Outerwear Suggesting Daily Weather Email with Settings Website

Project Report

Mitch Grogg

grogma01@students.ipfw.edu

CPET499 - Web Systems

Submitted to: Professor Paul Lin

Contents

[Executive Summary 3](#_Toc437727402)

[Introduction 3](#_Toc437727403)

[Problem and Proposed Solution 3](#_Toc437727404)

[Web System Design 3](#_Toc437727405)

[System Architecture 3](#_Toc437727406)

[Software Tools 5](#_Toc437727407)

[System Hardware Specification 5](#_Toc437727408)

[Software Design 5](#_Toc437727409)

[Software Flowcharts 5](#_Toc437727410)

[Software Description 6](#_Toc437727411)

[Overview 6](#_Toc437727412)

[Security 6](#_Toc437727413)

[Web Pages 7](#_Toc437727414)

[Software Code 7](#_Toc437727415)

[Website 7](#_Toc437727416)

[Email Service 8](#_Toc437727417)

[Software Testing and Validation 8](#_Toc437727418)

[Website 8](#_Toc437727419)

[Time and Cost 12](#_Toc437727420)

[Conclusion 13](#_Toc437727421)

[Works Cited 14](#_Toc437727422)

[Figure 1 - System Architecture 4](#_Toc437725958)

[Figure 2- MySQL Database Schema 4](#_Toc437725959)

[Figure 3 - Client Side Login/Settings Flowchart 5](#_Toc437725960)

[Figure 4 - Server Side Email Service Flowchart 6](#_Toc437725961)

[Figure 5 - HTTPS Header 8](#_Toc437725962)

[Figure 6 - Website Not Logged In 9](#_Toc437725963)

[Figure 7 - Registration Page Duplicate Username 9](#_Toc437725964)

[Figure 8 - Registration Page Register Success 10](#_Toc437725965)

[Figure 9 - User Settings Page 10](#_Toc437725966)

[Figure 10 - User Settings Input 11](#_Toc437725967)

[Figure 11 - Weather Email Test 12](#_Toc437725968)

[Table 1 – PHP files 7](#_Toc437727394)

[Table 2 - Time and Cost 12](#_Toc437727395)

# Executive Summary

This project allows a user to set up email alerts at predefined times that gives them the current and forecasted weather conditions for the day given their location. The system uses user-defined parameters to suggest what kind of coat they should wear for the day. For example, if one of the parameters is “wear a down parka” if the temperature is 25-40 degrees Fahrenheit, the email will include that suggestion if the temperature outside is in that range. This project consists of two main systems to accomplish this objective. The first is a website that a user can login to and set their parameters. The second is a server side email service that will generate the HTML emails and send them according to the user settings.

The website is implemented using PHP for the server side scripting and HTML, CSS, and JavaScript for the client-side scripting and markup. The email service is implemented using Python and scheduled using Cron. The database used by the system is MySQL. All of these components run on an Ubuntu 14.04 Linux server, which is hosted by Digital Ocean cloud hosting.

# Introduction

Online weather reporting and forecasting are a very popular demographic. According to the online Web Analytics site Alexa.com, Weather.com is the 34th most popular website in the United States. This project seeks to use email alerts to deliver personalized information to users based on the current and forecasted weather conditions. This removes the need for a user to visit an external website. This system is great for people in climates or in seasons where the weather changes drastically from day to day as it includes a suggestion on what to wear for the day.

# Problem and Proposed Solution

The problem addressed by this project is the need for a user to go to a website on a regular basis to obtain general information. For example, many people visit a weather website every morning to determine how they should what outerwear should be worn for the day based on the outside temperature. This system removes the need for the user to visit a website by delivering an email at a predefined time every day. Additionally, with the inclusion of personalized parameters for how to dress, the user is also given specific information that they care about. Given that they will also have the ability to login and edit their parameters at any time, they can adjust parameters as necessary. For example, if the user wears a down parka when it is 45 degrees out because their parameter was set in that range and it is too warm for their parka; they can change that parameter to avoid overdressing the next time.

