

Poverty Management App and Hospital Text Platform

Undergraduate Honor Research Distinction Thesis

Presented in Partial Fulfillment of the Requirement for the
Degree of Bachelor of Science with Honor Research
Distinction at The Ohio State University

By

Shuai Wang

Department of Electrical and Computer Engineering

The Ohio State University

2015

Advisor: Prof. Kevin Passino

Copyright by

Shuai Wang

2015

Abstract

There are two parts of this honors research: designing an Android App for managing daily finances to avoid poverty, and a text platform for patients who have left a hospital after being admitted for a suicide attempt.

The Android App is designed especially for poor people to help them manage daily income and living expenses. This app will contain a cover page, information input page, income and suggested spending plots, a suggestion page, as well as a user manual. Users have to input daily income, and desired wealth and daily expenses, then they will be given a suggestion page that contains the suggested living expenses which is calculated by a proportional-integral-derivative controller (PID controller). Furthermore, all information will be stored and the user can choose to plot these data for analysis. The integral and derivative parts of the PID controller will be treated as discrete functions and specified by previous inputs of the user. This app will be implemented by Phonegap, which is a mobile development framework, using web application technologies like HTML, CSS, JavaScript as well as JQuery. The data will be stored using LocalStorage, which is given by the PhoneGap API. The PID controller is implemented by JavaScript with local stored data and plot functions executed by jqPlot. The text platform for hospitals is designed for reminding patients about their appointments as well as psychological suggestions for patients to help prevent suicide. This platform will contain a log in system, which provides limited access to protect the security of the patients' information. The user (health care professional) can input specific messages and choose the patient to send the information to. This platform is a web application and implemented by HTML, CSS, JavaScript and Ruby on Rails. The function of sending SMS to cell phones is implemented by Rails mailer.

Acknowledgement

I would like to thank Prof. Passino for his patience and help in guiding me in a good direction and teaching me the knowledge that I am unfamiliar with. It is Prof. Passino who gave me the opportunity to join his research and some advice for career development. At first, he introduced a general idea of the research and specifies the details later when I got the knowledge, which did make me feel comfortable to start. He treated me as his graduate students and gave me enough space to finish the research based on my pace. The most meaningful things that I learned from the research are not only the ability for implementing mobile and web applications but also the idea that engineers should create technologies to help people.

In addition, I want to thank Chuning Luo and Mary Scherer for their useful and valuable suggestions and help when I got stuck especially in the process of building web applications. Sometimes it was just one piece of their suggestions that really counts. We set up meetings when I finished part of the project and they helped me test and came up with ideas to optimize my work. It was Chuning who found out the potential crash of the database of the Text Platform web application.

Vita

EDUCATION

The Ohio State University, Columbus Ohio August 2012-May 2015

B.S. Electrical and Computer Engineering (Honor Program)

B.S. Applied Mathematics (Physics Track)

Jiangsu University of Science and Technology, China September 2010-July 2012

Major: Electrical Engineering (Undergraduate Study)

Field of Study: Computer Engineering

Contents

Abstract	3
Acknowledgement	4
Vita	5
Chapter 1: Introduction to Poverty Management App	7
1.1 Introduction	7
1.2 Introduction to Design Process	8
Chapter 2: Poverty Management Model and App	9
2.1 PID Controller and Poverty Management Model.....	9
2.2 Simulation of Poverty Management Model in MATLAB	10
2.3 PhoneGap and Mobile Application Technologies	13
2.4 Poverty Management App Prototype	16
Chapter 3: Summary	19
3.1 Conclusion of PID Controller Model and App	19
3.2 Future Work of the Project.....	19
Chapter 4: Introduction to Hospital Text Platform	21
4.1 Introduction to Text Platform.....	21
4.2 Introduction to Rails.....	21
Chapter 5: Web Application Implementation	22
5.1 Introduction to Text Platform.....	22
5.2 Model: User & Patients	23
5.3 User Interface	25
5.4 Controller: Users, Patients, Static Pages & Sessions	31
5.5 Text Message	34
Chapter 6: Conclusion	37
6.1 Overall Conclusion	37
6.2 Future Work and Application of the Project	37
Appendix I: Project Schedule	38
Reference Page	39

Chapter 1: Introduction to Poverty Management App

1.1 Introduction

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable. The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. This feedback control theory can be used to build a financial management model then implemented by mobile application. With the development of mobile applications, the cell phone is more powerful and useful. Nowadays, there are various kinds of financial and money management Apps, for example Mint and Spendee. However, most of them provide a service like a personal accountant with beautiful user interface and charts. A more intelligent App which can have a scientific suggestion for daily expenses based on the user's own situation is necessary. Based on the PID controller, the Poverty Management App provides professional advice as well as plot functions, which complements the current shortcomings of current financial mobile applications. The mobile app provides functionality and personal evaluation as well as some popular microfinance institutions (e.g., Grameen Bank and SKS Microfinance) with more convenient services. According to the World Development Report 2015 by the World Bank, people make decisions based on three principles: thinking automatically, thinking socially, and thinking with mental models. Most people cannot create new models or methods to improve their lives but directly use mental models from their societies like social media and applications. This poverty mobile app provides people a complete model that users can fully rely on and this algorithm is compatible with any other electronic devices like laptops and portable to web applications as well. Moreover, this model can also be implemented on any type of cell phones, which enlarges the availability to more than 6 billion people.

1.2 Introduction to Design Process and IEEE Code of Ethics

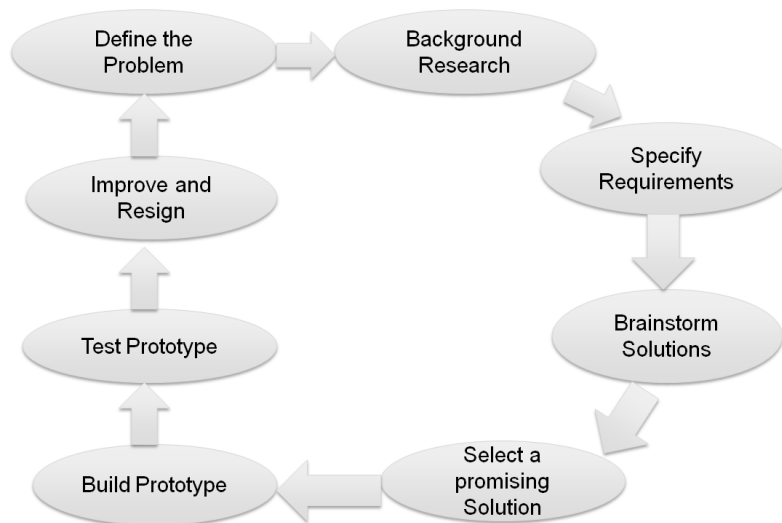


Figure 1.2a: Design Process Diagram

The development of the application follows a specific design process. Firstly, identify the need and problem. Then finish the background research to test the current solutions for the problem. Before implementing the technical part, specifying the requirements of the application is necessary. After the exploration of possible solutions, determine which solution best meets the original requirements. The next huge step is building a prototype and model the selected solutions. Given a prototype, a comprehensive test should be done and the last step is the optimization of current design and searching the future directions.

This application is built to help people manage their wealth, avoid poverty, and take responsibilities to the welfare of public. It provides a completely free service and well designed for charitable organizations and humanitarian engineering. It basically follows the IEEE code of ethics that *"To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment"* [7].

Chapter 2: Poverty Management Model

In this chapter, PID controller theory and web application technologies that were used to build the Android mobile application will be introduced. In addition, the result of simulation of PID controller model and the algorithm will be explained as well.

