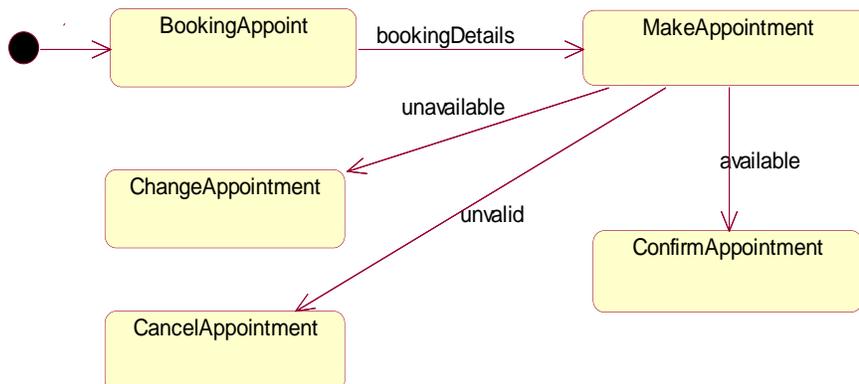
### Sequence System Diagram



## Collaboration and Patten Diagram

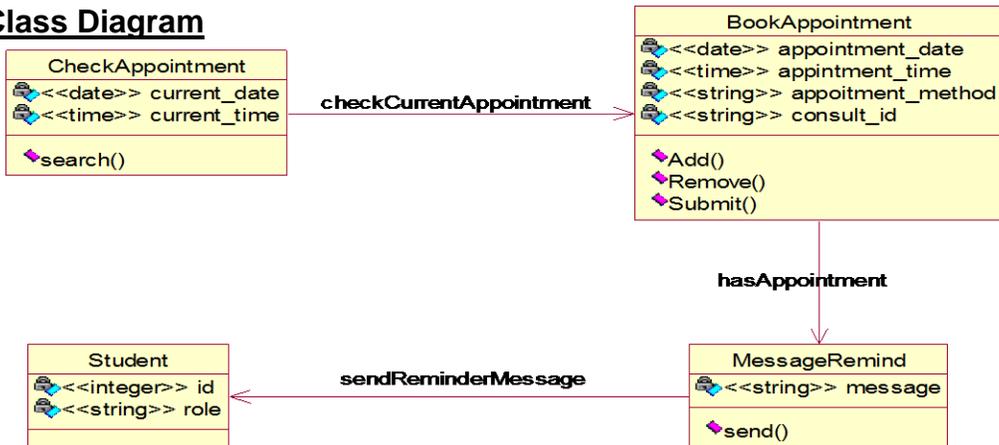


## State chart Diagram

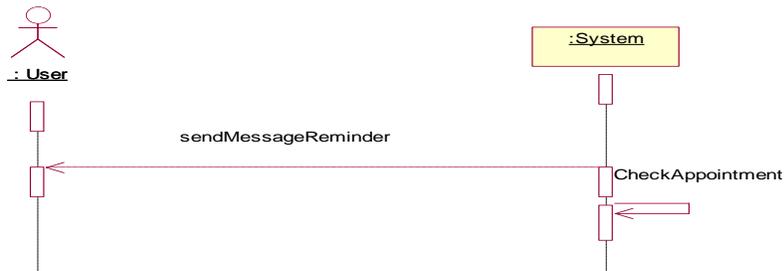


## Requirement 6: Appointment Reminder

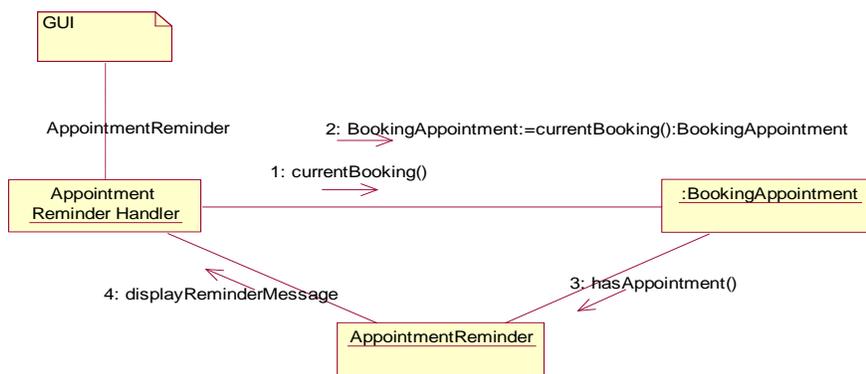
### Class Diagram



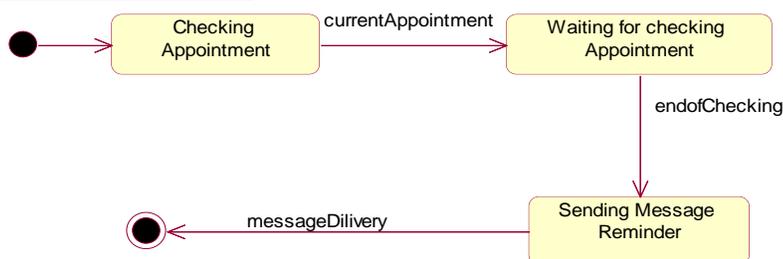
### Sequence System Diagram



### Collaboration and Patten Diagram

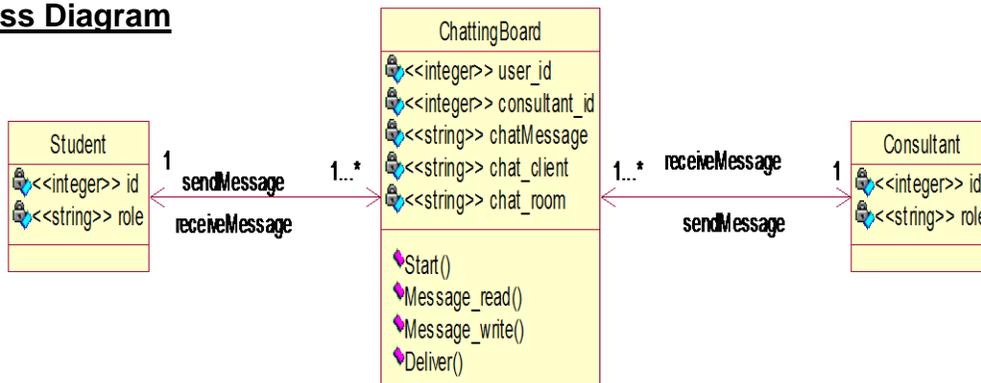


### State chart Diagram

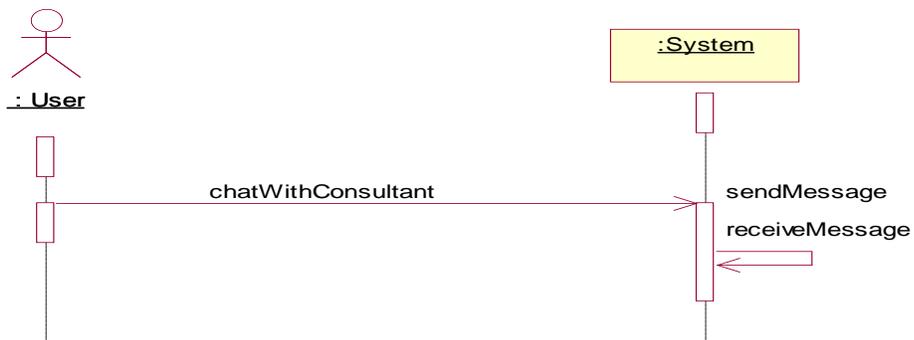


## Requirement 7: Chatting

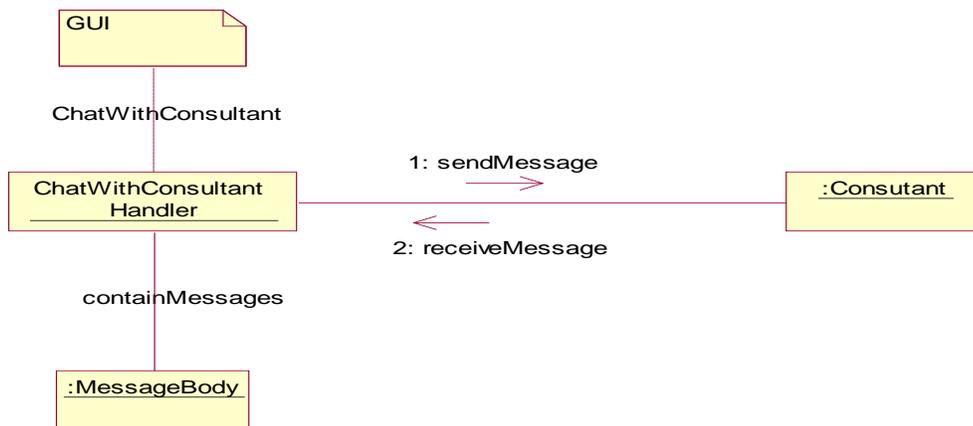
### Class Diagram



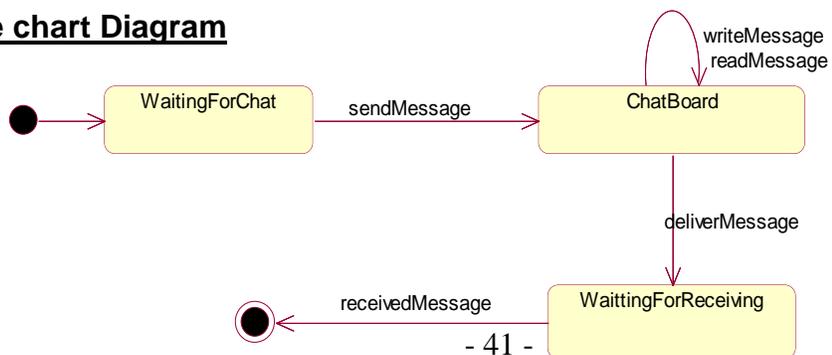
### Sequence System Diagram



### Collaboration and Patten Diagram



### State chart Diagram



## **2.3 Implementation**

### **Difficult according to my specialization**

The reason of choosing Ruby on Rails for the application was because it is a particularly good match for cloud computing, it's share-nothing architecture. Just toss off new instances of an application and it will just begin to run. It has less to do with Ruby itself, but more to do with smart architectural decisions. Ruby is known as having a high developer productivity rate and that's always good for business. And the cloud helps most business concerns in scaling. The dominant model for RoR application deployment is cloud, with platforms such as Slicehost (now part of Rackspace Cloud), Engine Yard and Heroku. Cloud services such as New Relic, FiveRuns and Scout provide the de facto standard monitoring and management frameworks, and cloud-based GitHub is the standard code version and developer collaboration tool for the RoR generation. Ruby is their first, or early programming language, as it is growing up with cloud platforms.

The most difficult thing is to use Ruby, which is unfamiliar for building the application. This is a big challenge. Learning syntax, learning how to use it, especially when problems or errors arise; means that time is required for searching for a solution or to fix the errors. Ruby on Rails are supported by a rich ecosystem of gems that collectively provide just about any capability a developer can think of. This is great for building up a complex application quickly, but it also seems to have many bloated applications, where the number of gems in the application's Gemfile is disproportionately large, when compared with the functionality provided. Sometime it is hard to install the gem because it does not like the vision of gem. The factor is that the application is deployed to cloud platforms which always update the latest version. For example, the application was deployed to Heroku but since Feb 24, 2016, Ruby 2.0.0 security maintenance reached its end-of-life. Span Ruby 2.0.0 will no longer receive security updates. One must upgrade the application to a supported Ruby version 2.2.4, to ensure that it is running in a secure environment. Every time a new version is update,

errors arise and time is required to fix the errors and test each function to make sure everything is working. Since Heroku updated a new version of Ruby, the application was transferred to cloud 9. It is a marvelous online integrated development environment which enables developers to get started with coding immediately with pre-setup workspaces, collaborate with their peers with collaborative coding features, and web development features like live preview and browser compatibility testing.

### **Difficulties**

Certain aspects of the implementation were difficult; the areas that were the most difficult were related to the automated response. Immediately after a problem is posted, AARS uses Google Custom Search API to search the web for links to useful and relevant resources. System receives a result set from Google custom search API, and creates an array of Ruby objects called result. Then AARS automated display the result which is related to the post. Users can also rate links suggested by AARS by assigning a value on a integral scale from 1 (poor) to 5 (excellent). AARS uses its own custom algorithm to intelligently sort results that takes into account users' feedback and online behaviour, namely; the number of times a link or response has been rated, the average rating for a link or response and the number of times a link has been clicked. It was difficult to find the way of implementing it; the snippets of code for where they were implemented are as follows:

### **Class Result**

System receives a result set from Google custom search API, create an array of Ruby objects called result.

```
require 'data'
class Result
  attr_accessor :title, :link_url, :click_count, :average_rating,
  :number_of_ratings

  def initialize(title, link_url, click_count = 0, average_rating = 0,
  number_of_ratings = 0)
    @title = title
```

```

        @link_url = link_url
        @click_count = click_count
        @average_rating = average_rating
        @number_of_ratings = number_of_ratings
    end
end

```

## Class Data

The links and rating are saved in the a database and used sort algorithm to sort the result by the number of times a link or response has been rated, the average rating for a link or response and the number of times a link has been clicked.

```

require 'result'
class Data
  def self.parse(results)
    result_set = []
    results.items.each do |result|
      # item = SavedLink.find_by_link_url(result.link.to_s) ?
      SavedLink.find_by_link_url(result.link.to_s) : Result.new(result.title,
result.link.to_s)
      if SavedLink.find_by_link_url(result.link.to_s)
        link = SavedLink.find_by_link_url(result.link.to_s)
        item = Result.new(link.title, link.link_url,
link.click_count, link.average_rating, link.ratings.count.to_i)
        result_set << item
      else
        item = Result.new(result.title.to_s, result.link.to_s)
        result_set << item
      end
    end
    result_set # return the result set
  end
  def self.sort(results)
    results.sort_by! {|object| [object.average_rating,
object.number_of_ratings, object.click_count] }
    results.reverse!
  end
end
end

```

## Class SaveLink

```

class SavedLink < ActiveRecord::Base
  has_many :ratings

```

```

def sum_of_ratings
  ratings.pluck(:rating).inject(0) {|sum, number| sum += number}
end

def average_rating
  sum_of_ratings/ratings.count.to_d
end
end

```

### Class SavedLinksController

```

class SavedLinksController < ApplicationController

  # before_action :set_saved_link, only: [:update_rating]

  def update_link
    @saved_link = ""
    if SavedLink.find_by_link_url(params[:link_url].to_s)
      @saved_link =
SavedLink.find_by_link_url(params[:link_url].to_s)
      @saved_link.click_count == nil ? @saved_link.click_count = 1 :
@saved_link.click_count += 1
      @saved_link.save
    else
      @saved_link = SavedLink.create(title: params[:title].to_s,
link_url: params[:link_url].to_s, click_count: 1)
      @saved_link.save
    end
    redirect_to @saved_link.link_url.to_s
  end

  def index
    @saved_links = SavedLink.all
  end

  def update_rating
    @saved_link = ""
    if SavedLink.find_by_link_url(params[:link_url].to_s)
      @saved_link =
SavedLink.find_by_link_url(params[:link_url].to_s)
      @saved_link.ratings.build(rating: params[:rating].to_i)
      @saved_link.save
    else

```

```

        @saved_link = SavedLink.create(title: params[:title].to_s,
link_url: params[:link_url].to_s, click_count: 0)
        @saved_link.ratings.build(rating: params[:rating].to_i)
        @saved_link.save
    end
    redirect_to URI(request.referer).path
end

# private

# def set_saved_link
#   @saved_link = SavedLink.find_or_create_by(link_url: params[:link_url])
# end

# def saved_link_params
#   params.require(:saved_link).permit(:link_url, :title, :click_count)
# end
end

```

Another difficult is real time chat system which allows users chat with consultants. With the first beta of Rails 5 was released recently. The biggest new feature is Action Cable, which provides support for implementing WebSockets with a pair of libraries for JavaScript (for the client) and Ruby (for the server) WebSockets are a convenient way to stream data between the client and server, making it easy to build apps that require real-time message passing. A chat room is the usual example of such an app: without anyone having to refresh the page, a message sent from one user to appear for all other connected users. In the past, this was implemented by having each client poll the server for new messages. WebSockets replace polling with two-way channels that stream messages to where they're needed, as soon as they're created, avoiding the overhead and latency of continuous polling over HTTP. It was difficult to implement in rails 4, but with WebSockets in rails 5 makes the work easier. Application is used first beta of Rails 5. The snippets of code for where they were implemented are as follows:

The chat room in the system is built on ActionCable. First channel is added, which can be used to communicate, via websockets between the client and the server. This function follow Hector Perez Arenas's online tutorial<sup>2</sup>.

### Setting up chat channel

```
# app/channels/room_channel.rb

class RoomChannel < ApplicationCable::Channel
  def subscribed
    stream_from "room_channel"
  end

  def unsubscribed
    # Any cleanup needed when channel is unsubscribed
  end

  def speak(data)
    ChatMessage.create! body: data['message'], conversation_id:
    data['conversation_id'], user_id: data['user_id']
  end
end
```

When a client connects to the channel, the #subscribed action is called. This subscribes the client to a stream called room\_channel. Whenever data is broadcast to the room\_channel stream, it is pushed to the clients.

