Software Test Plan

Team: Red Alert
Sponsor: State Farm Insurance
Faculty Team Mentor: Han Peng

Team Members:
Sal Galan (Lead)
Calvin Harper
Myles Dailey
Nick Nannen

4/1/22

Version 1

Table of Contents

Table of Contents **1**

1.0 Introduction **2**

2.0 Unit Testing **3**

2.1 Unit Testing Module: Dashboard 4

2.2 Unit Testing Module: User Accounts 8

2.3 Unit Testing Module: Map 12

3.0 Integration Testing **16**

3.1 Page Navigation 16

3.2 Interactive Mapping Functionailty 17

3.3 Notification System (SMS/Email) 18

3.4 Automations 19

4.0 Usability Testing **20**

4.1 Usability Testing Module: User Login 21

4.2 Usability Testing Module: Dashboard Usage 23

4.3 Usability Testing Module: Saved Searches, Subsets, and Automations Page 32

4.4 Usability Testing Module: User Profile Page 34

5.0 Conclusion **37**

1.0 Introduction

Our team has been tasked with creating an agent-client notification web app that will allow agents to send notifications to their clients easily using a visual map interface. Our envisioned product will enable agents to tailor communications with their clients based on their clients physical location. Currently, agents have no easy way to visualize where their clients are located. Our solution will allow agents to send warning and reminder notifications to their clients in case of natural disaster or problematic weather. Furthermore, agents can also use our product to send clients notifications to remind their clients to renew their policies or to simply wish their clients happy holidays.

The idea to create this application originated from a natural disaster use case. Our sponsors at State Farm envision this program being used as a tool used by State Farm agents to send notifications to their clients in times of crisis based on their clients home locations. If there is a fire, storm, or other disaster that is predicted to impact a specific geographic area, a State Farm agent could use our program to view where their clients live in order to warn their clients of these disasters. With this in mind, our web applications notification features need to be accurate and reliable, which is our primary motivation for a rigorous testing plan.

Software testing is extremely important for building reliable software that is easy to maintain. Proper software testing can also ensure the longevity of a program since software testing promotes bug free software and decreases the amount of unknown and hard to find bugs in the code. At its core, software testing simply verifies that a program is doing exactly what it is intended to do. For example, this could be as simple as verifying that a calculator program is adding numbers correctly.

There are many different types of software testing, but the three types of software testing we will be utilizing are unit testing, integration testing, and usability testing. Unit testing is a very granular form of testing as this method of testing verifies the output from individual functions in the code. Programs can have thousands of functions and verifying that each one works correctly can cut down on the amount of time required to debug the program. This type of testing could verify that a function that adds two numbers always returns 4 when adding 2 + 2 for example. Integration testing is a less granular form of testing compared to unit testing as this form of testing is concerned with the correctness of data returned from different modules. Since a program can have thousands of functions, these functions are usually organized into modules. Most of the time it is required that one program module is able to interface or communicate with another module. Communication between modules can become complicated quickly because each module is not necessarily related to one another so ensuring that one module returns the correct output to another module is extremely important to verify. Lastly, we will utilize usability testing which is mainly focused with ensuring that our graphical user interface is easy and intuitive to use. This type of testing gathers empirical observations gained from real users to gauge how easy to use an application is. For example we will implement usability testing by

giving a set of users a list of tasks to complete and collecting their feedback to improve the application.

To test our program we will focus on three modules: Dashboard, User Accounts, and Map. For each module we will implement unit tests, integration tests, and usability tests. By testing these three modules we can ensure that core features such as notifications, automations, user map selection, and account authentication work properly. Our testing will primarily focus on the Dashboard and Map modules since the functionality that they provide is vital for the application to work properly. For example, the map needs to consistently select the correct clients when a user draws a shape on the map. Our automated notifications also need to be sent on the correct date every time. We will begin our testing outline with unit testing which will be followed by integration testing and finally usability testing. Each type of testing will be organized into modules which contain specific tests for each function in the module.

2.0 Unit Testing

Unit testing is a method of testing concerned with the correctness of function outputs. For example, let's say we wrote a function that adds two numbers called addTwo(). In order to write a unit test for this function, we would create a second function that calls the addTwo() method with two numbers that we absolutely know the sum of, let’s say two plus two. The unit test would take the result returned from addTwo(2, 2) and check to ensure that the returned sum was exactly 4. This is the essence of unit testing. Writing functions that test other functions output for correctness. Since our program is a web application, most functions do not return explicit outputs such as a number, but they do perform operations such as database writes and reads, displaying information to the user or specific to our project, and sending notifications to clients. Most of our testing will involve checking user input against information in the database to ensure information is saved, edited, and displayed correctly to users. We will be using a python unit testing library called unittest that comes bundled with Django. This library is very basic and provides functionality similar to the example previously described. When using the unittest library a developer created functions that call other functions and save their return values and then check to see if that return value is what the developer expected it to be. We will be testing three major modules in our web application: Dashboard, User Accounts, and Map. These three modules provide the basic functionality of our web application such as displaying client information to users, sending sms and emails, and plotting client addresses on a map. Some of our unit tests cannot be conducted using unittest such as verifying that the correct phone numbers and emails receive emails sent to them. Instead we will visually examine and confirm the correctness of function similar to this.