2.1 PID Controller and Poverty Management Model

The PID control scheme contains three terms, proportional, integral and derivative which are summed to calculate the output signal $y(t)$ by some manipulations. The result of these three terms are called manipulated variables and the final form of the PID algorithm is:

$$y(t) = K_p * e(t) + K_i * \int_0^t e(\tau) d\tau + K_d * \frac{d}{d(t)} e(t)$$

where $u(t)$ is the input function and $e(t)$ is the error, which is the subtraction of $u(t)$ and $y(t)$. K_p is proportional gain, K_i is integral gain and K_d is the derivative gain. All three gains are just constants; however, each of them controls different parts and collaborate with each other to make the error signal smaller and smaller. The proportional gain leads to a change of output based on the error signal. A too high proportional gain will make the system unstable; on the contrary, too low proportional gain will lead to insufficient response to the system disturbances. Therefore, in order to solve the problem, steady state error, which is made by proportional term, the integral term is used to take previous error into consideration. So the change of output will trend to more stable. However, since the integral term responds to the previous accumulated error, it may overshoot the setpoint value. Therefore, we need the derivative term to predict the system behavior and improve the stability of the system.

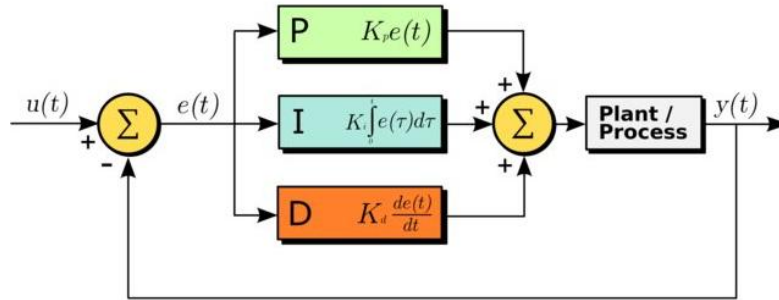


Figure 2.1a: PID Controller Scheme [16]

Based on the scheme explained above, a poverty management model is made. In this model, the input signal is desired wealth and the output signal is current savings. Error is the subtraction of desired wealth and current wealth, which will go into PID controller section and output a daily spending advice. After the consideration of daily income and daily spending, which is the suggested value made by PID controller; we can get a new current wealth.

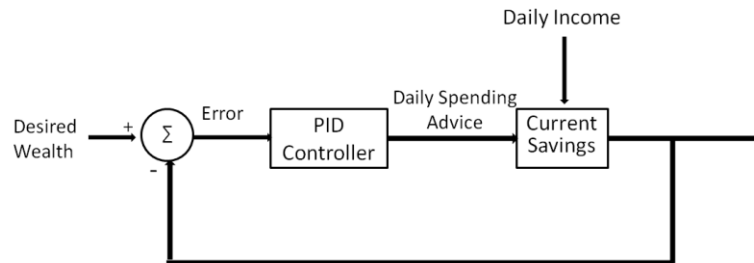


Figure 2.1: Poverty Management Scheme [13]

Theoretically, the signal in PID controller is continuous; however, in daily life we can only collect daily figures, which are discrete. Therefore, in order to implement the algorithm in the poverty management model, the proportional term is just the subtraction of desired wealth and current wealth. The integral term is the summation of all previous errors and the derivative term is the changing error divided by changing time, which is usually one day.

2.2 Simulation of Poverty Management Model in MATLAB

In order to check whether the model can help users accumulate money and speed up the process of achieving the desired wealth, some simulations are made in MATLAB. The first simulation is

the current wealth, which uses the PID spending strategy. A random daily income vector with value between 10 to 20 dollars was used to represent the daily income and the three gains of PID controller were chosen as $P=-0.316$, $I=-0.007$, $D=-0.547$ after manual adjustments. An important assumption is made that users will strictly follow the suggestion of poverty management model.

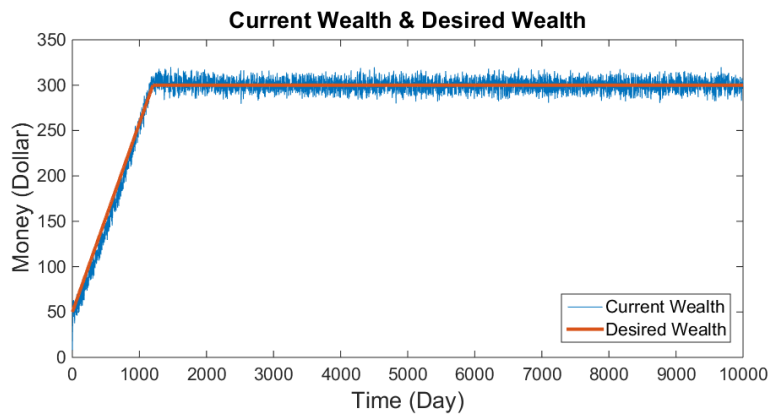


Figure 2.2a: Current Wealth vs Desired Wealth Graph

In the beginning 1000 days, the desired wealth is preset to grow lineally because people's desired wealth increases as their current wealth and they do not satisfy their current situation. After 1000 days, we set desired wealth to be stable, which is used to test whether the model can maintain the money in a high level. The result is obvious that current wealth always goes with desired wealth and people can achieve their goals finally. After 1000 days, the current wealth will oscillate around 300 dollars and keep stable. In addition, the speed of achieving desired wealth is not only affected by PID controller but some other parameters like minimum daily spending.

The minimum daily spending suggestion in this case is preset to be 10 dollars, which is used to simulate the minimum value that users have to spend every day. The PID controller just wants to speed up the process to achieve the desired wealth as fast as possible; therefore, chances are that its output is a negative value, which should be replaced by 10 dollars. In the daily spending advice histogram, most values are concentrated around 15 dollars, which is stable and reasonable.

Although the input is a random value between 10 to 20 dollars, the daily spending can be or even exceed 20 dollars sometimes to keep the current wealth going with the desired one. On the other hand, the change of minimum daily spending can significantly change the whole process. In reality, this value may be different everyday based on user's own situation.

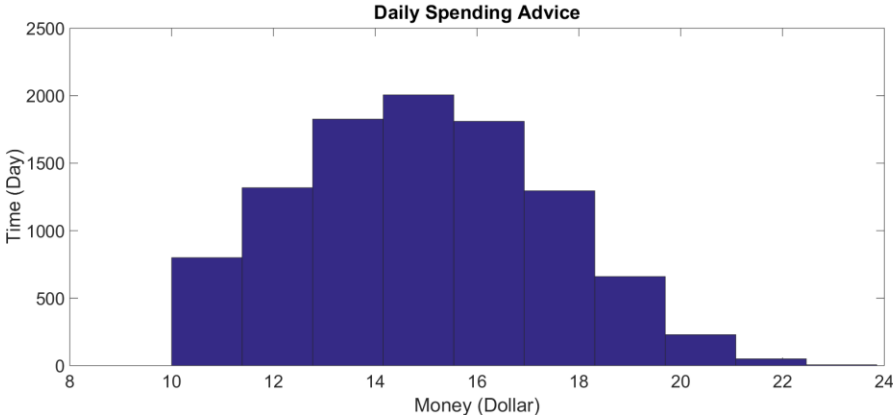


Figure 2.2b: Daily Spending Advice

To make the contrast, another simulation is made for random daily spending which means there is no such management model and users can spend as much as they want. Similarly, a random daily expenses vector with value between 10 to 20 dollars is used to represent the random daily spending. The daily income is the same with previous simulation. The result is quite random that it can accumulate some wealth to 100 or even 300 dollars, while it can also lead to big loss or negative wealth, which people should definitely avoid in reality.