The #speak action corresponds to a method in the client side code. When a user types a chat message a hits enter, App.room.speak is called on the client side, which in turn invokes this action on the server

### Client side

Rails has already generated some client side code for us. Let's start by handling the event when enter is pressed in the chat input field. Add this at the end of app/assets/javascripts/channels/room.coffee

```
$(document).on 'keypress', '[data-behaviour~=room_speaker]', (event) ->
  if event.keyCode is 13
```

---

<sup>2</sup> Arenas, H. (2015). *Rails 5 tutorial: How to create a Chat with Action Cable*. [online] Hector Perez Arenas. Available at: <https://hectorperezarenas.com/2015/12/26/rails-5-tutorial-how-to-create-a-chat-with-action-cable/> [Accessed 25 Apr. 2016]

```

jQuery ->
  text = $('#text').val()
  user_id = $('#user_id').val()
  conversation_id = $('#conversation_id').val()
  App.room.speak(text, user_id, conversation_id)
  event.target.value = ""
  event.preventDefault()

```

When you hit enter in the #room-speak input field, this pushes the content of the field to the chat channel by calling App.room.speak, which in turn sends it to the cable server.

```

App.room = App.cable.subscriptions.create "RoomChannel",

received: (data) -> # Called when there's incoming data on the websocket for
this channel
  $('#chats').append(data['message']);

speak: (message, user_id, conversation_id) ->
  @perform 'speak', message: message, user_id: user_id, conversation_id:
conversation_id

```

Active Job allows Rails application to work with common queries in a single interface. The job class is where the code that will execute by queue. There is a perform method which is called and sent whatever parameters were sent when the job was first enqueued

```

class ChatMessageBroadcastJob < ApplicationJob
  queue_as :default

  def perform(chat_message)
    ActionCable.server.broadcast 'room_channel', message:
render_chat_message(chat_message)
  end

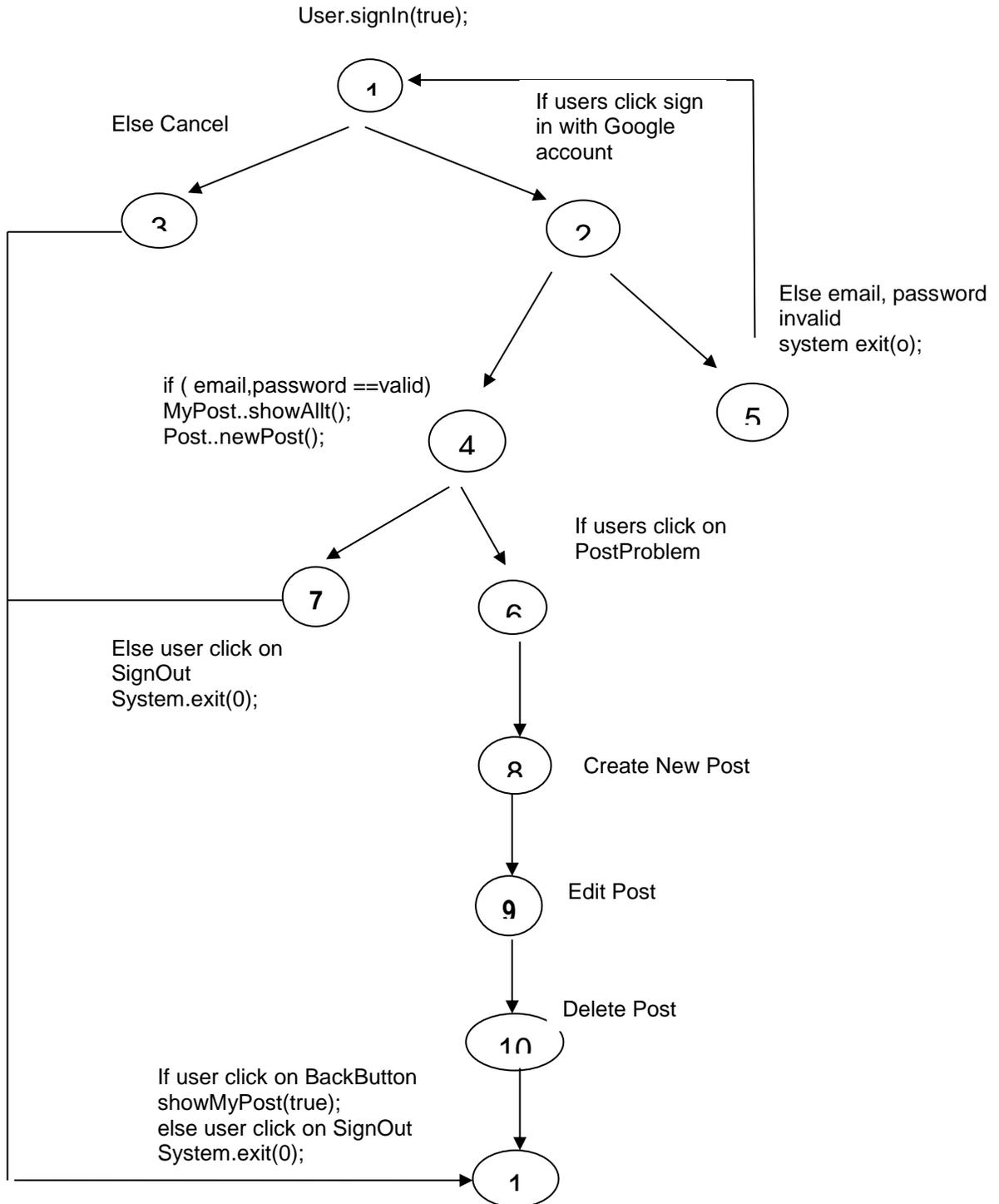
  private
  def render_chat_message(chat_message)
    ApplicationController.renderer.render(partial: "chat_messages/chat_message",
locals: { chat_message: chat_message})
  end
end

```

## 2.4 Testing

### 2.4.1. Testing plan:

#### Requirement < Post Problem >



1.) Determine the cyclomatic complexity

- $CC(G) = \text{Number}(\text{edges}) - \text{Number}(\text{nodes}) + 2$   
 $= 14 - 11 + 2 = 5$
- $CC(G) = \text{Number of Nodes with a condition} + 1$   
 $= 4 + 1 = 5$
- $CC(G) = \text{Number of regions} + 1$   
 $= 4 + 1 = 5$

2.) Determine a basis set of linearly independent paths

- path 1: 1 – 3 – 11
- path 2: 1 – 2 – 5 – 1
- path 3: 1 – 2 – 4 – 7 – 11
- path 4: 1 – 2 – 4 – 6 – 8 – 9 – 10 – 11

3) Prepare test cases that will force execution of each path in the basis set

Path 1 test case:

Test `SignIn(true)`. Expected result: users click cancel and system exit

Path 2 test case:

- Test `SignIn(true)`. Expected result: users enter but invalid email and password
- count  $\leq 3$  times then system exit;

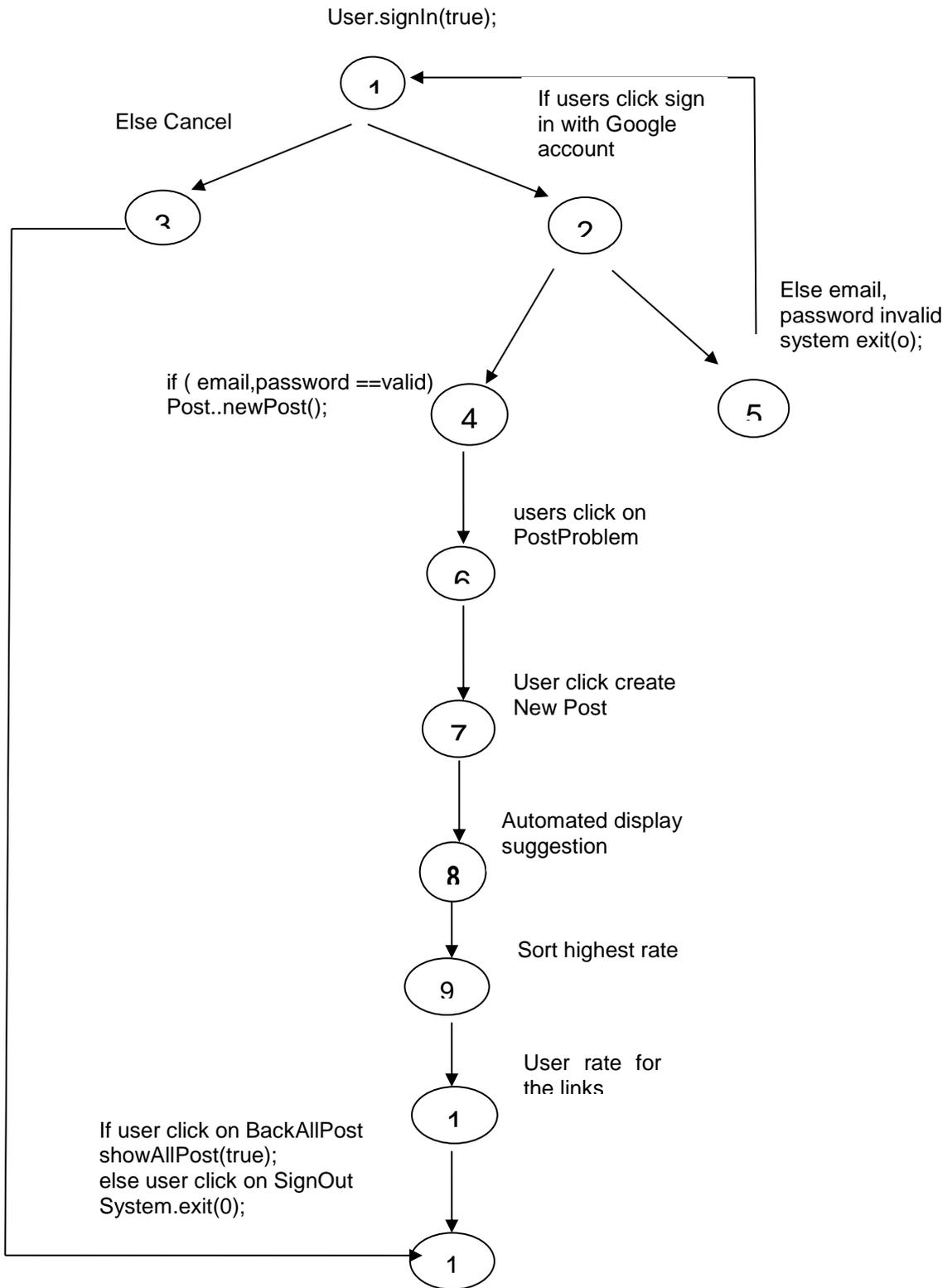
Path 3 test case:

- `SignIn(true)`.
- `PostProblem(true)`; then allow users to post their problems
- System exit or sign out

Path 4 test case:

- `SignIn(true)`.
- `PostProblem(true)`; then allow users to post their problems, edit and delete problems
- System exit or sign out

## Requirement < Automated Response >



1.) Determine the cyclomatic complexity

- $CC(G) = \text{Number}(\text{edges}) - \text{Number}(\text{nodes}) + 2$   
 $= 13 - 11 + 2 = 4$
- $CC(G) = \text{Number of Nodes with a condition} + 1$   
 $= 3 + 1 = 4$
- $CC(G) = \text{Number of regions} + 1$   
 $= 3 + 1 = 4$

2.) Determine a basis set of linearly independent paths

- path 1: 1 – 3 – 11
- path 2: 1 – 2 – 5 – 1
- path 3: 1 – 2 – 4 – 6 – 7 – 8 – 9 – 10 – 11

3) Prepare test cases that will force execution of each path in the basis set

Path 1 test case:

Test `SignIn(true)`. Expected result: users click cancel and system exit

Path 2 test case:

- Test `SignIn(true)`. Expected result: users enter but invalid email and password
- count  $\leq 3$  times then system exit;

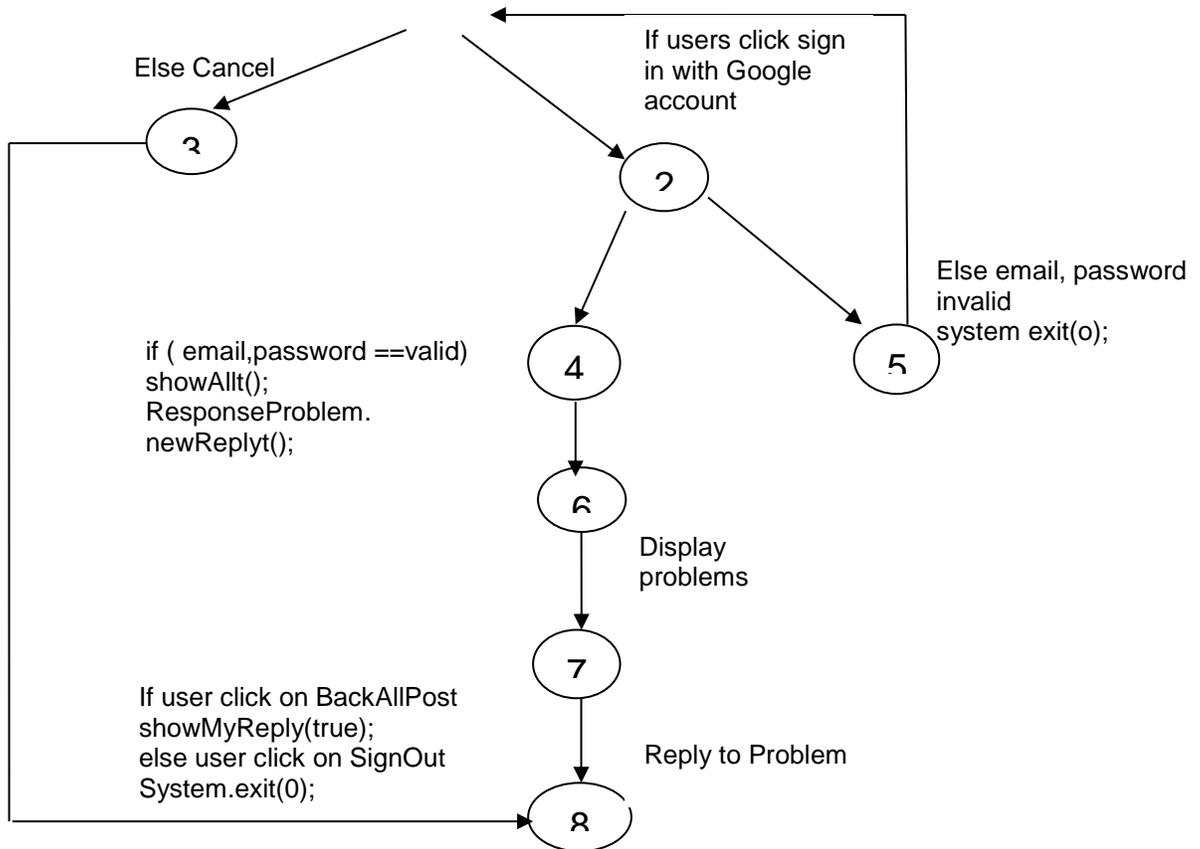
Path 3 test case:

- `SignIn(true)`.
- `PostProblem(true)`; then allow users to post their problems
- `AutomatedResponse(true)`. System automated display relevant resources.
- `AutomatedResponse(true)`. Then allow users rate for the link
- `AutomatedResponse(true)`. System sort and view the suggestion with highest average rate
- System exit or sign out

### Requirement < Response Problem >

User.signIn(true);





1.) Determine the cyclomatic complexity

- $CC(G) = \text{Number}(\text{edges}) - \text{Number}(\text{nodes}) + 2 = 10 - 8 + 2 = 4$
- $CC(G) = \text{Number of Nodes with a condition} + 1 = 3 + 1 = 4$
- $CC(G) = \text{Number of regions} + 1 = 3 + 1 = 4$

2.) Determine a basis set of linearly independent paths

- path 1: 1 – 3 – 8
- path 2: 1 – 2 – 5 – 1
- path 3: 1 – 2 – 4 – 6 – 7 – 8

3) Prepare test cases that will force execution of each path in the basis set

Path 1 test case:

- Test SignIn(true). Expected result: users click cancel and system exit

Path 2 test case:

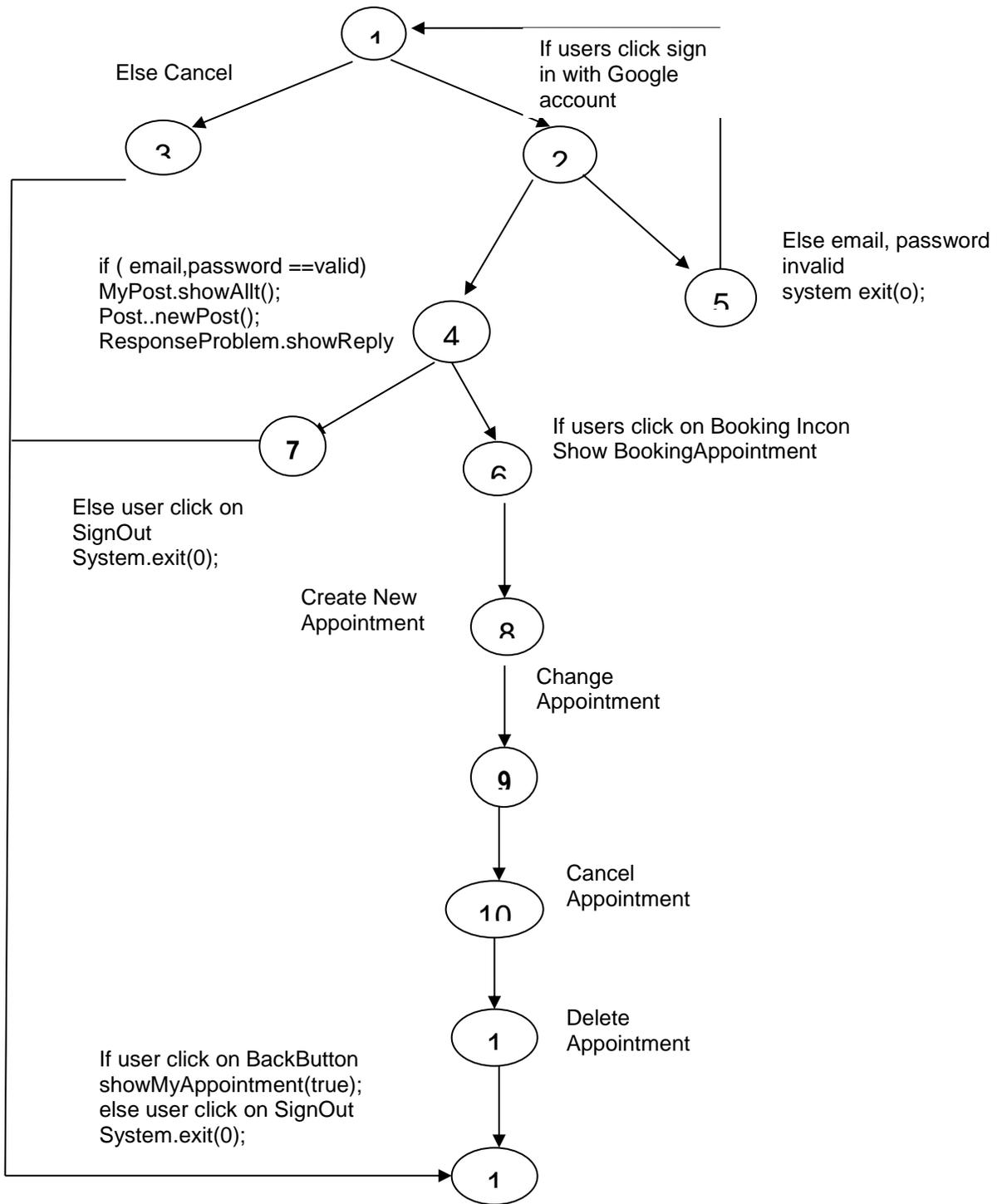
- Test SignIn(true). Expected result: users enter invalid email and password
- count <= 3 times then system exit;

