OPPORTUNITY

SUPPLIER DELIVERABLE 2A

Jake Runzer	Management plan
	UI Walkthrough
Zev Isert	UML
	Minimal System
	Proof
Claire Champernowne	Executive Summary
	Use Cases
Dylan Golden	Use Cases
	Error Handling

Table of Contents

Glossary

REST	Representational State Transfer, an architecture style for designing networked applications, which relies on a stateless, client-server, cacheable communications protocol.
API	Application Program Interface, specifies how software components should interact.
Opp	Event created by the Opportunity application referring to the meeting of all preconditions for a given Opportunity notification.
Trigger	The condition or conditions that must be satisfied for the Opp to produce a notification.
SSD	Solid State Drive, a persistent data storage device using integrated circuits and designed without moving parts.
UI	User Interface, the set of design choices for an information device of which a human being may interact with.
iOS	iPhone Operating System, a popular mobile operating system designed by Apple Inc.
OAUTH	Open authentication. A system to provide secure authentication to an application using sign in information from the providing system.

Executive summary

This report describes the mobile application, Opportunity. Specifically, the interface, use interaction, functional specifications, minimum deliverable, and plans for implementation and management will be covered.

Opportunity can be used for various functions. Users will utilize the app to help them:

- Remember or plan activities that they want/need to do.
- Fill stretches of free time with appropriate activities
- View and select appropriate local events

Users interact with Opportunity in a variety of ways. In Opportunity, a user can:

- Create an account using email, Twitter, or Google
- Create custom Opps
- Delete or disable previously created Opps

Though the user will only be interacting with the Opportunity app, the system as a whole is comprised of several parts:

- IOS mobile application
- Backend server
- Database
- SQLite website

To be considered minimally acceptable, the following triggers must be available in Opportunity:

- Time
  - Represented by the application's ability to specify time in hours or use generalizations such as sunrise and sunset
- Weather
  - Represented by the application's ability to query a weather API, yet restricted to current weather at present location of device
- Availability
  - Represented by the application's ability to query user's calendar to calculate availability
- Location
  - Represented by the application's ability to determine location, yet restricted to street addresses or geographic coordinates.

Opportunity aims to be a flexible service capable of improving the user's productivity.

Functional specifications

Opportunity will be used by people who have difficulty remembering or planning activities that they want or need to do. It allows users who have identified stretches of free time to fill it with appropriate activities. Opportunity will be able to show local events to the user letting them decide what they want to do with their time. Users can do all the things they want to do when the timing is perfect. It also allows users to spontaneously meet up and do activities with their friends. Opportunity is aware of its surroundings location in the world. It uses GPS along with data to determine if an Opp's parameters are met or not. It also monitors the weather; observing the conditions. The application also is aware of the locations of other users and Opps that they choose to share. Opportunity will be aware of local events, store inventories, and ski resort snow levels.

For the minimal implementation, an iPhone will be used as hardware and data will be stored on the phone itself. We will be using the built in GPS, accurate to within 10 meters of location. The application is meant to be dependable so that business and power users can rely on it. For the minimal implementation security is not a high requirement since everything is stored on the phone and none of the information is given out or stored on a server, so creating accounts will not be available. Opps in the minimal implementation will include conditions based on time/date, calendar, specific locations and weather based settings.

For all implementations minimal battery usage will be a top priority, as the application will be running in the background all the time. Depending on the Opps which are enabled, the application will determine what data to collect.

The goal implementation for Opportunity involves a mobile application developed for iOS. The server and database will be deployed onto a DigitalOcean Ubuntu server 14.04 virtual machine which has 1GB memory, 1 CPU core, and 30GB SSD storage. The server can dynamically scale to accommodate a large flux in demand. The response time to the server should be less than 2 seconds. For the goal implementation, security will be a requirement since the application will have users sharing data and their locations (if the user wants to share that data) as well as the information on the servers will have to be secure. For the stretch goals contacts (users meeting up to do things), hours of stores and general locations will be added to the Opps settings. The stretch goal will have the application using stores' inventories, sales, nearby events, and ski resort snow levels being added as conditions for the Opps.

System Model

This section describes some simple usage scenarios for which a normal user may experience. Furthermore, some UML diagrams are provided to demonstrate how the system is laid out and how it functions.

Use Cases

Below are some sample use cases for which FixCode anticipates the average user will encounter. The Opportunity app naturally requires that external parties provide data which may be used to provide the user with timely notifications based on their surroundings. The use cases provided below reference the term actor frequently. In this context an actor is any external party that has influence on the behavior of Opportunity.

Such actors include the following

- Weather API
- Ski Resort API
- Event API
- Calendar API
- Google and Twitter OAUTH APIs
- Location service

