Requirements Document

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

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

12/3/21

Version 1.2

Accepted as baseline requirements for the project:

Client:_____

Client:_____

Team Lead:_____

Table of Contents

1.0 Introduction

The insurance industry in the United States is enormous. In 2020, companies providing life insurance made a combined revenue of over 800 billion dollars in the United States alone. This is not very surprising given most homeowners, renters, and car owners have a desire to secure their belongings in case of a major accident. Companies like State Farm, Berkshire Hathaway, Progressive, and Allstate generate more than 30 billion dollars each in total insurance premiums paid by customers in 2020 with many other companies closely trailing behind. Property, home, and auto insurance companies are huge and growing considering that citizens care about ensuring their property will be replaced or fixed in the event of a disaster.

State Farm is the largest property, casualty, and auto insurance provider in the United States. Like most insurance companies, State Farm Insurance generates most of their revenue by selling insurance policies to customers. When a customer buys an insurance policy, that customer has to pay a monthly premium in order to keep their insurance policy. If a customer incurs some type of damage such as a car accident, a home catching on fire, or a family member passing away that customer will receive some benefit from State Farm in accordance with the customers insurance policy. In order to generate a stable revenue stream, State Farm and most other insurance companies must intake more premiums from customers than payouts for insurance policies. In order to sell policies to customers, State Farm employs agents which are independent contractors who are responsible for acquiring new clients. Agents are solely paid on commission which is earned by onboarding new customers and selling them insurance policies. This is why the agent to customer relationship is so important to State Farm as a company and their agents.

Our clients are Glenn Austin who is a Technology Analyst and Hans Yeazel who is a Technology Manager at State Farm. Glenn Austin is in charge of managing employee infrastructure. This means Glenn is in charge of projects that aim to improve the educational development of employees at State Farm. For example, Glenn’s team works on projects that provide additional educational education to employees such as developers, data analysts, and even consultants.

Similar to Glenn, Hans Yeazek works with State Farm product and development teams to ensure that each team is on the path to a quality solution. Hans also works with product planning teams to provide direction and leadership when discussing a solution.

In total, State Farm employs over 19,000 insurance agents and handles over 84 million insurance policies. State Farm also employs over 55,000 thousand internal employees that are not State Farm agents. Among the thousands of non-agent employees, State Farm employs lawyers, software developers, data analysts, paralegals and many other types of people with a broad range of skill sets.

2.0 Problem

Insurance does not sell itself which is why State Farm agents are integral to ensuring State Farms success. A State Farm agent is an independent contractor who sells insurance to people in their local area. Because State Farm agents' sales performance closely correlates with an agent's yearly salary, it is important that agents are consistently selling insurance policies and maintaining positive relationships with customers. Agents are the driving force behind State Farm's insurance policy sales which is why it is paramount that agents have the proper tools available to them to accrue and maintain customers.

Important to every business is acquiring new customers on a regular basis. One of the best ways to do this is to provide groundbreaking customer service to existing customers so that existing customers recommend the company to their friends and family. This could be accomplished by providing a simple way for State Farm agents to communicate with their clients in a way that is similar to communicating with friends and family.

Since State Farm employs over 19,000 insurance agents and handles over 84 million insurance policies, it is impossible for each agent to build a personal relationship with each client. The problem that our team will solve is that agents do not have a tool that allows them to easily and efficiently select clients to send group notifications too. Furthermore agents have absolutely no simple way to visualize where their clients live! This means agent's can't tailor their client communications based on client location in case of natural disasters or extreme weather. Agents strive to be the best neighbors they can be but they currently don't have all the tools available to be the best neighbor. Currently State Farm agents have tools to search for specific clients based on attributes such as birthdate or address, but this tool does not enable swift mass communications as agents still need to manually enter each client's contact information into their computer to send them a message. This is time consuming and painstaking, especially when an agent needs to send the same message to 20 or more clients.

The problem that agents are facing is similar to a contact list app that can not be used with the phone's texting or email apps! This means people would need to pull up their contact list and manually enter each recipient's contact information to send them a text message. That would be such a monotonous pain! This is what State Farm agents have to do when they want to message more than one of their clients at a time.

State Farm Agents Current Workflow for Sending Messages to Clients

1. Search for clients using attributes such as age or policy type.
2. Manually enter a client's contact information into an email or text.
3. Repeat step two until all clients have been entered into the message.
4. Send the text message or email to the client.

Problem with Client's Current Workflow:

- Manually entering an agent's client information is tedious and time consuming.

- Agents can’t search for clients based on location.
- Creating group messages to send to clients takes too long.
- Agents can’t visualize where all of their clients live.

3.0 Solution Vision

To address the lack of simple and efficient communication tools available to agents our team’s solution is to build a user based web application that allows agents to send texts and emails to groups of clients based on their clients personal attributes. For this project, the most important client attribute is their location. The focus of our web application will be a single page that will contain a digital map that displays each client's home address as a pin on the map. State Farm agents will be able to search for clients just as one would search for restaurants on google maps by panning around the city. Agents would also have the ability to select clients based on zip code or by dragging their mouse over the area of clients they would like to select. In addition to this digital map we will also implement a search bar that will allow agents to search for specific clients based on attributes such as age, location, or policy type. When an agent selects a client, that client will be transferred to a pool of selected clients. When an agent is ready to send their message to the selected pool of clients they type in their message, choose text or email and the message is sent in the selected format.

Specifically our solution will provide the following features:

- A State Farm agent’s client information will be provided by State Farm databases.
- Each agent will have their own user account with username and password.
- Agents will easily be able to send notifications to 20 or more clients in under 5 minutes.
- A digital map which will allow agents to use their mouse to draw an area on the map to select a group of clients to notify.
- A search bar interface that can search through an agent's client list using any attribute provided by State Farm databases such as age, location, policy type, or birthdate.
- An agent will be able to construct a group notification using the search interface and the digital map.
- Group clients into a subset that can be selected to easily send notifications to groups of two or more clients. For example, agents could create a veterans client subset so that on Veterans Day or related holidays, agents could select the subset to send a notification to.
- Send automated messages to specific clients or subsets.

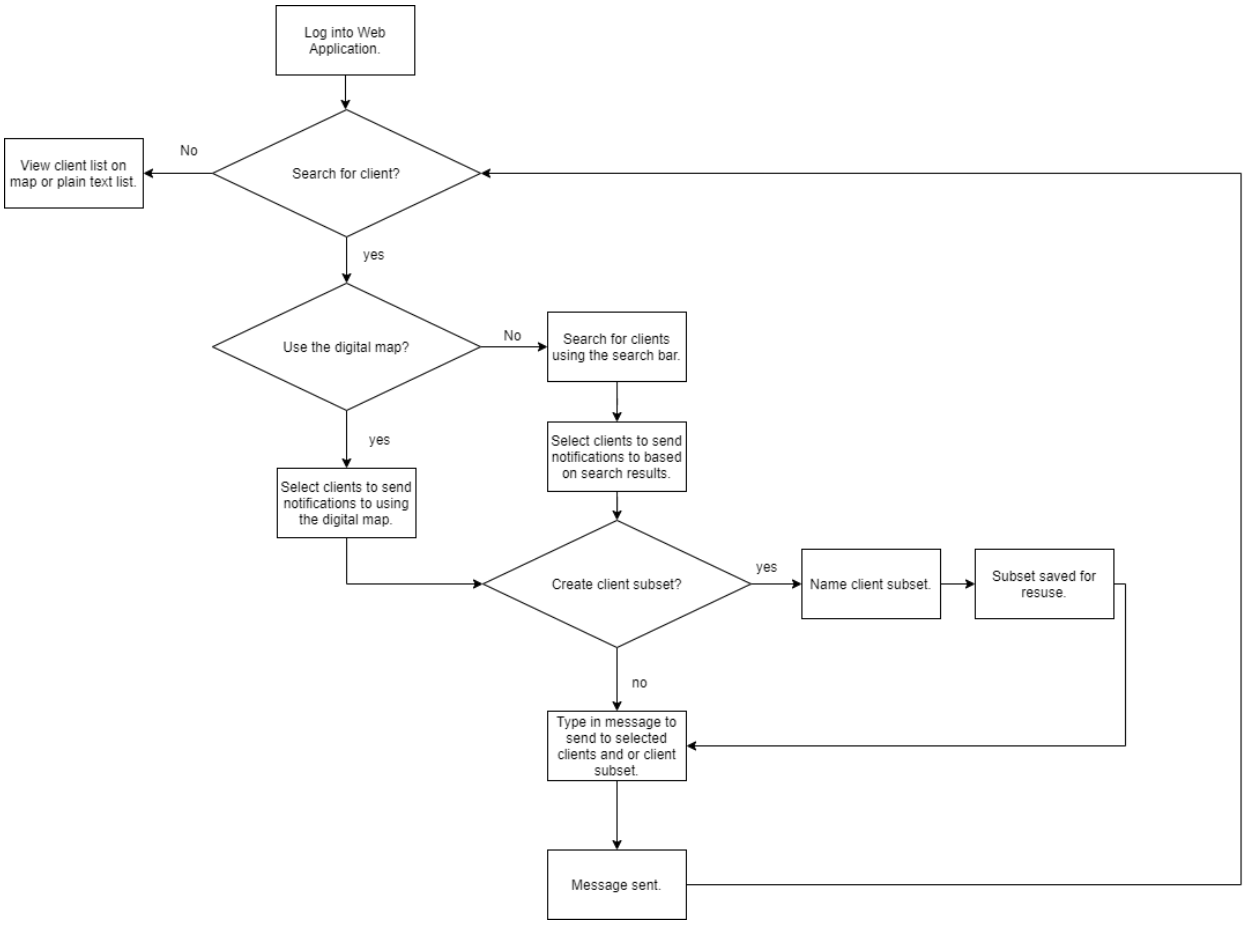

Figure 1. Diagram outlining how agents will use the web app.