2.1 Unit Testing Module: Dashboard

The dashboard module of our web application is responsible for fetching data and displaying that data to the user. This is the most important module of our web application as it provides the primary interface for users to interact with all features available on our website. The dashboard module initializes an Agent's list of clients, the map, the search bar, saved client subsets, and saved searches. The dashboard also provides the only interface for sending notifications and creating notification automations. It is vital to our web applications success that each of these processes return accurate results to the user.

Units being tested in this module:

- show_dashboard
- create_client_list
- send_message
- save_automation
- markOneTimeAutosAsInactive

Unit 1 Name

show_dashboard

Unit Description

Fetch an agent's complete list of clients and format the client data into an easy to read JSON format for the front end to display to the user on the dashboard website page. This function's primary responsibility is to fetch and convert an agent's list of clients into a JSON array to be used in the front end of our website for searching, sending notifications, creating subsets, and using the map. The accuracy of the formatted data is paramount to the dashboard page functioning properly.

Unit Tests

Test 1: Verify that the amount of clients shown to a user is the exact amount of clients associated with the user's account.

Bounds	- User has a valid account that can be used to login. - User is viewing the login page. - User has 23 clients saved in the database.
Input	- User logs into their account by entering their email and password
Expected Output	- User is redirected to the dashboard page after correctly entering their email and password. - 23 clients are displayed to the user in the search results box and on the map.

Test 2: Verify that users cannot access the dashboard before logging into a valid account.

Bounds	- User is viewing the login page but has not been authenticated by the website.
Input	- User clicks the dashboard link from the website toolbar.
Expected Output	- The user is redirected to the login page with a message that reads "You need to login to view that page!".

Unit 2 Name

create_client_list

Unit Description

For testing and proof of concept purposes our system creates a list of clients for a user behind the scenes. This function is responsible for creating 23 clients for the currently logged in user and is not a function that users interact with directly. Instead, this function is called from behind the scenes so data is ready for use by users. Each client is created with attributes such as a name, address, phone number, email, and most importantly an associated user id.

Unit Tests

Test 1: Verify that the clients generated by the system are only associated with the currently logged in user.

Bounds	- User has a valid account that can be used to login. - User is viewing the login page. - User does not have any clients in the database.
Input	- User logs into their account and is redirected to the dashboard.
Expected Output	- Each client user_id attribute in the database is associated with the currently logged in user. - Each client document in the database has a valid value for each field and no fields are empty.

Unit 3 Name

send_message

Unit Description

Send a message to the clients selected by the user on the dashboard page. Clients in our database contain a phone number or email associated with a capstone team member so we can verify that messages are sent and received properly.

Unit Tests

Test 1: Verify that only clients selected to receive a notification are the clients receiving the notification.

Bounds	- User is logged into their account. - User is viewing the dashboard web page.
Input	- The user selects two clients and sends a notification to those clients using the send notification interface on the dashboard web page.
Expected Output	- Each client the user sends a message to receives a notification of the correct type (sms or email) and the correct recipient receives the message.

Unit 4 Name

save_automation

Unit Description

Take input sent from the front end to the backend and create an automation document in the database. After the automation document is created in the database, schedule the automation to be sent on the specified date.

Unit Tests

Test 1: Verify that automations are saved to the database properly.

Bounds	- User is logged into their account. - User is viewing the dashboard web page.
Input	- The user selects two clients and selects the create automation button in the send notification area of the web page. - The user creates their automation by entering information for the same fields used to send a notification in addition to selecting a date to send the message on.
Expected Output	- An automation document is created in the database with the same information and clients selected by the user.

Test 2: Verify that automations are scheduled to be sent on the specified dates.

Bounds	- User has previously created an automation.
Input	- A list of automation ids that are currently scheduled to run at a specific date and time.
Expected Output	- A scheduled automation exists with the same id as the automation previously created by the user.

Unit 5 Name

markOneTimeAutosAsInactive

Unit Description

Fetch a list of automation that are scheduled to be sent only once and verify if the automation has been executed or not. If the automation has already been executed, mark the automation as complete.

Unit Tests

Test 1: Verify that one time automations are not sent more than once.

Bounds	- User is logged into their account. - User is viewing the dashboard web page.
Input	- User opens the automation creation interface. - The user enters the required information to create a one time automation and selects the "Create one time test automation button".
Expected Output	- A one time automation is created and scheduled by the system. The automation executes after 1 minute and does not execute again.