Path 3 test case:

- ResponseProblem(true); then allow users to answer the problem
- System exit or sign out

### Requirement < Booking Appointment >

User.signIn(true);



1.) Determine the cyclomatic complexity

$$\begin{aligned}
 - \text{CC}(G) &= \text{Number}(\text{edges}) - \text{Number}(\text{nodes}) + 2 \\
 &= 15 - 12 + 2 = 5
 \end{aligned}$$

- $CC(G) = \text{Number of Nodes with a condition} + 1$   
 $= 4 + 1 = 5$
- $CC(G) = \text{Number of regions} + 1$   
 $= 4 + 1 = 5$

2.) Determine a basis set of linearly independent paths

- path 1: 1 – 3 – 12
- path 2: 1 – 2 – 5 – 1
- path 3: 1 – 2 – 4 – 7 – 12
- path 4: 1 – 2 – 4 – 6 – 8 – 9 – 10 – 11 – 12

3) Prepare test cases that will force execution of each path in the basis set

Path 1 test case:

- Test `SignIn(true)`. Expected result: users click cancel and system exit

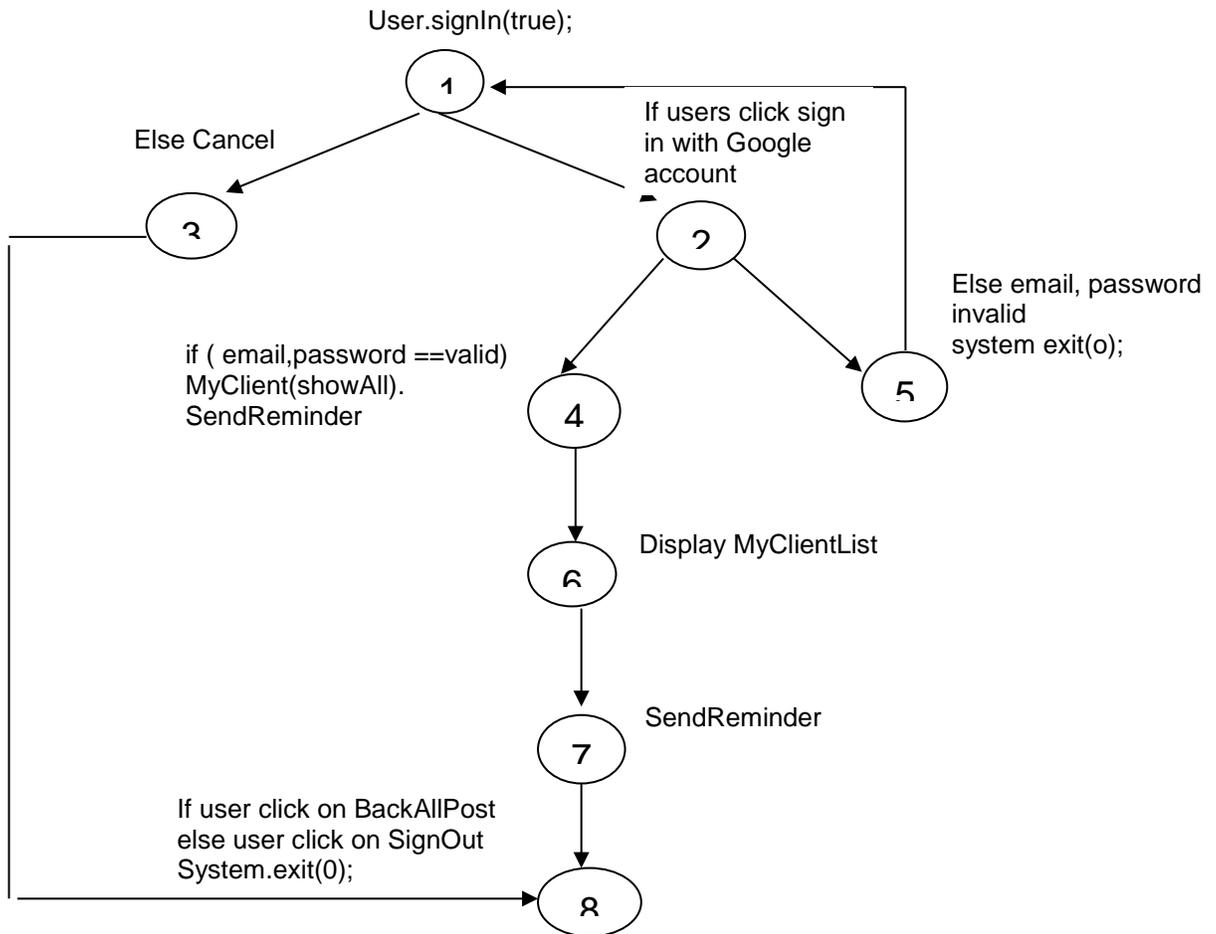
Path 2 test case:

- Test `SignIn(true)`. Expected result: users enter but invalid email and password
- count  $\leq 3$  times then system exit;

Path 3 test case:

- `SignIn(true)`.
- `BookingAppointment(true)`; then allow users to book an appointment
- `BookingAppointmentmen (true)`; Users can change appointment
- `BookingAppointmentmen (true)`. User can cancel appointment
- `BookingAppointmentmen (true)`. User can delete appointment
- System exit or sign out

## Requirement < Appointment Reminder >



1.) Determine the cyclomatic complexity

- $CC(G) = \text{Number}(\text{edges}) - \text{Number}(\text{nodes}) + 2 = 10 - 8 + 2 = 4$
- $CC(G) = \text{Number of Nodes with a condition} + 1 = 3 + 1 = 4$
- $CC(G) = \text{Number of regions} + 1 = 3 + 1 = 4$

2.) Determine a basis set of linearly independent paths

- path 1: 1 – 3 – 8
- path 2: 1 – 2 – 5 – 1
- path 3: 1 – 2 – 4 – 6 – 7 – 8

3) Prepare test cases that will force execution of each path in the basis set

Path 1 test case:

Test SignIn(true). Expected result: users click cancel and system exit

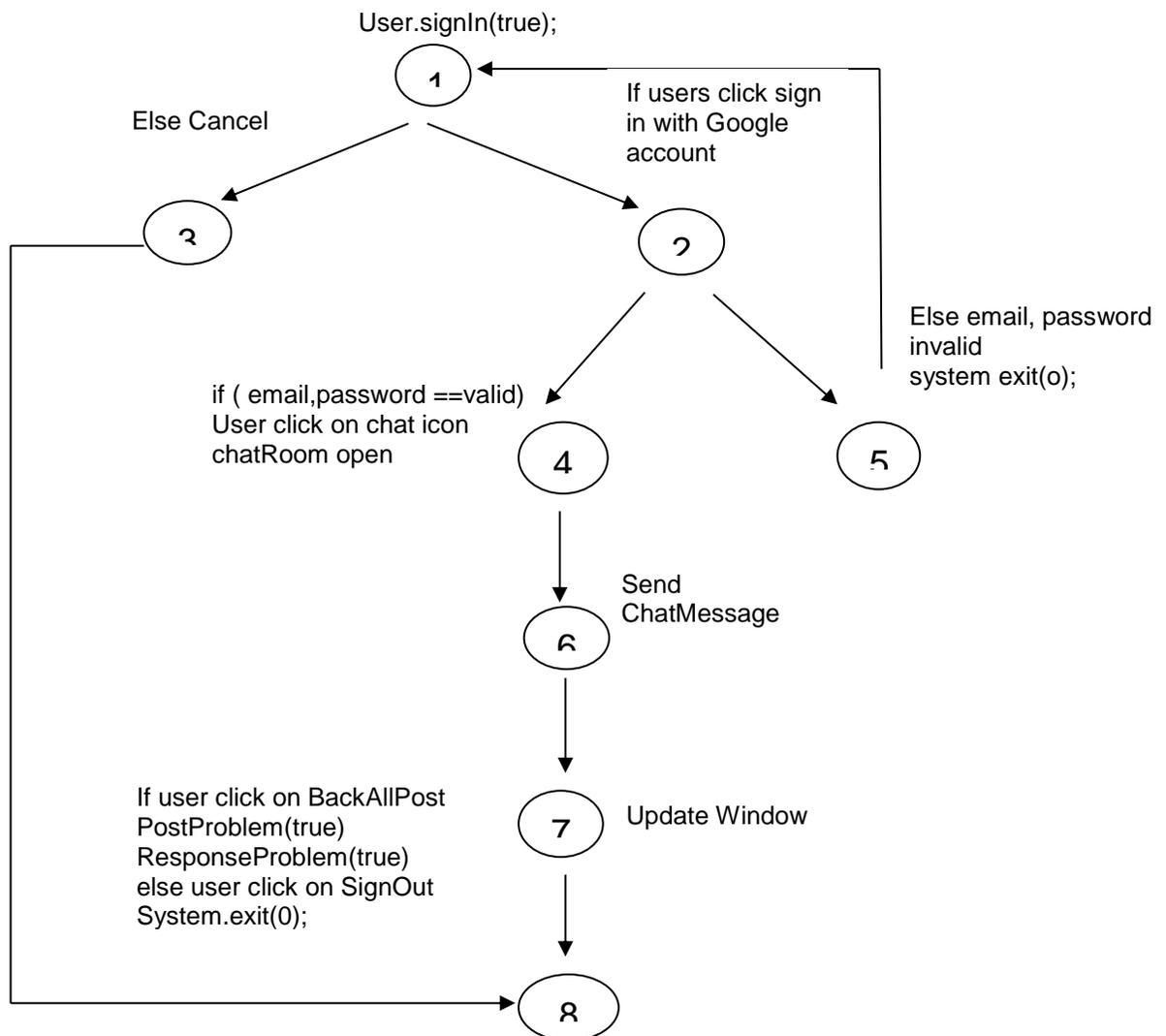
Path 2 test case:

- Test SignIn(true). Expected result: users enter but invalid email and password
- count <= 3 times then system exit;

Path 3 test case:

- AppointmentReminder(true); then allow consultant to send appointment reminder to user.
- System exit or sign out

### Requirement < Chatting >



1.) Determine the cyclomatic complexity

- $CC(G) = \text{Number}(\text{edges}) - \text{Number}(\text{nodes}) + 2 = 10 - 8 + 2 = 4$
- $CC(G) = \text{Number of Nodes with a condition} + 1 = 3 + 1 = 4$
- $CC(G) = \text{Number of regions} + 1 = 3 + 1 = 4$

2.) Determine a basis set of linearly independent paths

- path 1: 1 – 3 – 8
- path 2: 1 – 2 – 5 – 1
- path 3: 1 – 2 – 4 – 6 – 7 – 8

3) Prepare test cases that will force execution of each path in the basis set

Path 1 test case:

Test `SignIn(true)`. Expected result: users click cancel and system exit

Path 2 test case:

- Test `SignIn(true)`. Expected result: users enter but invalid email and password
- count  $\leq 3$  times then system exit;

Path 3 test case:

- `ChatRoom(true)`; then allow users to chat to each other
- System exit or sign out

## 2.4.2. Unit Testing

By default, every Rails application has three environments: development, test, and production. The database for each one of them is configured in `config/database.yml`.

Rails creates a test folder for us as soon as we create a Rails project using rails new Anonymous Automated Response System (AARS).

### Unit Testing for Models:

Rails will generate a default test for any Models or Scaffolds AARS generate, here is an example: The default test stub in `test/models/post_test.rb` looks like this:

```
require 'test_helper'
class PostTest < ActiveSupport::TestCase
  # test "the truth" do
  #   assert true
  # end
end
```

- `test_helper.rb` specifies the default configuration to run our tests. This is included with all the tests, so any methods added to this file are available to all your tests.
- The `PostTest` class defines a *test case* because it inherits from `ActiveSupport::TestCase`. `PostTest` thus has all the methods available from `ActiveSupport::TestCase`.

## Running Test

Running a test is as simple as invoking the file containing the test cases through rake test command.

```
$ rake test test/models/post_test.rb
.
Finished tests in 0.009262s, 107.9680 tests/s, 107.9680 assertions/s.
1 tests, 1 assertions, 0 failures, 0 errors, 0 skips
```

## Functional testing for Controllers

Controllers handle the incoming web requests to one's application and eventually respond with a rendered view. Functional testing tests the following type of functionalities of the controllers:

- Is the web request successful?
- Is the user redirected to the right page?
- Is the user successfully authenticated?
- Is the correct object stored in the response template?
- Is the response redirected as expected ?
- Is the expected template rendered?
- Is the routing as expected
- Does the response contain the expected tags?
- Is the appropriate message displayed to the user in the view?
- Is the information displayed right?
- Is the layout displayed right?

If a check fails or error then it is a sign that something goes wrong, to find these problems, certain functions are set up to detect these errors. The default assertion

messages which provide just enough information to help pinpoint the error has failed.

```
$ ruby unit/post_test.rb -n test_should_not_save_post_without_title
Loaded suite -e
Started
F
Finished in 0.102072 seconds.

  1) Failure:
test_should_not_save_post_without_title(PostTest) [/test/unit/post_test.rb:6]:
<false> is not true.

1 tests, 1 assertions, 1 failures, 0 errors
```

Now that AARS has used Rails scaffold generator for resource, it has already created the controller code and tests. Take look at the file `post_controller_test.rb` in the `test/controllers` directory

```
class PostControllerTest < ActionController::TestCase
  test "should get index" do
    get :index
    assert_response :success
    assert_not_nil assigns(:posts)
  end
end
```

In the `test_should_get_index` test, Rails simulates a request on the action called `index`, making sure the request was successful and also ensuring that it assigns a valid `posts` instance variable.

### Using Rake for Test :

- rake test : runs all tests in the test folder.

```
rake test
Run options: --seed 44145
# Running:
.....
Finished in 0.606734s, 11.5372 runs/s, 21.4262 assertions/s.
```

```
7 runs, 13 assertions, 0 failures, 0 errors, 0 skips
```

- rake test:controllers: Runs all the controller tests from test/controllers

```
rake test:controllers
Run options: --seed 12183
# Running:
.....
Finished in 0.558399s, 12.5358 runs/s, 23.2808 assertions/s.

7 runs, 13 assertions, 0 failures, 0 errors, 0 skips
```

## 2.5 Graphical User Interface (GUI) Layout

The user interface will be shown in a web browser. The user will be able to access the GUI from an internet enabled device. The application will have a navigation bar at the top. All problems and answers will be available to all users without any identification. Users will be able to ask questions or post their problems by signing in with the Google account. Once users sign in, they will be able to see their own posts and their information.

The system will automatically suggest possible answers to queries immediately after users post their problems or their questions. Users can also rate links suggested by assigning a value on a integral scale from 1 (poor) to 5 (excellent). AARS uses its own custom algorithm to intelligently sort results that takes into account users' feedback and online behaviour, namely; the number of times a link or response has been rated, the average rating for a link or response and the number of times a link has been clicked. The consultant can interact with users problems by posting answer(s) or giving advice

The application will have a booking appointment system if the user wishes chat personally with the counsellor or consultant. The system also automatically removes an appointment after user has seen a consultant. In addition, this application also provides a chatting functionality.

### Posting Problem

The screenshot shows the AARS web application interface. At the top, there is a navigation bar with the AARS logo and links for 'All Posts', 'My Posts', and 'Post Problem'. On the right side of the navigation bar, it says 'Welcome Thuy Linh Vo', 'User', and 'Sign Out'. Below the navigation bar, the main heading is 'Post Your Problem'. The form contains a text area with the description 'I always blame myself when something goes wrong' and a 'Create Post' button.

## Automated Response and suggestion

The screenshot shows the AARS interface with a dark header containing the logo and navigation links: All Posts, My Posts, My Appointments, Post Problem, Welcome Thuy Linh Vo Thi, User, Sign Out.

The main content area has a blue header: **Your Problem: How to avoid stress during an exam?** with a 'Back to All Posts' link.

Below this, a section titled '10 Suggestions' displays two suggestions:

- Managing Stress During Exams**  
[http://oasis.health.arizona.edu/health\\_topics/mental\\_health/managingstressexam.htm](http://oasis.health.arizona.edu/health_topics/mental_health/managingstressexam.htm)  
 Click Count: 0 Average Rating: 5.00 Number of Ratings: 1  
 Rate this link  
 Rating: 1 (selected) [Rate this link button]
- Deal with Exam Stress**  
<http://www.cboe.nic.in/examstress.htm>  
 Click Count: 0 Average Rating: 0.00 Number of Ratings: 0  
 Rate this link  
 Rating: 1 (selected) [Rate this link button]

An appointment can be arranged if the user wishes to chat personally with the consultant. “Calendar” icon is for booking appointment with consultant who has given the answers. “Envelop” icon is for sending an email to a consultant and “chat” icon is for chatting with consultant.

The screenshot shows the AARS interface with a dark header: AARS, All Posts, My Posts, My Appointments, Post Problem, Welcome Thuy Linh Vo Thi, User, Sign Out.

The main content area has a blue header: **how to deal with anger and depression?** with the date '21 Apr 2016 Thuy Linh Vo Thi'.

Below the header, there is a 'Destroy' button and a section titled '1 Answers' with an 'Answers' icon.

The answer text reads: "Anger, pain and depression are three negative experiences so closely bound together it can sometimes be hard to know where one ends and the other begins. Pain is a complex phenomenon that has emotional and physical components. The emotions play a huge role in the experience of pain, and pain is intimately associated with depression. It's long been known that the psychic pain of depression feeds anger. But just as often, anger fuels depression. A powerful emotion physiologically and emotionally, anger often feels good—but only for the moment. It can be a motivating force that moves you to action. But there are good actions and bad ones; it's vital to distinguish between the two. Many people confuse anger and hostility. Anger is a response to a situation that presents some threat. Hostility is a more enduring characteristic, a predisposition, a personality trait reflecting a readiness to express anger. Anger is usually anything but subtle. It has potent physiological effects. You feel it in your chest. You feel it in your head. You feel it coursing through your body." dated '21 Apr 2016'.

## Booking Appointment

The screenshot shows the AARS interface with a dark header: AARS, All Posts, My Posts, Post Problem, Welcome Thuy Linh Vo Thi, User, Sign Out.

Below the header, there are two main sections:

- New appointment**: A form with a 'Name' field containing 'Thuy Linh Vo'.
- Calendar**: A section titled 'These appointments have been booked already' with a calendar for 'Feb 2016'. The calendar grid shows dates from 01 to 07.

Consultants can either view the list of their client appointment or send reminder to students or clients.

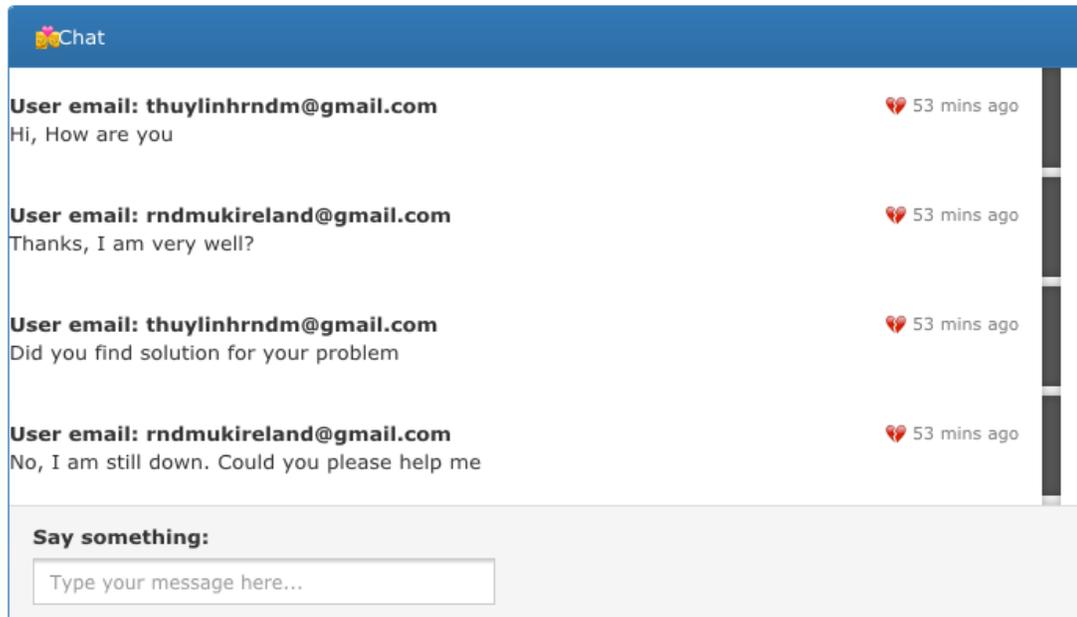
The screenshot displays the AARS web application interface. At the top, there is a navigation bar with the AARS logo, links for 'All Posts', 'My Answers', and 'My Clients' (which is active). On the right side of the navigation bar, it says 'Welcome Thuy Linh Vo!', 'Consultant', and 'Sign Out'.

Below the navigation bar, there are two main sections:

- Listing client appointments:** This section contains a table with columns for Name, Email, Day, and Time. Each row has a 'Send Reminder' button. The data shown is as follows:

Name	Email	Day	Time	Action
Thuy Linh Vo Thi	rndmukire...	26 Apr 16	13:20	Send Reminder
Thuy Linh Vo Thi	rndmukire...	02 May 16	12:30	Send Reminder
- Monthly Calendar:** This section shows a calendar for April 2016. The days of the week are listed as Mon, Tue, Wed, Thu, Fri, Sat, and Sun. The calendar grid shows dates from 28 to 01. The date 26 is highlighted with a blue '1'.

**Chat room where user can talk with consultant**



## 2.6. Users Testing:

Usability testing is a technique used in user-centered interaction designed to evaluate a product by testing it on users. It is the best way to understand how real users experience the application. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system. According to Jakob Nielsen "Usability is like cooking: everybody needs the results, anybody can do it reasonably well with a bit of training, and yet it takes a master to produce a gourmet outcome."<sup>3</sup>

Jakob Nielsen defined usability is a quality attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process.

Usability is defined by 5 quality components<sup>4</sup>:

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?

---

<sup>3</sup> Nielsen Norman Group. (2009). *Anybody Can Do Usability*. [online] Available at: <https://www.nngroup.com> [Accessed 30 Apr. 2016].

<sup>4</sup> Nngroup.com. (2012). *Usability 101: Introduction to Usability*. [online] Available at: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/> [Accessed 30 Apr. 2016].

- Efficiency: Once users have learned the design, how quickly can they perform tasks?
- Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it to use the design?

To conduct a usability test, begin by identifying the target audience. The target audience will consist of one or more user groups. For example, a group of students sign in with Google account and another group called consultants, sign in with Google account. Each user group is given tasks to perform during testing, that reflect their different usage patterns. For example, after signing in, students post a problem, send an email to a consultant, book an appointment, chat with the consultant, view their own posts and appointments. Consultant after signing in, can post the answers or give advice, check their clients, send a reminder to students and chat with a student.

Users were asked use the application and test all functionalities which belong to their areas. After answering the following questions:

- What do you think the purpose of this system is?
- Did you find what you were looking for?
- How did you find the design of the system?
- How are the icons used in the system?
- If you could only change one thing about this page, what would you change? Why?

There were 20 people involve in the testing system. As the result, most users think the purpose of the system is very helpful, in particular for young people, who have many questions in their lives, it is very hard for them to find the advice or answers, or for those who scare of go face to face. The good thing is users agree that they could find some advice from suggestions on the system. From the user's points of view, layout of the system is very simple, easy to use and clear and looks

professional. Users suggested that it would be good to have some instruction, because there is no post problem in the main menu. Users would like to use “Sign in with Gmail” instead of Google icon.

## **2.7 Evaluation**

To evaluate if the application is working in deferent platforms and different browsers. One must test the output of application against the information on the test server. If the information is correct then application is giving the correct information.

To evaluate whether the application is working in deferent cloud platforms or not, one must test whether output of the application is the same in each platform. Is the application working in both cloud 9 and Heroku cloud platform? There were problems with email and chat in Heroku because it requires account verification by having a credit card on file provided. Verified accounts may add any add-on to the application. Add-ons are third-party cloud services that provide out-of-the-box additional services for the application. However, it makes the full log stream available as a service - and several add-on providers have written logging services that provide things such as log persistence, search, and email and SMS alerts. For this reason, verification account in order email and chat functionalities are working in Heroku platform.

To evaluate the application with end users, users must be asked to use and interpret how the application could be used and how useful they think this application might be in their lives. As the result, there are possible feedbacks. There are 98% users think, the system is very useful, and they can get help anytime and anywhere. There are 85% users like the layout. There are 25% users suggest adding more colour in the layout. There are 25% users confuse the icons used. Their feedbacks help improve the performance of the system.

To evaluate whether the system has bugs or not, involves in an on-going test that take place at all stages of development. This has been completed with black box and white box testing.

### **3. Conclusions**

Anonymous Automated Response System has many advantages which can be brought to any organization. The system helps organizations improve their which can satisfy the customer's need with immediate effect. services For example "Samaritans", "ReachOut", " SpunOut" or " Childline" , etc can improve their services by the users' access to an immediate response to their queries. In the school situation, students have opportunities to share their problems without identifying themselves and teachers understand that students are be able to help find solutions before it is too late. For the Employees working in companies who deal with personal problems, especially those involved in relation with their colleagues or their employers. Users may find it difficult to discuss problem with the people concerned. They maintain that if they could access this application it would be a great help to improving personal job satisfaction. As a result they would be able to focus clearly on the overall objectives of the organization and thus improve its productivity. The system also economizes on the use of time, in that it will be available at anytime and anywhere.

There have been difficulties and challenges during the development stage of this project as discussed in the implementation section of this document. However this has been an interesting project and gave great job satisfaction when the application was sucessfully completed. In this document the requirements have been laid out, the design and architecture has been completed. The system has been tested in accordance to the testing plans laid out for each function, unit testing and users testing and the results for these tests have been documented in the evaluation section of the document. Screen shots of the GUI for different aspects of the system are supplied in the Graphical User Interface (GUI) Layout section.

#### **4. Further development or research**

Additionally a mobile and/or tablet friendly version of the application could be developed in the future to adapt to the currently expanding mobile device market.

Ruby on rail can also create mobile application using Ruby motion. RubyMotion is a tool that allows developers to write (code) an iOS app in Ruby. RubyMotion is a revolutionary toolchain for iOS, Android and OS X development. The most attractive part of the RubyMotion is to code an app (Cross platform app) in ruby. RubyMotion helps quick developing cross-platform native apps for iOS, Android and OS X.

RhoMobile Suite, based on the Rhodes open source framework, is a set of development tools for creating data-centric, cross-platform, native mobile consumer and enterprise applications. It allows developers to build native mobile apps using web technologies, such as CSS3, HTML5, JavaScript and Ruby. Developers can deploy RhoMobile Suite to write an app once and run it on the most-used operating systems, including iOS, Android, Windows Phone, Windows Mobile, Windows CE and Windows 8. Developers control how apps behave on different devices. RhoMobile Suite consists of a set of tools for building, testing, debugging, integrating, deploying and managing consumer and enterprise apps. It is a built-in Model View Controller pattern, an Object Relational Mapper for data intensive apps, integrated data synchronization, and a broad API set.

## 5. References

- Devcenter.heroku.com, (2015). *Getting Started with Ruby on Heroku | Heroku Dev Center*. [online] Available at: <https://devcenter.heroku.com/articles/getting-started-with-ruby#introduction> [Accessed 10 Nov. 2015].
- Gist, (2012). *Steps to set up a new Rails app, initialize a git repo, push to Github and deploy to Heroku*. [online] Available at: <https://gist.github.com/JennDudley/2493288> [Accessed 10 Nov. 2015].
- Hartl, M. (2015). *Ruby on rails tutorial*. 3rd ed. Upper Saddle River, NJ: Addison-Wesley.
- Heroku.com, (2015). *Build apps on Heroku: the innovative PaaS & leading dev experience*. [online] Available at: <https://www.heroku.com/platform> [Accessed 10 Nov. 2015].
- Intridea.github.io, (2015). *OmniAuth | Multi-provider Authentication for Web Applications*. [online] Available at: <http://intridea.github.io/omniauth/> [Accessed 3 Nov. 2015].
- Railscasts.com, (2015). *#241 Simple OmniAuth - RailsCasts*. [online] Available at: <http://railscasts.com/episodes/241-simple-omniauth?autoplay=true> [Accessed 10 Nov. 2015].
- Academy, T. (2014). *How to Send Emails in Rails - a Comprehensive Tutorial*. [online] Gotealeaf.com. Available at: <http://www.gotealeaf.com/blog/handling-emails-in-rails> [Accessed 13 Nov. 2015].
- Cse.google.com, (2015). *Sign in - Google Accounts*. [online] Available at: <https://cse.google.com/cse/setup/basic?cx=007092960594765751254:ajyutvxvgwe> [Accessed 20 Nov. 2015].
- Google Developers, (2015). *Introduction*. [online] Available at: <https://developers.google.com/custom-search/docs/tutorial/introduction> [Accessed 20 Nov. 2015].
- GitHub, (2015). *excid3/simple\_calendar*. [online] Available at: [https://github.com/excid3/simple\\_calendar](https://github.com/excid3/simple_calendar) [Accessed 27 Jan. 2016].
- GitHub, (2016). *wiseleyb/google\_custom\_search\_api*. [online] Available at: [https://github.com/wiseleyb/google\\_custom\\_search\\_api](https://github.com/wiseleyb/google_custom_search_api) [Accessed 27 Jan. 2016].
- Mejia, A. (2011). *Ruby on Rails Architectural Design - Adrian Mejia's Blog*. [online] Adrianmejia.com. Available at:

<http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/>  
[Accessed 4 Dec. 2015].

Morin, M. (2014). *Cloud Computing and Ruby: Interviewing Hampton Catlin*.  
[online] About.com Tech. Available at:  
<http://ruby.about.com/od/reviewsevents/p/hcatlin2.htm> [Accessed 11 Oct.  
2015].

Akita, F. and Akita, F. (2016). *Fixing DHH's Rails 5 Chat Demo | AkitaOnRails.com*. [online] Akitaonrails.com. Available at:  
<http://www.akitaonrails.com/2015/12/28/fixing-dhh-s-rails-5-chat-demo>  
[Accessed 25 Apr. 2016].

Arenas, H. (2015). *Rails 5 tutorial: How to create a Chat with Action Cable*.  
[online] Hector Perez Arenas. Available at:  
<https://hectorperezarenas.com/2015/12/26/rails-5-tutorial-how-to-create-a-chat-with-action-cable/> [Accessed 25 Apr. 2016].

DesignersLib.com. (2015). *10+ Free Bootstrap Chat Box Templates - DesignersLib.com*. [online] Available at:  
<http://www.designerslib.com/bootstrap-chat-box-templates> [Accessed 25  
Apr. 2016].

Morin, M. (2014). *Cloud Computing and Ruby: Interviewing Hampton Catlin*.  
[online] About.com Tech. Available at:  
<http://ruby.about.com/od/reviewsevents/p/hcatlin2.htm> [Accessed 11 Oct.  
2015].

Muhammed, A. (2015). *10 Best Free Bootstrap Chat Templates | Designrazor*.  
[online] Designrazor.net. Available at: <http://www.designrazor.net/best-free-bootstrap-chat-templates/> [Accessed 25 Apr. 2016].

Nngroup.com. (2009). *Anybody Can Do Usability*. [online] Available at:  
<https://www.nngroup.com/articles/anybody-can-do-usability/> [Accessed 30  
Apr. 2016].

Nngroup.com. (2012). *Usability 101: Introduction to Usability*. [online] Available  
at: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>  
[Accessed 30 Apr. 2016].

Docs.rhobile.com. (n.d.). *Rhobile | Welcome To RhoMobile Suite*. [online]  
Available at: <http://docs.rhobile.com/en/5.4/guide/welcome> [Accessed 3  
May 2016].

Rubymotion.com. (n.d.). *Guides | RubyMotion*. [online] Available at:  
<http://www.rubymotion.com/developers/guides/> [Accessed 3 May 2016].

## **6. Appendix**

### **6.1. Project Proposal**

#### **6.1.1. Objectives**

The application is for a social communication that provides a service which automatically suggests possible answers to queries immediately after users (e.g. students, employees...) post their problems or their questions. It also allows other users (e.g. counselors or consultancy or teachers...) to interact with them by posting answer(s) or giving advice. The posts and the answers are saved and displayed to all users without any personal information. The users can see their own post with their information and the other users only see the problem and the answer without any personal information.

An appointment can be arranged if the user wishes to speak or chat personally with the counsellor or consultant. The application will have a booking appointment if users wish to talk with consultant. The API will make provision to modify, create and display calendar events as well as work with many other calendar-related objects. In addition, this application will use Gmail API for chatting and speaking functionalities. Moreover, the application will economize on the use of time and in that it is available at anytime and anywhere.

This application will have the following characteristics.

- Login System: a system that the users will login to and get access to the information that they are allowed to see.
- Simply Layout: a simply layout that is easy to understand and will not overload the users with unnecessary information.
- Dynamic Scaling: a layout the can be used and look good on a PC, Tablet or Smartphone.

### **6.1.2. Background**

The reasons I have chosen to do this project are the following: First of all I want to use my knowledge, skills and new technologies which I have learnt to create something useful that can be of benefit to the users. The reason I chose the topic because I found that many people especially young people have problems communicating with friends, parents, families, colleagues and school. They have no one with whom to share their concerns or no one they can trust.

I was encouraged by a group of consultants who were longing to help people to discover more meaningful ways of communicating. So I decided to build the application that would allow the users freedom to share their own problem and discuss with consultants

### **6.1.3. Research**

I found in my research that there are some organizations that have a webpage which helps people who have problems with friends, families, school or work. Their services allow the users to make a call from a mobile or from a landline. Ideally there is someone to receive the call, listen and give advice. However problems may arise in that a caller may have to wait a considerable time for someone to answer the call or the organization may have set specific times when the calls will be answered. This arrangement may not suit the immediate need of the caller.

The following gives details of a few such organizations dealing with a cross section of society who is facing different types of problems, for example,

- “Samaritans” who provide a network of people you could 'ask' about anything;
- “ReachOut.com” helps young people get through tough times by providing quality mental health information and covering issues that can impact on mental health, “ReachOut.com” takes the mystery out of mental health;
- The website, “SpunOut.ie”, carries a range of health information for young people, including mental health, sexual health, exam stress and general lifestyle information. “SpunOut” also has an extensive online directory allowing site visitors to search for supports and services in their area;

- Childline.ie provides a free and confidential listening service to children and young people up to the age of 18. The Childline helpline is open every day, 24 hours a day and Childline Online Chat is open every day from 10am to 10pm.

In studying the above examples it seems to me that an improvement is needed, to be an improvement to facilitate the users' access to an immediate response to their queries

In the light of this information I decided to develop a response system which automatically suggests possible answers to queries immediately after users (e.g. students, employees...) post their problems or their questions. It also allows other users (e.g. counselors or consultancy or teachers...) to interact with them by posting answer(s) or giving advice. The posts and the answers are saved and displayed to all users without any personal information. The users can see their own post with their information and the other users only see the problem and the answer without any personal information.

An appointment can be arranged if the user wishes to speak or chat personally with the counsellor or consultant. The application will have a booking appointment system. In addition, this application will use Gmail API for chatting and speaking functionalities. Moreover, the application will economize on the use of time, in that it is available at anytime and anywhere.