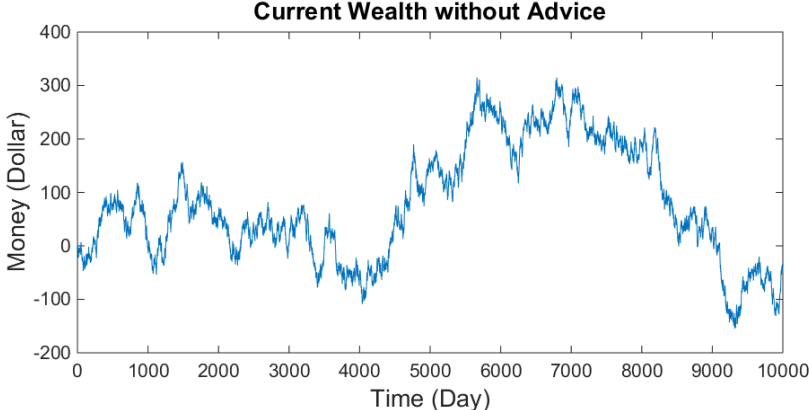


Figure 2.2c: Current Wealth without Advices

2.3 PhoneGap and Mobile Application Technologies

There are several ways, programming languages, and frameworks to develop a mobile application. Usually, mobile apps are developed by native tools for example like Java and Objective-C because these programming languages offers more flexible implementation and better performance, however, these approaches are much more expensive and can only been used in a single specific platform. Changing platform leads to a complete rewrite of the whole project. Therefore, in order to implement the poverty management model into an Android mobile application, we use a mobile development framework called PhoneGap, which can build apps by open web standards to cut down on development time by using web development skills. In addition, due to its cross-platform feature, the mobile application can be installed into almost all platforms like IOS, Android, and Windows Phone.



Figure 2.3a: Current Wealth without Advices

Almost all current mobile platforms support web applications, which can be implemented by HTML, CSS and JavaScript. Although these client-side scripts have limited access to the capabilities of the browser, PhoneGap will provide a native shell that solves this problem and make the mobile app have the access to deep level of the native devices like the physical file system by various application programming interfaces (API). The native shell provided by PhoneGap is actually different for each platform, while its core device functionality is consistent, which makes cross-platform development accessible.

Hypertext Markup Language (HTML) is a markup language for describing web documents and pages. HTML documents are described by tags, which are enclosed by angle brackets and it separates “content” (information of the page) from “presentation” which is described by JavaScript and CSS. HTML builds the skeleton of pages like the page title, header, footer and the information in body.

```
<HTML>  
  
<html>  
<title>HTML</title>  
<body>  
This is HTML!  
</body>  
</html>
```

Figure 2.3b: Sample HTML Skeleton [6]

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. It handles the presentation of the mobile app pages like the font of words, the color of the background and the theme of whole pages. Most commonly, CSS is combined with the markup languages HTML and JavaScript build a web application.



Figure 2.3c: Elements Described by CSS [14]

JavaScript is a dynamic programming language that is used in web and mobile application. The client-side script made by JavaScript can interact with the user, alter the information that needs to be displayed and communicate with server-side database asynchronously. JavaScript describes the function of the pages, for instance, grabbing the content that the user inputs and implementing the calculation inside of the PID controller.

```
String.prototype.trim =  
function ()  
{  
  return this  
    .replace (/^\s+/, "")  
    .replace (\s+$/, "");  
}
```




Figure 2.3d: Sample JavaScript Skeleton [3]

Another language that is used in the app is jQuery, which is a cross-platform JavaScript library. In addition, there is a plotting and charting plugin of the jQuery JavaScript framework called jqPlot, which can produce line, bar, and pie charts with many features. The graph of daily income, current wealth, and suggested daily spending can be plot by jqPlot.

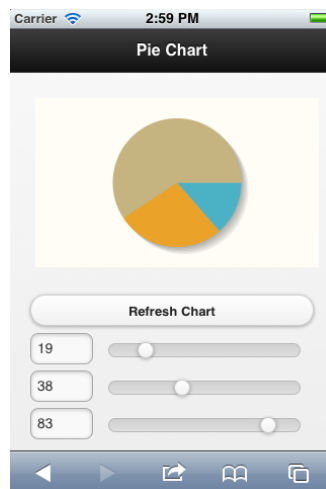


Figure 2.3e: Sample jqPlot Interface [10]

2.4 Poverty Management App Prototype

In order to help more people to manage their wealth using the PID controller model, a poverty management app prototype has been made with the basic functions like a personal accountant. Different from the most common financial management mobile applications, this app will put more emphasis on the implementation of PID controller and give daily spending advice to users rather than beautiful user interface. When users start the app, there will be a cover page, which contains a logo and two buttons. The first one is the user's manual that basically introduces the whole app and some instructions, and the second choice is starting managing the money. Both of them will link to a specific page, which is implemented by HTML link tag.

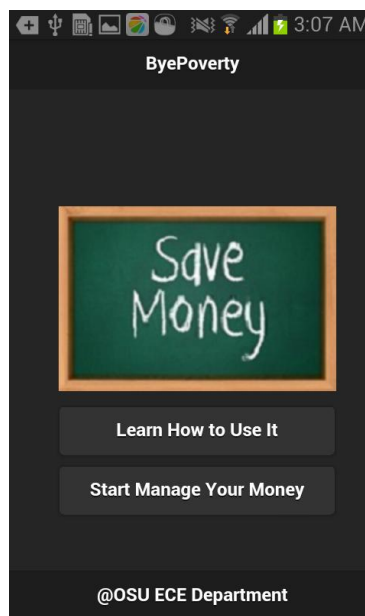


Figure 2.a: Cover page

The main page of this app will be the information input page that contains a form for user's daily information. The daily income, daily minimum spending and date will be stored in the corresponding vectors in background database used in PID controller. The daily expense is not used in PID controller, but it is necessary for the system to detect whether users follow the suggestion and update their current wealth. After users click the submit button, a suggestion page

will pop out and give the daily spending advice to user and users can also check their advice in the suggestion page. The footer of this page contains several links to the information pages of previous income, wealth and suggested daily spending. The information that users input will be grabbed after the click of the submit button by JavaScript. In the background system, all information will be stored by local (session) storage which is much simpler than SQL Lite database and usually used to keep small amounts of data between application launches. JavaScript will follow and implement the algorithm discussed in Chapter 2.1 then show the suggested daily spending to users. In addition, since the values of the three gains of PID controller are small, a fast JavaScript library for arbitrary-precision decimal arithmetic is used to compute the result of PID controller.

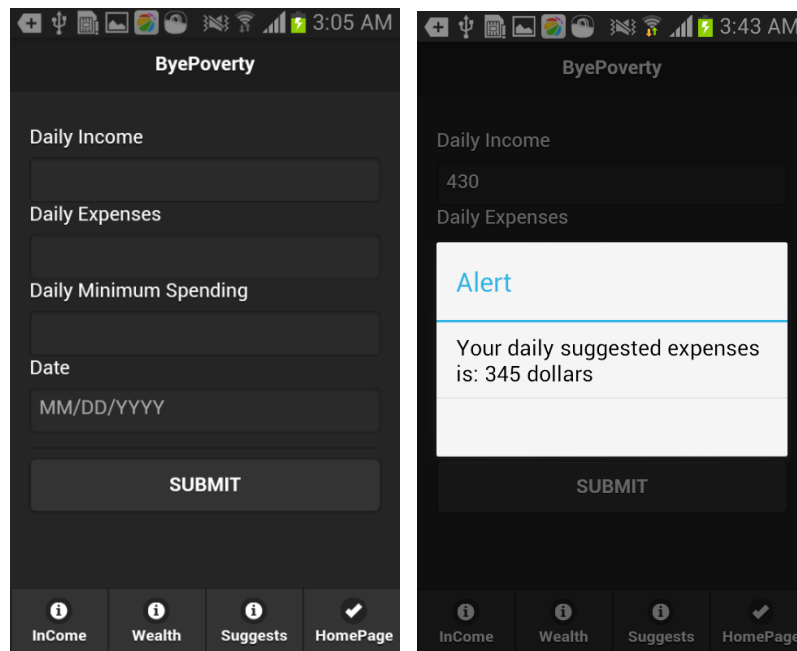


Figure 2.4b: Information input page Figure 2.c: Daily Spending Advice

In order to keep track of the user's spending behavior; their daily income, wealth and previous daily spending advice are all stored in background database and can be plotted by jqPlot. In each section, after the click of "Start Plot" button, a line chart will show up.