2.2 Unit Testing Module: User Accounts

The user account module of the website is responsible for creating new user accounts, authenticating user accounts, and editing and displaying user account information. This module allows users to view and manage the automations, subsets, and searches they have created. Users can alter automation execution dates, subset names, and saved search queries. Unit testing for this module will rely on verifying that information saved to the database is correct.

Functions being tested in this module:

- show_profile_page
- show_automations
- userLoginPage
- createNewUserForm

Unit 1 Name

show_profile_page

Unit Description

Show the user their profile page and provide a way for the user to edit their profile information.

Unit Tests

Test 1: Verify that user information is correctly displayed to the user.

Bounds	- User is logged into their account.
Input	- User clicks the “profile page” link located in the site navigation bar.
Expected Output	- The user information displayed on the profile page exactly matches the user information saved in the database.

Test 2: Verify that user information is correctly saved after editing their profile.

Bounds	- User is logged into their account.
Input	- User clicks the “profile page” link located in the site navigation bar. - User selects the “Edit Profile” button. - A User changes their name, last name, and agent code. - User selects the save changes button.
Expected Output	- The information entered while editing the user’s profile exactly matches the information saved in the database.

Unit 2 Name

show_automations

Unit Description

Display a list of user created automations and show a status for information which describes if the automation has been executed or not. Display a countdown timer indicating how many days, hours, and minutes are left until the automation is executed.

Unit Tests

Test 1: Verify that information for each automation is correctly displayed on the show automation page.

Bounds	- User is logged into their account.
Input	- User clicks the automation button located in the site navigation bar.
Expected Output	- Each user created automation is displayed with data that exactly matches the automation information saved in the database.

Test 2: Verify that automation edits are saved to the database correctly.

Bounds	- User is logged into their account.
Input	- User clicks the automation button located in the site navigation bar. - User selects the automation they want to edit. - The automation is displayed in a modal window to the user. - The user selects the “edit” button and changes the field that is editable.
Expected Output	- The automation information saved in the database matches the changes the user made to the automation.

Unit 3 Name

userLoginPage

Unit Description

Display the login page to the user and correctly show any important information as a pop up to the user. For example, if the user tries to navigate to any page except the login page before being authenticated they are redirected to the login page and displayed a popup warning them of why they were redirected. This module also authenticates user accounts and creates a session for the user.

Unit Tests

Test 1: Verify that users are displayed a popup on the login page upon completing a specific action.

Bounds	- User is not logged into their account. - User is on the website login page.
Input	- User clicks the dashboard page.
Expected Output	- The user is redirected to the login page and is displayed a message warning them that they need to log in to view that web page.

Test 2: Verify user accounts are authenticated correctly.

Bounds	- User is not logged into their account. - User is on the website login page.
Input	- User enters an invalid email and password.
Expected Output	- The user is displayed a warning message indicating that there was no account found with the credentials they entered.

Unit 4 Name

createNewUserForm

Unit Description

Display a form for creating a new user account. This page ensures that the information entered by the user is valid and can be used to create an account. If the user has entered bad information a warning message is displayed to the user letting them know the information they entered was incorrect.

Unit Tests

Test 1: Verify that users are displayed a popup on the login page upon completing a specific action.

Bounds	- User is not logged into their account. - User is on the website login page.
Input	- User clicks the create new account button. - The user enters invalid information such as entering letters instead of numbers in the agent code and phone number fields. - The user leaves the email, and birthdate fields blank. - The user enters mismatched passwords that do not meet the password requirements.
Expected Output	- The user is notified that they need to enter only numbers into the agent code and phone number fields. - The user is also notified that they are required to enter an email and birthdate in order to create an account. - Lastly, the user will see a list of requirements that have not been met with their currently entered password including that the passwords do not match. - The user is not able to submit their new account form.

2.3 Unit Testing Module: Map

The map module is responsible for operations regarding the interactive map in the dashboard. Operations and functionality that the map module provides includes rendering the map itself, plotting clients on the map, selecting clients by drawing around them, as well as all intermediary logic for the functionality provided such as clearing drawings and toggling specific selected clients.

Functions being tested in this module:

- plotClientSearchresults
- clearMapDrawings
- drawPolygonOnMap
- toggleSpecificPin

Unit 1 Name

plotClientSearchresults

Unit Description

Takes the search results from the user and refines the clients plotted on the map based on what is searched. By default, all clients are plotted on the map until a specific search query is made to narrow the results.

Unit Tests

Test 1: Verify that all clients are plotted on the map while there is no search query

Bounds	- User is on the dashboard page - User has not typed anything into the search query - Clients attached to the user exist in the database
Input	- None
Expected Output	- All clients attached to the user will be plotted and displayed on the map

Test 2: Verify that the plotted clients on the map are narrowed down based on a given search query

Bounds	- User is on the dashboard page - User has typed something into the search bar - Clients attached to the user exist in the database
Input	- User clicks on the search bar - User types something into the search bar
Expected Output	- Only clients that are found based on the search query are plotted onto the map

Unit 2 Name

clearMapDrawings

Unit Description

Remove any existing drawings that are on the map. Afer a user is done using the drawing feature, they can clear the map and get rid of any and all drawings by pressing the clear drawing button that is located directly below the map.