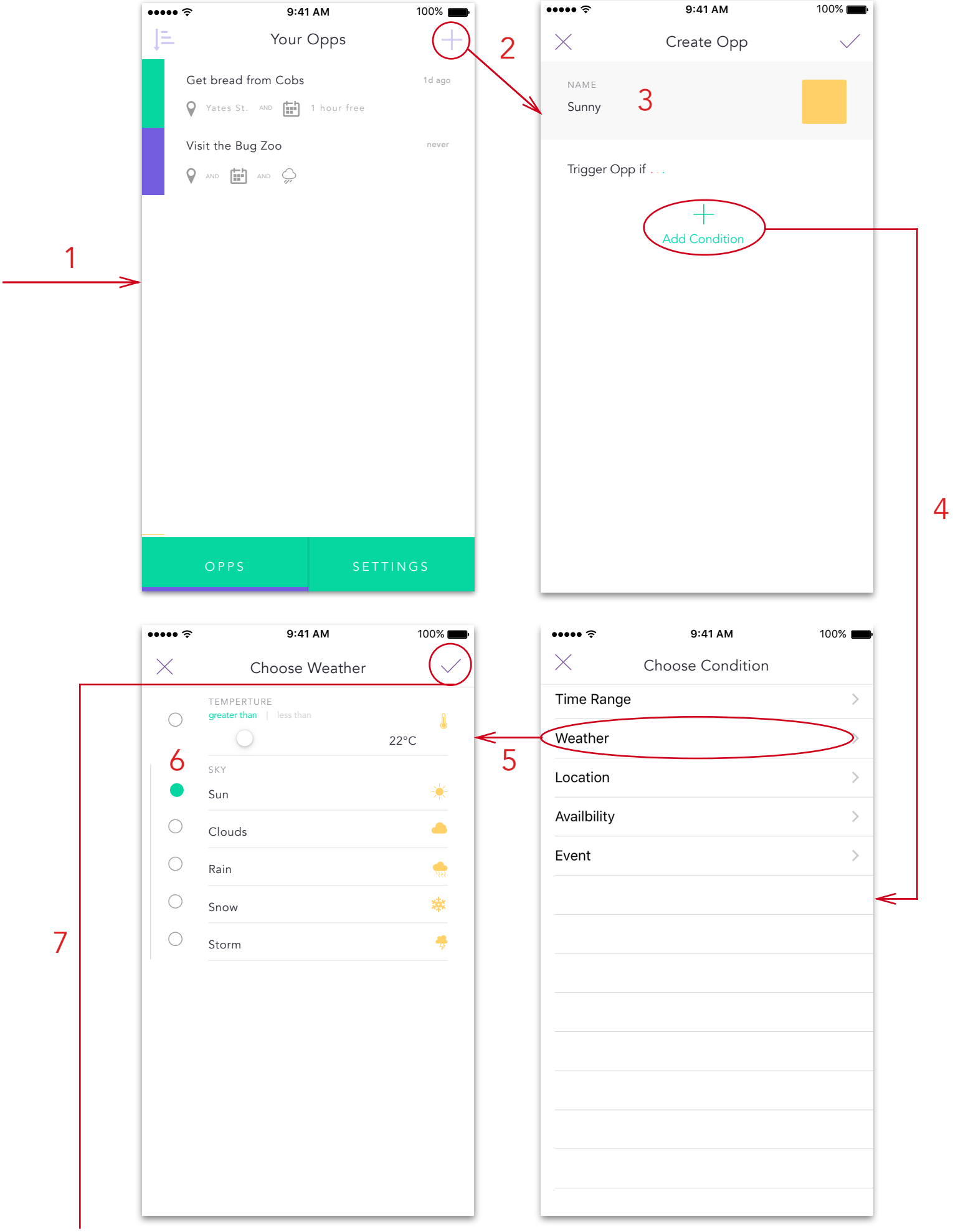
The weather API (open weather map) will affect the app by letting the app know the weather conditions for a given time. The ski resort API (weather2) affects the app by sending information (snow report and weather) about a specific ski resort. The event API (EVENTful) will send information about local events when the application requests for this information. A calendar API (Google Calendars) will send information about when the user has an appointment from which Opportunity will calculate when the user is free. Opportunity will allow users to sign in using Twitter or Google account or to create a new account specifically for Opportunity using the user's email. Opportunity is also affected by the location of the phone, as location data is retrieved using a the built in location services.