Figure 2.4d: Plot of Current Wealth

Chapter 3: Summary

3.1 Conclusion of PID Controller Model and App

Based on the input of the amount of daily income, daily expenses, desired wealth and minimum acceptable daily spending, the PID controller can give user an appropriate way to spend. The PID controller model can redistribute the wealth over time; it intelligently moves incomes from good to bad days in order to meet the user's desired wealth. It can predict by its derivative part when the user needs to save in times of a consistent deficit and when to spend in order to satisfy basic living requirements. Based on this model, an Android mobile app has been made, which can calculate the daily spending value automatically.

3.2 Future Work of the Project

There are several parts in the poverty management model that need to be optimized. The first one is the maximum daily spending for the result of the PID controller. As mentioned above, in the simulation part we directly use 10 dollars for minimum daily spending but there is no restriction to the maximum daily spending. For example, there are some times the controller will give a large daily spending much more than daily income, which contradicts with the principle that we want a stable process.

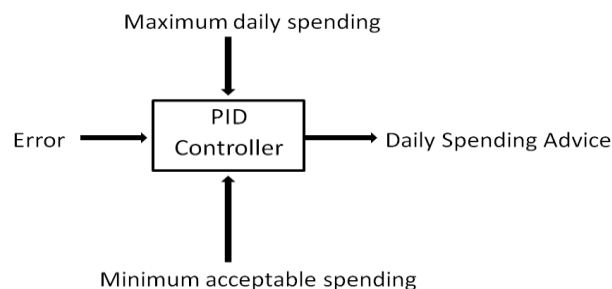


Figure 3.2a: Optimization of PID Controller Model

Then, the second part that needs to be optimized is the daily income and spending pattern. In the simulation part, we use a random vector with value 10 to 20 dollars to represent the daily income, however, usually people have relative stable income instead of a random one. Furthermore, we should consider more income patterns like financial investments besides the salary. On the other hand, we cannot assume the user can strictly follow the daily spending advice. In reality, they may have to pay bills at the same day every month or donate money to the church, which may significantly increase the spending value for a specific day.

What's more, the mobile app system needs to estimate user's spending or income if necessary. Chances are that users will not input their income or spending every day, which is required in the PID controller part. The system should estimate somehow based on users' previous information.

Chapter 4: Introduction to Hospital Text Platform

4.1 Introduction to Text Platform

Nowadays, most patients make appointments with doctors by phone or email. Chances are that they may forget about their appointments, which will not only waste their own time in rescheduling but also decrease the efficiency of hospitals. Furthermore, for some patients with mental illnesses, daily communication is essential; however, talking with a doctor every day is not always possible. The SMS platform can remind patients of their own appointments as well as make some necessary psychological (e.g., to seek out appropriate family support) by sending them text messages. The text platform also contains a user login system for doctors to manage their patients and a search system to find the specific patient.

4.2 Introduction to Rails

The text platform is not only implemented by client-side technologies like HTML and CSS but also the server-side language, Ruby on Rails. Therefore, more complicated functions can be used like user login system, user management system as well as the message schedule system. Rails is a web application development framework that is written in Ruby language and used to make the development of web application to be much easier and convenient. It contains two major guiding principles: Don't Repeat Yourself (DRY) and Convention Over Configuration. DRY claims that the application should re-use as much code as possible rather than writing similar codes in the application. DRY is aimed at reducing the repetition of the information and making the whole application more clean and extensible. Then convention over configuration is a software design paradigm that allows the most decisions made by Rails rather than the developers to gain simplicity.

Chapter 5: Web Application Implementation

In this chapter, a MVC framework will be introduced at first. Then the whole web application will be explained in the respect of model, view and controller. Lastly, the text message and schedule function will also be discussed.

5.1 MVC Framework

The Model View Controller (MVC) design pattern divides the implementation of web application into three parts that cooperate closely. Rails provides three different libraries to implement the function of these three parts.

The Model part contains the data of application, which can be related with MySQL database and handles validation and transactions. A library in Rails called ActiveRecord can provide an interface and binding between the scheme in MySQL database and programming languages to manage the database records. The model can only communicate with Controller part, which means it has no knowledge about the user interface. Therefore, it can be reused several times, which follows the Don't Repeat Yourself (DRY) principles. The View part will take responsibility for creating user interface and presenting the information grabbed from Controller part to users. Usually they are made by HTML and described by CSS to provide a specific format and style of the web pages. In Rails, a library called ActionView will define the templates of data presentation; however, it is passive and cannot do any processing. The Controller part is basically a central process unit that receives events from users through the View part, communicates with Model parts and transfers the new information to View after some process. In the Controller part, programmers can implement large quantities of functions to satisfy the requirement of the web application like searching, sorting or sending and scheduling messages.

In Rails, there is a library called ActionController, which is a data broker among Model, View and Browser. When the Browser makes a request, the Web Server will receive it and use Routes to find out a specific Controller to serve it. The Dispatcher will pass the parameter and any other information to controller as well.

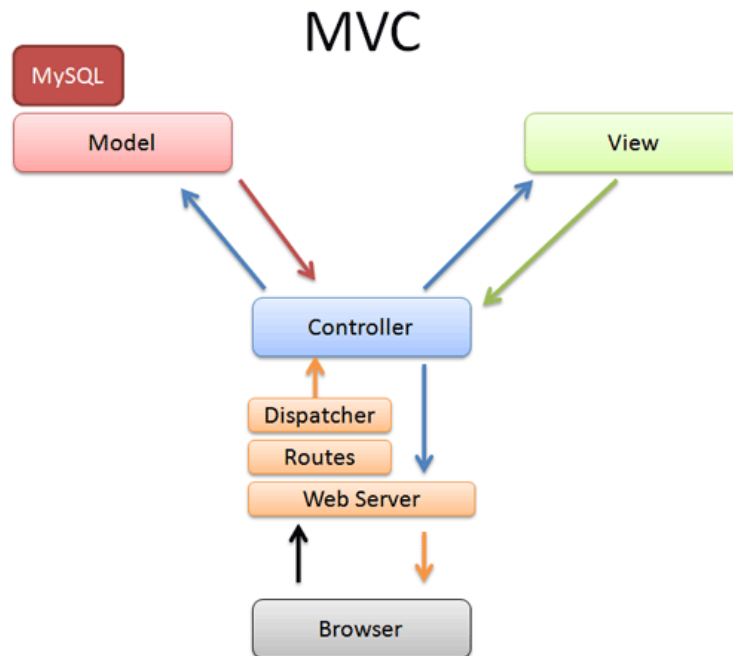


Figure 5.1a: Diagram of MVC

5.2 Model: User & Patients

There are two models in this Text Platform web application, user and patients. User model is used to represent the doctors and it has several entries like email, password_digest, name, phone, password and password_confirmation. Most entries have their own restrictions based on their format and length; furthermore, some entries have more requirements. Email and name cannot be null, which means users have to input this information, otherwise, they cannot register successfully. There is a special entry called password_digest, which is used to store the hashed password. When users input the passwords, the background system will keep the passwords in an

encrypted vault by converting the plain code into hashed password. Therefore, even if the web system has been hacked, this confidential information will not be released.