In addition to the convenience of the aforementioned system, this system could actually be profitable through including non-invasive marketing in the emails. For example, if the system detects that it is very cold outside, it could append a short advertisement for a warm parka. Since the audience for this system is people who want to prepare how they dress for the outside conditions, the target demographic would already be known and advertising could be tailored.

# Web System Design

## System Architecture



Figure 1 - System Architecture

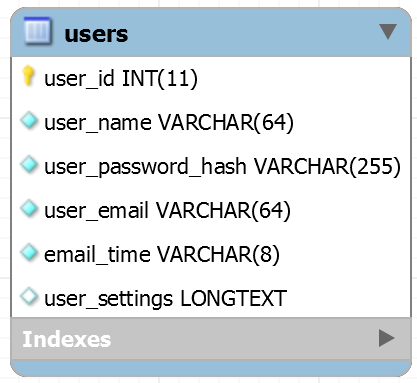


Figure 2- MySQL Database Schema

The system uses a LAMP (Linux, Apache, MySQL, PHP) stack to serve the website. The website consists of a home/user settings page and registration page. The user settings include the necessary form for input of the user’s parameters. The parameters include email, location (city and state), temperature ranges with specified clothing, and email time.

The server side email system uses Python and an SMTP email library for building and sending the emails. The emails are formatted as HTML to allow for rich text formatting and icons and images to be embedded. To obtain the necessary weather information, Wunderground.com, a free weather API such, is used. The user data to be shared between the website and the email system is stored in a MySQL database with a schema illustrated in Figure 2- MySQL Database Schema.

The client side web pages use the Bootstrap 3 [1] open source front-end design framework. This includes CSS classes for mobile-first design. Thus, allowing the webpages to be formatted for both mobile and desktop screens.

## Software Tools

The following software tools were utilized in the development of the website. Vim and Sublime Text 3 text editors were used for writing and editing code. XAMPP was used as the development server on a local Linux laptop. Vim was used for command line code editing and Sublime Text 3 was used for editing in a GUI environment. MySQL Workbench and phpMyAdmin were used for database administration. FileZilla FTP Client was used to transfer files to the production server. Chrome DevTools was used for analysis and debugging of client side code.

## System Hardware Specification

The server system hardware requirements were selected based on the base (least expensive) server available from cloud hosting provider Digital Ocean [2]. The system on which the production website runs is a 512MB, 1 CPU, 20GB SSD Disk Ubuntu 14.04 Linux Server.

# Software Design

## Software Flowcharts

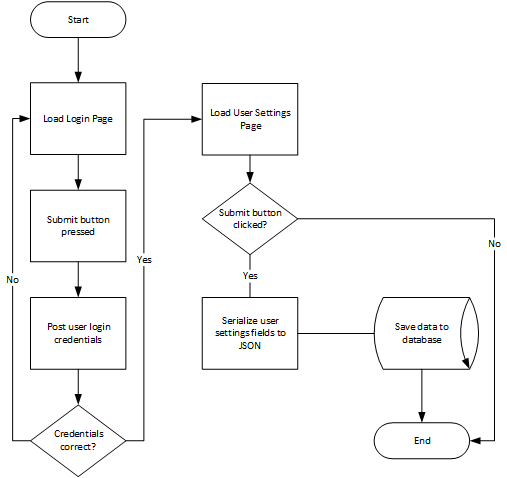


Figure 3 - Client Side Login/Settings Flowchart

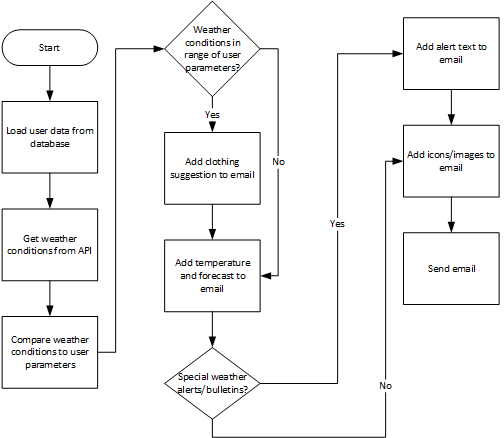


Figure 4 - Server Side Email Service Flowchart

## Software Description

### Overview

The login and registration section of this system is designed based on a basic login script framework called php-login-minimal [3]. The author explains reasons for using the script as follows,

In the PHP world every beginner tries to build login systems from scratch, doing all the typical mistakes, usually going from saving plain text passwords to using (horribly wrong) MD5 hashing. This script tries to give beginners a usable code base with a fully implemented user authentication (‘login’) system, preventing less-experienced developers at least from the worst security issues.

This login script was used because I wanted to implement a login system, but did not want to expose my user’s personal information if I made a mistake in the PHP coding of the website. For implementations of security, it is usually best to use off-the-shelf solutions rather than to try to reinvent the wheel, which is what I did here.

The php-login-minimal code includes three PHP views that are included. This system builds upon the basic included functionality which is login, register, and logout. Additionally, the included code was not styled at all. The HTML/CSS styling that was added is based upon the Bootstrap 3 library. The use of this script was very helpful in that it assured that I was securing my user’s information and also was a great learning platform since I had to expand upon code that used good, object-oriented PHP practices.

### Security

In addition to using a well-respected base framework (php-login-minimal), I also implemented HTTPS security. This enables TLS version 1.2 (Transport Layer Security) on all connections to the site, thus encrypting all information being sent between the server and the client.

To accomplish this, a certificate was purchased using a company called SSLMate. This company has a program that is installed on a Linux server to download the SSL certificates and keys. Once downloaded, a DNS route is generated and installed to verify that the person who downloaded the keys and certificates has authorization for the indicated domain.

After receiving the certificates and keys from SSLMate, the Apache server was configured to use the SSL certificate. The full Apache configuration file is available in the project files at “project\_website\_code\\_apache\_configuration\000-default.conf.” An overview of what this file does is redirects all traffic on port 80 to port 443. Port 443 is used for SSL traffic. Then, within the tag <VirtualHost \*:443>, the server name and website document root are specified and the location of the SSL Certificate Key, SSL Certificate, and SSL Certificate Chain are provided.

### Web Pages

As pictured in Figure 3 - Client Side Login/Settings Flowchart, the website is only based on two actual pages; the home page (index.php) and the registration page (register.php). When navigating to index.php, the content that is displayed to the user is dependent on whether or not the user is logged in based upon the PHP \_SESSION superglobal.

If the user is not logged in they see the page logo and a description of the service and navigation bar displays a form to login or register. If the register button is pressed, the user is directed to the register.php page, which loads the register.php view. If the user is logged in when navigating to index.php, they see the form for their settings and the navigation bar at the top of the screen gives them the option to log out. Additionally, attribution for the logo on the site belongs to Ande Myers (spouse of the author).

## Software Code

### Website

Upon navigating to index.php, this PHP file loads the needed files and classes. First it loads the following files.

Table 1 – PHP files

|  |  |
| --- | --- |
| **File** | **Description** |
| config/db.php | Contains the MySQL database login information |
| classes/Login.php | Contains the business logic for receiving and processing “login” POSTs. |
| classes/Settings.php | Contains the business logic for receiving and processing “settings” POSTs. |

Following the inclusion of the required files, a new “Login” object is instantiated and saved in an object variable. Finally, the isUserLoggedIn method is called. If the user has been successfully logged in, the they are shown the views/logged\_in.php file and if they have been logged in, they are shown the views/not\_logged\_in.php file.

When a user has been logged in, the view they are shown is a form containing their settings. These settings include Email, City, State, Email Time, and five Temperature Ranges. The Temperature Ranges each include a Low, High, and Outerwear input. This allows for five temperature and corresponding outerwear fields to be specified. There is a submit button at the bottom of this page. When JavaScript on the page first serializes the user settings into JavaScript Object Notation (JSON) and saves it to a hidden HTML form element. Then the JSON settings and email time are inserted into the user’s fields in the database in user\_settings and email\_time respectively. Each time the user settings page is accessed, the data for the form is retrieved from the user’s database record and inserted into the page. The “user\_settings” JSON data is parsed and inserted using JavaScript.