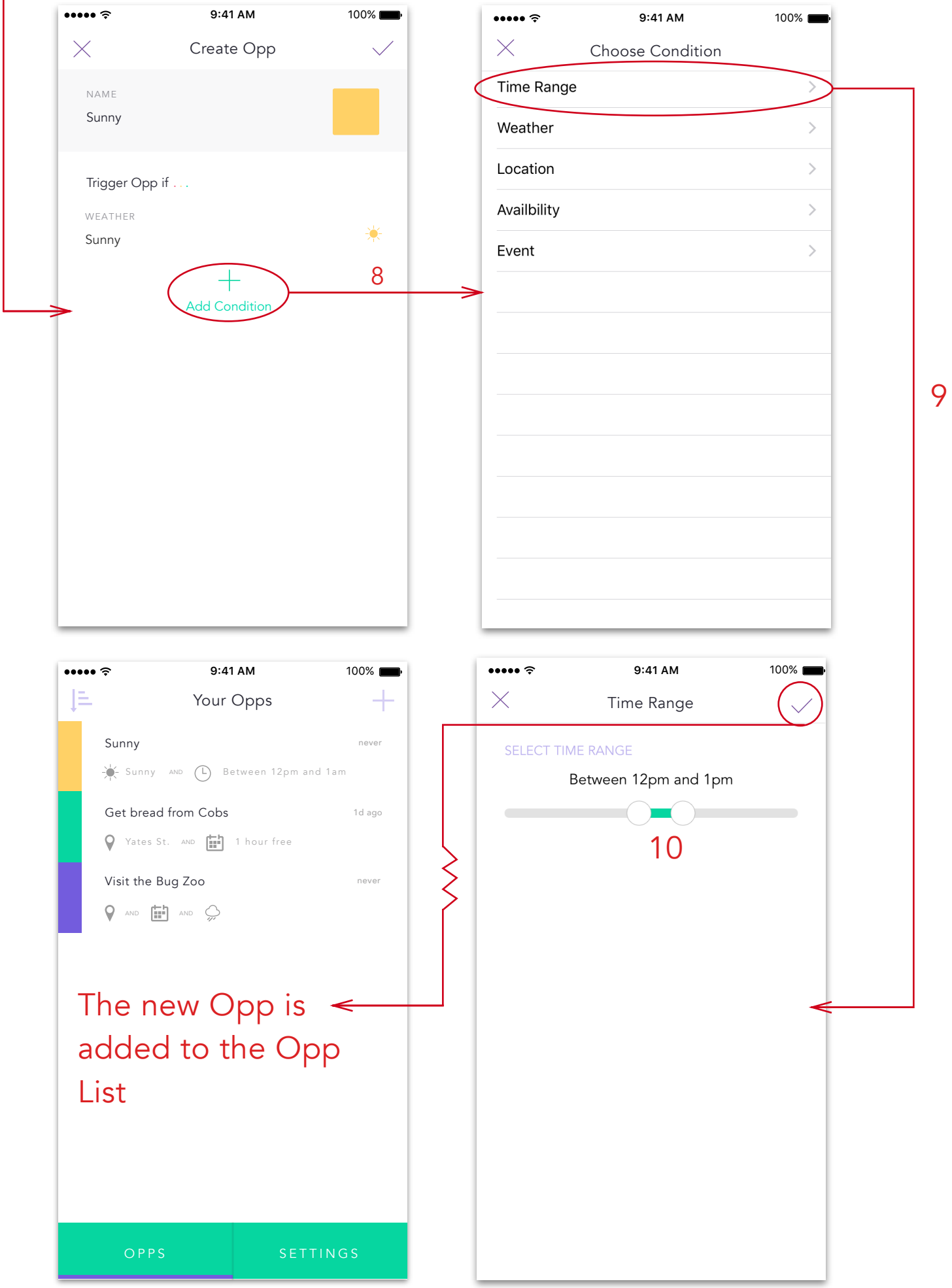
Use Case 1

ID	CREATE AN OPP
EXAMPLE	User wants to create a new Opp to be notified when the weather is sunny at the noon hour
ACTORS	Weather API
PRECONDITIONS	The user has already created an Opportunity account
BASIC STEPS	1. The use case begins after the user has already opened Opportunity. 2. User selects (+) allowing them to create a new Opp. 3. Names their new Opp 'Sunny Day'. 4. User selects the (+) 'Add Condition' button. 5. User selects 'Weather' from the available conditions. 6. User selects 'Sunny' as the weather state. 7. User selects the (✓) to add the condition. 8. User selects the (+) 'Add Condition' button again. 9. User selects 'Time range' from the available conditions. 10. The user adjusts the time range slider to read "Between 12pm and 1pm" 11. User selects the (✓) button again to finish creating the Opp.
POST CONDITIONS	The user has created an Opp, they will be notified if it is sunny and between 12pm and 1pm.

A user interface walkthrough is provided on the following page for this use case.

Walkthrough for use case sunny and between 12pm and 1am




9:41 AM
100%
Create Opp
NAME
Sunny
Trigger Opp if . . .
WEATHER
Sunny
Add Condition
8
Choose Condition
Time Range
Weather
Location
Availbility
Event
9
Your Opps
Sunny
never
Sunny AND Between 12pm and 1am
Get bread from Cobs
1d ago
Yates St. AND 1 hour free
Visit the Bug Zoo
never
AND AND
The new Opp is added to the Opp List
OPPS
SETTINGS
Time Range
SELECT TIME RANGE
Between 12pm and 1pm
10

Use Case 2

ID	ADD A CONDITION TO EXISTING OPP
EXAMPLE	User wants to add further conditions for their local ski resort
ACTORS	Weather API
PRECONDITIONS	An existing use case exists for new snow at a ski resort, user logged in
BASIC STEPS	1. The user selects the existing Opp from their list of Opps. 2. The user clicks the (+) add condition button, 3. Then selects “Weather” from the options. 4. From the available choices the user selects “Sunny” 5. User selects (✓) to add the condition 6. The user taps on the title of the Opp to change the name to “Blue Bird Powder Day” 7. User selects (✓) again to update their changes to the Opp
ALTERNATIVE STEPS	3) User can select any type of condition at this step
POST CONDITIONS	The existing Opp has a new condition in its triggers, it now must have snowed recently and be currently sunny.

Use Case 3

ID	LOCATION TRIGGER
EXAMPLE	A location-based Opp is triggered near a grocery store
ACTORS	Location Services
PRECONDITIONS	The user has created an Opportunity account, and an Opp to be triggered when they are near an appropriate store exists
BASIC STEPS	1. User is near an appropriate store 2. Opp's preconditions are met 3. A Notification is created 4. The user expands the notification drawer on their device 5. <user's response> User acknowledges the Opp and dismisses it by selecting 'dismiss'
ALTERNATIVE STEPS	At <user's response> the user could alternatively choose to 'snooze', allowing the Opp to trigger again the next time the conditions are met
POST CONDITIONS	User has been reminded that they are near a grocery store

Use Case 4

ID	DELETE
EXAMPLE	User wants to delete an Opp they have previously created
ACTORS	None
PRECONDITIONS	The user has a previously created Opp
BASIC STEPS	1. User selects the Opp they want to delete 2. User selects ‘delete’, destroying the Opp 3. The user observed that their Opp is no longer present in the Opp list
ALTERNATIVE STEPS	2) The user may also disable the Opp by selecting ‘disable’
POST CONDITIONS	The user has deleted an Opp

Use Case 5

ID	SIGN UP
DESCRIPTION	User wants to create a new account after downloading Opportunity
ACTORS	Google or Twitter OAUTH API
PRECONDITIONS	The user has downloaded Opportunity
BASIC STEPS	1. User opens Opportunity and selects ‘sign up’ 2. <Sign in method> User chooses to sign up with email and enters their name, email, and desired password 3. The user is presented with a tutorial on the basic usage of the Opportunity app
ALTERNATIVE STEPS	At <Sign in method>, user could alternatively choose to sign in using a Google or Twitter account
POST CONDITIONS	The user has created a new Opportunity account