Unit Tests

Test 1: Verify that when the user clicks on the clear drawing button, all drawings are removed.

Bounds	- User is on the dashboard page - User has existing drawings on the map
Input	- User clicks on the clear drawing button
Expected Output	- All existing drawings are removed from the map

Unit 3 Name

drawPolygonOnMap

Unit Description

Allows users to select clients on the map by drawing specific shapes around them. Drawing is only active when the draw button below the map has been clicked. Shapes are drawn “point by point”, users click on the map to make a line from the previous point to the place they just clicked. To finish a shape and complete the drawing, the user must click on the starting flag.

Unit Tests

Test 1: Verify that shapes can be drawn on map and clients that are inside of the shape are selected

Bounds	- User is on the dashboard page - Drawing mode has been selected and enabled
Input	- Click on the map to initialize the start flag(first point in the shape) - User clicks on the map to add new line(s) to their shape - User clicks on the start flag to complete their shape
Expected Output	- Color of shape and start flag changes from red to green - If there were clients inside of the drawing, they are now selected - Drawing mode is disabled

Unit 4 Name

toggleSpecificPin

Unit Description

Toggles a specific client pin when it is clicked and ensures that they are moved into the selected clients list, or removed from the selected clients list if they are already selected.

Unit Tests

Test 1: Verify that a client pin is toggled to selected and put into the selected clients list when clicked

Bounds	- User is on the dashboard page - Client pins exist on the map - Client pin being clicked is not already selected
Input	- User clicks on a specific client pin
Expected Output	- Pin that was clicked will go from blue(unselected) to red(selected) - Client is added to the selected clients list

Test 2: Verify that a client pin is toggled to un-selected and removed from the selected clients list when clicked

Bounds	- User is on the dashboard page - Client pins exist on the map - Client pin being clicked is already selected
Input	- User clicks on a specific client pin
Expected Output	- Pin that was clicked will go from red(selected) to blue(unselected) - Client is removed from the selected clients list

3.0 Integration Testing

After unit testing is performed to determine if the individual components perform as intended, integration testing is used to test the interactions between the units. In other words, it tests whether different parts of the system pass the necessary information correctly between each other. This is critical to the Red Alert system, since we need to ensure that data is accurate and usable for alerting clients. Integration testing will involve testing the inter-functionality of the Red Alert Web Pages, Interactive Map, Notification System, and the Automations.

3.1 Page Navigation

Due to the nature of our project being a web application; we will conduct a simple test to make sure that every web page is up to date and that our links to pages work correctly. This will help ensure that our Dashboard, Automations, FAQ, and all other pages in our navigation bar correspond with the correct webpage. The purpose of this module is to ensure that all directories on the website route the user to the intended destination.

When conducting this test, a user will log in into their respective Red Alert account, and click on the different links from the navigation bar. They will visit the Profile page, Dashboard, Automations, and FAQ page. Once they are able to visit each webpage and use the navigation bar to alter between pages with no errors, then our Page Navigation test will be completed.

Model 1 Name

Page Navigation

Model 1 Description

This model will test the ease and functionality of our navigation bar and web pages.

Model 1 Tests

Test 1: User is able to transverse webpgaes easily.

Bounds	- Tester is logged into an account - Tester is on the profile page
Input	- Tester's will select between the different page links on the navigation bar.
Expected Output	- Tester's will revisit the selected webpage without errors.

3.2 Interactive Mapping Functionailty

To test the functionality of our Interactive Map, the tester will simply need to log into their account and visit our dashboard page. Within their account the tester should have clients loaded into their database based on a unique account ID. If this is the case these clients should display pins on the map of their location. If the tester is a new user and has no clients, then they will need to select the button, “generate client list”. This button will generate mock database information to prove that our interactive map is functioning correctly. Our interactive map needs to display clients, live filter clients, as well as outline selected geographical locations for messaging.Overall this is one of the more emphaisied models that our project contains.Once you have the selected client that you want, the map will save this selections and utilize them in sending notifications or when setting automations.

Model 1 Name

Interactive Mapping Functionailty

Model 1 Description

This model will test the functionality of our interactive map. The map should have integration with our backend database. The map will also allow for live filtering of elements to narrror map searches.

Model 1 Tests

Test 1: Tester is able to utilize our interactive map.

Bounds	- Tester is logged into an account. - Tester is on the dashboard page.
Input	- Tester will select a pin on the map to see if data is stored correctly with clients. - Tester will filter live searches to check and see if the map is live filtering those searches. - Tester will test the selection of multiple clients to be used for automations or notifications.
Expected Output	- Testers will be able to ultilize our interactive map for viewing clients.