Figure 1, outlines how agents will use the web application to view their list of clients or send notifications to specific clients or subsets of clients.

In order to acquire and search through a specific State Farm agents list of clients, our solution application will need access to State Farms client databases. With this information, our application will be able to generate lists of clients with some type of shared attributes. Our application will also be able to generate an accurate visual representation of each client's home address. Most of our applications computation will involve searching through existing databases and plotting client addresses on a map. With these features in hand, agents will be able to send warning messages to clients who are in the path of a thunderstorm or fire quickly because they will be able to see where each of their clients live relative to a possible disaster. Agents will also be able to send automated messages to clients that could say happy birthday, happy holidays, or happy veterans day.

Undoubtedly with an application like this, State Farm customer satisfaction will consequently increase so will State Farm’s annual insurance policy gains.

4.0 Project Requirements

In order to increase customer satisfaction as well as the satisfaction of our user's, the agents, we have created several requirements that we will be looking to as guides while we develop this project. These requirements range from the functional requirements that outline what functions our software will be performing to how well we will be aiming for our project to perform. Any specifications that our client has specifically requested in relation to their environment and compatibility with their existing systems will also be outlined in this section.

4.1 Functional Requirements:

State Farm handles millions of insurance policies, so our system must give State Farm agents the ability to easily search through their list of clients using State Farms existing client databases. State Farm records client information such as birthday, home address, age, and many other characteristics which must be searchable in order to provide agents a quick way to manage and send notifications to their clients. Because each agent has their own private list of clients they manage, each agent will need to have their own user account so that each agent has an individual space to use our application. Because insurance agencies like State Farm record sensitive client information our application will also need to be secure.

Note: "DR#" - "Domain Requirement Number #", "FR#" - "Functional Requirement Number #"

- **DR1 - Agent Accounts**
  Each State Farm agent will have their own user account with a username, password, and customizable profile picture.
  - **FR1 - Agent Accounts**
    - Each agent will need to create a user account to use the application. An agent's password will be stored securely inside our systems database and will not be stored in plain text.
      - Passwords will be hashed using a hashing function provided by a Python library available in the Django framework.
    - If an agent forgets their password there will be a system in place which will allow the agent to reset their password using the email they used to create their user account.
      - A website backend function will send an email to the user using smtp email server which will direct the user to a webpage where they can reset their password.
    - Because agents using the application will be doing so individually, agent accounts will be limited in personalization features. An agent's account

will primarily be used to manage the agent's email, password, and some application settings.
        - Once an agent is logged into the web application, their list of clients will be automatically loaded into the site as well so the agent can quickly get to work creating notifications to send their clients.
            - Upon logging in, our website will query the database for the relevant clients to display to the agent.

- **DR2 - Every Feature in One Dashboard**

  From a single page, State Farm agents will be able to search through their list of clients, select clients from a map, type notifications, automate notifications, and send notifications without having to navigate to an intermediary page to accomplish these tasks.

    - **FR2 - Every Feature in One Dashboard**
        - Our application will give agents the ability to view and search through their list of clients.
            - We will use some type of javascript or Django library that provides the ability to search through a list of database records.
        - The application will also provide a digital map which will give agents a visual representation of their list of clients.
            - Using Nominatim and Leaflet, our web app will query the database for an agent's clients and plot each individual client on the digital map.
        - The agents list of clients, the search interface, and the digital map will all be available from a single dashboard web page.
        - The dashboard view will also give agents the ability to add clients to a notification pool, through the search interface or digital map, so that agents can send the clients in the notification pool a notification.
        - This web page will also contain tools to allow agents to save a specific search query.
            - Once an agent has saved a search query, that agent will easily be able to select the saved search query from their list of saved searches in order to make the same search again.
            - Agents will also be able to delete saved searches if they do not need them anymore.

- **DR3 - Ability to search and select subsets of clients**

  Agents will be able to filter what clients they see on the dashboard map or they will be able to see all of their clients. Agents can also drag their mouse around any group of clients to select them to send notifications to.

- **FR3 - Ability to search and select subsets of clients**

  Searching and identifying their clients based on specific criteria, whether it be by address or geographical location, plays a critical role in the functionality of our software. The interactive map in our software can be considered as the keystone to the whole project; There is a slew of different “low level” functionality that the interactive map portion is expected to handle in order to provide the appropriate data that an agent might be looking for. At its core, our software is a simple and easy to use notification system, and our methods for searching and retrieving data should reflect that.