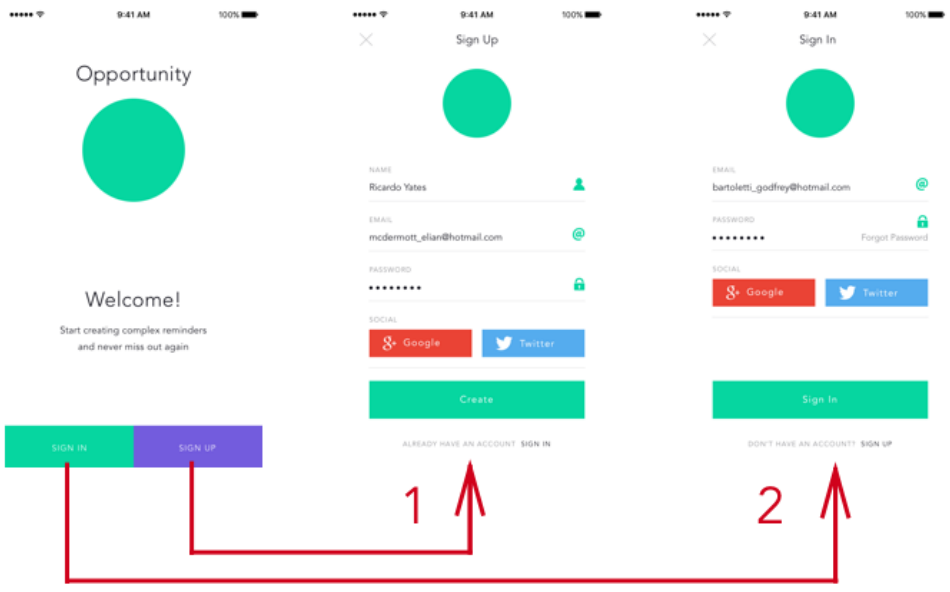


Figure 1 – Example sign-up user interface

UML Diagrams

UML is used as a tool to represent the structure and behavior of portions of the Opportunity system. The following diagrams are provided:

- Opp Dependency Component Diagram
- Opportunity App Class Diagram
- Opp Trigger State Sequence Diagram
- Opportunity Deployment Diagram

Opp Dependency Component Diagram

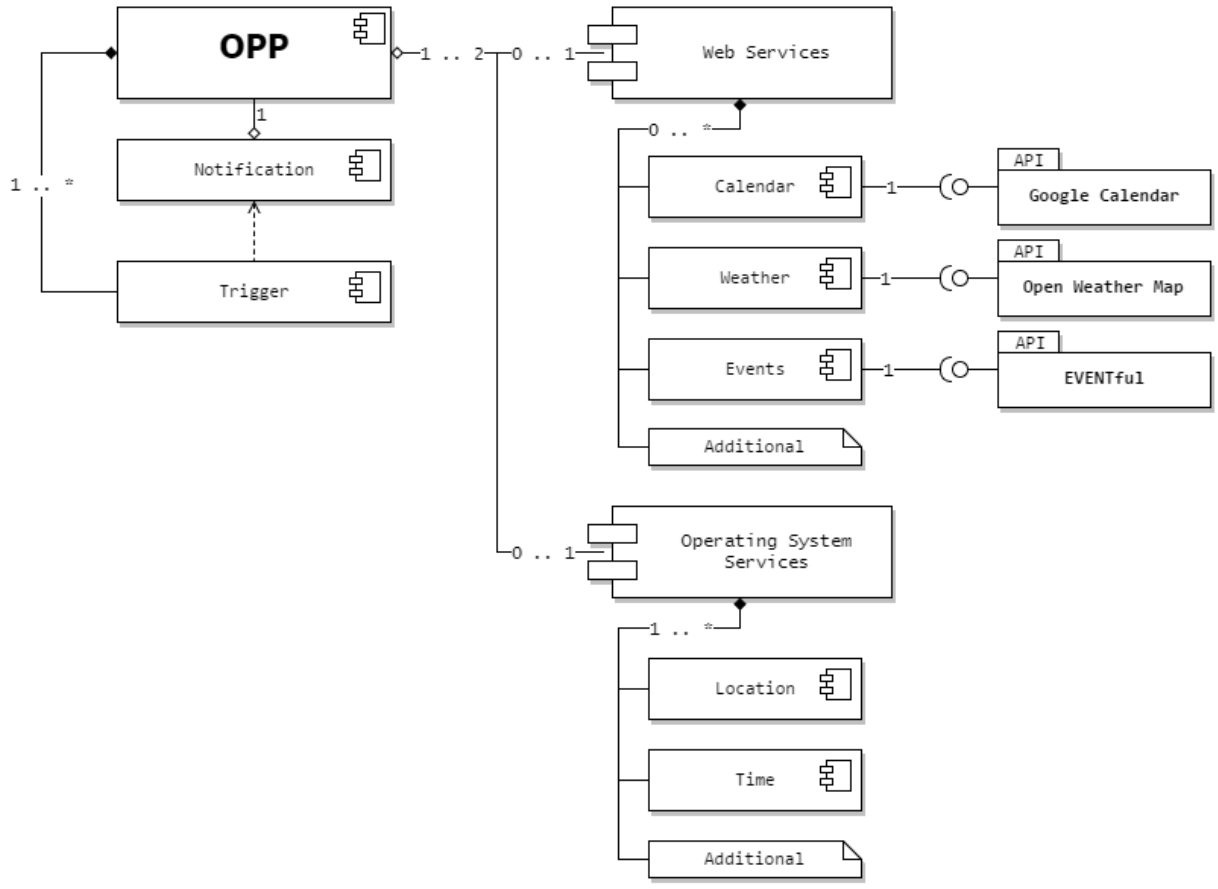


Figure 2 – Component relations for an Opp

Figure 2 depicts the relations of high level system components that an Opp utilizes and depends upon. To summarize the diagram, an Opp is an entity that may produce notifications. The state of an Opp is determined by the triggers it contains. State checking is elaborated in the Opp Trigger State Sequence Diagram section. To check the states of a trigger, the Opp depends on either or both of the services encompassed in the Operating System or Web Service components. These components provide methods to check the state of a trigger. Services wrapped by the Web Service component further depend on functionality provided by an internet API.

