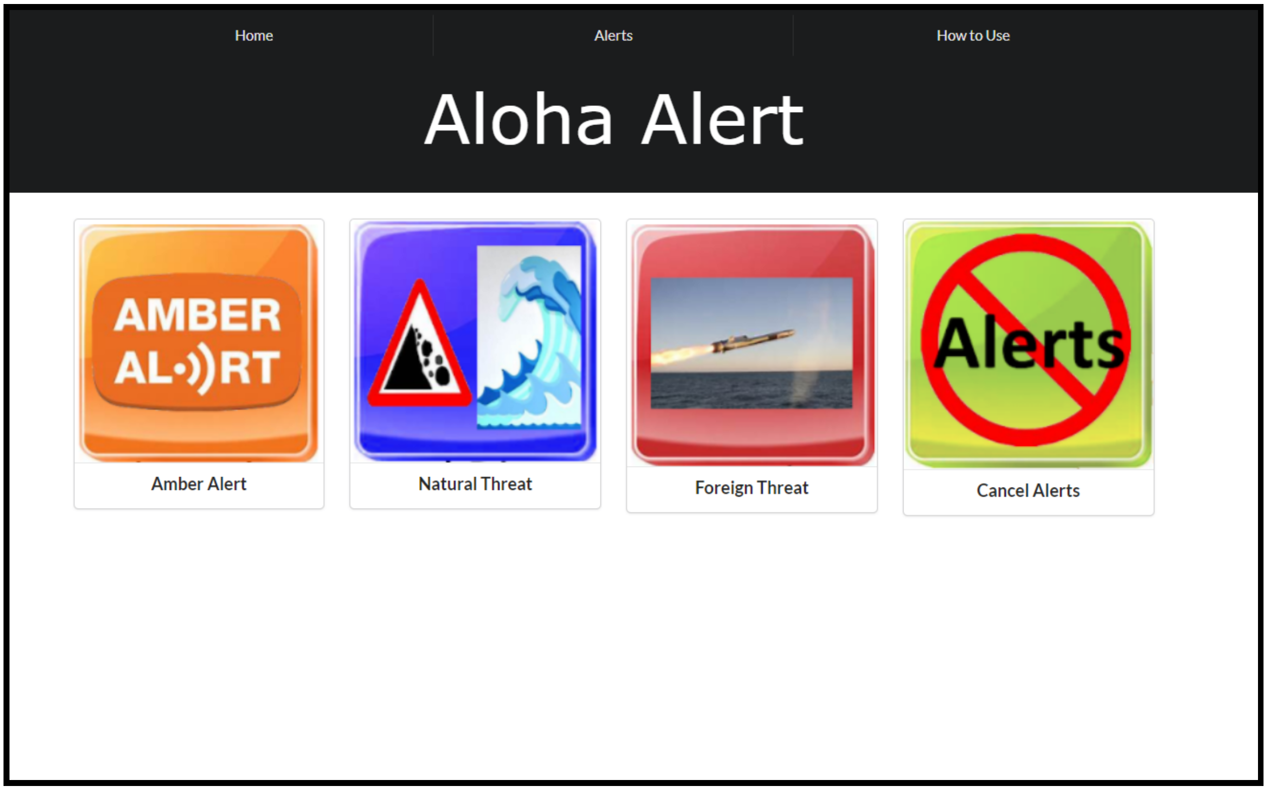
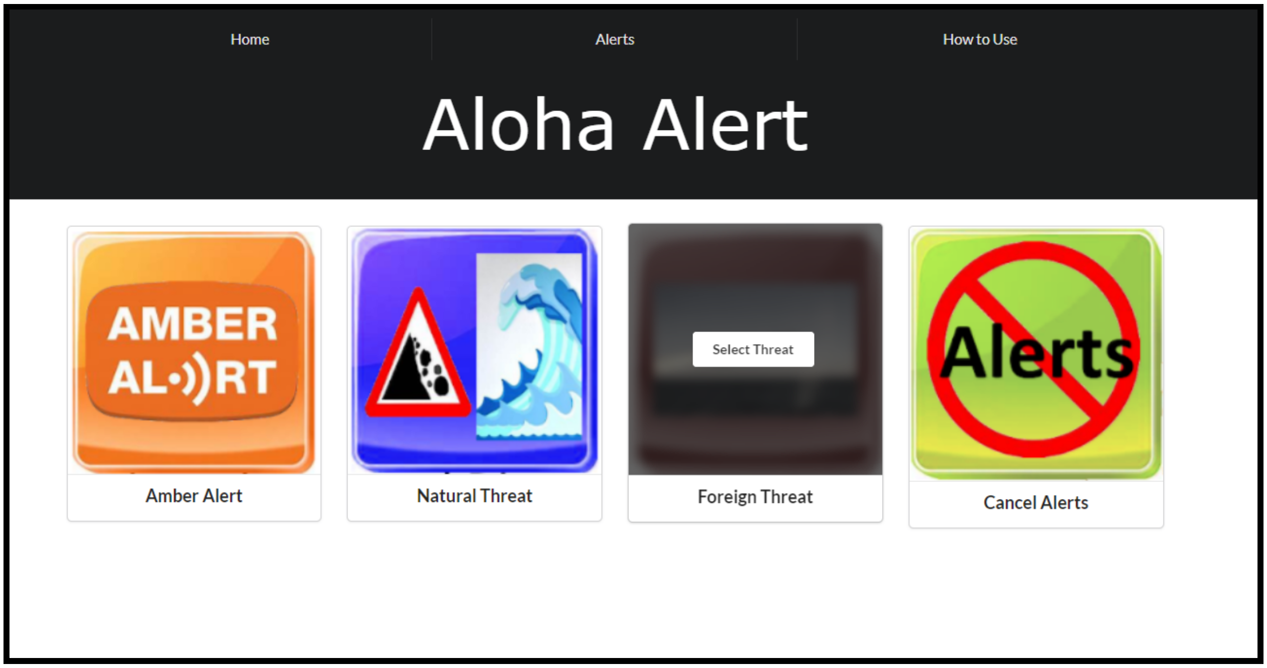
Project: Aloha Alert
Members: Sean Takafuji and Jason Len
Primary Language: Python
GitHub Link: https://github.com/ics414-buffalo
Estimated Lines of Code: 5000 LOC (not including lines in Libraries and Frameworks being utilized)
Reflections from Prototyping:

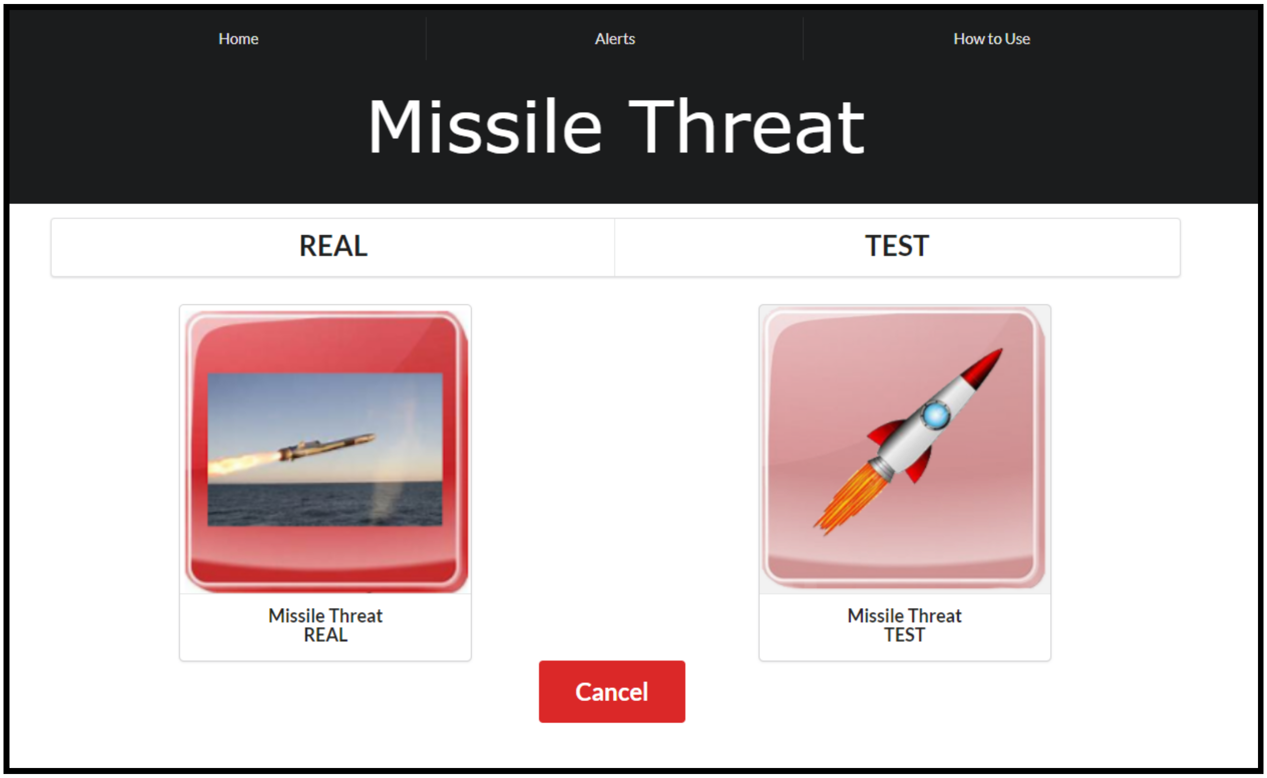
In doing the paper pro-typing, we received a lot of feedback, as described in the write up for that assignment. In talking things over as a group and with Vincent, we have come up with a list of ideas that we are considering:

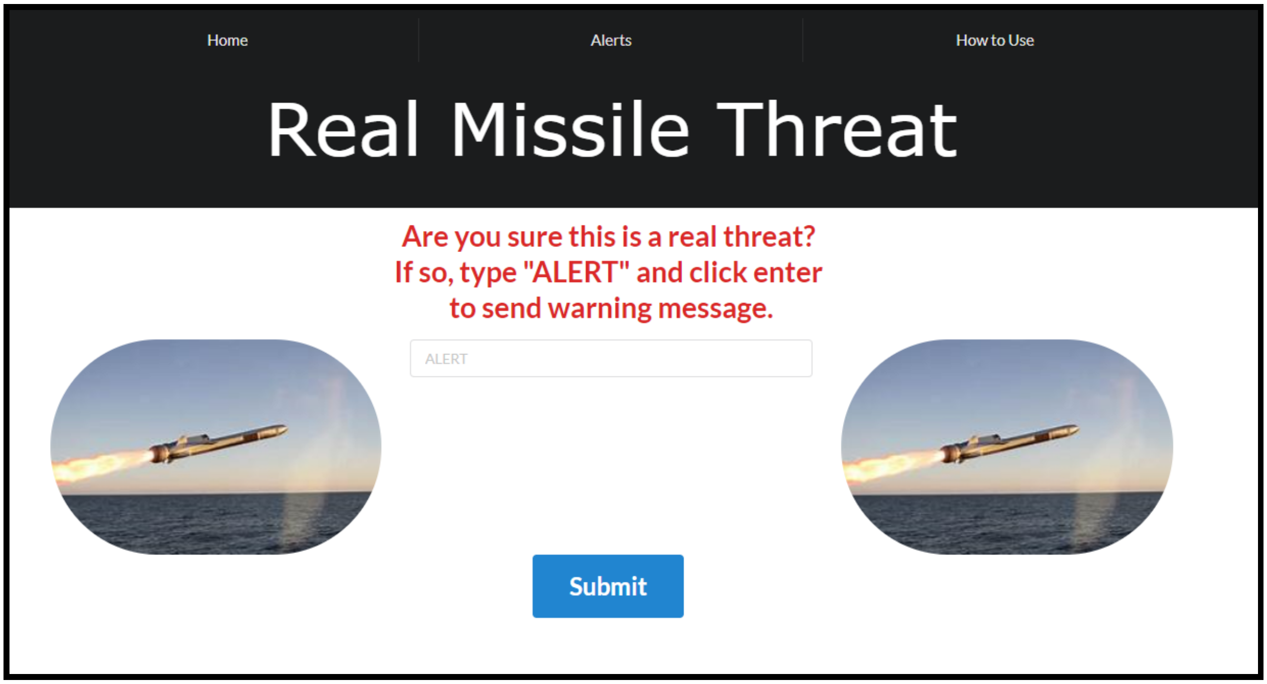
- Changing the color scheme test button
- Changing the picture of the test button
- Changing the picture real missile attack button
- Add in a confirmation page for after cancellation
- Adding in a "user input" text message to be sent on demand
- Implementation of a second user for confirmation - sent to a different interface
- Implementation of List Current Alerts Screen
- Randomization of typed word to send to improve user focus when sending a message
- Video recording activation when system is activated
- I/O Implementation of the various devices
  - Data Messaging
  - Video Recordings

Screen Captures: (at least 2)




Home
Alerts
How to Use
Aloha Alert
AMBER
AL·))RT
Amber Alert
Natural Threat
Select Threat
Foreign Threat
Alerts
Cancel Alerts

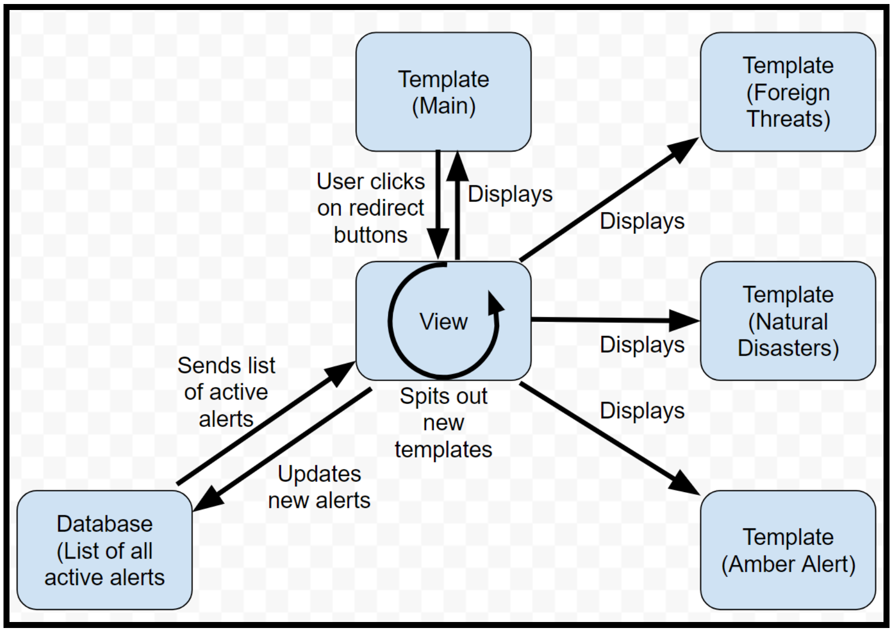

Home
Alerts
How to Use
Missile Threat
REAL
TEST
Missile Threat
REAL
Missile Threat
TEST
Cancel



Real Missile Threat

Are you sure this is a real threat?
If so, type "ALERT" and click enter
to send warning message.

Architecture Design:


Template (Main)
Template (Foreign Threats)
User clicks on redirect buttons
Displays
Displays
View
Template (Natural Disasters)
Displays
Sends list of active alerts
Spits out new templates
Displays
Updates new alerts
Database (List of all active alerts
Template (Amber Alert)

Architecture Description:

The Templates are the various web pages of the application and serve as the Front-End component that the user will interact with. The View routes the user to the appropriate Templates depending on the type of the alert they are trying to send out. The general hierarchy of the templates is

- The Main Screen
    - Alert Category 1 (Amber Alerts)
        - Real Alert
            - Confirmation Screen
                - Message Sent Screen
        - Test Alert
            - Confirmation Screen
                - Message Sent Screen
    - Alert Category 2 (Natural Disasters)
    - ….
    - Cancel Alerts
        - List of Alerts with appropriate Cancel Buttons
            - Message confirming cancellation

At any point in time, the user may return 1 page back or jump immediately to the home screen.
Cancellation of **any** alert is available via the Cancel Alerts pathway.
The database contains the list of active alerts with the respective warning types that were selected upon creation.
The database also contains a history of **all** alerts that were ever sent on the system to allow for review.

Implementation Language/Platform:

Our system is called Aloha Alert and is a web application built on the Django Web Framework.
Django is a Python framework that utilizes a Model-View-Template system.
The Model will be built using SQLite to hold the list of all active alerts that the system is handling
The View (what is traditionally known as the Controller in an MVC framework) is built using Python to control the routing and the logic behind the application.
The Template (what is traditionally known as the View in an MVC framework) is built using HTML and CSS with several components provided by the Front-End Framework Semantic UI.

The implementation is well-suited for this application as the alert options are all structurally similar to each other. Given proper modularity on our part, it is relatively simple to add new types of alerts and to break down various alerts into their own category. Choosing various warning systems is also very simple as they can be added in with a simple refactor of a line of code.