I have spoken with people from different backgrounds about my idea and have asked them what they would want, if they were to use the application. Teachers in secondary school, for example agree with me that the students have problem with studies, friends and families. Students feel they have nobody to talk to or trust, they are afraid to share their problems and as the result they keep them to themselves until perhaps it is too late to solve the problems. Teachers think the application will give students opportunities to share their problems without identifying themselves. This refers only to a group of teachers who will answer the

questions or give an advice. This will also help teachers themselves understand students and be able to help them find solutions before it is too late.

Employees working in companies also have to deal with personal problems, and those evolved in relation to their colleagues or their employers find it difficult to discuss these with the people concerned and they maintain that if they could access this application it would be a great help to improving personal job satisfaction. As a result they would be able to focus clearly on the overall objectives of the organization and thus improve their productivity.

#### **6.1.4. Technical Approach**

To build the application I will be using the Ruby on Rails where the framework is rails and the syntax is ruby.

GitHub is a Web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

The application will connect to database using SQLite3 and PostgreSQL. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. PostgreSQL is an object-relational database management system (ORDBMS) with an emphasis on extensibility and on standards-compliance.

Google Custom search API will be used for automated response and suggestion. When users click on the post button, the post will save in the database, display to user, send to search bar and automatically display possible suggestion.

For the front end design I will use foundation. The web application will be host on cloud platform using Heroku which is a platform as a service (PaaS).

Simple Calendar API will be used for display events of the days, weeks and months. Gmail API as a calling, chatting system and for sending email reminds about appointments.

I plan to take the following technical Approach:

Capture the requirements

Research what needs to be gathered.

Research How to gather different data types

Research how to summarize and display information

Design the Project

Create a Front end website application by using foundation

Learn Ruby syntax for the need of the project

Search the use of each API and how to connect them together

Use HTML, CSS, JavaScript to connect APIs

Learn to build, deploy and manage application on Heroku and Cloud 9 platforms.

#### **6.1.5. Special resources required**

- Google Account, Gmail API, Google Custom Search API, Calendar API, GitHub account, Cloud 9 and Heroku Account.
- Ruby 2.15 and Rails 4
- Sublime Text 3
- Learn ruby syntax and ruby on rails.

#### **6.1.6. Technical Details**

To build project I will use Ruby on Rails which is an open source web application framework, it is a full-stack framework. Ruby on Rails uses a well-known software engineering patterns and principles such as active record pattern, convention over configuration (COC), don't repeat yourself (DRY) and model-view-controller (MVC).

The application will connect to database using SQLite3 and PostgreSQL. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. PostgreSQL is an object-relational database management system (ORDBMS) with an emphasis on extensibility and on standards-compliance.

For a version control system, I will use Git which allows tracking the history of a collection of files and includes the functionality to revert the collection of files to another version and to use Github to host my repository.

Sublime Text is a sophisticated text editor for code, markup and prose. It is an amazing piece of software that is a clean, functional, and fast code editor.

The Rails community provides a wealth of plugins as Ruby Gems that simply add to project Gemfile and install. For example, the application used OmniAuth for the sign in with Google. It simply added gem *"omniauth-google-oauth2"* in the Gemfile, then bundle install.

The application "sign in" will use simple OmniAuth. It is a library that standardizes multi-provider authentication for web applications. It was created to be powerful, secure, and flexible. Any developer can create strategies for OmniAuth that can authenticate users via disparate systems. For example: Google, Facebook, Twitter,... The reason of using OmniAuth is because most users don't like to sign up for websites. They've already signed up to so many; using different usernames and password that trying to remember them is sometimes impossible.

Foundation is used for the front end design. Foundation is a framework to build the front-end, or client-facing part, of a website or web application. It lets me quickly prototype and create sites and apps that work on any device. HTML front end will be responsible for displaying the information on multiple device types. This service will be written in JavaScript, JQuery, CSS3 and HTML.

The application also uses Gmail API for sending email to book appointments. The Gmail API is a RESTful API that can be used to access Gmail mailboxes and send mail. For most web applications (including mobile apps), the Gmail API is the best choice for authorized access to a user's Gmail data.

Heroku allows deploying directly from popular tools like Git, GitHub or Continuous Integration (CI) systems. First install the Heroku Toolbelt. This provides you access to the Heroku Command Line Interface (CLI), which can be used for managing and scaling the applications and add-ons. A key part of the toolbelt is the heroku

local command, which can help in running applications locally. Once installed, use the heroku command from command shell.

#### **6.1.7. Evaluation**

To evaluate, if the project is working in deferent platforms and different browsers. Result will be testing the output of my project against the information on the test server. If the information is correct then my project is giving the correct information.

To evaluate the application with an end user, I will ask a few people I know, to use and interpret how the application could be used and how useful they think this application might be in their lives.

To evaluate if my project has bugs: This will be an on-going test that will take place at all stages of development. I will do black box and white box testing.

## **6.2. *Project Plan***

ID	Task Mode	Task Name	Duration	Start	Finish	Predec	Resource Names
1	 	<b>Project Proposal</b>	<b>16 days</b>	<b>Mon 14/09/15</b>	<b>Sat 03/10/15</b>		
2		Research and write Project Proposal	12 days	Mon 14/09/15	Tue 29/09/15		Thuy Linh
3		Review Project Proposal	1 day	Wed 30/09/15	Wed 30/09/15		Thuy Linh
4		Reflective Journal	1 day	Thu 01/10/15	Thu 01/10/15		Thuy Linh
5	 	Submit Project Proposal			Fri 02/10/15		Thuy Linh
6	 	Submit Reflective Journal			Sat 03/10/15		Thuy Linh
7		<b>Requirement Specification</b>	<b>25 days</b>	<b>Mon 05/10/15</b>	<b>Fri 06/11/15</b>		
8		<b>Introduction</b>	<b>5 days</b>	<b>Mon 05/10/15</b>	<b>Fri 09/10/15</b>		
9		Purpose	1 day	Mon 05/10/15	Mon 05/10/15		Thuy Linh
10		Project Scope	2 days	Tue 06/10/15	Wed 07/10/15		Thuy Linh
11		Definitions, Acronyms and Abbreviations	1 day	Thu 08/10/15	Thu 08/10/15		Thuy Linh
12		User Requirement Definitic	1 day	Fri 09/10/15	Fri 09/10/15		Thuy Linh
13		<b>Rquirements Specification</b>	<b>12 days</b>	<b>Mon 12/10/15</b>	<b>Tue 27/10/15</b>		
14		<b>Functionion requirements</b>	<b>6 days</b>	<b>Mon 12/10/15</b>	<b>Mon 19/10/15</b>		
15		Use Case Diagram	3 days	Mon 12/10/15	Wed 14/10/15		Thuy Linh
16	 	Requirements	3 days	Thu 15/10/15	Mon 19/10/15		Thuy Linh
17		<b>Non-Functional Requirement:</b>	<b>6 days</b>	<b>Mon 19/10/15</b>	<b>Mon 26/10/15</b>		
18	 	Performance/Response time requirement	1 day	Mon 19/10/15	Mon 19/10/15		Thuy Linh
19		Availability requirement	1 day	Tue 20/10/15	Tue 20/10/15		Thuy Linh
20		Recover requirement	1 day	Wed 21/10/15	Wed 21/10/15		Thuy Linh
21		Robustness requirement	1 day	Thu 22/10/15	Thu 22/10/15		Thuy Linh
22		Security requirement	0.5 days	Fri 23/10/15	Fri 23/10/15		Thuy Linh
23		Reliability requirement	0.5 days	Fri 23/10/15	Fri 23/10/15		Thuy Linh
24		Maintainability requiremer	0.5 days	Mon 26/10/15	Mon 26/10/15		Thuy Linh
25		Reusability requirement	0.5 days	Mon 26/10/15	Mon 26/10/15		Thuy Linh
26		<b>Interface requirements</b>	<b>8 days</b>	<b>Tue 27/10/15</b>	<b>Thu 05/11/15</b>		
27	 	GUI	3 days	Tue 27/10/15	Thu 29/10/15		Thuy Linh
28	 	Application Programming Interfaces (API)	3 days	Thu 29/10/15	Mon 02/11/15		Thuy Linh
29		System Architecture	1 day	Tue 03/11/15	Tue 03/11/15		Thuy Linh
30		System Evolution	1 day	Wed 04/11/15	Wed 04/11/15		Thuy Linh
31	 	Submit Requirement Specification			Fri 06/11/15		Thuy Linh

32		★	▸ <b>Project Analysis &amp; Design</b>	21 days	Fri 06/11/15	<u>Fri 04/12/15</u>		
33	🚫	★	Architecture Design	2 days	Fri 06/11/15	Mon 09/11/15	Thuy Linh	
34	🚫	★	Logical View	8 days	Mon 09/11/15	Wed 18/11/15	Thuy Linh	
35	🚫	★	System Design	8 days	Wed 18/11/15	Fri 27/11/15	Thuy Linh	
36	🚫	★	Database Design	1 day	Fri 27/11/15	Fri 27/11/15	Thuy Linh	
37	🚫	★	Reflective Journal	1 day	Sat 28/11/15	Sat 28/11/15	Thuy Linh	
38		🚫	Submit Reflective Journal			Sat 28/11/15	Thuy Linh	
39	🚫	★	User Interface Design	5 days	Sun 29/11/15	Thu 03/12/15	Thuy Linh	
40		🚫	Submit Project Analysis & Design			Fri 04/12/15	Thuy Linh	
41		★	▸ <b>Prototype</b>	85 days	Sat 05/12/15	<u>Thu 31/03/16</u>		
42	🚫	★	Sign In	3 days	Sat 05/12/15	Tue 08/12/15	Thuy Linh	
43	🚫	★	Posting Problem	5 days	Sat 12/12/15	Thu 17/12/15	Thuy Linh	
44	🚫	★	Automated Response	20 days	Thu 17/12/15	Wed 13/01/16	Thuy Linh	
45	🚫	★	Response Problem	6 days	Wed 13/01/16	Wed 20/01/16	Thuy Linh	
46	🚫	★	Booking Appointment	5 days	Wed 20/01/16	Tue 26/01/16	Thuy Linh	
47		★	Reflective Journal	1 day	Wed 27/01/16	Wed 27/01/16	Thuy Linh	
48		★	Mid-point Presentation Documentation	5 days	Thu 28/01/16	Wed 03/02/16	Thuy Linh	
49		🚫	Submit Mid-point Presentation Documentation			Thu 04/02/16	Thuy Linh	
50		🚫	Submit Reflective Journal			Thu 04/02/16	Thuy Linh	
51		★	Styling and Font-end	8 days	Thu 04/02/16	Mon 15/02/16	Thuy Linh	
52		★	Appointment Reminder	11 days	Tue 16/02/16	Tue 01/03/16	Thuy Linh	
53		★	Chatting System	20 days	Wed 02/03/16	Tue 29/03/16	Thuy Linh	
54		🚫	Project Code complete			Thu 31/03/16	Thuy Linh	
55		★	▸ <b>Project Final report</b>	29 days	Fri 01/04/16	<u>Wed 11/05/16</u>		
56	🚫	★	Implementation	6 days	Sat 02/04/16	Fri 08/04/16	Thuy Linh	
57	🚫	★	Testing	5 days	Sat 09/04/16	Thu 14/04/16	Thuy Linh	
58		★	Evaluation	1 day	Fri 15/04/16	Fri 15/04/16	Thuy Linh	
59	🚫	★	Checking, Bug fixing	10 days	Sat 16/04/16	Thu 28/04/16	Thuy Linh	
60		🚫	Project final Documentation and code			Tue 10/05/16	Thuy Linh	
61		★	<b>Project Presentation</b>	3 days	Wed 18/05/16	<u>Fri 20/05/16</u>	Thuy Linh	
62		★	Project Showcase	1 day	Wed 25/05/16	<u>Wed 25/05/16</u>	Thuy Linh	

### **6.3. Requirement Specification**

#### **Table of Contents**

- 1 Introduction
  - 1.1 Purpose
  - 1.2 Project Scope
  - 1.3 Definitions, Acronyms, and Abbreviations
- 2 User Requirements Definition
- 3 Requirements Specification
  - 3.1 Functional requirements
    - 3.1.1 Use Case Diagram
    - 3.1.2 Requirement 1 <Log In>
    - 3.1.3 Requirement 2 <Posting Problem>
    - 3.1.4 Requirement 3 <Automated Response>
    - 3.1.5 Requirement 4 <Response Problem>
    - 3.1.6 Requirement 5 <Booking Appointment>
    - 3.1.7 Requirement 6 <Chatting System>
    - 3.1.8 Requirement 7 <Appointment Reminder>
  - 3.2 Non-Functional Requirements
    - 3.2.1 Performance/Response time requirement
    - 3.2.2 Availability requirement
    - 3.2.3 Physical environment requirement
    - 3.2.4 Robustness requirement
    - 3.2.5 Security requirement
    - 3.2.6 Reliability requirement
    - 3.2.7 Maintainability requirement
    - 3.2.8 Portability requirement
    - 3.2.9 Resource utilization requirement
- 4 GUI
- 5 System Architecture
- 6 Process Flow Diagram
- 7 System evaluation

### **6.3.1. Introduction**

#### **6.3.1.1. Purpose**

The purpose of this document is to set out the requirements for the development of a social communication that allows users to interact with each other by posting question(s) and answer(s), chatting or talking that is all happening in one location with some extra tools to make communication more convenient.

This specification describes the functional and performance requirements for this project. This document also defines technical terminology and illustrates factors that affect software performance, reliability and security.

#### **6.3.1.2. Project Scope**

The scope of the project is to develop a social media communication that provides a service which automatically suggests possible answers to queries immediately after users (e.g. students, employees...) post their problems or their questions. It also allows other users (e.g. counselors or consultancy or teachers...) to interact with them by posting answer(s) or giving advice. The posts and the answers are saved and displayed to all users without any personal information. The users can see their own post with their information and the other users only see the problem and the answer without any personal information.

An appointment can be arranged if the user wishes to speak or chat personally with the counsellor or consultant. The application will have a booking appointment if users wish to talk with consultant. The API will make provision to modify, create and display calendar events as well as work with many other calendar-related objects. In addition, this application will use Gmail API for chatting and speaking functionalities. Moreover, the application will economize on the use of time and in that, it is available at anytime and anywhere.

### 6.3.1.3. Definitions, Acronyms, and Abbreviations

User	Any person who will interact with the system. This could be individual person who would like to share their problems and longing for help. This could be individual person who would like to help or give an advice or Individual person who maintain, repair the system or monitor any issues with the system
Roles	Roles are a powerful tool that allow a collection of users into a single unit against which they can apply permissions in a database.
API	Application <b>P</b> rogram <b>I</b> nterface - API is a set of routines, <i>protocols, and tools for building software applications.</i> <i>The API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components. API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together</i>
Ruby on Rails	Ruby on Rails is a web development framework written in the Ruby programming language
HTML	Hyper Text Mark-up Language
MVC	Model–View–Controller is a software architectural pattern for implementing user interfaces <ul style="list-style-type: none"> <li>- A model stores data that is retrieved according to commands from the controller and displayed in the view</li> <li>- Controller is a server-side component of Rails that responds to external requests from the web server to the application, by determining which view file to rend</li> <li>- A view generates an output presentation to the user based on changes in the model</li> </ul>
Heroku	Heroku is a cloud platform as a service (PaaS) supporting several programming languages
OmniAuth	OmniAuth is a library that standardizes multi-provider authentication for web applications. It was created to be powerful, secure, and flexible. Any developer can create strategies for OmniAuth that can authenticate users via disparate systems
Sublime Text	Sublime Text is a sophisticated text editor for code, mark up and prose. It is an amazing piece of software that is a clean, functional, and fast code editor.

### **6.3.2. User Requirements Definition**

The application was requested by a group of consultants who are longing to help people to discover a more meaningful ways of communicating. This application allows users freedom to share their own problems and discuss with consultants.

The application will be free to consultants. Users and consultants will not know each other. There is no personal information shown. It is a secret room. Moreover, the application will economize on the use of time and in that it is available at anytime and anywhere.

This system will require the following:

- A system that will run on multiple devices computer, iPad and mobile.
- System that has a secure login system.
- A system that has a web interface.
- A system that allows user to input and display information.
- A system that can store information about user problem.
- A system that user login can operate with different roles: user, consultant and admin.
- A system that can use automated display suggestion after user has posted the problem
- A system that has a booking appointment.
- A system that has a chatting and talking system.

The application should be user friendly and simplistic with its user interface, so that the different sections are easy to use.

### **6.3.3. Requirements Specification**

The system will allow for different users as follows:

**User 1:** Log in with Google account as user's role. Then users should be able to post their problems, view their own questions and list of all problems. The user should then be able to book an appointment and chat or talk with consultants or counselors as they wish.

**User 2:** Consultants log in with Google account. Consultants should be able to see all problems and manage to give the advice.

**User 3:** Admin log in with Google account. Admin should be able to see all the users, posts and have full permission on the post. This user will be a System administrator who will have user privileges to start and shut the system and monitor any issues with the system.

#### **6.3.3.1. Functional requirements**

- **Web Based Interface:** This system will have a web based interface so that it can be viewable on multiple devices: PC and tablets or phones
- **User Login:** This system will have a user login to ensure only the right users get to see the content. The user should be able to use an OAuth login to then login to all the available social media feeds. This system will have users' role that allow the collection of users into a single unit against which they can apply permissions in a database.
- **Automated Response:** This system will display automatically the result from outside resources which relate to users problem and give user some suggestions.
- **Chat System:** This system will have a chatting system that allow user to chat with consultants
- **Booking system:** This system will have a booking system so that user can make an appointment with consultant and a calendar which displays users' appointment. It will automatically remove the appointment when it is over.
- **Posting system:** This system will allow users post problems or questions.
- **Responding system:** This system will allow consultants or counsellors to answer the problem which users have posted.
- **Reminder system:** This system will automatic remind the users about their appointments.

### 6.3.1.1. Use Case Diagram

The Use Case Diagram provides an overview



### 6.3.1.2. Requirement 1 <Log In>

#### **Description & Priority**

This function is to allow a user to securely login to Anonymous Automated Response System.