Opportunity App Class Diagram

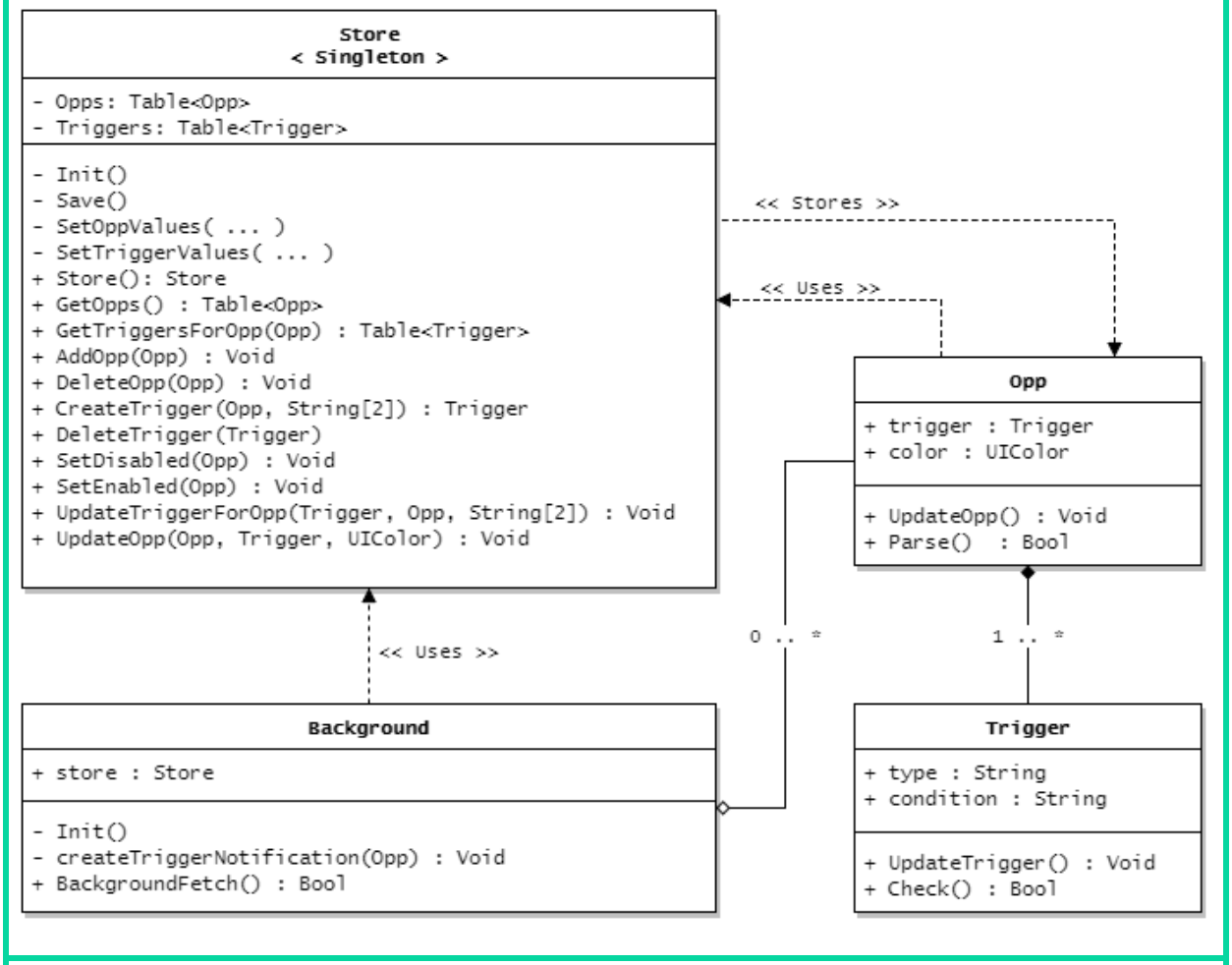


Figure 3 – Opportunity App functional class diagram

The diagram in figure 3 shows the classes and their fields and methods. The store class is persistent, as this is the class that manages local storage for Opps and triggers, it behaves in accordance to the singleton pattern; any other class is able to obtain its instance. The trigger class is part of the composition of the Opp class, in that an Opp is useless without at least one trigger. The background class is responsible for checking the state of an Opp when the Opportunity application is not in the foreground. Note that this diagram does not depict implementation classes used in support of the iOS platform, such as view controller and visual component classes.