When a user navigates to the Registration page (register.php) by clicking on the “Register” link the code in the register.php file is executed. This includes connecting to the MySQL database and including the Registration.php class and instantiating a Registration object to handle form POSTs. Finally, the register.php view is loaded and the user is presented with a typical registration page.

The Registration page includes Username, Email, and Password (with confirmation). Upon clicking the “Register” button, a “register” POST is submitted to the server and if the username or email are not already taken, a new record is created in the database with their information.

### Email Service

The email service consists of a Python program called send\_emails.py and a file containing settings called settings.py. The Cron program, which allows for a specified program to be run on set schedule, is used to automatically spawn the send\_emails.py program every 15 minutes (HH:00, HH:15, HH:30, HH:45).

When send\_emails.py starts, it first connects to the data base and selects the user settings for users whose settings specify the current time to receive their email. For example, if users Rita and Percy are signed up for emails at 12:15, `select user\_settings from users where email\_time=”12:15”` will be executed and Rita and Percy’s settings will be returned and stored in an array.

Once the users who are signed up for the current time are received, the program then iterates through the array. Each user’s data is processed and sent and an email is sent based on current weather data. The current weather data is obtained using the Wunderground.com weather application programming interface (API). The conditions, forecast, and alerts APIs are used. The Conditions API contains the current weather condition, the Forecast API contains the three day forecast, and the Alerts API contains any weather alerts or emergencies. All of these API calls are done using the urllib2 library which makes HTTP requests. The APIs return the data in a JSON format. Examples for each of the API calls are below, using Fort Wayne, IN as an example city and <api key> as a placeholder for the actual key.

http://api.wunderground.com /api/< api key>/forecast/q/IN/Fort\_Wayne.json

http://api.wunderground.com /api/< api key>/conditions/q/IN/Fort\_Wayne.json

http://api.wunderground.com /api/< api key>/alerts/q/IN/Fort\_Wayne.json

The send\_emails.py program first uses the conditions API data to compare the current temperature to the ranges that were input by the user on the User Settings page. If the current temperature falls within that range, their selected outerwear is added to the email text. The email text contains the following information:

* Icon indicator of the current weather conditions (as returned by the API)
* The current temperature
* Outwear suggestion
* Indication if it is raining or windy
* Forecast
* Weather alerts, if any

This email is formatted as HTML, which allows for rich text and inclusion of links to images (weather icons). After building the text of the email, the email is sent to the user’s email using the smtplib module (simple mail transfer protocol) and a Google Gmail account.