3.3 Notification System (SMS/Email)

Another key feature that Red Alert must test using integration testing is our Notification System. A large part of our project is being able to assist State Farm Agents in providing a simple and effective way for them to communicate with clients. Our notification system allows for Agents to send messages directly from our web application. These messages can be mass commuicted via selected clients or sent to individual clients. The messages can also be for social reasons or if needed sent as an emergency to warm clients. These social and emergency notifications can vary based on events happening in that area. For example, if an agent wanted to warn people in Florida of a Hurricane, they could simply select the desired clients and send out a mass communication. If an Agent would like to just send a social happy birthday message, they could also do this.

Model 1 Name

Notification System (SMS/Email)

Model 1 Description

This model will test the functionality of our notification system. The Notifcation system should have integration with our backend database. The System should allow for the query to be selected with clients from our database. From here, when sending a message we will verify that its delivery type (SMS or Email) is correct along with the destination (Email, and Correct Phone Number linked to Client).

Model 1 Tests

Test 1: Tester is able to send SMS and Email Notifications.

Bounds	- Tester is logged into an account. - Tester is on the dashboard page.
Input	- Tester will select clients to send a notification too. - Tester will fill out the desired type of Notifcation (SMS/EMAIL). - Tester will send the message.
Expected Output	- Testers will be able to verify the message and that it uses the correct delivery type along with the correct desired location.

3.4 Automations

The last integration test we will be conductuing has to deal with sending automated messages to clients.The Automated system will allow State Farm Agents to send recurring messages to selected clients. These messages are flexible to the given agent but we envision them being ultilzed for reminders regarding policies, deadlines, or for set appointments. The flow of our Automations is very similar to our Notification System. The Agent will select the desired clients and instead of sending a notification, they will select the “Automations Button" located next to the send notification button. Once selected, the Automations Button will prompt a modal to input the given message along with when the recurrence of that message should be sent. Once completed, if the Agent wishes to view, modify, or delete the automation they can visit the Automations tab within our navigation bar and it will reroute them to the Automations page. This is where the Agent can perform edits, view, or delete the automations.

Model 1 Name

Automations Systems

Model 1 Description

This model will test the functionality of our automations system. The automations system should have integration with our backend database. This model will follow a similar flow as the notifications system. The automations system should allow for the query to be selected with clients from our database. Once the Agent has their desired client(s) they will select the Automations Button and create an Automation for the selected client(s).

Model 1 Tests

Test 1: Tester is able to send SMS and Email Notifications.

Bounds	- Tester is logged into an account. - Tester is on the dashboard page.
Input	- Tester will select clients to send a notification too. - Tester will fill out the desired type of Notifcation (SMS/EMAIL). - Tester will send the message.
Expected Output	- Testers will be able to verify the message and that it uses the correct delivery type along with the correct desired location.

4.0 Usability Testing

Usability testing is an important part of the testing process as it tests how easily and effectively the end user is able to interact and carry out the functions of the software application. This set of tests will ensure that a user can not only access all intended parts of our application, but they are also able to do so in an efficient and effective manner. As such, we assume that our end users, which are the State Farm Agents in our case, are not particularly technically savvy and we designed our project as such. Given this assumption, our tests should reflect an easy to learn yet effective flow of our application for the user.

Our usability testing plan will consist of a few modules that are designed to test the user-friendliness of our application. As stated above, our end users are State Farm agents that we assume for testing purposes don't have an extremely high level of technical proficiency. The effect that this will have on our tests in this section will be that they will be more skewed towards testing two properties; how easily and timely the user can learn and effectively use our software and if our software allows the user to access all of the functionality that they should be able to access according to our requirements specification document. Should these tests find that this isn't the case, then the consequences would be extremely detrimental to the overall usability of our application. Such consequences may include inability to access certain functionality that may severely limit a user's ability to perform key tasks in our application or a bad user flow that is complicated for the user to understand and/or takes too much time to complete. Another consequence of bad design in this area would be a steep learning curve for new users that complicates employee onboarding. While this product has been developed before in other forms, we believe that the novelty of our software lies in its inclusion of an interactive map tool that allows agents to select clients on a map with a drawing tool. While we believe that this feature is very useful and effective, we also realize that it may be unfamiliar to many and will need to be clear in its purpose in order to be used effectively, especially by users who are new to our software.

Our approach to testing will be similar to that of a guided survey. We will bring in testing candidates one at a time and walk them through each of our tests with little to no guidance and record how fast and how comfortable they seem with the software. We will then give them a survey with questions asking for details about each part's useability. With our observations coupled with the feedback of the user, we are confident that we will be able to gather significant intelligence about our software's usability. Since we want to simulate users with little to no technical experience, we will look for testers that reflect those qualities in order to get accurate results similar to those that our target user-base would give.

Given the information above, we have designed our usability testing pattern in a way that tests individual modules for user-friendliness and effectiveness. The models below detail the specifics of our testing pattern for this section.