Opp Trigger State Sequence Diagram

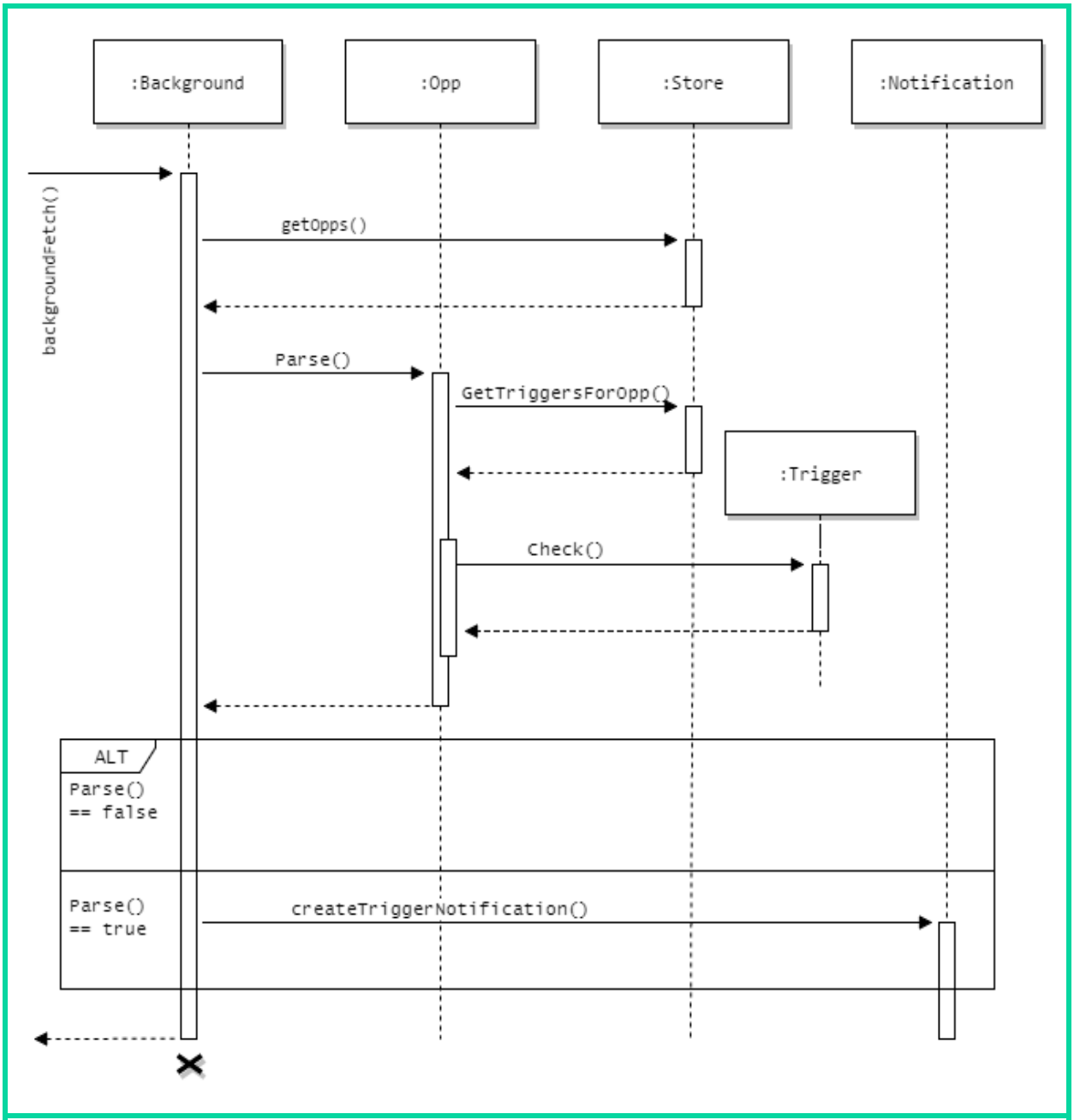


Figure 4 – Trigger checking sequence diagram

The diagram in figure 4 shows the sequence which occurs when the system invokes the background class. This occurs when the system allocates CPU time to Opportunity app when it is not in the foreground, at time the `backgroundFetch()` function is invoked and acts as the entry point at which Opp triggers can be checked. This may occur via push notification from either of the Service Components (Web and Operating System) or on a set interval timer. The background class calls `opp.Parse()` for each of active Opps returned by `store.getOpps()`, if this unction returns true, all of the Opp's triggers have been met, and a notification may be produced.

Opportunity System Deployment Diagram

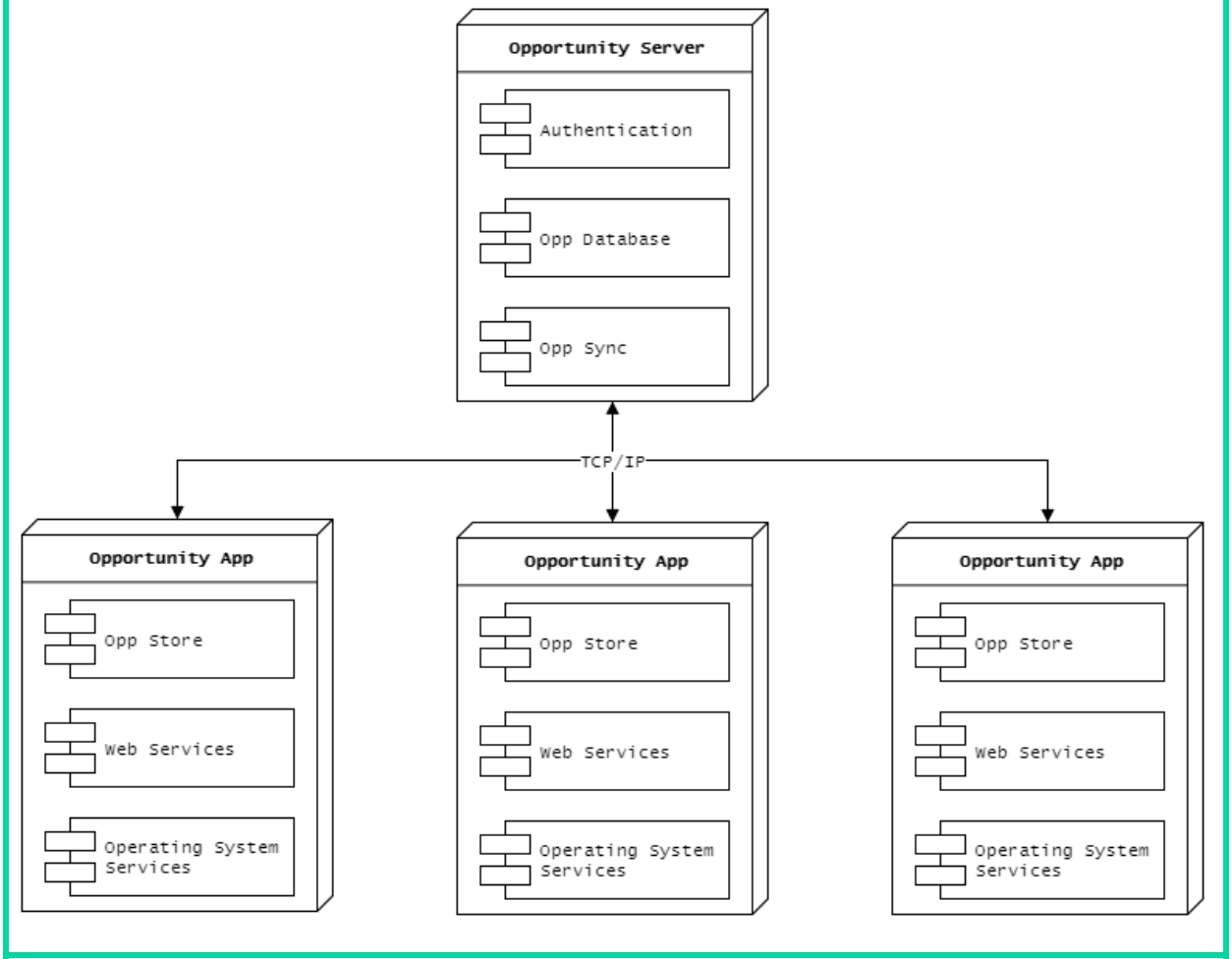


Figure 5 – System deployment diagram

As discussed in the Management Plan section below, the ideal deployment of the Opportunity app involves a central server to allow users to create accounts with Opportunity such that they can back up their Opps from devices they own and synchronize Opps across multiple devices. The minimal system involves only deploying the “Opportunity App” component seen in the deployment diagram in figure 5.

Management plan

There are four main components required for the goal implementation of the Opportunity system. These components are:

- mobile application
- backend server
- database
- website

The user will operate Opportunity through the mobile application. The app is where the user will create and login to an account, create Opps, and be notified of triggered Opps. The application will contact the server through a REST API to authenticate users and create and delete Opps. The server will store user and Opp information in a database. Details on these components are detailed below.

Mobile Application

The application will be designed using Sketch, an interface design tool, and prototyped using Invision, a prototyping platform. Using these tools before actually coding the app will allow us to quickly iterate between designs to efficiently decide on the design with the best user experience.

The mobile application will be developed for the IOS platform. The currently designed app interface can be seen in the UI walkthrough. An Android application may also be made if time permits. The IOS app will be developed using the Swift programming language on the XCode development IDE. The interface will be created using XCode storyboards and coded using view controllers for each view. User info and Opps will be stored locally on the device using a core data database. The app will use background app refresh to wake the app up at specific times, to check the weather, determine the location, and check availability on the user's calendar. Each time background app refresh wakes the app the device's battery will be drained. Therefore, it is important to only wake the app when absolutely necessary. The app will communicate with the server via a REST API. Using GET and POST requests to specific endpoints, the app will upload user and Opp data to the server.

User Permissions

The application will require the following permissions to be accepted by the user.

Location services

In order to provide the user with location based Opportunity notifications, the application should be able to access the device location in the background.

System notifications

Notifications are the fundamental service used to communicate the availability of an Opportunity to the user when the Opportunity app is not in the foreground.

App Icon

The app icon was created based off the Opportunity logo and is seen in figure 6. It follows the style of many other popular IOS apps available, such as YouTube, Reddit, Photos, and Safari. The icon can be seen on the IOS home screen in figure 7.

Figure 6 – Application icon

Figure 7 – Application icon on home screen

Opps

Opps are created and triggered using the mobile application. This section will detail how each condition in the Opp will be technically implemented. Many of the conditions require waking the app in the background periodically to check data. These checks will be squished into one check for all Opps to save device battery.

Location

There are two components to location in Opportunity. The first is a specific address the user wishes to be notified at. When the user has an Opp with a location condition, the app will use the device functions to be woken when the device approaches a specific location. Device battery will drain when the location is checked with GPS, so the location checks for all Opps that use location will be combined into one. The second component of location is to trigger an Opp when the user is near a general location, such as grocery store, clothing store, or gas station. This is not a minimum requirement as a more complex use of Google or an external API is needed to find a list and location of stores and businesses.

Weather

We are using an external API to query for weather data for the user's location. The API we are using is the OpenWeatherMap API (http://openweathermap.org/api). The free plan allows for 60 request per second and 50 000 requests per day. This is within our expected demand. When the user creates an Opp that includes a location condition, the app will periodically check the weather at the user's location to see if they meet the Opps conditions.

Availability

In the app the user will be app to connect a Google calendar. Authentication with Google will be done using secure OAuth2 using pre-existing libraries. The app will refresh the calendar once every hour to pull in new events. Within the app, the calendar events will be analyzed against the user's Opps to determine if the conditions are met.

Events

We will be using the Eventful API (https://api.eventful.com/) to get event information. This API will provide the app with information about concerts, festivals, sports, etc. The user will be able to search for upcoming events and the application will alert the user near the time of the event.