  - **FR3.1 - Constraints related to individual data elements**

    Agents will be able to use the search bar to easily identify and retrieve specific clients based on a variety of criteria that the agents have control of. We should be able to search the client database based on any number of attributes that might be attached to their “profiles”; But for right now, the two basic criteria that a client can be identified by in the search bar is by their street address specifically, or a larger subset of clients can be identified by the policies that they are “subscribed” to. Other criteria can, and likely will be implemented, but these are the fundamental search aspects that our sponsor has requested. After a search has been completed, agents will have the ability to save and store the search query to be re-executed at a later date. The list of clients acquired from any given search can also be saved into a “named subset”. Agents will be able to modify(edit), or delete, the named subsets as needed. Considering that this data might be useful outside of our software, the stored subsets can be exported for external use.

  - **FR3.2 - Geographic constraints using interactive map**

    Although the search bar can take care of constraints based on individual data elements, our interactive map will provide agents with the ability to make search queries based on geographical data. Agents will be able to use drawing tools to select and outline regions of the interactive map that they would like to retrieve client data from. When the region is defined, the program will return the subset of clients, if any, that fall within the outline. To further refine the results of this query, the subset can also be pruned and modified based on the same criteria that can be used in the search bar. For example, if a specific region was expecting to receive thunderstorms within the next few hours that pose the risk of flood damage, an agent might consider outlining the surrounding area and notifying all clients within that area that have a policy that contains flood

damage. The interactive map also has the same ability to store “named subsets” of clients; This could be particularly useful to save and notify a portion of clients that live in a geographical region that is prone to disasters such as forest fires, flooding, tornados or any other relevant phenomenon.

- **DR4 - Ability to select a variety of actions to apply to a searched subset**

  Agents will be able to select clients on the map and choose to send them notifications, create automations using the client selection, create client subsets, and save query searches.

  - **FR4 - Ability to select a variety of actions to apply to a searched subset**
    - Our solution will prioritize having a feature that allows the State Farm Agents to select a variety of actions to apply to a searched subset.
      - This means that the agents should be able to select a group of clients based on information such as age, location, policy type etc.
    - Once the agents have this subset selected they should be able to use all the functions we have implemented in our system to assist in communicating with these clients more efficiently.
      - This means they should be able to send notifications, update accounts, or be able to modify the entire subset easily.

- **DR5 - Ability for customer to opt out of notifications, or opt out of some notifications, e.g., all but “emergency”**

  By default, all of an agent's clients will receive a notification when an agent signs up to use our web app and clients will immediately be asked if they want to opt in or out of receiving future notifications. Each notification sent to clients will also provide clients the ability to opt in or out of receiving future notifications. Clients can also choose to only receive specific types of notifications such as emergency notifications at any time.

  - **FR5 - Ability for customer to opt out of notifications, or opt out of some notifications, e.g., all but “emergency”**
    - Another feature we will be implementing is the ability to opt out of our service. We do understand that the service we provided could be assisting millions of people; however with this being said our product must allow for the user to opt out of the service if they choose they no longer need the service.
      - We will be using a website sms server and an email smtp server to send notifications to clients. We will also use these tools to

determine if a client has opted in our out of receiving notifications from our app.

- **DR6 - Ability to prioritize messages that are sent, e.g., emergency, account-related, social.**

  Any message sent using our web app will be labeled as emergency, account-related, social, reminder, or other types of categorizations. Clients will be able to respond to notifications to specify which types of notifications they want to receive.

  - **FR6 - Ability to prioritize messages that are sent, e.g., emergency, account-related, social.**
    - Continuing, another feature we have implemented is the ability to prioritize the notifications and messages being sent to clients.
      - This feature will allow State Farm agents the necessary ability to keep their clients informed and up to date.
    - Our system will have an automated system for the pre-made notifications, however we did want the agents to have the ability to send notifications right away if needed.
      - The system will allow for the agents to create their notifications and prioritize based on how they see fit. For example, State Farm agents can prioritize emergency requests, account-related situations, or social events how the agent see's fit.

- **DR7 - Creating Automations for Client Notifications**

  Agents will have the option to create notifications that are automatically sent to a specific set of clients at a recurring interval or on a specific date and time.

  - **FR7 - Creating Automations for Client Notifications**
    - Agents will be able to use their saved search queries or client subsets to create automations.
      - Automations are predefined messages that are sent out when a specific time interval occurs. For example, agents could create a veterans client subset and then create an automation that automatically sends the message "Happy veterans day, thank you for your service!" every Veterans Day.
    - An agent could use a saved search query that includes all clients with an auto insurance policy
      - An agent could create an automation that sends a monthly text message to each client in the saved search query reminding them to perform routine maintenance on their vehicles.

- **DR8 - Integration with client's existing systems**

  Our web app will use MongoDB as the database because it is the same database that State Farm uses to store client information.

  - **FR8 - Integration with client's existing systems**

    One of our biggest focuses when we started to think about this project is how closely we could replicate the conditions in which this piece of software could be implemented by our clients. We kept this requirement in the back of our minds from the beginning and weighed it against most, if not all, decisions we made when deciding things such as our tech feasibility choices. To aid in our understanding of how our client's company, State Farm, currently handles relevant tasks such as client data storage and agent-to-client communication, we asked our clients Glenn Austin and Hans Yeazel questions as to what database frameworks and languages do they use and what sort of information the keep with each client. This information will allow us to replicate their databases as closely as possible in order to effectively test our system in the most accurate environment as we possibly can.

    - **FR8.1 - Database Compatibility Constraints**

      We chose MongoDB as our database framework primarily because that is one of the frameworks that State Farm currently uses to manage client data in their back end. Choosing this database framework will allow as seamless transition as possible if our clients decide to implement our project into their existing system. It also allows us to more effectively test our system in the context of our clients considering that we do not have access to their actual database given that it would be a huge security risk. We plan to go into even more detail in future meetings with our clients to gain a better understanding of the internal structure of their database detailing things such as what specific data fields they record for clients, possible access restrictions, and typical syntax used in their databases. Details like this will only further help us mirror the way their system operates.

    - **FR8.2 - Front-End Design Constraints**

      Not only do we want our system to integrate well with our client's existing database, we also want our product to look and feel like it belongs at State Farm. This includes things like our color palette for the front-end as well as an organizational structure that is best for the client. The colors we are looking to use are

primarily red and white as those are the colors of our client's company. We may also look to give it a bit more diversity in color by adding some greys into the mix. This will help our product feel right at home in the State Farm environment and give the front-end a more exciting look than a more mundane-looking interface.

- **FR8.3 - Interface Constraints**

  While color is important for a product, a more important aspect is how our layout is done. We want our layout to be the best possible fit for the agents that will be using it. So we plan to try out a few different layouts along our development process and gather as much feedback as we can on what works and what could be improved. This information will not only make the final product more comfortable and user-friendly for our clients, but will also hopefully improve the speed and efficiency that the agents using our system will be able to achieve when using the product.

Overall, we want to make the integration of our project as seamless as possible for our clients so that not only do they have an easier time integrating it into their existing systems, but also that we have an accurate portrayal of what their systems look like so that we may develop an effective product on top of it. If we accomplish this requirement well then our product will run easily alongside our client's database architecture and will also look and feel like a product worthy of State Farm's use.

4.2 Performance Requirements:

Our project has a specific emphasis on speed and performance as some of its use cases include some emergency cases. On top of this, it is also very important to our project that agents using our system are able to communicate with their clients as fast as possible no matter the situation as the quality of customer service depends directly on this aspect of the project. There are some performance bottlenecks that we expect this project to have based on some limited options as far as frameworks go and we will address those in this section as well. Another important non-technical requirement is the user performance. This includes things such as how fast a user is able to pick up and use the different parts of the system effectively and additionally how difficult it is to teach users how to use our system. All of these aspects will determine the overall performance of our project so mitigating performance issues and bottlenecks anywhere we can is critical to our success.

Speed:

Speed is important in any software system and ours is no exception. The responsiveness of our software is one of the key things our software revolves around as communication is more useful the faster it is accomplished.

- **Front-End:**

  One of the most important performance requirements we have is that lag be minimized as much as possible. Given that our front-end frameworks of AngularJS and Bootstrap have proven to be fast and reliable, we expect our webpage to take no more than 5 seconds to load completely on a decent internet connection (about 25 Mbps). While most web pages have shorter load times than this, we expect that the embedded GIS (geographic information system) interface will take a relatively considerable time to load into the web page. In fact, we expect that our main performance bottleneck for this project, and therefore our biggest challenge to overcome, is going to be the GIS framework and how to implement it in a way that maximizes both the framework's performance as well as our project's performance as a whole.

- **GIS Interface:**

  The reason that we believe this aspect of the project to be the most time consuming performance-wise is because of the version of the GIS framework we are implementing. As GIS frameworks are relatively expensive and well-used pieces of software, most GIS frameworks require a monthly payment to be able to utilize them to their fullest extent. The framework we chose, opting to minimize cost for this project, is Leaflet + Nominatim. It can do what we need it to do in our project for free but at the cost of its performance. This GIS framework allows one free call to the GIS servers per second, which is severely limiting to our performance. However, despite its drawbacks, Leaflet + Nominatim is very promising for the functionality of our project, theoretically integrating with our project very well.

- **Database System:**

  Despite the slower speed of the GIS framework, we expect our database framework to be able to execute a fair number of queries (say 100) in under 1 second. Database speed is a very crucial part of our project as the number of queries could range from a handful of client information queries to thousands depending on what the query is for, how common an attribute is among clients, the number of entries that are viable candidates for the search, and the overall number of clients that an user agent might be responsible for just to name a few. This variation in number means that our database system should be able to easily handle requests and queries that are both small and large in number. Additionally, it should be able to be scaled easily to account for multiple users at one time. While our chosen database framework, MongoDB, is already fast; we will be

working to further increase its efficiency while implementing and testing to ensure the maximum performance output.

- **Communication System:**

  Lastly, our communication systems, such as email and text notifications, should also be fast. We're aiming for the time for sending both emails and text messages to be under 10 seconds. This part of our project can be argued to go hand in hand with the GIS system in terms of importance of speed. This is because while the GIS system allows agents to find clients more easily, the communication system speed determines how fast they might receive the information being sent. This makes it one of the top priorities in terms of speed and optimization.

Overall, speed is incredibly important to this project given that one of its primary functions is that it be able to act as an emergency alert system between agents and clients. Its other function of simply being an easy-to-use and reliable form of communication between the two parties also puts an emphasis on speed and performance. We are shooting for total time for one use of our system to be under 1 minute for a simple search and notification being sent, provided the agent knows what they will be sending and the parameters for which they are searching. This also assumes that the agent using the system is well trained in how to use our project, which we also deem an important non-functional requirement.

Training:

Along with the speed and technical performance of our system, we also aim to make our system easy to learn and to use. Having these qualities in mind when implementing our project would not only reduce the overhead of a company's training time for the system but also increase efficiency in how fast a user could use the system as well as how effectively. By making our system user-friendly, we increase the performance of our system even more by combining efficient use with efficient performance.

One of the biggest examples of overhead in a company is the training required to onboard agents that are new to a system so that they may use it effectively. While one may not immediately think of the ease of training as a performance requirement, it is a very important aspect of our software as the time that it takes to learn our system is something that we would like to minimize. One of our goals that we've had with this project is that we make it easy to use on our client's side. This extends to the training aspect of our system where if we succeed in our goal then not only will the system be easy to use, but also easy to learn as well. This will help with the time that it takes to become familiar and eventually master the use of our system. We expect that the most difficult part of our system to learn will be understanding the feature set of the customer search functions. We expect its wide range of attributes to search for clients will be the most complex to learn from a training standpoint.

Given this information, we are aiming to create a very simple system interface and would like it's average learning time to be under 3 hours. This goal is based on the expectation that while the person learning the system may not have complete mastery over said application at the end of the training, they should have a firm grasp on all features and be able to perform the base functions of the system with little to no help.

Documentation:

Another important non-technical requirement for our project is our documentation. Not only is this important for our development as a team, but it is also important for readability from outside developers if they are to quickly integrate our system into their existing infrastructure. Our documentation includes things like the readability of our code and an organized project repository.

- **Code Readability:**

  Readability of code is important in any project, especially if the project is one that many people are working on. The ability for the entire team to understand what a particular part of code does within a few minutes of reading it is imperative for the speed of development. It is also important to our team that we are all on the same page in our development and are all able to to explain most or all parts of our project. While it is true that readability greatly benefits our team with the development of our project, it also ties into one of our functional requirements. By making our code readable, we further achieve our goal of having our application interface well with our clients existing system. When integrating our system with theirs, it is important for code to be readable so that their technicians are able to easily translate our code into something that works better with their system if the need arises. Furthermore, they can modify and update our system if they want to apply an update or modification after implementation. The readability contributes to much of this project and we strive to develop code that can be read by most developers with ease.

- **Repository Organization:**

  Another part of documentation is the state of the project's repository. For a clear and understandable repository, we plan to craft well-commented push requests, pull requests, and such so that the history of development is clear and to make the job of the quality assurance individual easier. It is also imperative that files are saved in places that you would expect them. For example, you wouldn't expect to find a CSS file for our web application's front-end in a folder labeled "Database". This further strengthens organization and intuitive design of our repository so that a member of our team can easily find and edit a file they are looking for with little to no direction, an important quality that helps speed up development. This, again, also benefits our client's as they integrate for the same reasons.

While our functional requirements are of great importance, our performance requirements are also very valuable. We strive to have our project run as efficiently and as quickly as possible as that is a key requirement for our system overall. We will also hold user-friendliness as a high priority to mitigate the inevitable overhead of training agents to use our system in a way that maximizes their effectiveness and to further improve the performance of our system from the user's side. Lastly, we will maintain a clean and readable codebase that is further supported by an organized and well-documented repository for the improvement of development and documentation of our project as a whole.

4.3 Environmental Requirements:

Our sponsor has not specified or imposed any “Environmental Requirements” and has been clear that they are mostly expecting us as a team to create our environment as necessary. Although, through the nature of our software expected to be developed using free tools and resources, our choice of GIS framework is limited to cost-free options.

Many GIS frameworks typically involve API calls that are more often than not very expensive, depending on the usage and amount of calls you intend to make. We’ve researched our best free alternatives and have come to a feasible solution that will integrate with our project while maintaining no pricey overhead costs.