#### **Use Case**

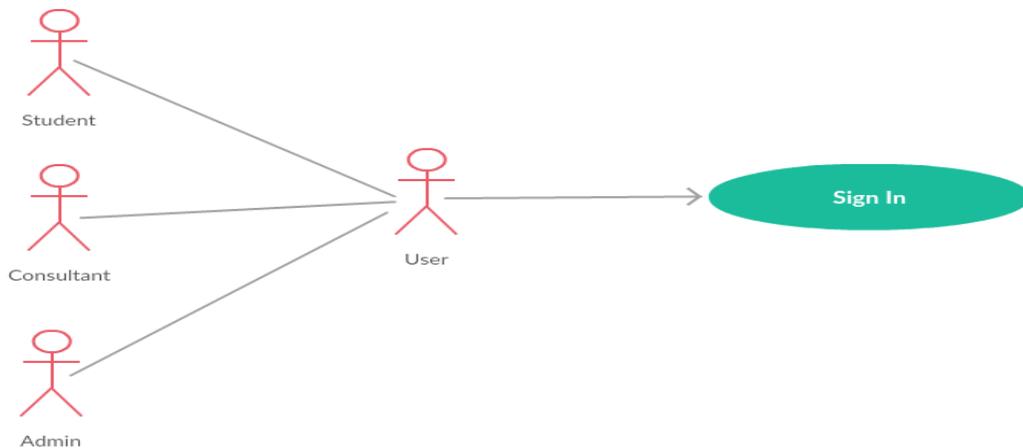
#### **Scope**

The scope of this use case is a log in system for user

#### **Description**

This use case describes the user secure when they login to Anonymous Automated Response System

#### **Use Case Diagram**



#### **Flow Description**

#### **Precondition**

The system is in initialisation mode. User must have Google account.

#### **Activation**

This use case starts when a user wishes to log in to Anonymous Automated Response system.

#### **Main flow**

- The system connects Google
- The user enters email and password (See A1)

- The system validates the entered email and password and logs the actor into the system

### **Alternate flow**

A1 : <Invalid email or password>

- The system displays error message
- The user enters email and password again
- The use case continues at position 3 of the main flow

### **Termination**

The system stores all the login information

### **Post condition**

The system goes into a wait state. If the use case was successful, the actor is now logged into the system. If not, the system state is unchanged

### **6.3.1.3. Requirement 2 < Posting Problem >**

#### ***Description & Priority***

This function is to post the problem which the user wants to ask about or needs an advice on. It will be stored in the database.

#### ***Use Case***

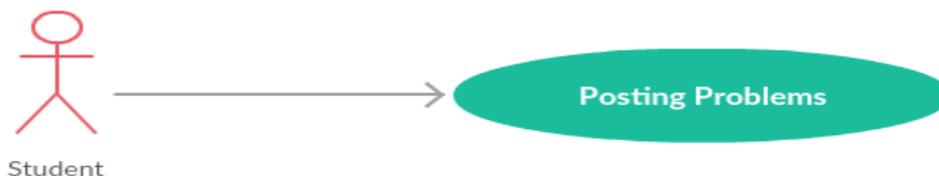
#### **Scope**

The scope of this use case is to post a problem and store it in the database

#### **Description**

This use case describes what users do when they have a problem and need help.

### **Use Case Diagram**



### **Flow Description**

### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system.

### **Activation**

This use case starts, when a user presses the button to post his/her own problem

### **Main flow**

- The system identifies the user role and displays the post page
- The user clicks on the button “post your own problem”
- The system reloads the page and opens the new post page
- The user enters problem and clicks the button “post your problem”
- The system displays the message and says that the post is created successfully

### **Termination**

The system stores all the post into database. The use case terminates when the user exits

### **Post condition**

The system goes into a wait state

### **6.3.1.4. Requirement 3 < Automated Response >**

#### ***Description & Priority***

This function is automated it displays to user some suggestions which are already given from the other web site, while user is waiting for the answer from a consultant.

#### ***Use Case***

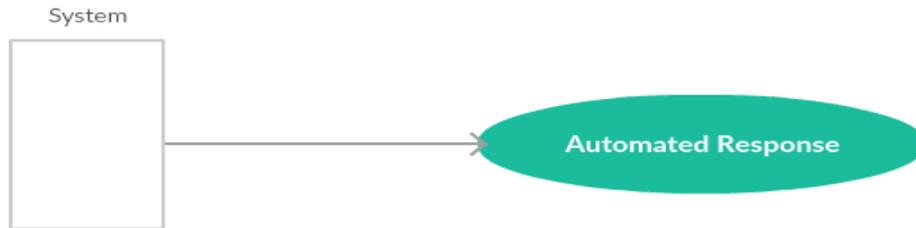
#### **Scope**

The scope of this use case is to automatically display some recommendations which are related to user’s problem.

### **Description**

This use case describes what the system does when users are waiting for the answer to the problem.

### Use Case Diagram



### Flow Description

#### Precondition

The system is in initialisation mode. User must login to Anonymous Automated Response system.

#### Activation

This use case starts when a user presses the “created post” button.

#### Main flow

- 1- The user posts the problem
- 2- The system makes calls to the Google Custom Search API and renders a list of relevant resources which are given from other website.
- 3- The system automated displays the web pages which are related to the post.
- 4- The users can rate for the links
- 5- The system save the rating and that links
- 6- The system views the results which are sorted and ordered and only the most relevant and useful hyperlinks are retained sort
- 7- The system gives the suggestion according to the highest average rate, highest number rate and most number of time users click on the links.

## Termination

The use case terminates when the user exits

## Post condition

The system goes into a wait state

### 6.3.1.5. Requirement 4 < Response Problem >

#### *Description & Priority*

This function is to let the consultant give an advice or an answer to the problem which the user has posted.

#### *Use Case*

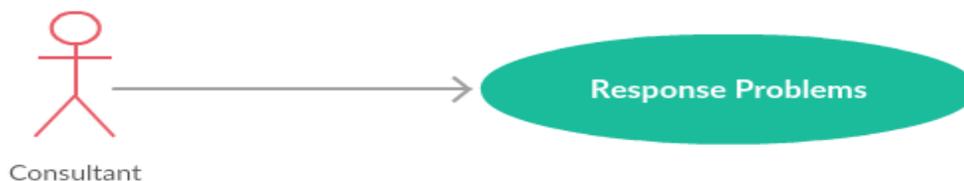
#### **Scope**

The scope of this use case is to give an answer to the problem

#### **Description**

This use case describes what the consultant does when he/she gives the answer for the problem.

#### **Use Case Diagram**



#### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system.

#### **Activation**

This use case starts when a user presses the answer button

#### **Main flow**

- The system identifies the user role and displays the post page.
- The consultant clicks on the answer button.
- The system reloads the page and displays the answer page.
- The consultant enters the answer and clicks to post the answer button.

- The system displays the message saying that the post has been created successfully
- The system sends the alert to the user, saying that user's question has been answered.

### Termination

The system stores all the post into a database. The use case terminates when the user exits

### Post condition

The system goes into a wait state

### 6.3.1.6. Requirement 5 < Booking Appointment >

#### Description & Priority

This function allows user to book an appointment with consultant.

#### Use Case

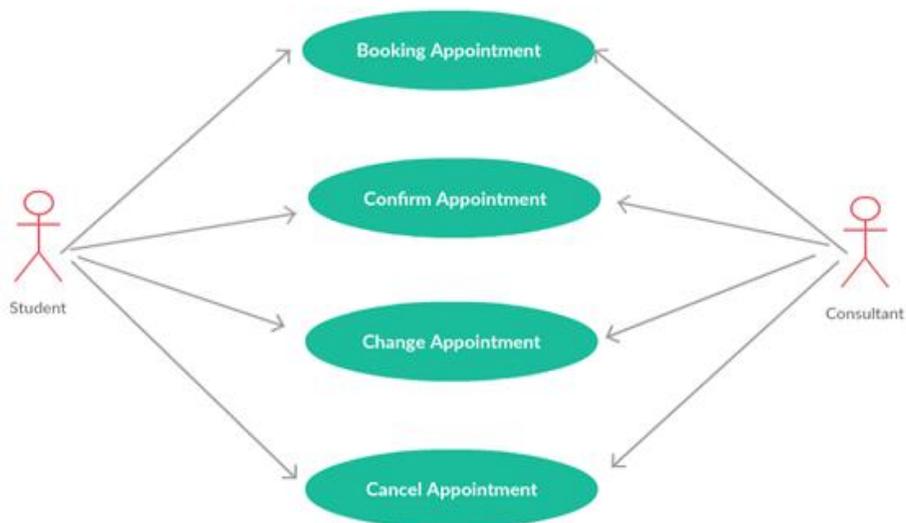
#### Scope

The scope of this use case is to book an appointment with consultant.

#### Description

This use case describes how to book an appointment with consultant for more help.

#### Use Case Diagram



## **Flow Description**

### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system.

### **Activation**

This use case starts, when a user presses the booking appointment button

### **Main flow**

- The system displays the booking window.
- The user wants to make a new appointment (See A1)
- The system displays new booking appointment window
- The user wants to cancel an existing appointment (See A2)
- The system displays cancel booking appointment window
- The user wants to change an existing appointment (See A3)
- The system displays result window

### ***Alternate flow***

A1. Make a new appointment:

- The system displays possible appointment times, dates and email of consultants
- The user chooses date and time available
- The use case continues until the user submits booking button or want to release it.

A2. Cancel an appointment:

- The system displays the old booking appointment with time and date.
- The use case continues until the user click cancel button or the user wants to release it.

A3. Change an appointment

- The System shows the change appointment window where the users can change the date and time for theirs appointment.
- The use case continues until the user click make change button or the user wants to release it.

### **Termination**

The system stores all the post into database. The use case terminates when the user exits

### **Post condition**

The system goes into a wait state

### **3.3.1.7. Requirement 6 < Chatting System >**

#### ***Description & Priority***

This function allows user to chat or talk with consultant.

#### ***Use Case***

#### **Scope**

The scope of this use case is to chat or talk with consultant for more help.

#### **Description**

This use case describes how user operate with the system.

#### **Use Case Diagram**



#### **Flow Description**

#### **Precondition**

The system is in initialisation mode. User must login to Anonymous Automated Response system and book the appointment with consultant.

#### **Activation**

This use case starts when a user presses the chatting button

### **Main flow**

- The system displays the chat window
- The users send chat messages
- The system update chat window
- The system reloads the chat window.

### **Termination**

The use case terminates when the user exits

### **Post condition**

The system goes into a wait state

### **3.3.1.8. Requirement 7 < Appointment Reminder System >**

#### ***Description & Priority***

This function is to remind users about their appointments.

#### ***Use Case***

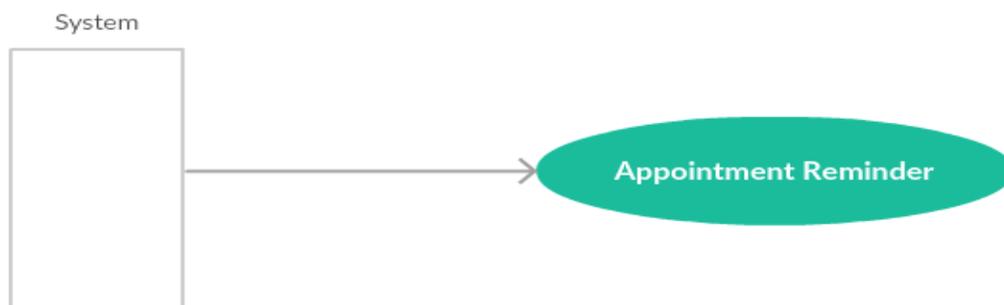
#### **Scope**

The scope of this use case is to remind users about their appointments. Appointment reminders allow system to automate the process of reaching out to users in advance of an upcoming appointment.

#### **Description**

This use case describes how the System Appointment Reminders work in the Anonymous Automated Response System.

### **Use Case Diagram**



## **Flow Description**

### **Precondition**

The system is in initialisation mode. User must book appointment in Anonymous Automated Response system

### **Activation**

The use case starts when a user submits a booking appointment.

### **Main flow**

- The system stores booking information in the database.
- The system checks the appointment list and configured time, in advance of the appointment.
- The system sends out a reminder

### **Termination**

The use case terminates when the user exits

### **Post condition**

The system goes into a wait state

## **6.3.2. Non-Functional Requirements**

### **6.3.2.1. Performance/Response time requirement**

The server itself will require a high speed connection in order to process all of the information that the users send to it in a timely manner.

The specifications of the server's hardware should include that of a powerful processor to allow for faster processing of the data received from users over the web so that the server responds to the user with as short a delay as possible.

### **6.3.2.2. Availability requirement**

Availability is a measure of how often the application is available for use. More specifically, availability is a percentage calculation based on how often the application is actually available to handle service requests when compared to the total, planned, available runtime. The formal calculation of availability includes repair time, because an application that is being repaired is not available for use.

### **6.3.2.3. *Physical environment requirements***

The system will require a server on which to operate .The server will be hosted on a free cloud service called Heroku. Heroku is a platform as a service (PaaS) that enables developers to build and run applications entirely in the cloud.

### **6.3.2.4. **Robustness requirement****

The App shall continue to work correctly even after it has taken invalid input from the user and will notify the user of any errors that may occur.

The App shall continue to work if one part of it fails or throws an error.

### **6.3.2.5. **Security requirement****

The system will require a password to access private parts of the site or server.

The system is also auto backed up to GitHub to deal with any potential data loss due to possible hardware failure.

### **6.3.2.6. **Reliability requirement****

The system should be up and running at all times.

The system should reset a user session if a crash happens.

The system's web service should have two or more connections to ensure if one is down then the others can take up the slack.

### **6.3.2.7. **Maintainability requirement****

The server used to host on cloud the service will need to have as little down time as possible and as a result, must be easily maintained.

The system should have admin access to the underlining code so that fixes or changes can be made

The system should have admin access to the database so that fixes or changes can be made

### **6.3.2.8. **Portability requirement****

The system is a web based app it will be available for systems that support modern web browsers.

### **6.3.2.9. **Resource utilization requirement****

The system needs an admin to maintain and manage the system. It also needs a back-up server, just in case the cloud server is out of service.

### 6.3.3. Interface requirements

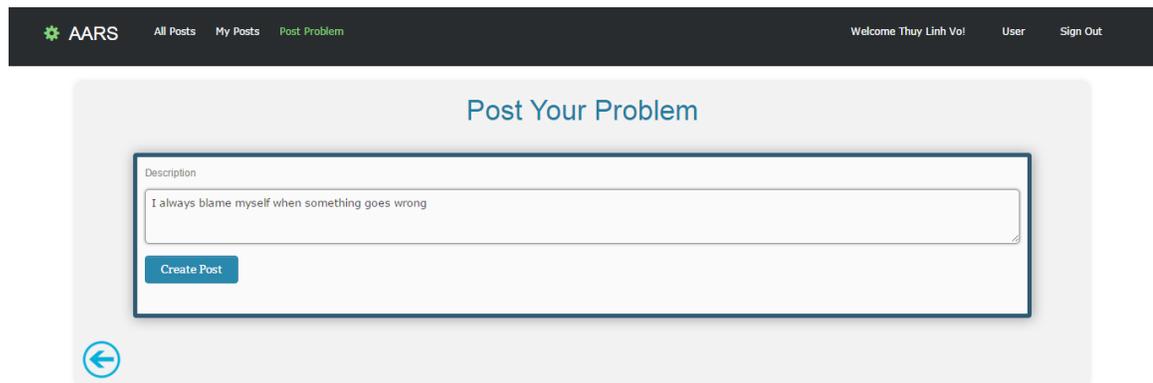
#### 6.3.3.1. GUI

The user interface will be shown in a web browser. The user will be able to access the GUI from an internet enabled device. The application will have a navigation bar at the top. All problems and answers will be available to all users without any identification. Users will be able to ask questions or post their problems by signing in with the Google account. Once users sign in, they will be able to see their own posts and their information

The system will automatically suggest possible answers to queries immediately after users post their problems or their questions. The consultant can interact with users problems by posting answer(s) or giving advice

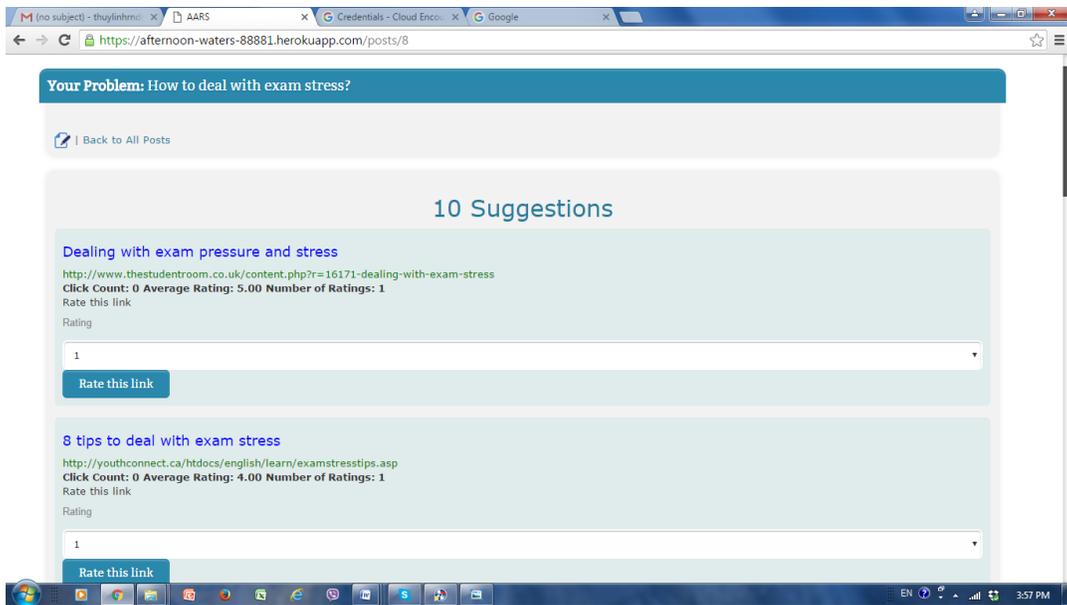
The application will have a booking appointment system if the user wishes to speak or chat personally with the counsellor or consultant. The system also automatically removes an appointment after user has seen a consultant. In addition, this application will use Gmail API for chatting and speaking functionalities.

#### Posting Problem

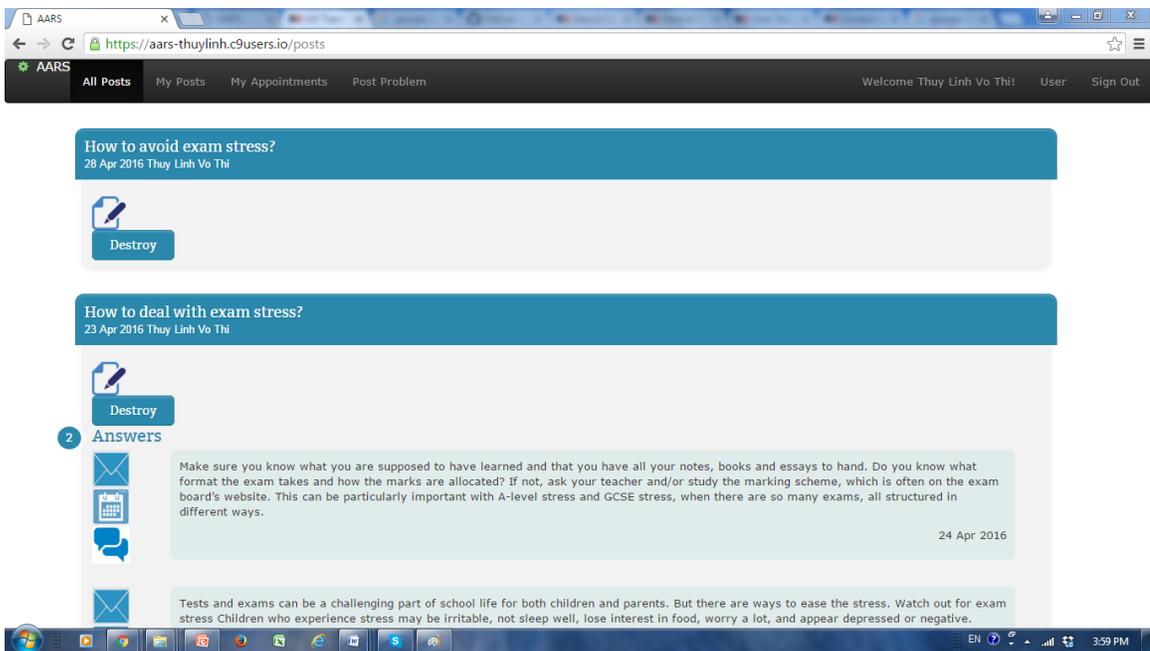


The screenshot shows the 'Post Your Problem' form in the AARS application. The form is titled 'Post Your Problem' and is located within a dark navigation bar. The navigation bar includes the AARS logo, 'All Posts', 'My Posts', and 'Post Problem' (the active page). On the right side of the navigation bar, it says 'Welcome Thuy Linh Voi', 'User', and 'Sign Out'. The form itself has a 'Description' label and a text input field containing the text 'I always blame myself when something goes wrong'. Below the input field is a blue 'Create Post' button. A blue circular arrow icon is visible in the bottom left corner of the form area.

#### Automated Response and suggestion



An appointment can be arranged if the user wishes to speak or chat personally with the counsellor or consultant.



## Booking Appointment

### New appointment

Name  
Thuy Linh Vo

Email

Time  
29 -  
January -  
2016 -  
10 -  
09 -

Create Appointment Cancel

### Calendar

These appointments have been booked already

Feb 2016

Mon	Tue	Wed	Thu	Fri	Sat	Sun
01	02 12:30	03	04	05	06	07
08	09 15:00	10	11	12	13	14
15 10:00 12:30	16	17	18	19	20	21
22	23	24	25	26	27	28
29	01	02	03	04	05	06

Consultant can interact with user's problems by posting answer(s) or giving advice

Your Problem: I always blame myself when something goes wrong

Post Answer

### Answers

Thuy Linh Vo Thi - 27 Jan 2016

There are two types of people in the world, according to Rhonda Britten. Those who blame others for their problems, and those who blame themselves. When I was suffering from depression, I spent a lot of time blaming myself for just about everything, I was a victim of my own circumstances and it certainly kept me feeling low and sad. It was hard for me to be accountable or see the big picture. If you find yourself blaming others or yourself for everything, it can be hard to let go. But with more awareness and a daily practice of self-compassion, it is possible. No matter what, blaming yourself (or others) for situations keeps you unhappy because you feel like you have no control. A healthier alternative is to go beyond the blame by looking at your life and obstacles as an opportunity to take action. When we move past blame, we are able to take responsibility and release the guilt attached to self-blame.

Thuy Linh Vo Thi - 27 Jan 2016

1. Reframe What You Should Do Many of us pressure ourselves to do things we don't want to do. "I should workout," "I should call them back," "I should do more in my day." We end up shoulding all over ourselves. Instead of saying "should," start saying "could." "I could" is more empowering, freeing and expansive. It gives you permission to feel more joy in the moment. 2. Look at the Big Picture Every situation we experience is part of a bigger plan. When you can look at setbacks and opportunities for growth, life becomes easier and there is less pressure. Look at the blessing in each lesson. Instead of blaming yourself for a situation, look for the silver lining. Ask yourself: what could this situation teach me? 3. Trust Yourself Many folks blame themselves after the fact, most often because of regret or denial. This happens because we don't trust ourselves. We make choices for the wrong reasons. But when you learn to trust yourself, you will have more confidence. This eliminates self-doubt. You will be aligned with your true self and innermost desires. That naturally creates a compassionate experience and helps you remove self-blame.

## List of client appointment

### Listing client appointments

Name	Email	Day	Time
Thuy Linh Vo	inialosul...	22 Jan 16	12:00
Thuy Linh Vo	inialosul...	22 Jan 16	16:11

### Monthly Calendar

Jan 2016

Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	31	01	02	03

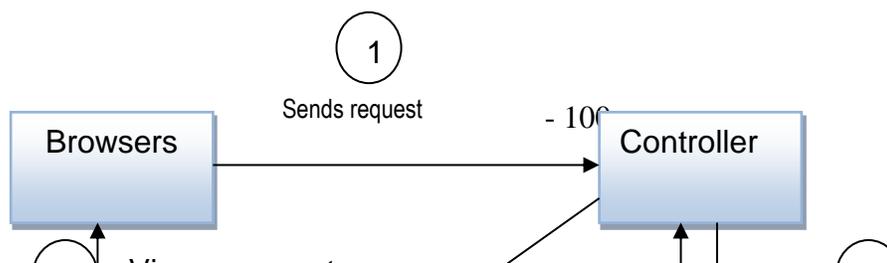
### 6.3.3.2. *Application Programming Interfaces (API)*

- API will be created
- The application will use Google login API, simple Calender API for the booking system, Google Custom search API for automated response and suggestion.
- The system's web service will be based on ruby on rails

### 6.3.4. System Architecture

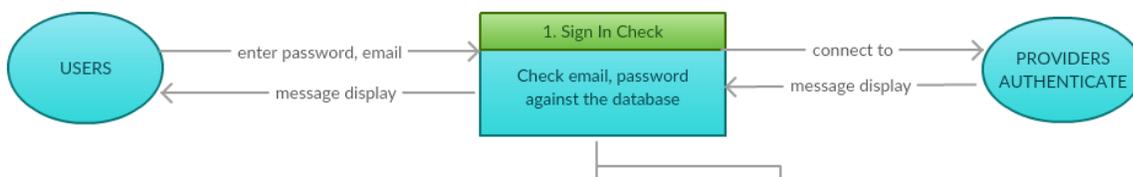
Use a class diagram to outline the structure of the system. Explain briefly why you have chosen this architecture. You might want to use Visio or Rational Rose to create these.

The application is implemented using the Model-View-Controller (MVC) architecture.



When interacting with a Rails application, a browser sends a request, which is received by a web server and passed on to a Rails controller, the controller interacts with a model, which is a Ruby object that represents an element of the site and is in charge of communicating with the database. Controller Invokes view and View renders to browser screen.

### 6.3.5. Process Flow Diagram

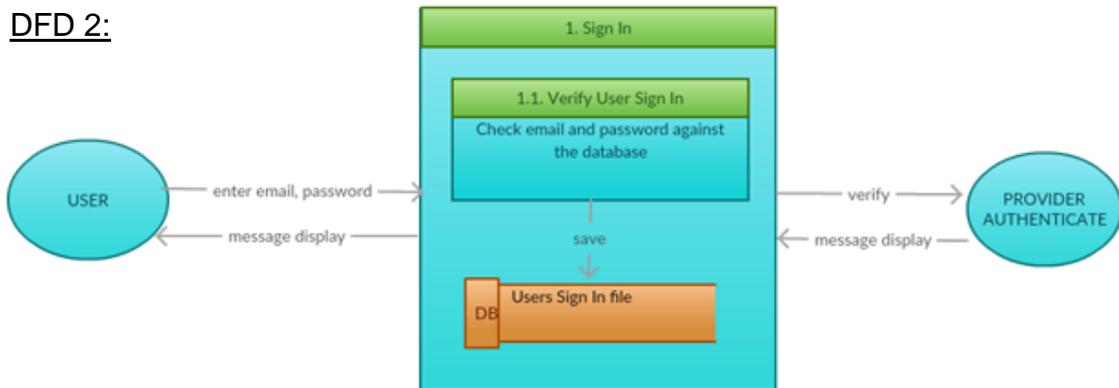


***Use-Case: sign in***

**DFD 1:**



DFD 2:

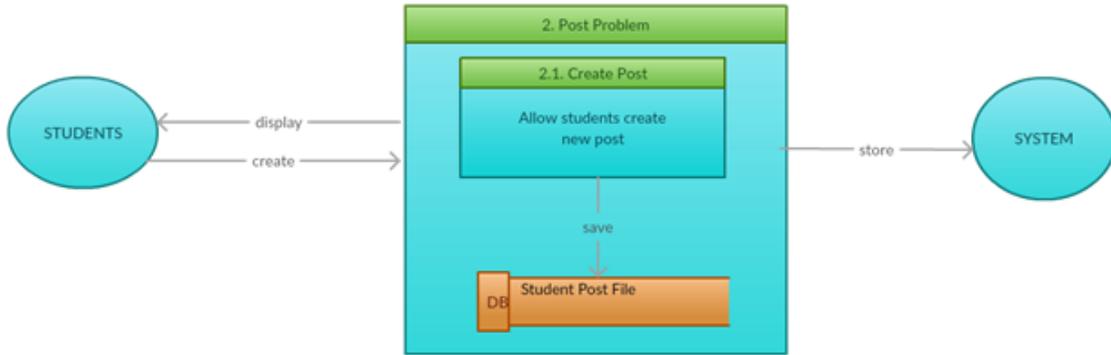


***Use-Case: Posting Problem***

DFD 1:

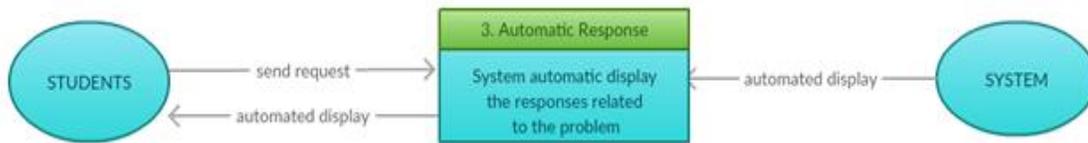


DFD 2:

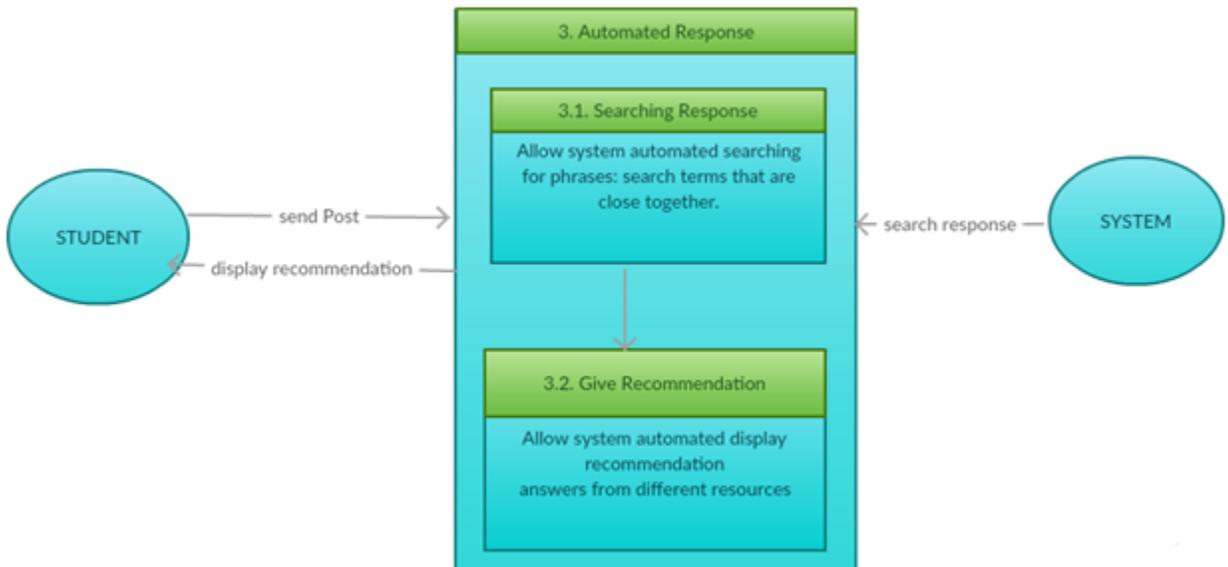


**Use-Case: Automated Response**

DFD 1:



DFD 2:

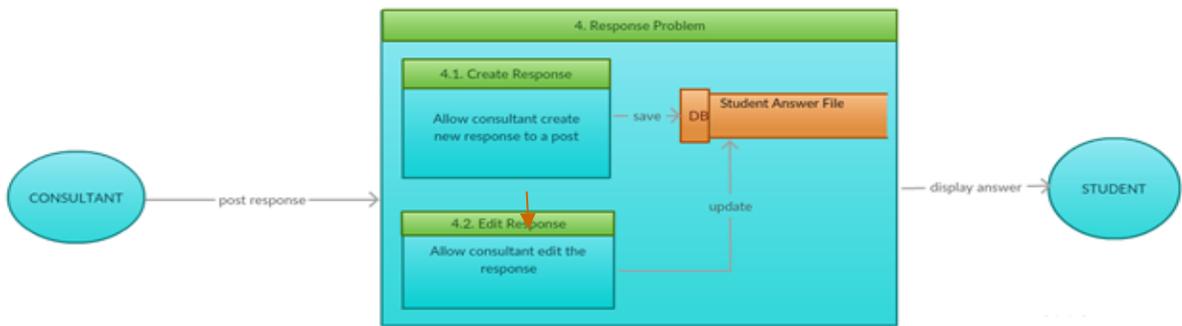


**Use-Case: Response Problem**

DFD 1:



DFD 2:

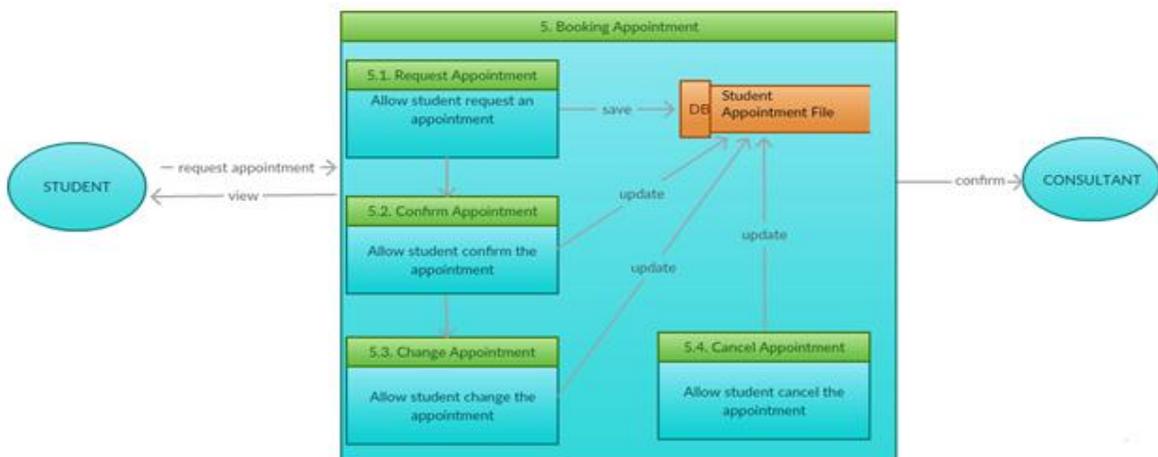


***Use-Case: Booking Appointment***

DFD 1:



DFD 2:

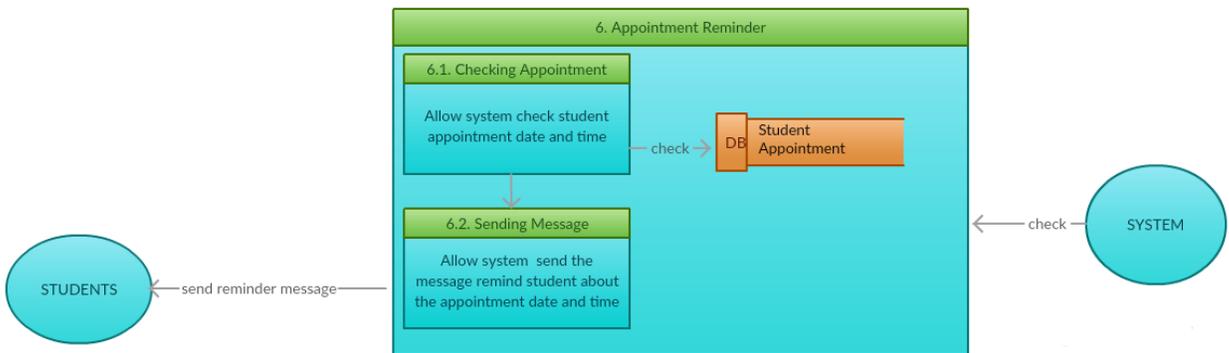


**Use-Case: Appointment Reminder**

**DFD 1:**



**DFD 2:**

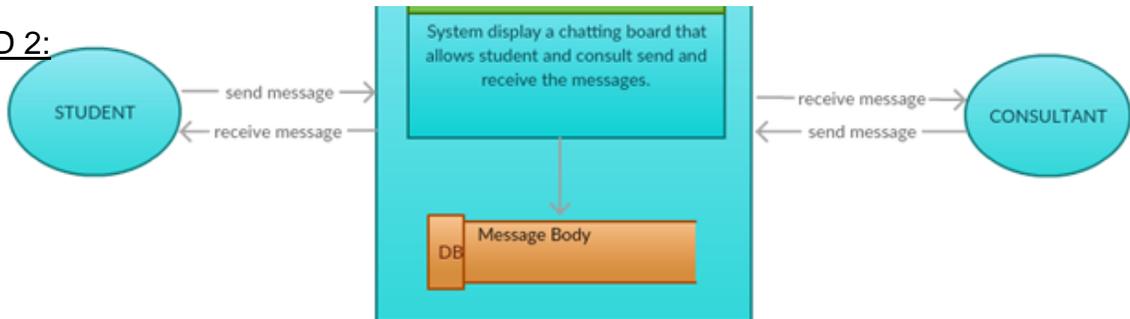


**Use-Case: chatting**

**DFD 1:**



**DFD 2:**



### **6.3.6. System Evolution**

Additionally a mobile and/or tablet friendly version of the application could be developed in the future to adapt to the currently expanding mobile device market. It can be run with android and ios.

## **6.4. Monthly Journals**

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme : BSc in Computing

Month: September

#### **My Achievements**

For the month of September I had to set up my environment for my application. I decided to use PHP for my application and have XAMPP installed. I learned step by step how to use PHP.

I researched information for my project proposal project.

I came up with my project idea and got the Project Proposal completed and handed in to Moodle

#### **My Reflection**

I felt that my initial idea was workable and I was happy to complete the proposal project early.

However, I found, it difficult as I didn't know whether or not my idea would make the project possible. I was to discover that I couldn't find the resources to support

the application so I have to change to an alternative. I was distressed and spent a lot of time trying to research other possible resources which might suit my project. I knew that I had to do my project based on a cloud application, because I am in the Cloud Stream. This highly influenced my decision about what project I wanted to do. I took a lot of time to research what I needed and what resource would support my project.

Selecting a new direction for my project using PHP with which I am not familiar meant that I would have to spend time to study PHP in relation to the project.

### **Intended Changes**

Once my Project Supervisor has been assigned. I will try to see him/her on a weekly basis.

I will also try to source out a real client, document their requirements for the proposal, and based on that information, create requirement specifications.

Create front-end using Foundation

Sign up with Heroku cloud platform, APIs keys

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme : BSc in Computing

Month: October

In the month of October, I set up the environment for my application. I wanted to use PHP for my application but I could not find information to support it, so I decided to use Ruby on Rails for the application which I installed. I also focused on the Requirement Specification document. This included all the details of the

application. I found it very difficult because I had to relearn UML for use cases and class diagram.

### **My Achievements**

This month, I was able to finish the Requirement Specification document.

I also have learnt ruby syntax online and followed a tutorial in “*Agile Web Development with Rails*” and “*Ruby on rails tutorial*”. I started the omniauth on ruby with Google and Facebook login and had foundation installed for the front end. I also did market research about my application to find its feature.

### **My Reflection**

I felt disappointed that I had wasted time finding information about PHP and setting up PHP as the environment for my application. If I chose Ruby on Rails from the beginning I could have had more time to learn Ruby which I am unfamiliar with. I now have to learn everything from scratch, but at least it works for me. I find Ruby is very powerful language.

I felt that doing certain parts of the Requirement Specification was difficult, like Non-functional requirements, class diagram and System Evolution. I spent a lot of time doing it and researched how to do it. Once I had it done I felt I had accomplished something significant.

After meeting with the project supervisor, I asked myself what make my application more advantages than the present one or is any other application the same as mine? For this reason, I researched and compared with the other. I also asked people from different backgrounds, what they thought about my idea and what they expected, if they used this application. I discovered that there are many strong points in my application but there are also many challenges to meet the users’ requirements.

### **Intended Changes**

Next month, I will try to have the SignIn page for Anonymous Automated Response system and push it into Heroku cloud platform.

### **Supervisor Meetings**

Date of Meeting: every week

Items discussed: Features of the application

Action Items: Research and compare my application with other application.

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme : BSc in Computing

Month: November

During the month of November, I focused on the Analysis Design document. This included all activities, which help the transformation of requirement specification into implementation. This is the intermediate stage, which helps human-readable requirements to be transformed into actual code.

### **My Achievements**

This month, I was able to finish the Analysis Design document. I have also started programming the prototype with post and response system and then deployed it into Heroku platform.

I have gained the knowledge of ruby syntax and HTML/CSS. I have learnt how to build an application on ruby on rails and understand how to use Model, View and Controller in Rails.

### **My Reflection**

I felt that doing certain parts of the Analysis Design document were difficult, such as Logical View, Software Architecture, Communication Architecture, System Design, ... I have no idea what they are. It took me a lot of time to review Software Engineering subject which I did in second year. I spent lots of time doing it and finding the way to apply it in my application. Once I had it done I felt like I had accomplished something significant.

I feel I have finally got going on my project and from now on will be heavily involved in my application. Having finished the use cases and the preparation of the application, I feel confident that I will develop this application with no problems.

The meeting with my supervisor was very helpful. We met as a group. Each person shared about their project and the difficulties in implementing it. The supervisor and members of the group gave advice and recommendations. I was appreciative of their help. Now if they discover anything that relates to my project, they draw my attention to it.

My supervisor suggested some functionalities that would make my application more professional. I am spending more time in researching and finding the best way to proceed with my project.

### **Intended Changes**

Next month, I will try to work on SignIn, Post and Response pages.

### **Supervisor Meetings**

Date of Meeting: every week

Items discussed: Prototype

Action Items: Research and implementation of the application

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme : BSc in Computing

Month: December

I need to submit a Distributed System's Project on 12<sup>th</sup> December. I spent most of my time focusing on it and I am still working on how to build the chat system and booking system for the project.

### **My Achievements**

This month, I was able to finish Distributed System's Project which has used Ruby on Rails. So I have learnt how to use Geolocation API in Ruby, how to use curl sending a request through HTTP. This led me understand what is meant by API and how it works. I also learnt how to upload pictures and file in the Ruby on Rails project by using the gem file. I have learned how to build API application using Ruby on Rails.

### **My Reflection**

I feel I have finally got going on my project, having finished the use cases, analysis design and the preparation of the application. Using new language to build the project is a big challenge. There were some errors I didn't know how to correct. I spent a lot of time searching for a way to fix problems. I felt distressed. It took me two weeks to find out the errors. I have been working with Ruby on Rails since. I feel confident that I will develop this application with no problems.

This month, I didn't have much time for the project. I am preparing for the January exam, but I will try to have prototype done for the midterm presentation.

### **Intended Changes**

Next month, I will try to work on Automated Response, Booking Appointment prototype.

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme : BSc in Computing

Month: January

For the month of January, I spent most of my time focusing on prototype and midpoint presentation documentation.

### **My Achievements**

This month, I was able to do a booking system done and simple calendar which display all appointments events to users by days, weeks and months. I worked on styling the navigation menu which tells the user where they are in the application. Finally I got automatic response worked. I was able to have a midpoint presentation documentation done.

### **My Reflection**

I feel, I am making progress at last on my project. Sometime I find stressful because I have spent too much time finding out how to use Google search API which will automatic search possible information that related to users' post after they have submitted. It took me a month to look up different online tutorials. Finally, I found a tutorial which helped me to solve the problem. I am so happy about that. I found new information in Rails while I was working on styling, the navigation menu which tells the user where they are in the application. The link text colour is green if the current page is active. Otherwise the link text colour is white. To do this, I needed to check if the user was on the current page. Dynamically, I set the class of the list tag to "active" (so that the navigation link would appear green due to foundation css style sheets).I first created a helper method to check whether the current page was the active page

`ApplicationHelper#active_tab?` Takes a path string and compares it to the current path string. It returns true or false. If this method returned true I set the class of the navigation link to active. To do this I created another helper method `ApplicationHelper#list_class`. This method takes a path string. It is called `active_tab?` with the passed parameter and returns a string. It returns "Active" if true and "" if false.

A content\_tag helper in Rails lets you wrap one tag in another and allows you to pass options about the outer tag. I used this to wrap a link tag in a list tag. I used the helper methods above to set the class on the list tag if the current page was the active page.

I also worked on appointment form and Simple Calendar which views all appointments and events of users by day view, week view or month view. The appointment automatically disappears after the time has passed.

### **Intended Changes**

Next month, I will work on Chatting and appointment reminder

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme: BSc in Computing

Month: February

After midpoint presentation, I reflected on the midpoint feedback and focus on implementation of the rating for Anonymous Automated Response System.

### **My Achievements**

This month, I focused on implementation of the rating for Anonymous Automated Response System and have showcase profile completed.

### **My Reflection**

I feel that my progress has been slow. I spent time prepare for the midpoint presentation. After the presentation, I got feedback from the teachers; reflecting on that feedback; I focused on implementation of rating. I got the rating shown in Google Custom search result. The system searches possible information that related to users' post and display relevant result with highest rate first.

I have learnt how to show the rating in Google Custom Search result. It seems very simple but I have spent a lot of time finding the way to do it. Custom Search extracts a variety of structured data for use by structured search operators, including dates,

authors, ratings and prices; this is the same data available for use in custom snippets.

PageMap: A PageMap explicitly represents structured data as DataObjects with Attributes and values, encoded as an XML block embedded in a web page. Custom Search makes all well formed PageMap data available for structured search operators; it can also be used in custom snippets.

In controller:

```
@results = GoogleCustomSearchApi.search(@post.description.to_s)
```

In html.erb page if I put `<%= @results.to_yaml %>` it will display all the information related with what I have put in the search. The code below shows how to display rating in Google search result.

```
<% if item["pagemap"] %>
  <% if item["pagemap"]["aggregaterating">%>
    <p>Aggregate Rating: <%= item["pagemap"]["aggregaterating"] %></p>
    <p>Rating: <%= item["pagemap"]["aggregaterating"][0]["ratingvalue"]
%></p>
    <p>No of reviews: <%=
item["pagemap"]["aggregaterating"][0]["ratingcount"] %></p>
  <% end %>
<% end %>
<% end %>
```

### **Intended Changes**

Next month, work on the rating links

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme: BSc in Computing

Month: March

## My Achievements

This month, I have the rating for Anonymous Automated Response System, send email reminder done and have showcase profile completed.

## My Reflection

I feel that my progress has been slow. I spent time for Distributed System assignment and project.

I worked on sort result for a search, rating for Anonymous Automated Response System and send email to remind users about their appointments.

User submits a search query using Google custom search API. System receives a result set from Google custom search API, create an array of Ruby objects called result. System uses its own custom algorithm to intelligently sort results that takes into account users' feedback and online behaviour, namely; the number of times a link or response has been rated, the average rating for a link or response and the number of times a link has been clicked.

I got the send email reminder work. When consultant clicks button send, the email will send to user. I tried to set up email send automated a night before the appointment but I unfortunately I couldn't get it.

I learnt how to use the form tag to get the feedback from user. For example below:

```
<%= form_tag update_rating_path do %>
  <%= label_tag 'rating' %>
  <%= select_tag 'rating', options_for_select(Rating::VALUES) %>
  <%= hidden_field_tag 'title', result.title.to_s %>
  <%= hidden_field_tag 'link_url', result.link_url.to_s %>
  <%= submit_tag 'Rate this link' %>
<% end %>
```

And Action View Partials which render sub templates within the current controller that depends on a single object. With partials, you can extract pieces of code from your templates to separate files and also reuse them throughout your templates.

For example result.html.erb below:

```
<div class="inner-wrap">
  <h4><%= link_to result.title, update_link_path(title: result.title.to_s, link_url:
result.link_url.to_s), {:style=>'color:blue;', :class => "css_class"} %></h4>
  <div style="color:green"><%= result.link_url %></div>
```

```

    <b>Click Count: <%= result.click_count %></b>
    <b>Average Rating: <%= sprintf('%0.2f', result.average_rating) || "No ratings
yet" %></b>
    <b>Number of Ratings: <%= result.number_of_ratings %></b>
    <p>Rate this link</p>
    <%= form_tag update_rating_path do %>
      <%= label_tag 'rating' %>
      <%= select_tag 'rating', options_for_select(Rating::VALUES) %>
      <%= hidden_field_tag 'title', result.title.to_s %>
      <%= hidden_field_tag 'link_url', result.link_url.to_s %>
      <%= submit_tag 'Rate this link' %>
    <% end %>
  </div>

```

It is reused in another place, for example result.html.erb is reused in Posts/show.html.erb

```
<%= render partial: "results/result", locals: {result: item} %>
```

The links and rating are saved in the a database and used sort algorithm to sort the result by the number of times a link or response has been rated, the average rating for a link or response and the number of times a link has been clicked.

```

require 'result'
class Data
  def self.parse(results)
    result_set = []
    results.items.each do |result|

      # item = SavedLink.find_by_link_url(result.link.to_s) ?
      SavedLink.find_by_link_url(result.link.to_s) : Result.new(result.title,
result.link.to_s)

      if SavedLink.find_by_link_url(result.link.to_s)
        link = SavedLink.find_by_link_url(result.link.to_s)
        item = Result.new link.title, link.link_url,
link.click_count, link.average_rating, link.ratings.count.to_i
        result_set << item
      else
        item = Result.new result.title.to_s, result.link.to_s
        result_set << item
      end
    end
    result_set # return the result set
  end

  def self.sort(results)

```

```
        results.sort_by! {|object| [object.average_rating,  
object.number_of_ratings, object.click_count] }.reverse  
    end  
end
```

By using Action Mailer in rails allows me to send emails from my application using mailer classes and views. Mailers inherit from ActionMailer::Base and live in app/mailers, and they have associated views that appear in app/views.

By default rails tries to send emails via SMTP. It will provide SMTP configuration in environment settings /config/environments/production.rb. Before proceed it needs to save sensitive information such as username and password as environment variables. It will do so by using the gem figaro.

### **Intended Changes**

Next month, I will work on the chatting system and testing the system

### **Reflective Journal**

---

Student name: x11113065 - Thuy Linh Vo Thi

Programme: BSc in Computing

Month: April

### **My Achievements**

This month, I prepared for my exam, worked on chat room, have testing done and have poster designed.

### **My Reflection**

I feel that my progress has been slow. I spent time for my exam and Cloud project. After exam, I worked on a chat system for Anonymous Automated Response System. I found that the first beta of Rails 5 was released recently. The biggest new feature is Action Cable, which provides support for implementing WebSockets with a pair of libraries for JavaScript (for the client) and Ruby (for the server). WebSockets are a convenient way to stream data between the client and server,

making it easy to build apps that require real-time message passing. A chat room is the usual example of such an app: without anyone having to refresh the page, a message sent from one user to appear for all other connected users. In the past, implementation this by having each client poll the server for new messages. WebSockets lets me replace polling with two-way channels that stream messages to where they're needed, as soon as they're created, avoiding the overhead and latency of continuous polling over HTTP.

In order to build chat room in rails 5, I have to transfer my system which is used rails 4 into rails 5. After that I have to check all functions which are working in rails to make sure that they are working in rails 5.

I used online tutorial learn how to build chat room. I learnt that chat room in the system is built on ActionCable. First I added a channel which can use to communicate via websockets between the client and the server.

Rails has already generated some client side code for us. Let's start by handling the event when enter is pressed in the chat input field. Add this at the end of `app/assets/javascripts/channels/room.coffee`

Active Job allows Rails application to work with common queries in a single interface. The job class is where the code that will be executed by queue. There is a perform method which is called and sent whatever parameters were sent when the job was first enqueued.

I also worked on testing system, made testing plan for each function, preparing questions for user testing and correct the errors or change the layout according to users' suggestions. I learnt how to use unit testing in rails. By default, every Rails application has three environments: development, test, and production. The database for each one of them is configured in `config/database.yml`.

Rails creates a test folder for us as soon as we create a Rails project using rails new Anonymous Automated Response System (AARS).

Rails will generate a default test for any Models or Scaffolds AARS generate, here is an example: The default test stub in `test/models/post_test.rb` looks like this:

```
require 'test_helper'
class PostTest < ActiveSupport::TestCase
  # test "the truth" do
  #   assert true
  # end
end
```

Running a test is as simple as invoking the file containing the test cases through rake test command.

```
$ rake test test/models/post_test.rb
```

```
.
Finished tests in 0.009262s, 107.9680 tests/s, 107.9680 assertions/s.
```

```
1 tests, 1 assertions, 0 failures, 0 errors, 0 skips
```

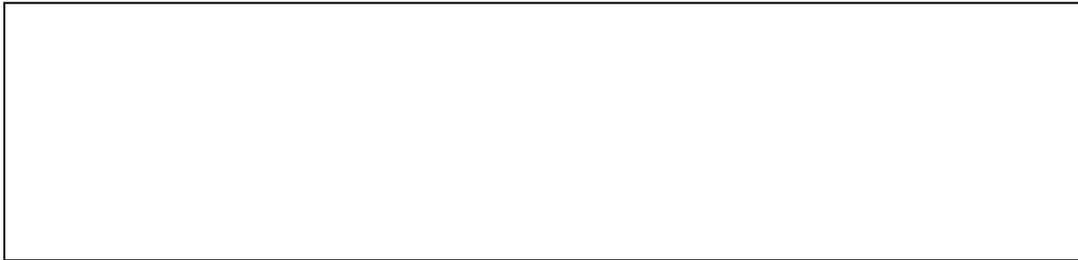
I worked on the report document and have it finished by the end of this month. I am really grateful that I have all the functions of the project working. There were a problem with chat and email when the application deploy to Heroku because these functions are supported by adds on which require verification account with credit card. Has account verify and fix the errors in Heroku which allow to send email and chat.

### **6.5. Survey**

Thank you very much for taking part in the testing of the AARS. Could you kindly take a few minutes to answer the questions below, please?

1. What do you think the purpose of this system is?

2. Did you find what you were looking for?



3. How did you find the design of the system?



4. How are the icons used in the system?



5. If you could only change one thing about this page, what would you change? Why?



6. What do you think of the colour used in the system?