4.1 Usability Testing Module: User Login

The user login part of our software is the first stop that a user, new or seasoned, will see when they visit our web application. It is also one of the most important as the user's account will have their personal subset of clients attached to it. This means that a user must be logged into their specific account to see their specific clients. This module will test how easily a user can create, log into, and log out of their account and relies on proper database functionality and management as well as good front-end design. Results will be based on both observance of user interaction as well as user feedback.

Functionality being tested in this module:

- Account creation
- User login

Model 1 Name

Account creation

Model 1 Description

This model will test both the ability for the user to access the account creation page as well as the efficiency that a user is able to achieve in the process of account creation.

Model 1 Tests

Test 1: User is able to quickly and successfully create a new account

Bounds	- User is connected to the internet - User is viewing the user login page
Input	- User selects the "Create new account!" button - User enter their details in all of the fields - User selects the "Create your account!" button
Expected Output	- Account is successfully created and stored in the database with all relevant data intact - User is redirected back to the user login page and a green

	popup stating that the account has been created appears

Model 2 Name

User login

Model 2 Description

This model will test both the ability for the user to login to an already created user account as well as logging out of the account. In this model we will be testing the functionality and ease of use for the process of logging into and out of an account.

Model 2 Tests

Test 1: User is able to quickly and successfully login to an existing account

Bounds	- User is connected to the internet - User is viewing the user login page - User already has a created account
Input	- User enter their username (email) and password associated with their account into the fields - User selects the "Log in!" button

Expected Output	- User is successfully logged in and is redirected to the dashboard area with the subset of clients associated with the account loaded in
	- Dashboard is correctly loaded with the following features:
	  - Toolbar
	  - Search bar
	  - Search filters
	  - Interactive map tool
	  - Search results
	  - Messaging module
	  - Selected clients
	  - Saved searches
	  - Saved Subsets

Test 2: User is able to quickly and successfully logout of an existing account

Bounds	- User is connected to the internet - User is viewing the dashboard page - User is logged into a created account
Input	- User selects the "Log out" button
Expected Output	- User is successfully logged out and is redirected to the user login page

4.2 Usability Testing Module: Dashboard Usage

The dashboard area of our web application is arguably the most important part of our project. It can be thought of as the hub of all of our functionality. It is the main page that the user will see upon successfully logging into their account and displays the searching tools, interactive map, messaging module, and the saved searches and subsets of clients. From this page, the user can access all other pages via the toolbar at the top of the screen as well as go through the entire process of searching and selecting clients, drafting an alert to send to them, and sending said alert. This module will test how easily a user is able to access and use these features and will rely on the majority of our application's features including our searching functions, database querying, and messaging functions to name the most prominent. As before, thoughtful front-end design will also be key to the success of these tests. Results will be based on both observance of user interaction as well as user feedback.

Functionality being tested in this module:

- Client searching
- Client selection
- Saving searches and subsets
- Creating an automation
- Sending alerts

Model 1 Name

Client searching

Model 1 Description

This model will test the client's searching ability that the user will use to find certain types of clients or individual clients. This feature set is imperative for the functionality of our application.

Model 1 Tests

Test 1: User is able to search clients using the search bar

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page
Input	- User types a letter or collection of letters into the search bar corresponding to an existing client
Expected Output	- The clients in the search results section reorder to have the most likely match at the top and less likely results below it in a descending fashion - Clients in the search results are shown to be plotted on the map - Some results deemed completely unlikely are removed from the results and the map

Test 2: User is able to search clients using filters

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page
Input	- User selects a city from the filters - User selects a policy type from the filters that corresponds to a client in the search results - User selects an age range from the filters that corresponds to a client in the search results
Expected Output	- Only clients whose data matches that specified by the filters that the client has selected will show up in the search results and on the map

Test 3: User is able to search clients by importing a saved search

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page
Input	- User selects a saved search from under the "Saved Searches" box

Expected Output	- Data from the saved search is translated to the filters and on the map and its results are correctly shown in the search results section

Test 4: User is able to select clients by using all searching options

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has saved a search
Input	- User selects a saved search from under the "Saved Searches" box - User selects a city from the filters - User selects a policy type from the filters that corresponds to a client in the search results - User selects an age range from the filters that corresponds to a client in the search results - User types a letter or collection of letters into the search bar corresponding to an existing client that is within the parameters of the filters
Expected Output	- Results from search bar, filters, and saved search import are correctly reflected in both the search results and on the map

Model 2 Name

Client selection

Model 2 Description

This model will test the functionality and effectiveness of the different methods by which the user can select the clients they wish to send alerts to. These features give the user options when trying to select clients to select the way that is best for their situation.