5.0 Potential Risks

When developing software there are many risks that can occur. One of the best practices to help combat the amount of risks you will face is to actively think of these potential risks while you are developing. This procedure helps outline key factors that need to be accounted for when developing. Some of the potential risks we have preemptively thought of are described below along with a table (figure 2) that outlines the risk, severity, likelihood, and migration step.

Risk	Severity	Likelihood	Mitigation
Customers Not Receiving Notifications.	**Moderate: We understand the need for a company to deliver the services they have promised to their clients. If the clients don't receive the notifications our service could lose its credibility.**	**Low**	**Required OPT-IN: This allows our solution to track which users should be receiving notifications. Our solution will allow users to OPT-OUT in the future if they would like to.**
Sensitive Information to the wrong user.	**High: Users safety and privacy must be held accountable within our software. This entails making sure personal information does not go to the wrong user.**	**Moderate**	**To assist in preventing the leak of sensitive information our solution will have restrictions to the automated messages and notifications. State Farm agents will also be allowed to modify certain notifications and messages.**
False Alarms and Incorrect Notifications.	**Moderate: False alarms and incorrect notifications can lead to panic and costly mistakes. If this happens our solution will lose credibility and interest from users.**	**Low**	**To help combat this risk, our automated messaging and notification system will be categorized by geographical location. This makes sure that all current OPT-IN users of our service within the given location can receive the same emergency notification and/or message.**
Data Leaks.	**High: Data leaks are becoming more common across the web, therefore data that is protected and private must remain intactly that way.**	**Moderate**	**To assist in protecting our software we will be utilizing security protocols such as using HTTPS, password hashing, and sanitizing database inputs to prevent SQL injection. We understand the reputation of the clients, therefore we have agreed to implement these securities with the intent that State Farm will add, modify, or remove the security features as seen fit.**

Figure 2 - This table outlines the key potential risks for our solution. With the table the risk, severity, likelihood, and mitigation plan are also stated.

- **Customers Not Receiving Notifications:**

  One of the key risks our team has discussed is making sure that customers who are eligible for the service actually receive the notification. We understand the need for a company to deliver the products they have promised to their clients. Therefore, if a user has signed up for the service they should be able to receive our emergency notifications and also separate notifications, such as birthday alerts or changes in policies that are happening. Our solution, will require users to OPT-IN. This will allow us to track and make sure verified users are receiving the notifications.

- **Sensitive Information to the wrong person:**

  Continuing, we understand the risks that come with developing a software that uses sensitive data. This means that we must account for the data and make sure that sensitive information does not go to the wrong person. To avoid conflict with sensitive data our notification and messages will be generic. This helps keep information safe as the only people who will have access to this information will be your State Farm agents.

- **False Alarms and Incorrect Notifications:**

  Next, another potential risk we have discussed is if our system starts sending out false alarm notifications, or the incorrect notifications to the clients. For example, it could be detrimental if a client receives a notification to evacuate the location, yet there is no emergency in that area. This could create panic and result in a costly mistake. Similarly, it is just as negative for clients to receive incorrect notification, such as the wrong emergency message. The action measures taken for fire safety are different from a hurricane alert, therefore the notification must be accurately represented.

- **Data Leaks:**

  Continuing, because our system will be using sensitive data, it is prevalent our entire system is secure. Data leaks are becoming more common due to simple bugs and the increasing amount of malware on the web. Having a secure system and rigorous testing will assist in combating these issues.

6.0 Project Plan

Our risks, along with other unforeseen issues, is something that we would like to mitigate as much as possible. One of the best ways to do this is by having a solid, detailed, and efficient project plan. This will help us stay on track, think about what problems we have to tackle and when, and be able to more easily work around any issues that pop up suddenly.

Moving forward, our number one priority is to begin working on a prototype and further individually test our technologies to ensure that our current plan is feasible, and will cause no

issues in the foreseeable future. We will begin by implementing a basic GIS web application that will serve as the foundation for the rest of our project. From there, we will make sure that we can modify and extract the specific data that will be required later on, such as outlining client subsets on the map. Our client has requested that our program is compatible with their current database; Testing of our chosen database, with mock data provided by our client, will take place to ensure that we can begin implementing relevant features without worrying about compatibility issues. Our web application should be sleek and easy to navigate, while also providing fast response times. The front end will need to be optimized for performance and ease of use in mind, but we don't expect to have many issues in this regard considering that we are using lightweight and efficient packages / libraries in our project. As seen in the provided gantt chart (*Figure 3)*, looking further down the line, once most basic functions are well integrated, we will be implementing our communication systems for notification purposes.