Model Name	Column	Type	Option	Description
User	email	String	presence, length{<=50}, format, uniqueness	Use as login name
	password_digest	String		Hashed password
	name	String	presence, length{<=50}	Name of the doctor
	phone	String	format	
	password	String	length{>=6}	Not record in database for safety
	password_confirmation	String		Not record in database for safety

Table 5.2a: Table of User Model

In the patients model, there are many more entries than user model and most of them are named straight forwardly. Similarly, there is a special entry called user_id, which is not the information input by users. Instead, it is automatically generated by the system when one patient model has been created and used to uniquely represent every single patient. Another two entries without the requirement of presence, note and img_url, are used to store some special notes made by doctors and the uniform resource locator (URL) of patient's photo. The birthday entry is also restricted

by the format requirement, which is described by regular expression. In other words, only inputs that follow the specific format will be accepted.

Model Name	Column	Type	Option	Description
Patients	user_id	integer		Use as login name
	name	String	Presence length{<=30}	Patient Name
	address_line1	String	presence, length{<=50}	Current Address, First Line
	address_line2	String	length{<=50}	Current Address, Second Line
	gender	String	Presence length{<=6}	Patient Gender
	age	integer	length{<=3}	Patient Age
	birthday	String	Presence, format	Patient Birthday
	note	string	length{<=50}	Note for Patient
	img_url	string	length{<=50}	Patient Photo URL

Table 5.2b: Table of Patients Model

5.3 User Interface

User interface is the interaction between users and the web system, which is also the only visible part to users in a web application. By clicking buttons, selecting menu or inputting messages, users can generate several events, which can be grabbed by controller and then get an expected

response. User interface is one of the most important parts because it determines how easy and efficient the application can work as planned. In this text platform application, the user interface contains static pages and dynamic pages. Static pages are delivered to user exactly as they stored, which means it cannot react or change based on the requests made by users. The skeleton of static page, for instance the homepage, is made by static HTML. The header is also called navigation bar that include several links to other pages for instance the login page and register page. The body part is a welcome page and a search box used to search patients by names.

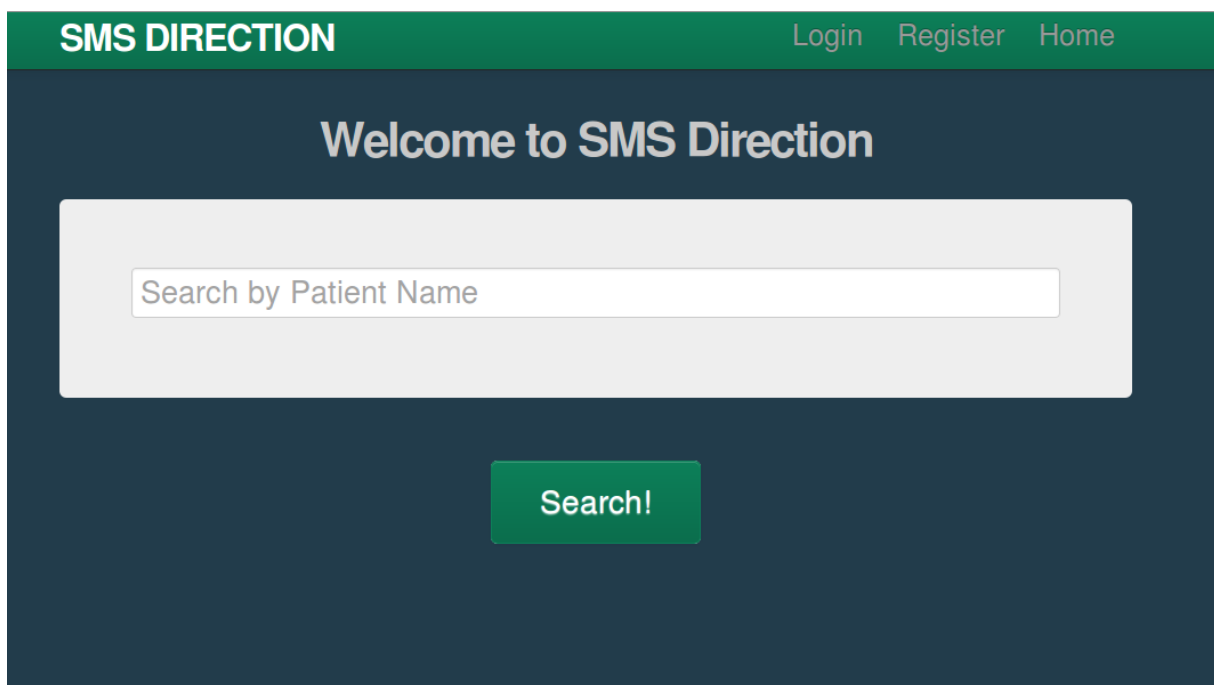


Figure 5.3a: Screenshot of Homepage

Rails provides a search function that can be defined in the controller of the patient's model. The name that user input will be grabbed and compared with the patients' names in the database and patients' information will be displayed to users if there is something matched. Similarly, since the user does not login, the navigation bar of this page is same with homepages and follows the guiding principle, reducing code repetition as much as possible. However, different from the homepage, the search result page is a dynamic page that can display different information based

on different requests. The number of matched patients in title will correspond with the result from database. In other words, it will not show any information if the patient's name is unfound.

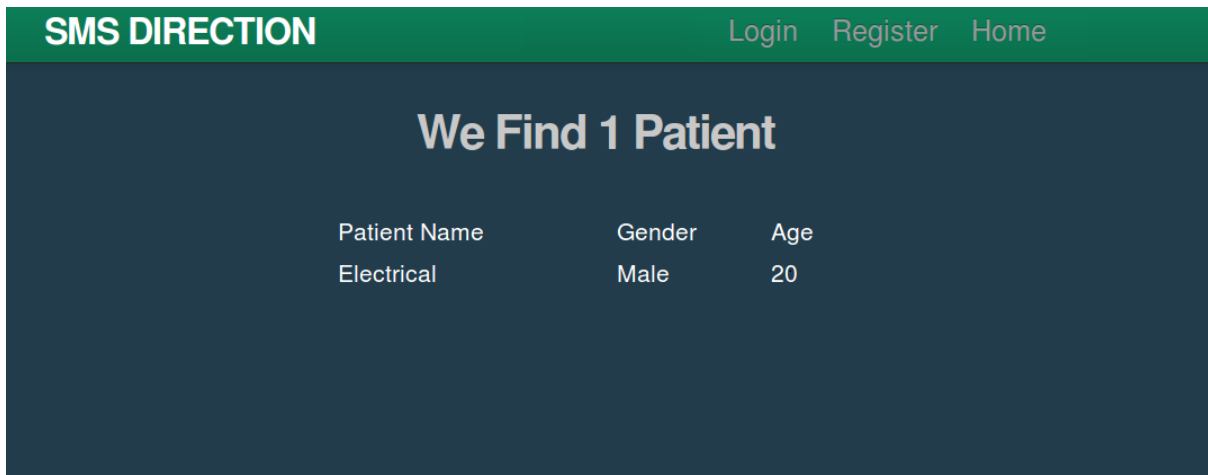


Figure 5.3b: Screenshot of Search Result

One of the most important functions in this web application is the login system, which can detect the status of users then display the corresponding pages. In order to have more access to posting patients or even scheduling messages for them, users have to register a personal account.

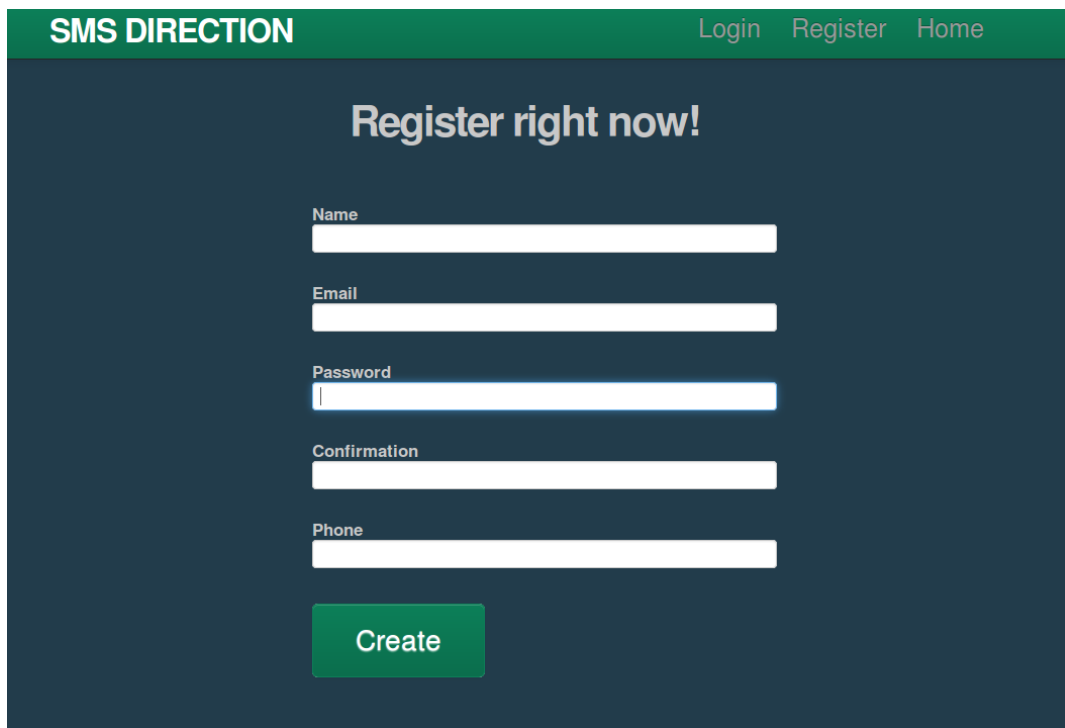


Figure 5.3c: Screenshot of Register Page

In the register page, users have to input some information such as their names, emails and so on. All these information will be grabbed and stored in the user model as explained above; therefore, they must satisfy the restrictions of the model. For instance, there is a uniqueness requirement about the email address so that a duplicate email will not be accepted and a warning page will pop out. In the SQL database, a model can be visualized as a relational table. Each row represents a certain user and each column has a column header that gives an indication of the meaning of the data in that column. After a new registration, a new row will be added into this table to represent a new user. As discussed in Chapter 5.2, the password will be stored as hashed characters instead of plain code. In order to uniquely identify every user, email entry is used as primary key that is also the default username. When a user try to log in, Rails will automatically grabs the input email and password then search the table for the matched email address and hashed password.

Name	Email	Password_digest	phone
Benjamin Bayer	Ben.1@osu.edu	!\$#var	61437316416
Chung-Cha Kim	Kim.1@gmail.com	\$@%#t	6143754409
Dick Davidson	Davidson.1@yahoo.com	\$_\$(\$,\$:[\$?])",	6143769854
Shuai	shuai@txt.att.net	LT? UP-N?&	6145958462

Table 5.3a: Relational Table of User Model

After login, users will be distributed a personal account where they can manage their patients and schedule messages for them. In addition, the navigation bar after login is quite different from previous pages, because there are some functions not available for guests. In the user account page, users can choose to update their own profile, which is basically the same page with register. However, after registration, users can never change their names and emails for the purpose of safety. Furthermore, there is also a list of patients added by current doctor as well as the options

to edit or delete the patients. Similarly, same with the search result page, this is also a dynamic page with the number of patients changeable.