Model 2 Tests

Test 1: User is able to select clients from the search result list

Bounds	- User is logged into an account - User has clients associated with their account

	- User is on the dashboard page
Input	- User will find a desired client or clients and click the "Select" button next to the client's data in the search results section
Expected Output	- The "Select" button on all selected clients will turn into a "Remove" button - The selected client or clients will appear under the "Selected Clients" section at the bottom of the page - The pin on the map that represents the selected client or clients is highlighted

Test 2: User is able to select clients using the map drawing tool

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page
Input	- User selects the green pencil icon below the map - User clicks on the map to select a starting point - User clicks around the desired selection of clients - User clicks the flag when he/she is happy with their selection and wishes to complete the selection
Expected Output	- The "Select" button on all selected clients will turn into a "Remove" button - The selected client or clients will appear under the "Selected Clients" section at the bottom of the page - The pin on the map that represents the selected client or clients is highlighted

Test 3: User is able to select clients by importing a saved subset

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page
Input	- User selects a saved subset of clients from under the "Saved Subsets" box
Expected Output	- Data from the saved subset is translated to the search results section, on the map, and the selected clients section

Test 4: User is able to select clients by using all selecting options

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page
Input	- User follows the steps outlined in tests 1, 2, and 3
Expected Output	- Clients selected are correctly reflected on the map, in the search results, and in the selected clients section

Model 3 Name

Saving searches and subsets

Model 3 Description

This model will test the functionality and effectiveness of saving searches that the user has performed as well as saving subsets of selected clients. These features will greatly enhance the efficiency of searching and selecting as clients can save searches they frequently make and subsets of clients that they frequently select.

Model 3 Tests

Test 1: User is able to save a search they have made

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has made a search
Input	- User selects the "Save Search" button
Expected Output	- Search data is stored in a container in the database - Saved search appears both on the dashboard under the saved searches box and on the saved searches, subsets, and automations page under the saved searches section

Test 2: User is able to save a subset of clients they have selected

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User selects the "Save Client Subset" button
Expected Output	- Selected client data is stored in a container in the database - Selected clients subset appears both on the dashboard under the saved subsets box and on the saved searches, subsets, and automations page under the saved subsets section

Model 4 Name

Creating an automation

Model 4 Description

This model will test the functionality and effectiveness of automation creation and to assess its ease of use and functionality. Automation creation is another quality of life feature that allows users to create alerts that will be sent to a specified subset of clients at a specified time in the future. They can be set to alert these clients once or on a recurring basis.

Model 4 Tests

Test 1: User is able to successfully create and send out an automation

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User selects the "Create Automation" button under the messaging section - User fills out all fields and selects their automation type, message type, and message priority - User selects create automation

Expected Output	- Green banner briefly pops up in order to let user know that their automation has been created - Automation object is created and stored and will send the alert at the specified time - Automation shows up along with its status on the saved searches, subsets, and automations page

Model 5 Name

Sending alerts

Model 5 Description

This model will test the functionality and effectiveness of alert sending. These tests will seek to test the range of alert options that we have available to the user. We will be testing each combination to ensure that the functionality of this model is at 100%.

Model 5 Tests

Test 1: User is able to send an SMS emergency alert to multiple users at once

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User fills out all of the message fields and selects SMS as their message type and Emergency as their message priority - User selects "Send Message"
Expected Output	- A green banner will pop up to let the user know that their message has been sent - The emergency variant of an alert will go out to each selected client's phone number

Test 2: User is able to send an SMS social alert to multiple users at once

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User fills out all of the message fields and selects SMS as their message type and Social as their message priority - User selects "Send Message"
Expected Output	- A green banner will pop up to let the user know that their message has been sent - The social variant of an alert will go out to each selected client's phone number

Test 3: User is able to send an email emergency alert to multiple users at once

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User fills out all of the message fields and selects Email as their message type and Emergency as their message priority - User selects "Send Message"
Expected Output	- A green banner will pop up to let the user know that their message has been sent - The emergency variant of an alert will go out to each selected client's email address

Test 4: User is able to send an email social alert to multiple users at once

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User fills out all of the message fields and selects Email as their message type and Social as their message priority - User selects "Send Message"

Expected Output	- A green banner will pop up to let the user know that their message has been sent - The social variant of an alert will go out to each selected client's email address

Test 5: User is able to send both an SMS and an email emergency alert to multiple users at once

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User fills out all of the message fields and selects Email and SMS as their message type and Emergency as their message priority - User selects "Send Message"
Expected Output	- A green banner will pop up to let the user know that their message has been sent - The emergency variant of an alert will go out to each selected client's email address and phone number

Test 6: User is able to send both an SMS and an email social alert to multiple users at once

Bounds	- User is logged into an account - User has clients associated with their account - User is on the dashboard page - User has selected at least one client
Input	- User fills out all of the message fields and selects Email and SMS as their message type and Social as their message priority - User selects "Send Message"
Expected Output	- A green banner will pop up to let the user know that their message has been sent - The social variant of an alert will go out to each selected client's email address and phone number

4.3 Usability Testing Module: Saved Searches, Subsets, and Automations Page

This module shows the functionality of the webpage responsible for the viewing and editing of saved searches, subsets, and automations that a user has created. This data is user-specific meaning that each user will only be able to access and view searches, subsets, and automations that they have created/saved. This module will test how easy and functional it is to navigate to this page, view its contents, and edit each type of data on this page. Results will be based on both observance of user interaction as well as user feedback.