Server

The server will be implemented if time permits. It is not an essential part of the system as the mobile application can function on its own. The server will be implemented using the Python programming language using either the Django or Flask framework. The server should be lightweight as it solely serves a REST API.

Authentication Server

authentication is one of our stretch goals. To implement this a backend server is required to make API calls to. The user will either create an account with social platform Google or Twitter, or with email and password. The server will generate the client with a token when valid credentials are provided. This token will be used in the header of all requests that change user Opps or account information. Passwords on the server will be encrypted with bcrypt, an irreversible encryption algorithm.

Opp Sync

The data for authenticated users will be stored on the server in a database. This data will be synced to the user's device to make sure they have the most up to date data. Storing user data on the server will allow multiple devices to be used for the same account. Also, if the user's device is lost or broken, they will easily be able to restore all data to a new device.

Database

The database will be SQLite. SQLite is a lightweight relational database that will allow us to quickly setup and configure it. We are not expecting to be storing large amounts of data so a more advanced database (PostgreSQL, MySQL, etc...) is not needed. The database will store user and Opp information which will be served by the server. The database will be hosted on the same machine as the backend server.

Website

The website is a small component of the system. It will describe what the app is and have a link to the app store where you can download the app. There will be screenshots of the interface and short descriptions of all the functions. This will be the main place to market the Opportunity system.

Minimal System by EOT

Should time restraints affect the progress of the management plan outlined above, a minimal implementation is forecasted herein for delivery by the end of the development term. FixCode assures that a minimally viable system composed of a self-supported application - that is, without dependence on any internal system servers - is produced by the development deadline. Such a system will consist of the ability to create Opps with triggers dependent on weather, user availability, and user location. As mentioned, the minimal implementation does not require the use of any servers internal to Opportunity to provide the described functionality. Furthermore, a minimally complete system will have a smaller set of accessible states within the application user interface.

Short list for triggered Opps

Below are trigger types considered as required in a minimally complete system.

Time

All Opps should have the ability to be triggered at a certain time. The user may specify the time in hours, or by general times, such as sunrise or sunset.

Weather

Any Opp should have the ability to query a weather API and react to the resulting data as one of the specified triggers. To refrain from over-extending the minimally viable system, weather based Opp triggers are restricted to the current weather at the present location of the device. The user should be able to specify triggers for weather events including active precipitation, and current temperature above or below a set value.

Availability

All Opps should have the ability to query the user's calendar. Minimal state triggers are if the triggered time doesn't conflict with an existing calendar event, or when a calendar event begins or ends.

Location

Any Opp should also be able to use device location as a trigger. In the minimal implementation only address or pin based locations will be provided, and will trigger the Opp when the device location is within a specified radius of the trigger location.

States accessible in minimal user interface

Due to the possibility that not all of the projected features make it into the delivered product, the outline below lists the portions of the app that the user will be able to navigate to in any implementation.

Opp list

Considered to be the main screen of the application, this screen shows the list of active Opps that are awaiting their trigger conditions to be satisfied.

Opp configuration

Opp creation and editing can be done using the same interface. This interface will likely consist of a set of a screens for viewing and configuring a selected Opp.

Error handling

Error states may arise during normal usage. FixCode has anticipated the following and will provide a convenient handler for each scenario.

A. User tries to create an Opp with 2 of the same conditions
B. User turns off location services
C. Notifications turned off
D. Trying to connect to Google Calendar but is not using Google Calendars

A

If a user wants to try and set an Opp that has 2 of the same conditions, the user will be unable to do this. Opportunity will grey out a condition that is already set for the Opp and thus only allow one type of condition for each Opp to be set. The tutorial will solve this by telling the user to set another Opp if they want to set an Opp with the same condition type.

B

If the user turns off location services on their phone, Opps that have location as a condition will not work. To handle this error a pop-up message will appear telling the user that location services is turned off for the application and to turn it back on if they want to receive notifications for Opps that have location as a condition.

C

If the user turns off notifications for Opportunity, the main functions will not work. To handle this error a pop-up message will appear, telling the user to turn notifications back on, so that Opportunity can function normally (see figure 8).

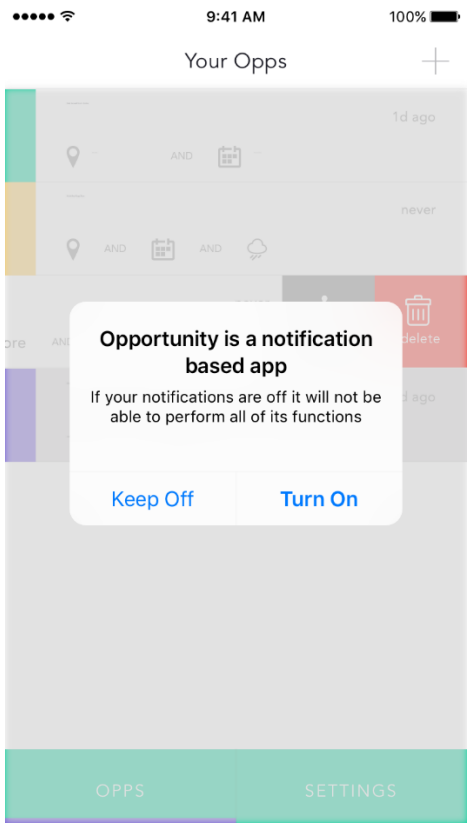


Figure 8 – Notifications off warning

D

If a user tries to connect a calendar app that is not google calendars, will cause an error for Opportunity. To handle this error Opportunity will not allow you to connect the non-Google calendar app and will have a pop-up come up that tells the use that Opportunity only works with google calendars. Opportunity does not need google calendars to be connected to work, however any conditions that use the user's schedule will not be available.