Utility Manager

Requirements and Design Document

Miles Crabbe
Ethan Downey
Chad Easter

Table of Contents

v. Ask User for Outage Status
vi. Broadcast Message
vii. Leaderboard Generation
viii. Settings

Change Log

- **Added Class Diagram** - Chad
- **Added Activity Diagrams** - Chad
- **Added Use Case Diagram** - Chad
- **Reorganized Use Cases into Individual Cases** - Miles
- **Added Functional and Non-Functional Requirements** - Ethan
- **Added Diagram Descriptions** - Ethan, Chad, Miles
- **Added Purpose** - Miles
- **Added Scope** - Miles
- **Added Goals** - Miles
- **Added Project Description** - Chad
- **Added Changelog** - Miles, Chad, Ethan
- **Added Design Diagrams** - Miles
- **Added Design Diagram Descriptions** - Chad

Introduction

Purpose: There are currently no universal, well-made Utility Outage Managers available to the public. This app is meant to be a free service to the public that allows them to alert other members of the public and their service providers of outages affecting them and their community. It is also meant for service providers to help gauge the nature of an affected area in hopes that these outages can be resolved more quickly.

Scope: Initially the scope of the app will be limited to the general area of Tuscaloosa. However, server-based data coupled with crowd sourced updates will allow the scope of the app to grow to a regional or national size. The primary utilities that will be monitored are water, natural gas, cable, power, internet, and telephone. Users will also be able to search reports based on providers of a given utility.

Goals:

- Provide an efficient and effective way for Users to alert the necessary service providers about outages.
- Provide a way for Companies to visualize an outage in order to better determine the location of the problem.
- Provide a way for Users to provide information like response time and efficiency of service providers to other Users.
- Create a user experience that is both enjoyable and rewarding to the user in order to foster more crowd sourced data and interaction.

Project Description

This project is intended to be a multi-faceted tool for utility services by helping customers and service providers alike. The app will be capable of storing crowd sourced data pertaining to outages of various types of utilities. This data can then be displayed on a map as a collection showing the affected areas.

This will be beneficial to customers to easily determine the scale of the outage and the speed at which it is being resolved, and it will also be beneficial to companies by displaying the information geographically allowing easier diagnosis of node failure, as well as helping the company directly help individual customers.

As a side benefit to storing data in this way, other data may be obtained about the outages on a more generalized note. This can include information such as a specific company's average turnaround for outages or even frequency of outages. This data could then be displayed for users to browse and compare when trying to decide which company they would like to receive services from.

Functional and Non-Functional Features

Functional Features

- Utility Tracker will give users the ability to report outages.
- Utility Tracker will allow users to see reported outages on a map interface.
- Utility Tracker will allow users to modify outage reports that they have previous created.
- Utility Tracker will be compatible with multiple types of utilities, including but not limited to: power, water, gas, cellular, and internet.
- Utility Tracker will allow users to see multiple layers of information on a single map, such as seeing power and internet outages at the same time.

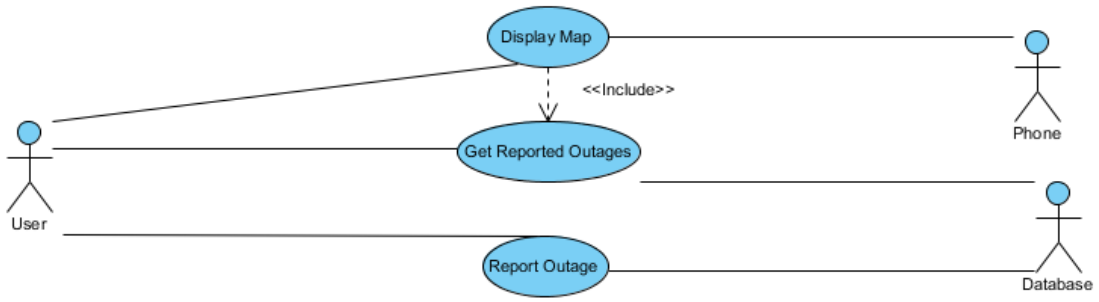
Non-Functional Features

- Utility Tracker will allow companies access to elevated privileges through “company accounts.”
- Utility Tracker will allow companies to modify outage reports in which they are tagged as the provider of the utility.
- Utility Tracker will allow easy visualization of outage statistics such as response time, percentage uptime, etc.
- Utility Tracker will allow users to see a history of outages within a certain time period.
- Utility Tracker will notify users of planned outages in their area that may affect them.
- Utility Tracker will blacklist users that abuse outage reporting.

Requirements Diagrams

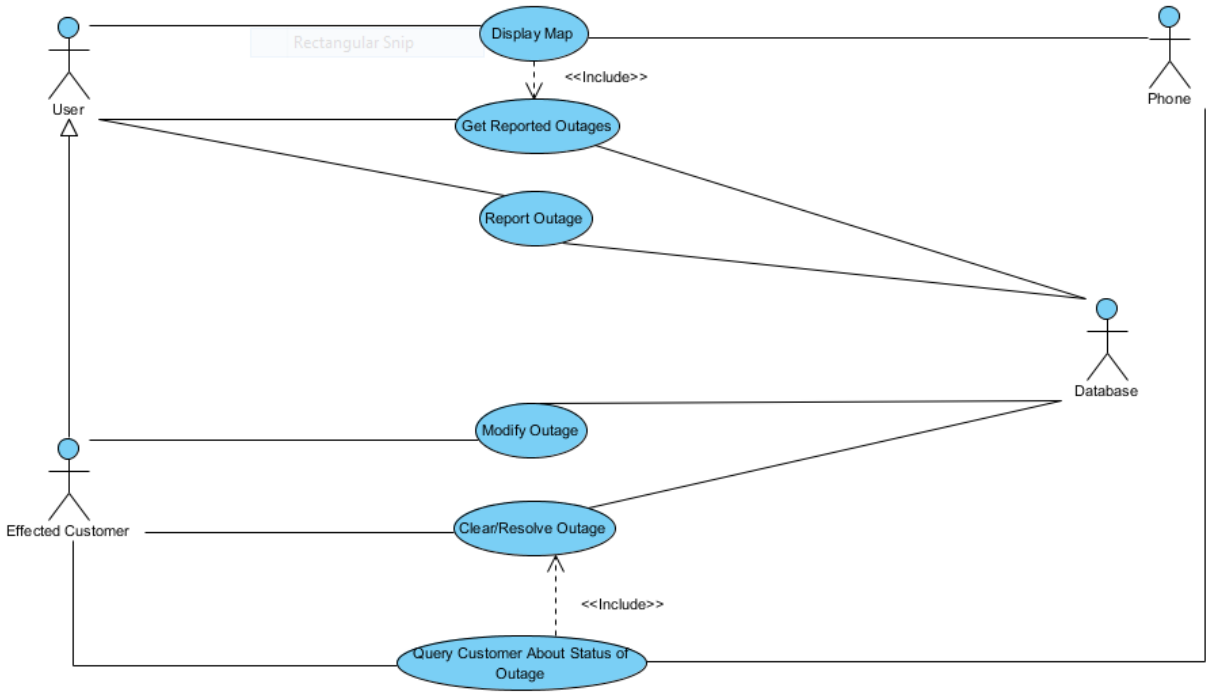
Use Cases

Basic User Use Case



- Basic Functionality for user will consist of displaying a map of outages, a list of outages in non map form, and the ability to report outages
- Displaying the map also calls the get outages process (if data is not already up to date on phone), so those uses are intertwined
- Retrieving reports will be filterable by several parameters: Provider, Time, Type, and lastly reports filed by the specific user
- Reporting outages will be a simple posting of a record to a server, all sorting will be handled server side and by the phone when receiving new reports

Outage Affected User

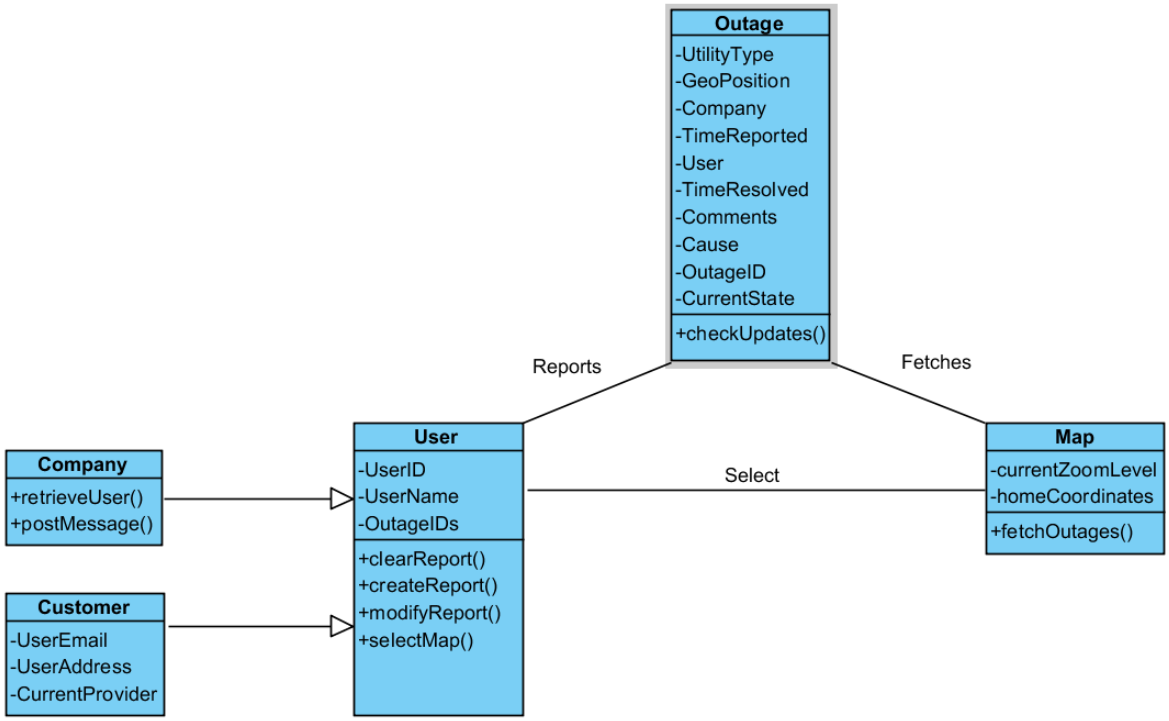


- Affected User has same abilities as a normal user but with additional functionality.
- Modify Outage allows an Affected user to modify comments or other updatable information He/She deems is worthy of being updated.
- Clear/Resolving an outage allows the Affected user to remove erroneous reports they submitted or mark a report as resolved.
- In conjunction with resolving the report, the main form of communication will be the phone automatically polling the user periodically (2 hours -> multiple days depending on type of outage) for updates about the outage.

Company User

- A Company user would have similar access as the Affected User but with less automatic reporting features.
- Major difference from other users is the Message Function
- Message function allow company to email or in-app message Affected Users of a particular outage
- Message function sends messages to server which client app requests on start up
- A Company User's app will not poll the user about resolving an outage
- Instead the Company User will have more options regarding how the report is handled when the User selects a report from a list

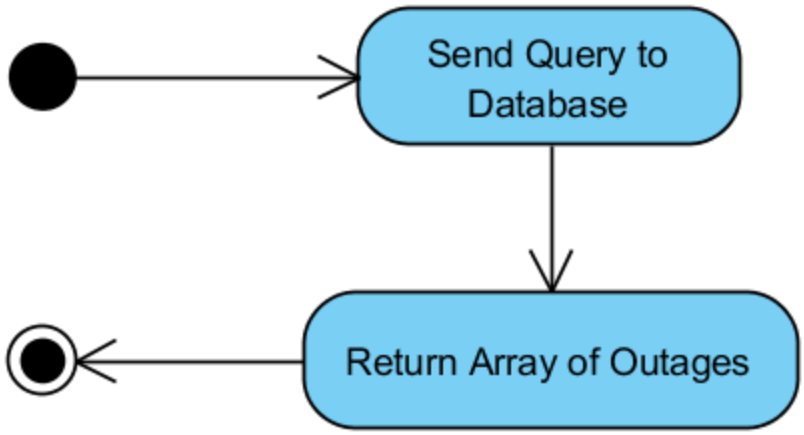
Class Diagram



- The Outage class contains all the data for a reported outage. It will represent an outage stored in the database.
- The Map class will render the outages as pins on a map using Google Maps API.
- The User class stores all user information that is shared between the Company and Customer classes.
- The subclass Customer adds a few fields that individual customers need to report outages.
- The subclass Company adds a few methods that allow a Company to manage Outages reported by its customers.

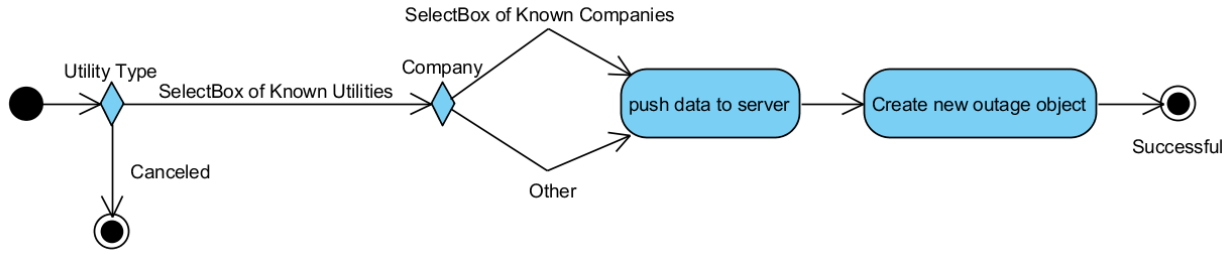
Activity Diagrams

Get Outages



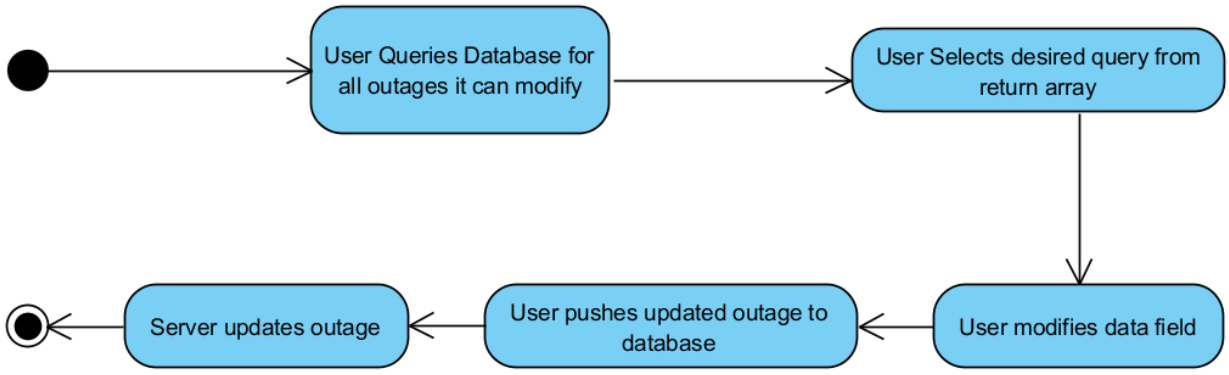
The activity “Get Outages” will support both both common “Users”, “Company users”, and “Affected Users” by allowing each of them to query a specified set of recorded outages allowing them to show only outages pertaining to them.

Create Outage



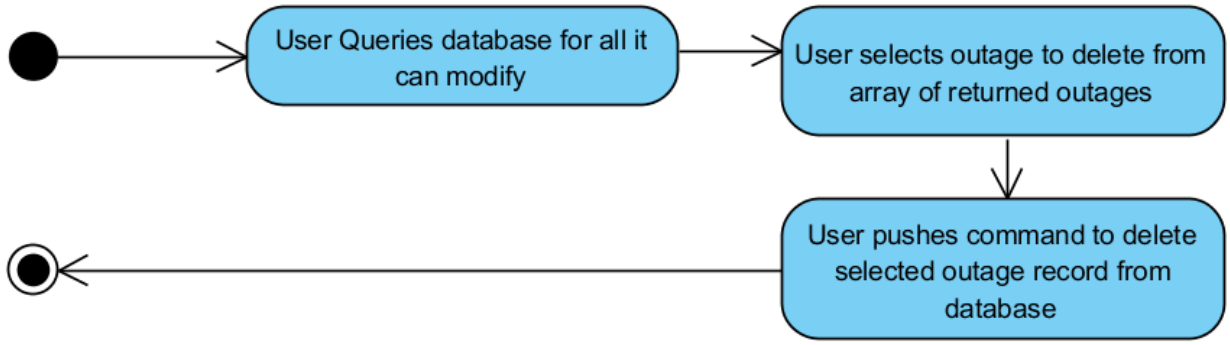
The user will be able to create a new outage by following a simple UI. It will first ask for the type of utility, and then will ask for the company that provides their utility, which will be automatically populated by the user's address and the utility type selected in the previous question. If the company is not listed, they may select "Other" which will allow them to type in a new company that will then become available to other users who report in the future. The phone will then take that data, and ship it off to the server for storage.

Modify Outage



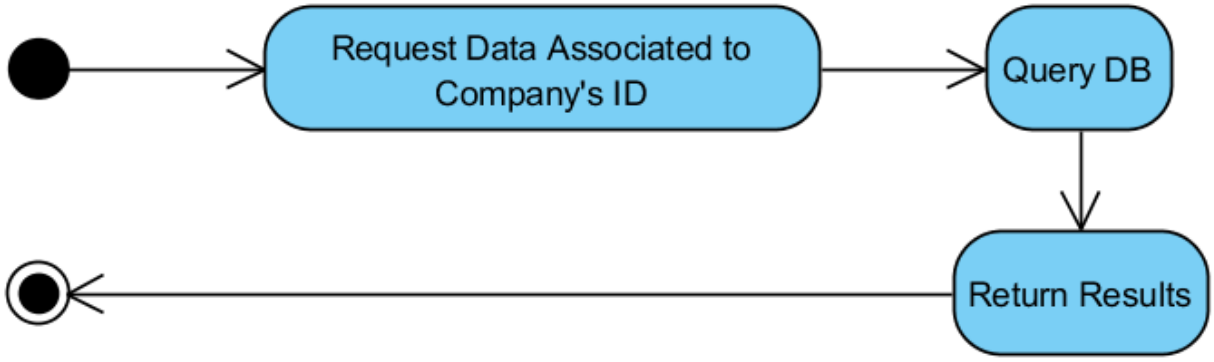
The user will only be able to modify outages that it has permissions to. This will be limited to outages created by that user, or if the user is “Company User”, they will be able to modify all outages associated to their company. The user will first query for outages that it has permission to modify, then after receiving those outages they may select one to modify. They may then modify appropriately, then send the outage back to the server. The server will then update the existing outage to match the data given by the user.

Delete Outage



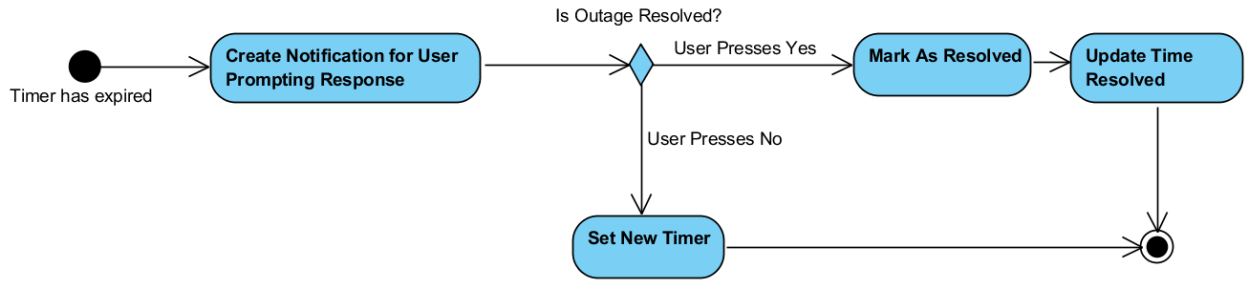
To delete an outage, the user may query the database for outages they own permissions to. They may then select one outage in specific, then push the change out to the server which will delete it from the database.

Get Customer Data for Company



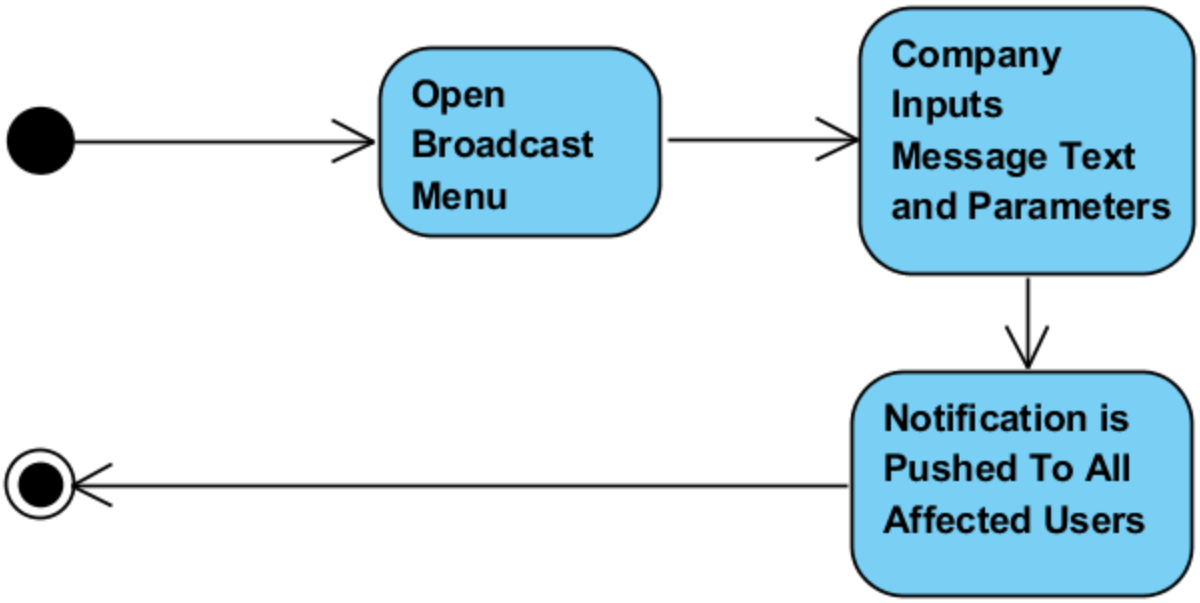
To allow a company to directly communicate with a customer, we will allow a customer's data to be retrieved from an outage. This may be done by requesting the user's data while looking at an outage related to that company. The database will then query for the user that created the outage, and use that information to query for that user's information. The server will then return key information to the phone for the company to use.

Outage Resolved Notification



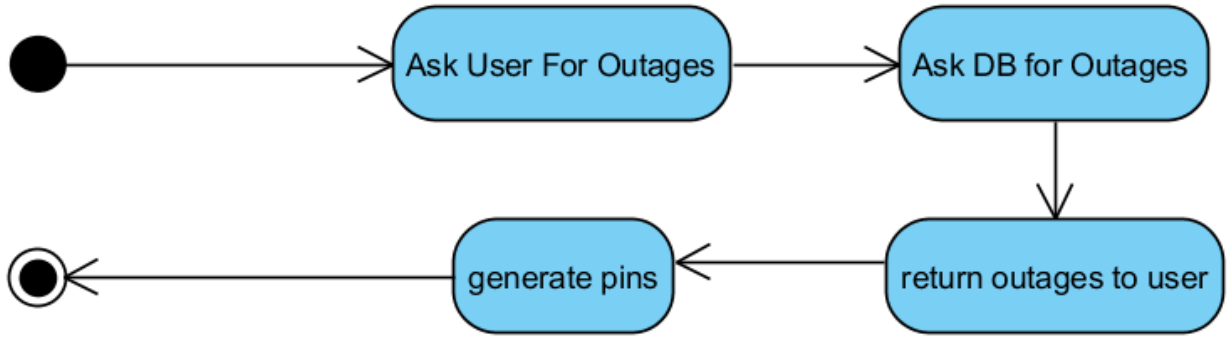
Attempting to maintain an accurate database, the app will set a timer on a reported outage. When the timer expires, it will automatically poll a notification to the user asking a simple “Yes or No” question of, “Is this outage resolved?”. If the User selects “No”, the timer will reset, and ask again once the new timer expires. If the User selects “Yes”, the app will mark the outage as resolved, stamp the outage with a time of resolution, and push these changes to the server.

Broadcast Message



The “Company Accounts” will be capable of pushing out notification to all registered users subscribed to their services. The company will be able to open their “Broadcasting” menu, input some text to push out, then send it to the server which can then push it to the respective users. This will be targeted for helping companies communicate in situations where they have a large number of customers suffering from an outage and would like to alert all of them directly to update their customers on the situation, or to let customers know about a planned outage ahead of time.

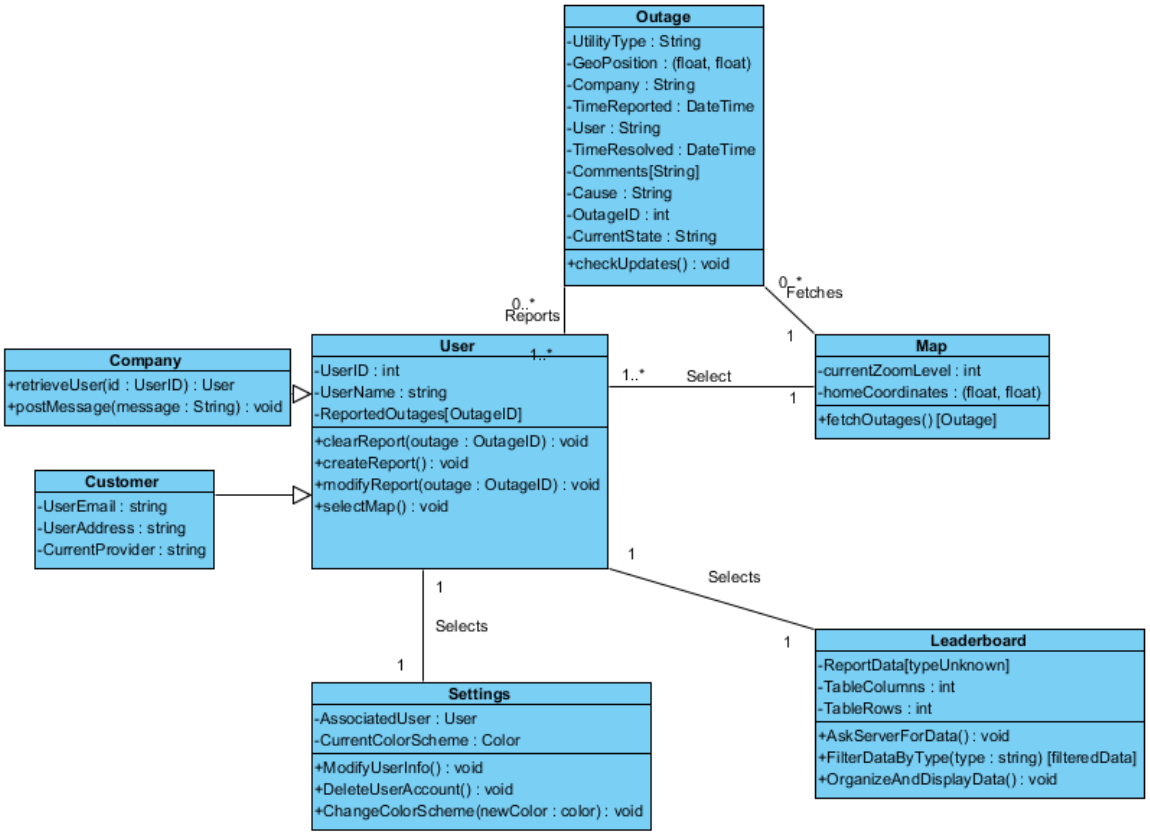
Generate Map of Outages



The main focus of the app is to allow a user to see outages of a specified type in a specified area. This will be done through polling a server for reported outages that pertain to their filters, then displaying them over a map. This will be done by first asking the user what filters to apply. The database will then run a query, return the related outages to the user's phone, then finally generate the outages as pins.

Design Diagrams

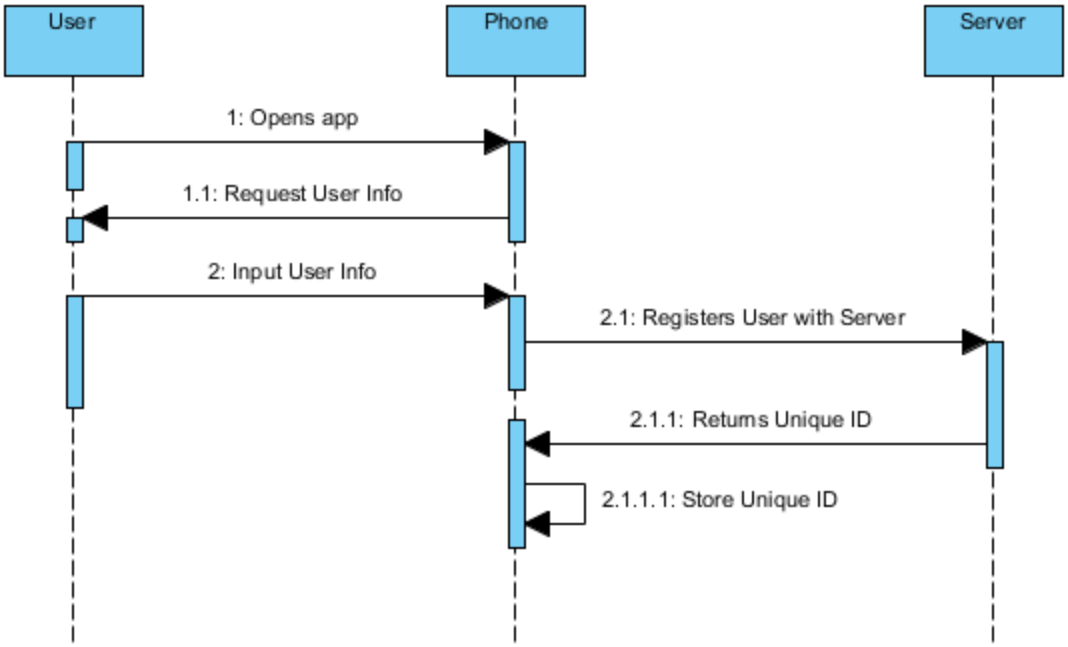
Detailed Class Diagram



Here, the class diagram has been extended and fully developed to match the intended final product's code. The extensions include the addition of a "Settings" class, as well as a "Leaderboard" class. To reflect on the multiplicities, there is only ever one relative map. This map can contain multiple outages, which are created by a single user. Inversely, a user may create multiple outages. And a many users may select one specific map. Additionally, one user may have their own settings, and one custom leaderboard to display.

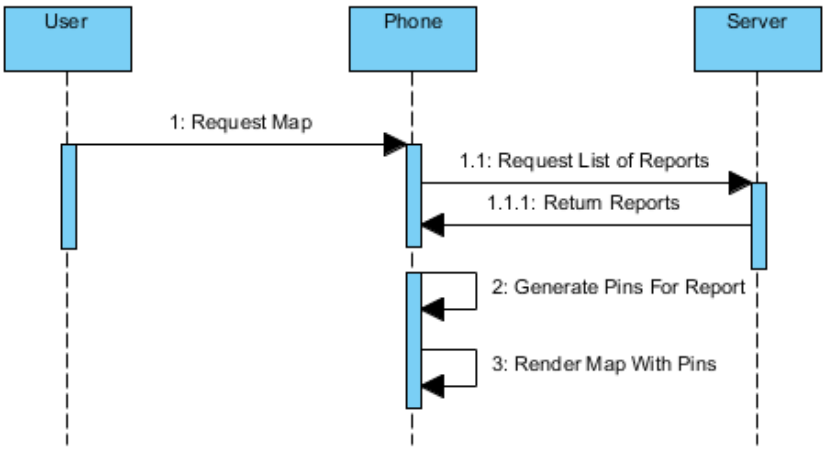
Sequence Diagrams

Initial Load



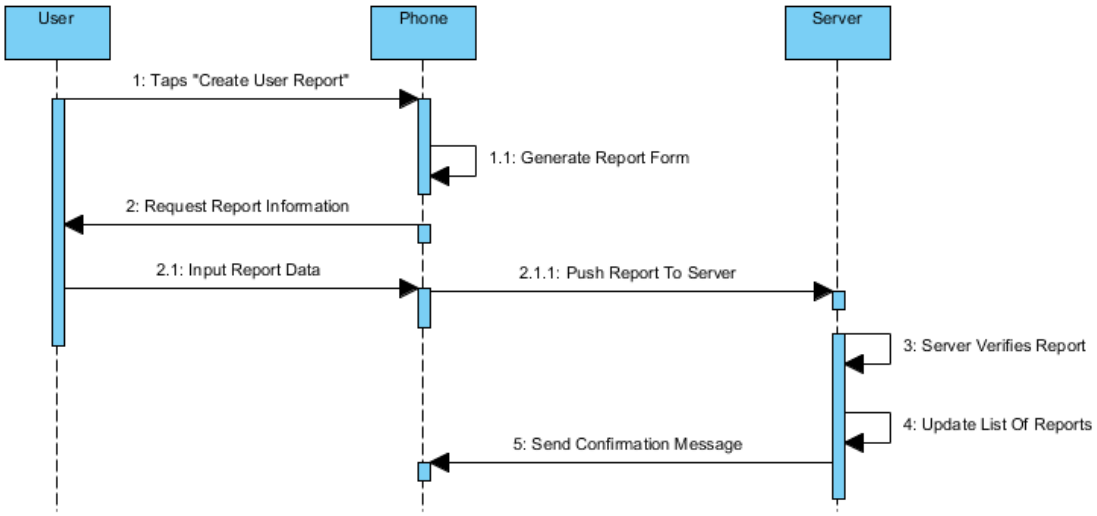
“Initial Load” represents the first time the app is ever opened by on the device. At this point the phone will request information to represent the user in the data. The phone will then register the user with the server, which will return a unique ID that can be stored on the phone. This unique ID is what will link future reports to this user.

Load Map



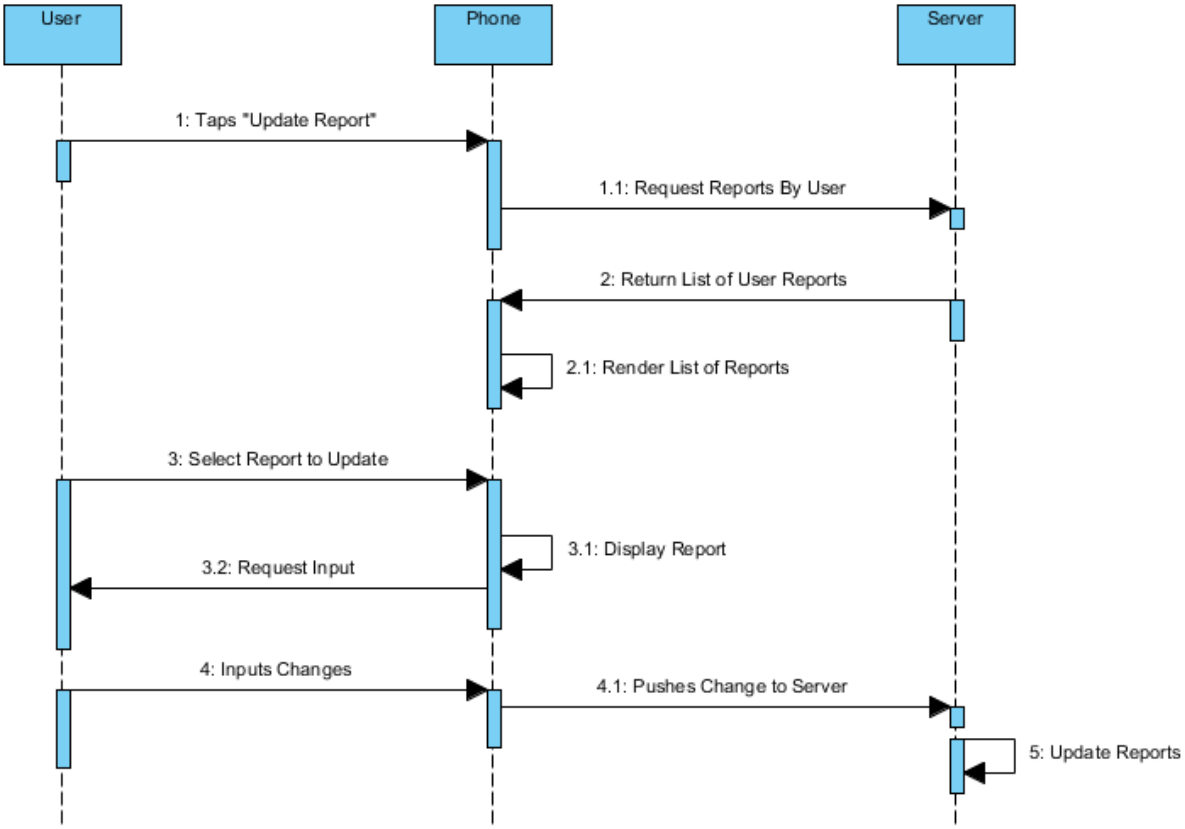
“Load Map” represents the activating of the app, which will then request the server for active outages that apply to the user’s filters. After returning said outages, the phone will then generate pins from the data. Then the Maps API will be initialized along with the overlaid pins.

Send Report



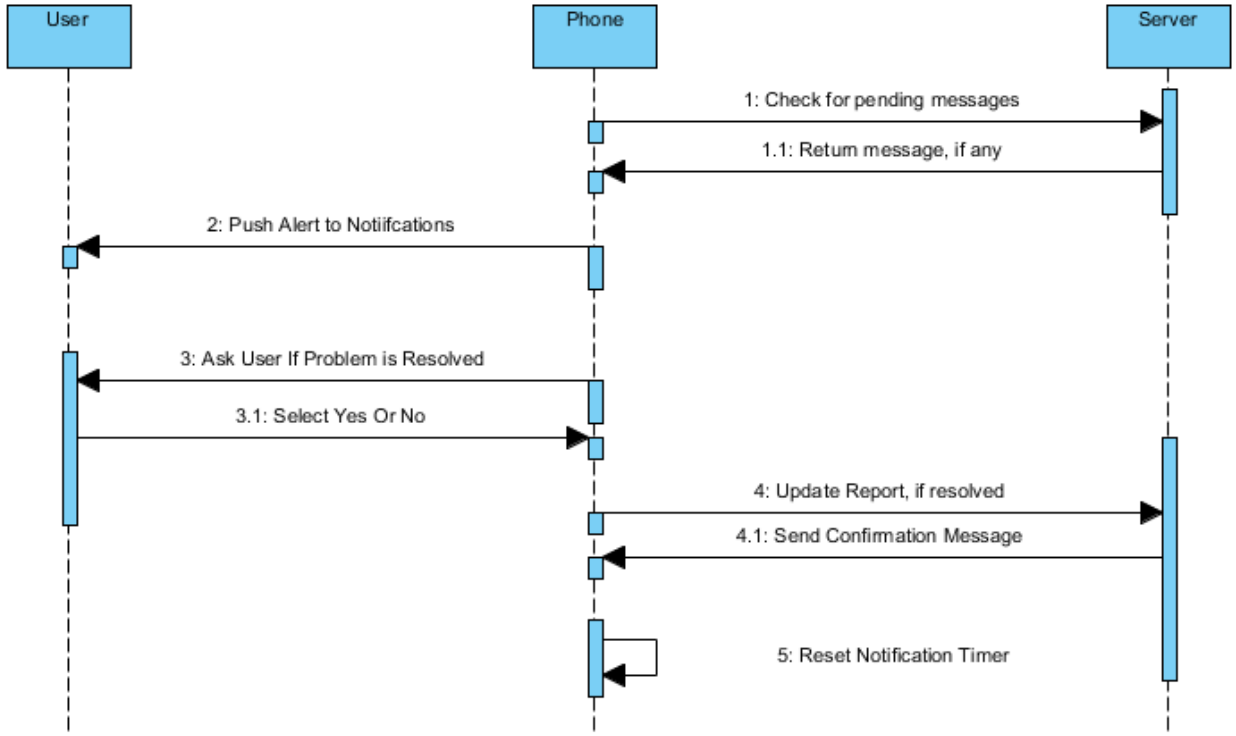
“Send Report” represents the creating of a new report by a user. When a user selects to do so, the device will generate a partially pre-filled form from the user’s personal data that was entered during “First Load”. The user will then fill out all remaining data, and modify any generated data that is not correct or applicable. The device will then push the new outage to the server where it will be validated for legitimacy, then added to the database of known outages. Upon completion, the server will return a confirmation to the phone.

Update Report



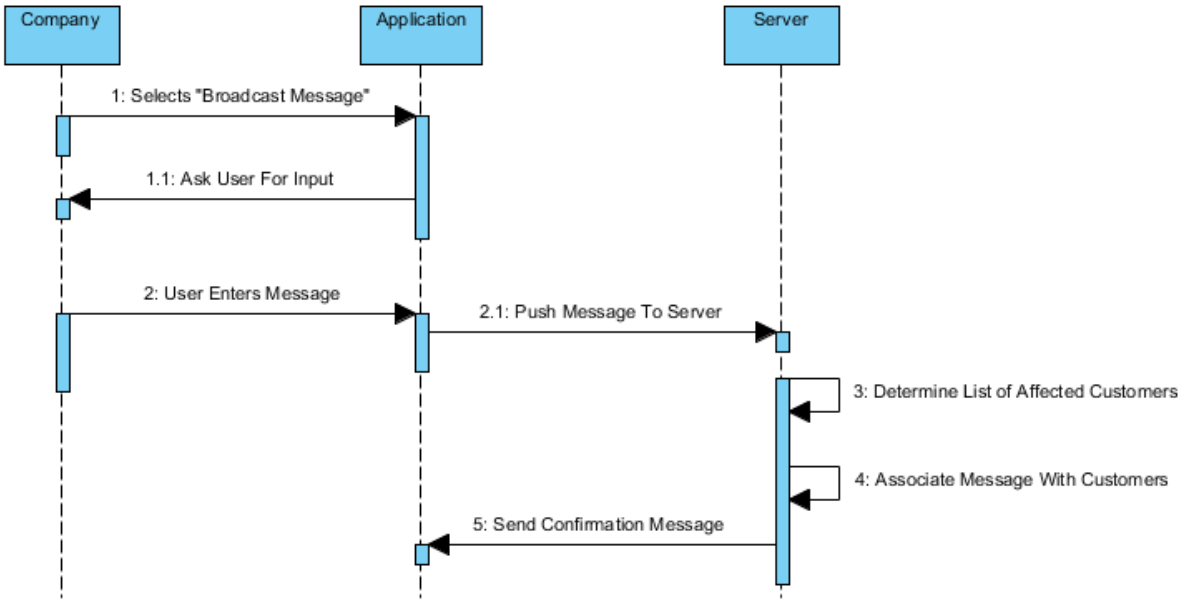
“Update Report” represents the action of a user modifying the existing data of a report already stored on the server. When a user selects to do this, the device will first request a list of all reports created by the user. Upon receiving the list, the device will display the outages available to be updated in a list. Once an outage has been selected, the server will then display, in an editable manner, the existing information related to that outage as received previously. The User then may change information as needed, confirm changes, and push the new data to server which will update its records.

Ask User for Outage Status



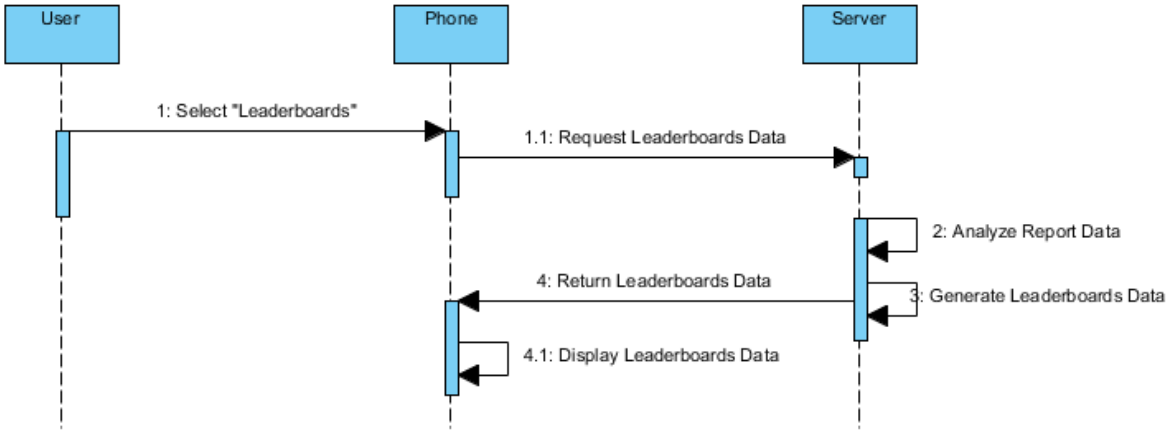
“Ask User for Outage Status” represents the devices ability to check periodically whether a pending outage is resolved, as well as check with the server to see if any messages have been broadcasted from any companies they are subscribed to. This will be automatically called periodically, in which case it will check with the server to see if the user has any pending broadcasted messages. The server will then return the message if any exists. The phone will then alert the user. Then the device will ask the user if any open outages are resolved. If the user selects “Resolved”, the device will then update the report on the database, be returned a confirmation, then reset it’s timer to run this loop again after a period of time.

Broadcast Message



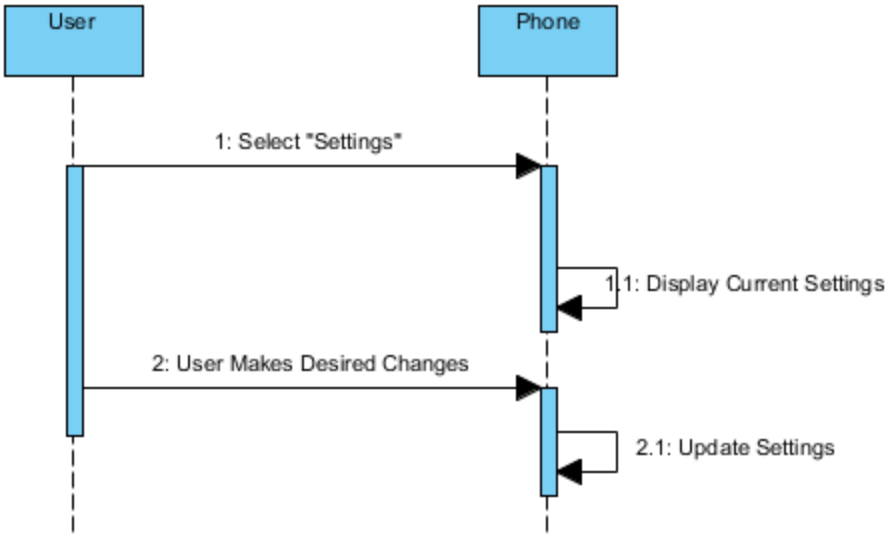
“Broadcast Message” represents a company’s ability to push out a message to their subscribers via the database. Upon requesting the device to do this, the device will request the message to be broadcasted, which will then be pushed to the server. The server will then parse the database to determine which users to assign the message to. It will then update each of these user’s data to direct their account to a given message, which will later be read when the user’s device requests any pending messages.

Leaderboard Generation



“Leaderboard Generation” represents the compiling of potential service providers into a filtered list. When a user selects to view leaderboards, the device will ask the server for relevant data. The server will then analyze the data, generate parsable data from the analysis, and will then send it back to the device which will then be displayed on the device for the user to see.

Settings



“Settings” represents the viewing and modifying of a user’s local settings on the device. This can be done when the user selects “Settings”. The phone will then retrieve current settings from internal storage and display them in an editable manner. The user can then make any changes desired, confirm the changes, and the phone will store the new settings.