# Software Testing and Validation

## Website

1. Navigate to website: <http://mitchbot.xyz>. Verify redirect to https site. **Pass - Figure 5**.



Figure 5 - HTTPS Header

2. Clear all cookies and navigate to <http://mitchbot.xyz>. Verify not logged in view shown. **Pass - Figure 6**.

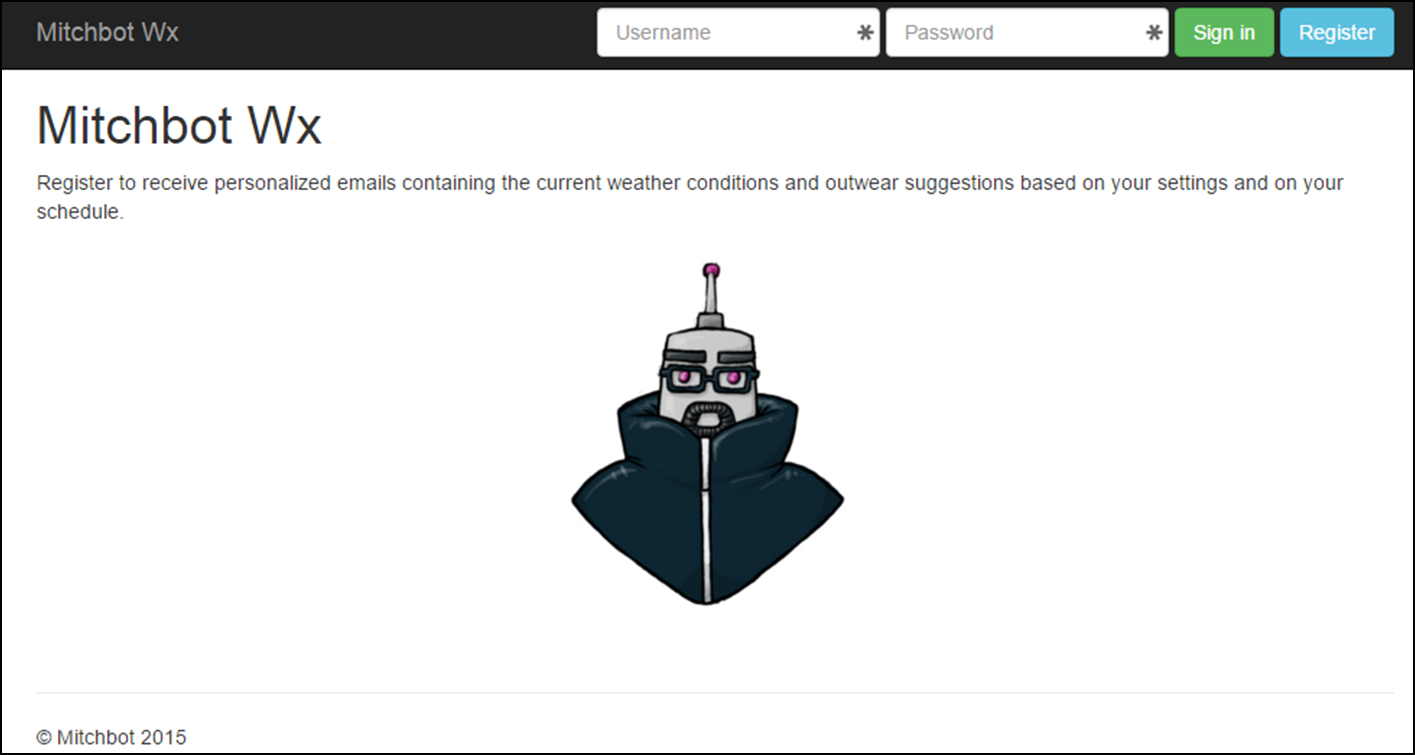


Figure 6 - Website Not Logged In

3. Click “Register” button. Register using known duplicate username “mitch”. Verify error message thrown. **Pass - Figure 7**.

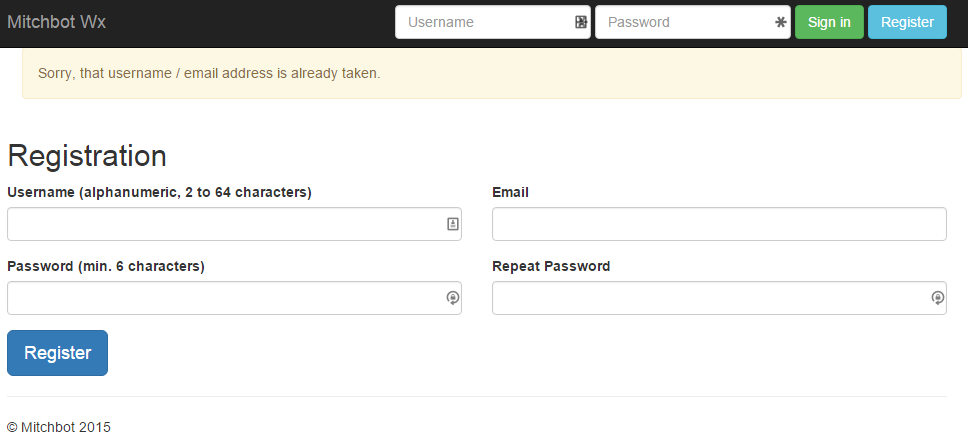


Figure 7 - Registration Page Duplicate Username

4. Register using new username “mitch” and email “mitchellgrogg@gmail.com”. Verify successful account creation. **Pass - Figure 8**.