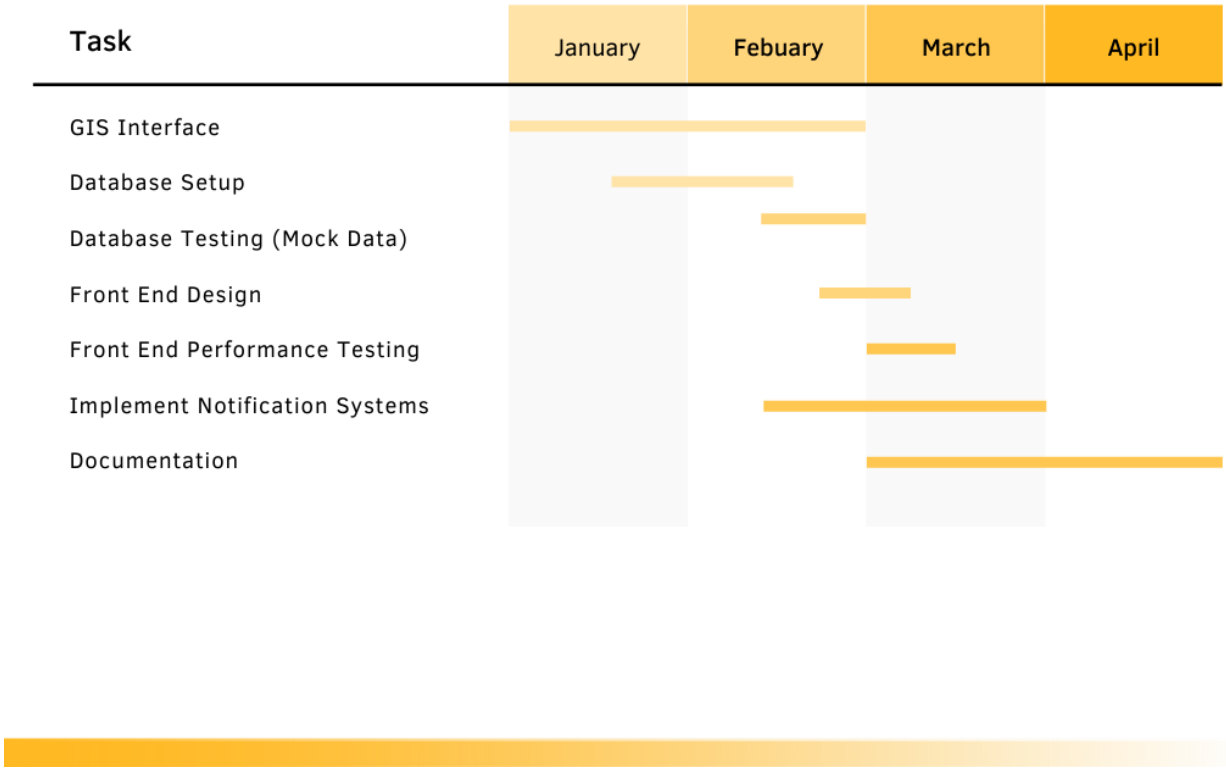

Figure 3 - This is our current schedule. We have outlined the most important parts of our implementation and plan to develop them in this logical order.

7.0 Conclusion

Connecting with one's customers is a very important part of any business to make the customer feel valued. While this is true for most if not all businesses, it is especially true in our client's, State Farm insurance, business model. State Farm emphasizes heavily their business to client relationship as their line of business is very much dependent on making their customers feel safe and protected. This is further supported by their tagline: "Like a good neighbor, State Farm is there." which shows how they wish their customers to view their company. As such, a good relationship with a customer hinges on an exceptional communication system. For someone to be perceived as a good and helpful neighbor, they need to communicate as one. This means communication should be quick, easy, and helpful.

Currently State Farm's communication system is less efficient and functional than it could be, lacking some features and tools that would make agent-client communication much more effective. This has led to our project; an alert system for agents that allows them to quickly search clients by a number of attributes and offering an interactive map tool that allows agents to quickly and easily select clients by location. This tool will help agents become more effective communicators with clients. Not only does it allow for day to day communications to clients about important notices and policy changes via email and text, but it also allows agents to quickly and easily warn clients of rapidly developing emergencies and other dangers they may not be aware of due to a lack of an early warning system. This takes communication to the next level and would really further elevate State Farm to being the "good neighbor" that it strives to be.

To achieve the functionality that we want out of this project, we have outlined a few requirements that will help us reach the project goals and help keep track of our progress. Some of the most notable of these requirements are the ability to search for clients based on a geographic interface and the integration with our client's existing infrastructure. We also recognize that with this project comes some risks that we'll have to manage such as data leaks and incorrect notifications being sent out. We will actively seek to mitigate these risks along the way and have tried to identify as many as we can now in order to prevent them causing issues in the future.

Overall, we see our project as one that has the potential to make the lives of both customers as well as State Farm's agents much easier by providing an incredibly useful interface for communication. Whether it's an emergency that erupts without warning or just a friendly reminder about when a customer's monthly payment is due, our goal is to give the agents of State Farm Insurance the tools they need to be the good neighbor that they are.