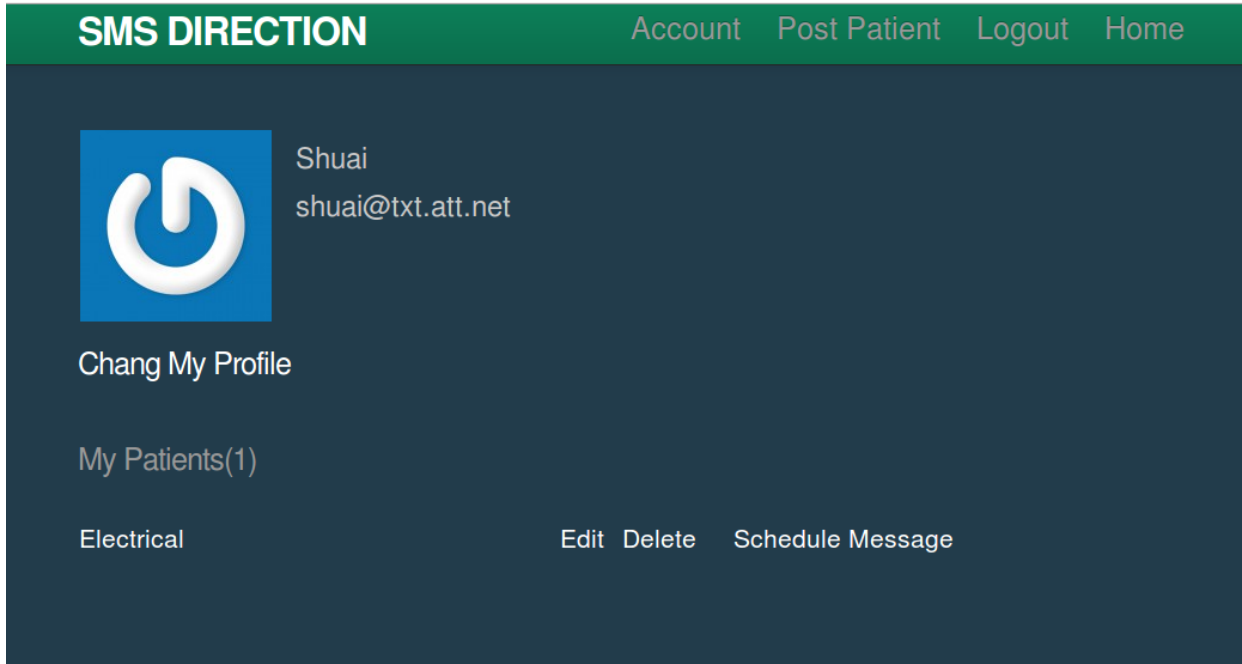


Figure 5.3d: Screenshot of User Account

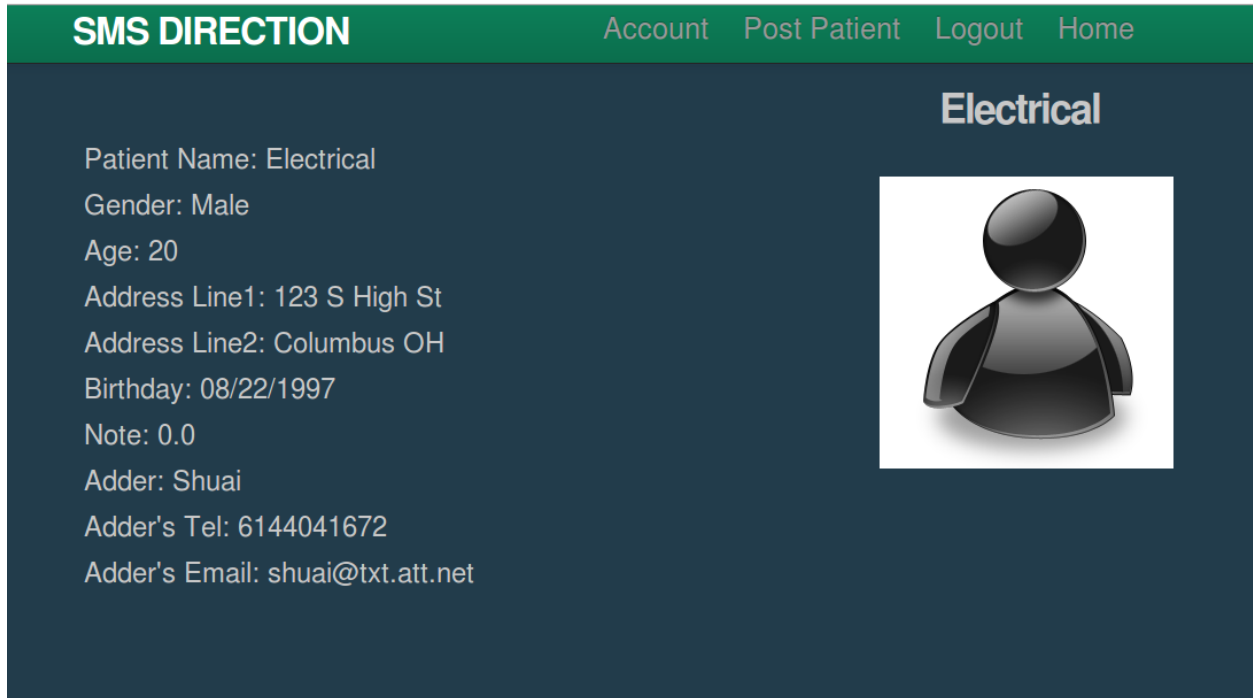


Figure 5.3e: Screenshot of Patient Info

To add a new patient, the doctor can click the “post patient” button any time after login. Add patient page is similar with the user registration page except for more information needed to be input. In the patient model, the special entry, user_id, which is automatically generated by system, is the primary key of the relation table. Therefore, there is no problem when patients have the same name by chance. Like the user model, patient model will store this information into a relational table as well.

The screenshot shows a web form for adding a patient. The form is set against a dark blue background. At the top, there is a green navigation bar with the text 'SMS DIRECTION' on the left and 'Account Post Patient Logout Home' on the right. Below the navigation bar, the main heading reads 'Add the information for your Patients!'. The form consists of several white input fields, each with a label to its left: 'Name', 'Gender', 'Age', 'Address line1', 'Address line2', 'Birthday', 'Note', 'Phone', and 'Image link for this patient'. The 'Image link' field has 'http://' pre-filled. At the bottom of the form is a green 'Submit' button.

Figure 5.3f: Screenshot of Add Patient Page

For user interface, the HTML will only build the skeleton and in order to make a precise and clean presentation, a framework called Bootstrap is used to create a nice design for this web application. Bootstrap contains several predefined CSS classes to create common components such as the typography, form, button, navigation bar and more.

5.4 Controller: Users, Patients, Static Pages & Sessions

Controller is the process unit and the only part that can communicate with Model and View. When routes choose which controller should be called, the corresponding controller would produce an appropriate output based on the request of users. In the Text Message Platform Application, there are four controllers in total to manage and implement the requests from users. Mostly, the controller will take actions after the communication with database, there are four common actions for interacting with database: create, read, update and destroy (CRUD) and all of them are predefined in Rails. Besides CRUD, another four actions are used in user controller as well. Action new will wake up when users want to register a new account, it will route to the registration page as expected. Similarly, edit and index also take responsibilities for routing pages. While, action show has more missions that it will not only route to the user profile page but also check the patient database and count the number of patients who are added by the current doctor.

Controller Name	Action	View	Description
User	index	index	User List
	new	new	User Register Page
	show	show	User Profile Page
	edit	edit	Edit User Profile Page
	create		CRUD for User Mode
	update		CRUD for User Mode
	delete		CRUD for User Mode

Table 5.4a: Controller of User

The create action can create new records in the database, which is triggered when a new user has registered. Similarly, the update and delete functions are used to edit or destroy the current records. However, in order to keep the safety, the update action has no access to the user's email and name.

The controller for patients is similar with users for most actions, while it still has a different actions, search. Action search will implement the search function with the input patient's name and route to the result page. Then the action new will route the current page to add patient page and wait for the submit button. In the patient model, a method of searching the database and returning an exact match of query is created and this searching function can be triggered in patient controller. Lastly, the Controller will pass the result to View part then display to users.

Controller Name	Action	View	Description
Patients	new	new	Post Patient Page
	show	show	Patient Profile Page
	edit	edit	Edit Patient Information
	search		Search the patients and return the results.
	Create		CRUD for Patient Mode
	update		CRUD for Patient Mode
	delete		CRUD for Patient Mode

Table5.4b: Controller of Patients

Static pages controller is simple and straight forward that it just routes current page to the homepage. The last necessary controller is sessions, which takes responsibility to detect the status of current users and routes to corresponding pages. Sessions are the idea that user's status need to be reserved across pages, which means the application must have the knowledge about whether the user has already logged in or not. However, the Hypertext Transfer Protocol (HTTP) is stateless, so a session controller is essential. In order to identify the user's status, Rails will store a secure and temporary cookie that contains the session hash information. When users make any requests, the session cookies will be included, which make it possible to track user's logged-in status. The create action would be triggered when a user has successfully logged in so that it can create a new session to remember the user's logged-in status. Furthermore, it will also choose a different navigation bar containing "post patients" to be displayed. Similarly, the action destroy can destroy the current session if the user has logged out or he closes the browser.

Controller Name	Action	View	Description
Static Pages	home	home	Return to the homepage
Sessions	new	new	Login Pages
	create		Create Session for user login
	destroy		Destroy sessions for user logout

Table5.4c: Controller of Static Pages and Sessions

5.5 Text Message

The core function of this web application is sending messages online, which could be an appointment reminder or psychological suggestions. However, instead of directly sending text messages to patients, it is easier to use a SMS gateway service to send messages by sending emails. Almost all telecommunication companies in the US including AT&T, Verizon and T-Mobile provide the Email-to-text Message service, which means users can send text messages by sending email to a specific SMS gateway address. For instance, the default email address for AT&T Company is the 10-digit wireless number followed by “@txt.att.net”. Since the patient model has already stored the phone number of every single patient, it is easy to send them email by concatenating their phone numbers and the suffix. In Rails, there is an action called Mailer that can send email from the web application by Mailer classes, functions and views. Similar with the controller, Mailer can be defined by the email sever host name, port number, username and the password. In other words, Mailer can get the access of an email server like a remote control program.

The screenshot displays the 'SMS DIRECTION' web application interface. At the top, there is a green navigation bar with the title 'SMS DIRECTION' and links for 'Account', 'Post Patient', 'Logout', and 'Home'. Below the navigation bar, the main content area shows a calendar for April 2015. The calendar is a grid with columns for days of the week (Sun to Sat) and rows for dates. The date 10 (Monday) is highlighted in blue and contains the text 'MessageAt 1043 try gmail'. The date 11 (Tuesday) is highlighted in yellow and contains the text 'Future'. To the right of the calendar, there is a form for sending a message. The form includes a 'Patient' field with the value '6144041672@txt.att.net', a 'Time' field, a 'Template' dropdown menu, and a 'Details' field. A 'Send' button is located below the form.

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30 MessageAt try gmail	31 1043	1	2 Future	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Figure 5.5a: Screenshot of Message Schedule Page

In this web application, there is a public email address of Gmail that is used to send emails for every user in order to keep the consistency of the message format. Its username, password, host name and port number have been preset in Mailer functions. In the Message Schedule Page, users can choose a given template or add more detail manually, which will be the content sent to patients by text messages. In addition, there is a calendar on the left side of the page, which contains the history of messages sent to patients. To implement the scheduler function, an daemon is used called Cron, which can run separately with the web application and execute the preset file periodically. Cron has the access to any specified file in the current system and can call the exactable files. A Cron-file has five fields for specifying the time followed by the command to be run at that interval.



Figure 5.5b: Cron Command Structure [2]

In our case, Cron is used to run the mailer function every single minute and the command should be the absolute address of Mailer functions. However, every email has scheduled delivery time stored in the database so that although the Mailer function has been called every minute, the function will not be implemented until the scheduled time. Therefore, a user can schedule several messages at a time, which are all stored and displayed in the left side calendar. Then as long as the message has not been sent, users still have chances to edit or delete the information by clicking the messages in the calendar.

The view of messages is designed with two versions, HTML and text format, to guarantee that all kinds of cell phones can receive the messages without any troubles. In addition, the doctor's name and subject of the messages will also be included like an email.

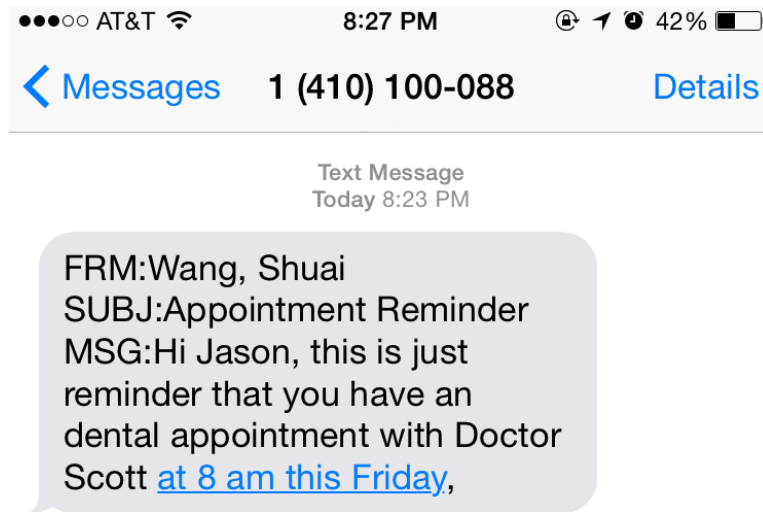


Figure 5.5c: Screenshot of Text Message

Chapter 6: Conclusion

6.1 Overall Conclusion

By Ruby on Rails, HTML and CSS, the text message platform application is built to manage patients in a more efficient and convenient way. Furthermore, doctors can schedule messages and send some information to patients. This web application can remind patients their appointments and provide psychological suggestions, follow-up surveys and so on.

6.2 Future Work and Application of the Project

Almost all basic functions for a text messages platform have been implemented in this application, however, an administrative system is still needed to manage the users account and patients information. As discussed in Chapter 5, in order to keep the safety of the system, normal users cannot change their name and email address. There should be an administrator, who has the access to all users' information including editing or deleting their profiles. In addition, the administrator should also be able to check all patients' information but cannot edit them, because only the specific doctor can manage his own patient. Another work needed in future is the receiving function that allows patients to reply the text messages so that doctors can get the response in time.

Obviously, this text platform application can be used in several areas such as the hospital and school. Due to the free SMS gateway service, this application can send out messages as many as possible without any charges. It is also well designed for charitable organizations and humanitarian engineering.

Appendix I: Project Schedule

May/2014-Aug/2014	Learn the basic idea of PID controller and learn the technologies of mobile application framework, PhoneGap.
Aug/2014- Nov/2014	Complete the first part of the research, Poverty Management App. Evaluate the performance and optimize its database and user experience.
Nov/2014-Jan/2015	Start the second part of the research, Text Message Platform. Build the basic web application, login and management system.
Jan/2015-Mar/2015	Complete the web application including the function of sending messages and scheduling messages.
Mar/2015-Apr/2015	Finished my undergraduate honor research distinction thesis and oral defense

Reference Page:

- [1] Chapman, R. Digital Control of Dynamic Systems. Belmont, CA: Thomson Learning, 2006. Print.
- [2] Cron_Syntax. Digital image.WebEnabled.N.p., 27 Jan. 2009. Web. 14 Apr. 2015.
<https://www.webenabled.com/setting-and-configuring-cron-jobs>.
- [3] Extension .js. Digital image.Wikipedia.N.p., 9 Apr. 2015. Web.13 Apr. 2015.<http://en.wikipedia.org/wiki/HTML>.
- [4] Hartl, Michael. Ruby on Rails Tutorial: Learn Web Development with Rails.N.p.: Addison-Wesley Professional, 2012. Print.
- [5] Harwani, B. M.PhoneGap Build: Developing Cross Platform Mobile Applications in the Cloud. N.p.: Auerbach Publications, 2013. Print.
- [6] HyperText Markup Language. Digital image.Wikipedia.N.p., 6 Apr. 2015. Web. 13 Apr. 2015. <http://bg.wikipedia.org/wiki/HTML>.
- [7] Institute of Electrical and Electronics Engineers, Inc.. (2006). Code of Ethics IEEE, <http://www.ieee.org/>. Retrieved at July 28, 2009, from the website temoa : Open Educational Resources (OER) Portal at <http://www.temoa.info/node/23284>
- [8]"JavaScript Tutorial." JavaScript Tutorial. W3C School, n.d. Web. 20 Aug. 2014.
<http://www.w3schools.com/js/>
- [9]"jQuery Tutorial." *jQuery Tutorial*.W3C School, n.d. Web. 20 Aug. 2014.
<http://www.w3schools.com/jquery/>
- [10] Miles, Troy. The Plot Chart.Digitalimage.The Rock N Coder.N.p., 28 May 2012. Web. 13 Apr. 2015. <http://therockncoder.blogspot.com/2012/05/jquery-mobile-charts.html>.
- [11] Myer, Thomas. Beginning PhoneGap. Indianapolis, In.: Wiley, 2012. Print.

- [12] "News." *World Development Report 2015 Explores "Mind, Society, and Behavior"* N.p., 2 Dec. 2014. Web. 14 Jan. 2015.<http://www.worldbank.org/en/news/feature/2014/12/02/world-development-report-2015-explores-mind-society-and-behavior>
- [13] Passino, Kevin. "Models, Dynamics, and Analysis of Poverty." *Humanitarian Engineering Creating Technologies That Help People*. Columbus: Bede, 2015. 70-95. Print.
- [14] Peter. CSS-Style.Digitalimage.Basic CSS Tutorial.N.p., n.d. Web. 13 Apr. 2015.
www.theinternetdigest.net.
- [15] Shotts, Kerri, and Suresh Mogre. *PhoneGap 3.x Mobile Application Development Hotshot: Create Useful and Exciting Real-world Apps for IOS and Android Devices with 12 Fantastic Projects*. 2nd ed. N.p.: Packt, n.d. Safari Books Online. 5 June 2014. Web. 07 Apr. 2015.
- [16] Urquizo, Arturo. A block diagram of a PID controller in a feedback loop.Digitalimage.Wikipedia.N.p., 6 Apr. 2015. Web. 13 Apr. 2015.
http://en.wikipedia.org/wiki/PID_controller.