Red Alert

NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing, and Cyber Systems

Team: Sal Galan • Calvin Harper • Nick Nannen • Myles Dailey
Sponsor: State Farm
Clients: Glenn Austin • Hans Yeazel

Team Mentor: Han Peng

Team Website:

https://ceias.nau.edu/capstone/projects/CS/2022/RedAlert/

What is the Problem?

Problem Summary

Imagine a scenario where a State Farm agent living in Superior, Arizona witnesses a lightning strike ignite a brush fire just outside their neighborhood. Without a project like ours, the agent would not have a way to view their clients on a map to determine if they are in harm's way. With the Red Alert web app, this agent was able to open the Red Alert website, select only clients that are located near the fire, and send them a text message alerting them of incoming danger in just a few minutes.

Specific Problems

- State Farm Agents do not have a tool to easily visualize client locations.
- No tool to send specific notifications based on geographic location.
- In the event of natural disaster, agents cannot quickly determine if their clients are in harm's way.

Solution Overview

Solution:

Red Alert is a web-based, agent-to-customer notification management portal. Red Alert can send messages to clients instantly, on specific dates, or at recurring intervals. Notifications can be sent as SMS, email, or both. Most importantly, agents can granularly select client subsets from the map by drawing any shape around the clients they want to select, making notifications extremely specific to a client's address.
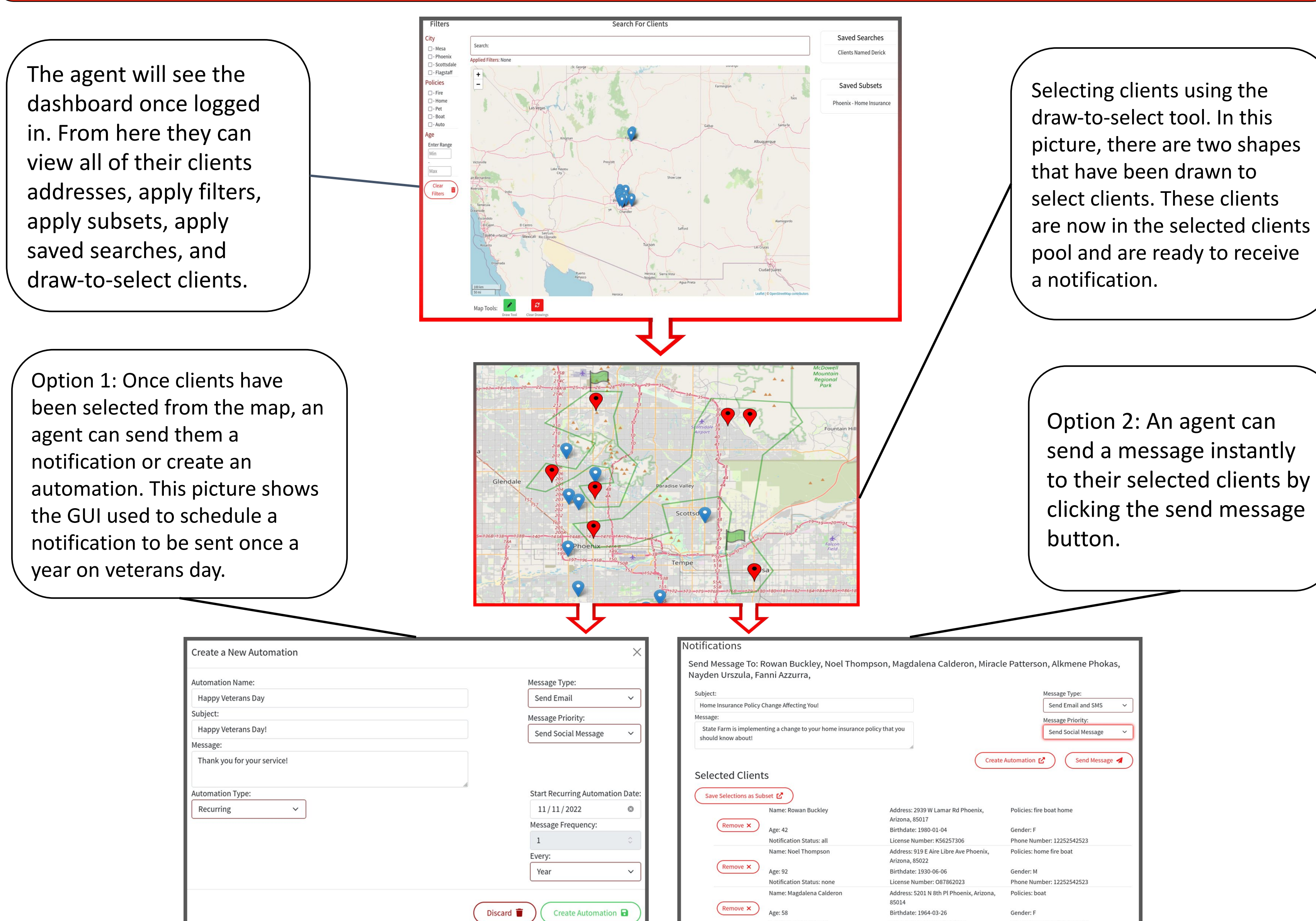
Challenges Encountered:

Draw-to-Select Clients

- How to calculate if an address is within a custom shape? Utilize longitude and latitude of client address and pretend that the map of the world is a huge coordinate grid.
- Utilized a library named point-in-polygon that can calculate if a point is within an n-sided polygon.

Sending SMS

- Sending large volumes of SMS requires an external API that is cheap and effective. The Twilio SMS API is a robust production solution that is also easy to use for development environments.