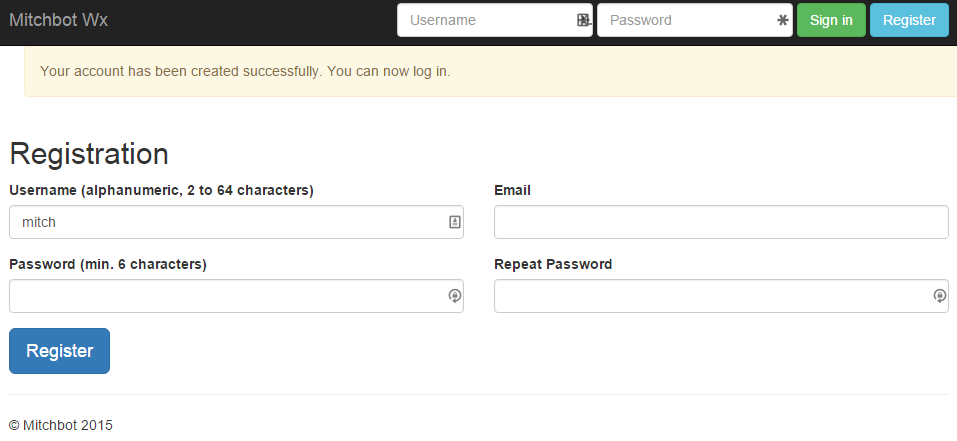


Figure 8 - Registration Page Register Success

5. Sign in using newly created user “mitch”. **Pass - Figure 9**.

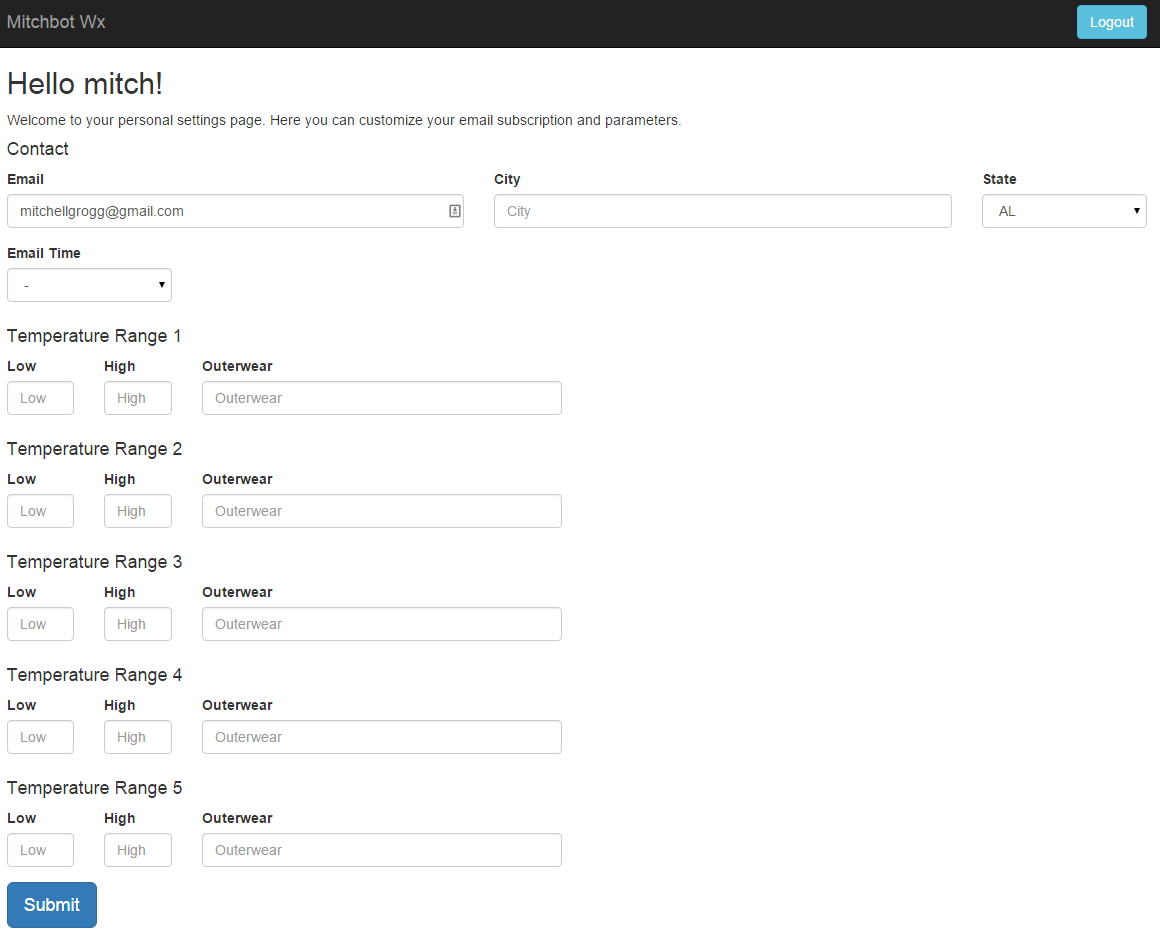


Figure 9 - User Settings Page

6. Fill out user settings (email time in next 15 minutes) and submit. Verify all settings are properly populated upon page reload after submit and that successful save message is displayed. **Pass - Figure 10**.

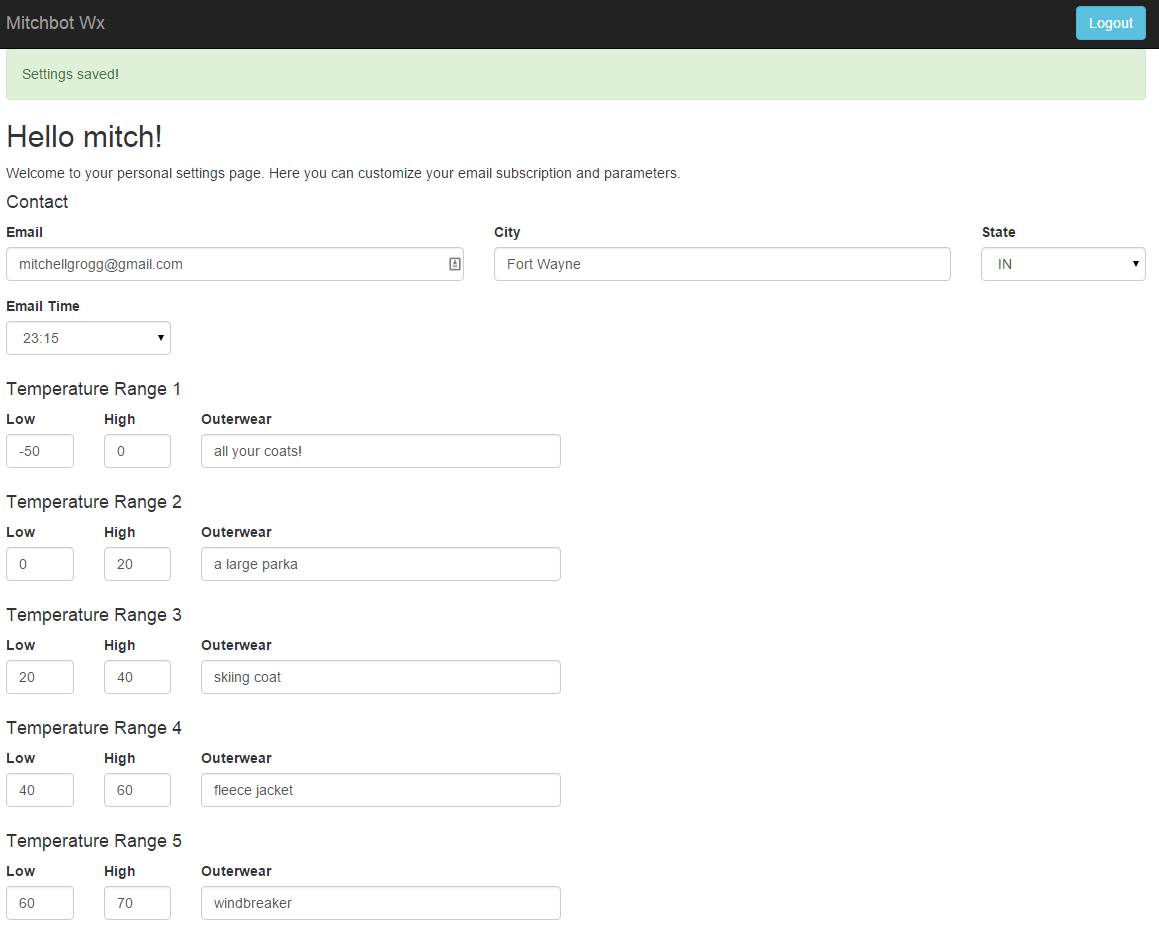


Figure 10 - User Settings Input

7. Verify email is received at scheduled time (11:15 PM) and indicates correct information based on user settings. **Pass - Figure 11**.

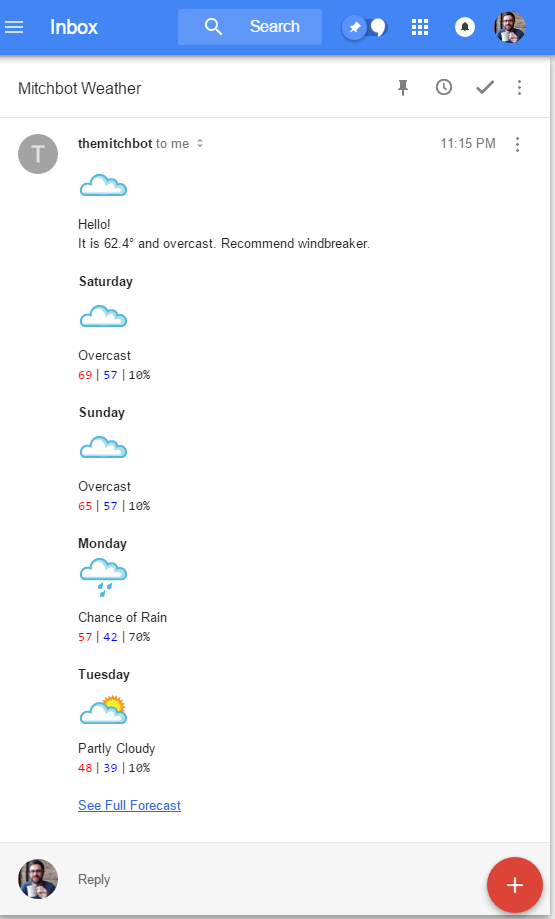


Figure 11 - Weather Email Test

# Time and Cost

Table 2 - Time and Cost



As referenced in Table 2, the time necessary to complete the project was significantly underestimated. Additionally, the need for and time to research and implement the HTTPS security was not foreseen when making the estimations. This ended up being a significant portion of the time required. I have a tendency to underestimate time required. This is a good reminder that I need to increase my estimations.

# Conclusion

The project was a success and I was able to implement both a website with working login, registration, and user settings pages. I was also able to interface the user input with a scheduled server side script to send emails based on the user input. Additionally, I implemented HTTPS security on the site. I hope to keep working on this site to make it better and will hopefully gain some users.

There are multiple improvements and additional features that can be added to the site. These include the addition of a “pause emails” and “delete accounts” options. Options such as these are important to not bother users with unwanted emails. Also, I would like to add the ability to let user choose the content of the emails. Last, I would like to improve the user interface of the temperature ranges with sliders or something similar. Overall, I am proud of how the site turned out and hope to continue adding features and maintaining it in the future.

# Works Cited

|  |  |
| --- | --- |
| [1] | Bootstrap, "Bootstrap," 10 December 2015. [Online]. Available: https://github.com/twbs/bootstrap. [Accessed 12 December 2015]. |
| [2] | Digital Ocean, "Digital Ocean," 2015. [Online]. Available: https://www.digitalocean.com/. [Accessed 12 December 2015]. |
| [3] | C. Panique, "php-login-minimal," 23 May 2015. [Online]. Available: https://github.com/panique/php-login-minimal. [Accessed 11 December 2015]. |