Functionality being tested in this module:

- Navigation to webpage
- Editing of data

Model 1 Name

Navigation to webpage

Model 1 Description

This model will test how easy it is for the client to navigate to the webpage. This should be easy and fast as users should be able to load pages quickly and not get lost in our web application.

Model 1 Tests

Test 1: User is able to quickly, easily, and successfully navigate to the saved searches, subsets, and automations webpage

Bounds	- User is logged into an account - User is on the dashboard page
Input	- User navigates the toolbar to find the "Automations" tab and clicks it
Expected Output	- The user is redirected to the page showing saved searches, subsets, and automations

Model 2 Name

Editing of data

Model 2 Description

This model will test how easy and effective it is for the client to edit a saved search, subset, and automation. This process should be easy to understand and fast to maximize efficiency and on-the-fly modifications to these collections.

Model 2 Tests

Test 1: User is able to edit/delete saved searches

Bounds	- User is logged into an account - User is on the Automations page - User has created a saved search
Input	- User clicks on a saved search they have made - User clicks "Edit" - User edits any of the searching information in the saved search - User selects "Save changes" - User clicks on the saved search again - User clicks "Edit" - User selects "Delete Search" - User selects "Yes" in the dialog box
Expected Output	- After the user saves the edits, a green banner will appear notifying them their changes were made and the saved search will reflect these changes - After the user deletes an saved search, a red banner will appear and notify the user that the saved search has been successfully deleted

Test 2: User is able to edit/delete saved subsets

Bounds	- User is logged into an account - User is on the Automations page - User has created a saved subset
Input	- User clicks on a saved client subset they have made - User clicks "Edit" - User edits any of the clients in the saved subset - User selects "Save changes" - User clicks on the saved subset again - User clicks "Edit" - User selects "Delete Client Subset" - User selects "Yes" in the dialog box

Expected Output	- After the user saves the edits, a green banner will appear notifying them their changes were made and the client subset will reflect these changes - After the user deletes a client subset, a red banner will appear and notify the user that the subset has been successfully deleted

Test 3: User is able to edit/delete automations

Bounds	- User is logged into an account - User is on the Automations page - User has created an automation
Input	- User clicks on an automation they have made - User clicks "Edit" - User edits any of the information in the automation - User selects "Save changes" - User clicks on the automation again - User clicks "Edit" - User selects "Delete Automation" - User selects "Yes" in the dialog box
Expected Output	- After the user saves the edits, a green banner will appear notifying them their changes were made and the automation will reflect these changes - After the user deletes an automation, a red banner will appear and notify the user that the automation has been successfully deleted - The list of automations will no longer contain the deleted automation

4.4 Usability Testing Module: User Profile Page

This module demonstrates the functionality and ease of viewing and editing a user's profile. A user may use these features for many things including moving from one agency to another or updating their address after moving. This module will test how easy and functional it is to navigate to this page, view its contents, and edit and save the user's data.

Functionality being tested in this module:

- Navigation to webpage
- Editing of data

Model 1 Name

Navigation to webpage

Model 1 Description

This model will test how easy it is for the client to navigate to the webpage. This should be easy and fast as users should be able to load pages quickly and not get lost in our web application.

Model 1 Tests

Test 1: User is able to quickly, easily, and successfully navigate to the user profile webpage

Bounds	- User is logged into an account - User is on the dashboard page
Input	- User will navigate the toolbar to find the Profile page - User will click the profile page
Expected Output	- User will be redirected to the profile page

Model 2 Name

Editing of data

Model 2 Description

This model will test the ease and functionality of a user updating their profile.

Model 2 Tests

Test 1: User is able to edit data fields in their profile

Bounds	- User is logged into an account - User is on the profile page
Input	- User will click on the "Edit Profile" button - User will change any fields they want

	- User clicks "Submit Changes"
Expected Output	- User's profile is updated according to their changes

5.0 Conclusion

Our team strives to create a quality application that is bug-free and works in the most efficient way possible. It consists of multiple parts, languages, and substructures that we want to work together as seamlessly as possible. Finally, we want our application to be extremely user-friendly given that we want our target users to know how to use our software with little to no guidance. Our team plans to ensure that all three of these core values we have are met by our application by implementing rigorous unit testing, integration testing, and usability testing. Our unit testing will help us determine if the functions we built in the code are working properly. Our integration testing helps us determine if we have any problems with the different parts of our software communicating and sharing data with each other. Our usability testing will help us determine what parts of our application design may need refactoring to be more user-friendly and efficient. These tests will help us to know if any bugs exist, if some parts of our code are not communicating properly, or if we simply overlooked a better option in our front-end design. We understand that our application could be used to save lives and prevent disaster through an inclement weather alert by a caring State Farm agent, and we're committed to building a product that's bug free and easy to use so that they can do just that.