How it works

The agent will see the dashboard once logged in. From here they can view all of their clients addresses, apply filters, apply subsets, apply saved searches, and draw-to-select clients.

Selecting clients using the draw-to-select tool. In this picture, there are two shapes that have been drawn to select clients. These clients are now in the selected clients pool and are ready to receive a notification.

Option 1: Once clients have been selected from the map, an agent can send them a notification or create an automation. This picture shows the GUI used to schedule a notification to be sent once a year on veterans day.

Option 2: An agent can send a message instantly to their selected clients by clicking the send message button.


Architecture

Feature Highlights

- **Draw to select clients on a map**
  - Draw any n-sided polygon/shape to select clients.
  - Use existing JavaScript library to calculate client addresses within n-sided polygon.
- **Automate notifications**
  - Send notifications on a specific date.
  - Send notifications on a recurring interval.
  - Uses the APScheduler library to send notifications at a later date.
- **Instantly send clients SMS or email notifications**
  - Twilio API used to send SMS.
  - Google SMTP server to send email.
- **Save client subsets and searches**
  - Save clients into a group. Select groups of clients in a single click.
  - Save useful search queries to avoid needing to retype them.

Future Works

Our future work includes preparing our web application to be used in a production environment. Currently, all client data available on the site is mock data created by the development team. In a production environment we would need to configure our application to fetch client data from State Farm servers to be used by agents. This will involve ensuring that agents can only view clients that are associated with the specific agent to maintain client confidentiality. Besides this, our web app is configured to fetch client data from a local database, so we would need to configure our system to properly format data imported from a remote State Farm database to be used in the frontend.
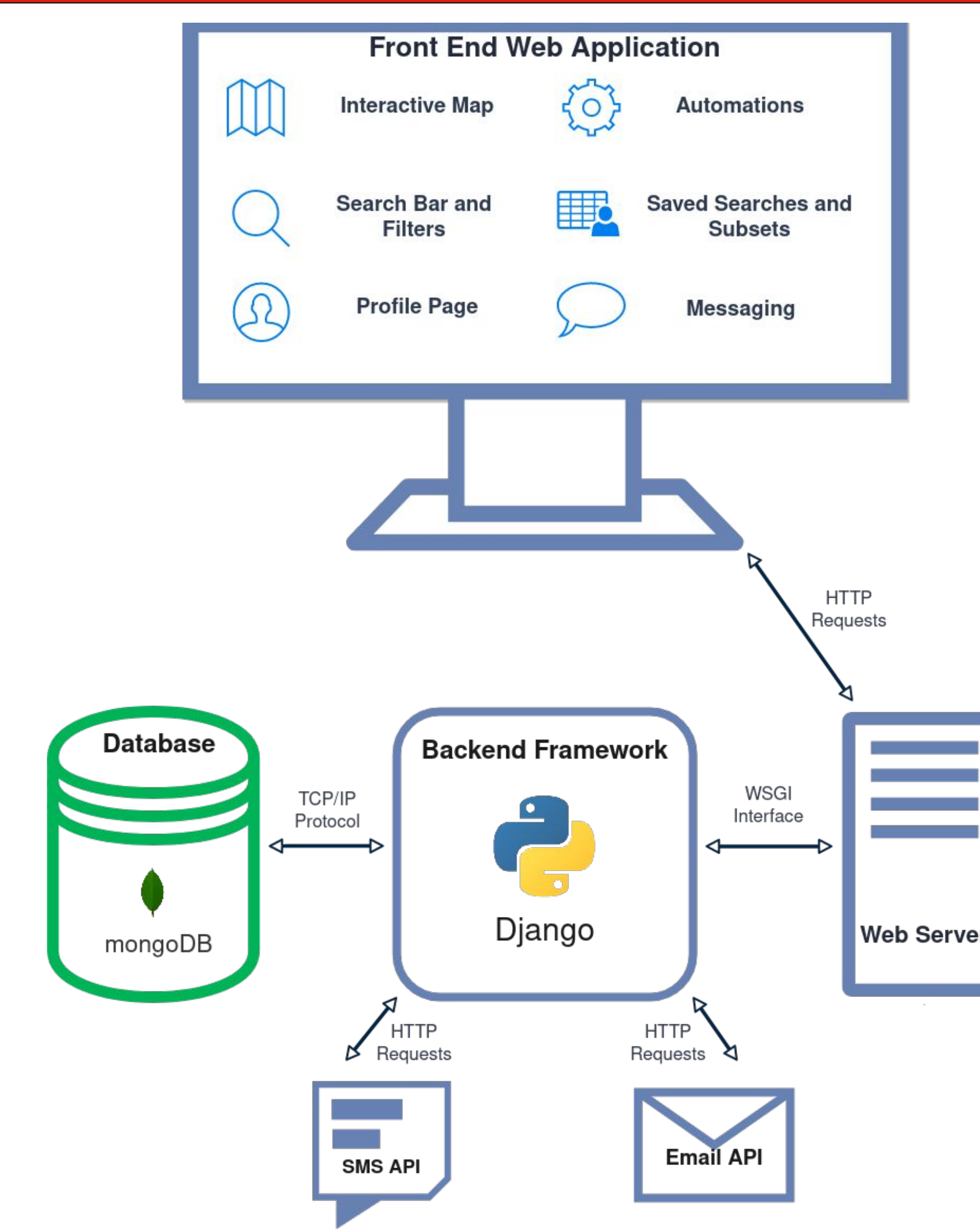
Technologies

Front-end:

Back-end: