FLOW

CS 470: Software Engineering, Fall 2016
Final Document
Instructor: Dr. Dan Grissom

Team Leader: Kennan Beard
Draft Leader: Jonathan Ming
Presentation Leader: George Vine
Team Member 4: Justin Rohweller
Team Member 5: Zachary Clark
External Graphics Designer: Daniel Koss

Table of Contents

Project Proposal

1 Problem Description

Water conservation is of paramount importance in a time of drought, not only for environmental reasons, but financial reasons as well. Droughts can devastate an area not only by depriving it of drinking water, but also by causing the death of flora and fauna and by increasing wildfire risk. On top of that, utility companies impose higher prices on water when it is scarce. 40 million Californians in particular are facing constant reminders from the state government and their water districts to cut down on water consumption, but they lack the ability to individually quantify their water consumption. And Californians aren't alone—people across the nation are missing out on opportunities to save on their water bills, since they don't know how they are using their water. This makes it much more difficult for them to effectively conserve water (and thereby save money).

2 Proposed Solution

Flow will allow its users to install smart meters on individual appliances in order to track water usage, which can be monitored in an app. This will allow users to track water usage for individual showerheads, faucets, or appliances. Additionally, *Flow* will be used to detect if a faucet has been left on by allowing the user to monitor their water consumption (and potentially close off such a faucet) even when away from home, which will increase home safety. By tracking individual appliances, the consumer will be better equipped to find water-greedy appliances and make changes that will conserve not only water, but also money.

3 Expected Business Value

Flow's primary method of monetization will be the cost of purchasing the smart meters. While it is also an option to charge users for the app in which the meter's data will be displayed, we believe it to be more appropriate to provide this app for free to those who have already purchased the meters. This aligns more closely with the practices of other smart home products, and we believe it will be more appealing to potential users.

3.1 Cost Saving Potential

In Los Angeles, the average residential utility customer (that is, each) uses 81.81 gallons of water per day.[1] This means that a family of 4 living in a house would use approximately 300 gallons of water daily (which translates to about 12 HCF/month). Based on current rates, such a

[1] http://projects.scpr.org/applications/monthly-water-use/los-angeles-department-of-water-and-power/

family would face a monthly water bill of about $75, or $900 annually.[2] Especially considering the fact that water rates are scheduled to rise 5% over the next 5 years for the typical household, such a family could save quite a bit simply by cutting back on water usage.[3] On a larger scale, the owners of multifamily apartment buildings and massive office buildings could stand to save thousands of dollars annually by providing their tenants with tools like *Flow* to help them save water. Therefore, it stands to reason that it would be very attractive to make a small, one-time investment on a product like *Flow* in order to save a large amount of money every year.

3.2 Environmental Conservation Market

In the US, there is a growing market of people who care about environmental sustainability and will spend money in order to feel like they are helping. These are the people who buy hybrid cars, organic foods, and solar energy systems for the sake of the environment. For example, according to a Bloomberg New Energy Finance report, "the sale of electric vehicles will hit 41 million by 2040, representing 35% of new light duty vehicle sales".[4] Additionally, the UK and Chinese governments currently both provide incentives for their citizens to choose electric vehicles. Similarly, organic food, once a $10 billion industry in 2005, has now increased to an over $30 billion industry (as of 2014). According to the USDA, organic food consumers are even willing to pay a premium for organic foods due to health and environmental concerns.[5] These same environmentally-minded people constitute a strong potential market for *Flow*. Water is a natural resource that replenishes more slowly than it is consumed, so the value for this kind of product will only continue to grow, especially in the eyes of such a market.

3.3 Smart Home Market

Another significant viable market for *Flow* can be found in the growing popularity of Apple HomeKit, Google Home, Nest, and related smart home technology. Current smart home products include cameras, lights, locks, and thermostats controlled by users' smartphones. This growing market still lacks any type of system that tracks or controls water consumption. Overall smart home revenue in 2016 has averaged out to a $10 million market, with energy consumption comprising about 10% of that market.[6] Adding *Flow* to Apple HomeKit or Google Home would be the next step for any smart home user looking to track and control their resource consumption.

2 https://d3n8a8pro7vhmx.cloudfront.net/ladwp/pages/41/attachments/original/1461708776/2016Rates_factsheet_web.pdf

3 http://www.myladwp.com/2016_2020_rate_request

4 http://www.nasdaq.com/article/the-rise-of-electric-vehicles-by-the-numbers-cm595564

5 http://www.ers.usda.gov/topics/natural-resources-environment/organic-agriculture/organic-market-overview.aspx

6 https://www.statista.com/outlook/279/109/smart-home/united-states

4 Alternate Solutions

Driblet is one alternate solution to *Flow*, and it uses a very similar hardware device to measure water usage from individual pipes and provide the data to an app.[7] However, not only is Driblet currently unavailable on any market, but the company has also expressed that they aim to focus on corporate and business-level installs, whereas *Flow* will target individual and household use primarily.[8]

Other alternatives such as Dropcountr[9], 30by30[10], and MeterHero[11] either require cooperation from utility companies to install meters on the residence's main waterline (Dropcountr) or require the user to manually input their water usage (30by30 & MeterHero).[12] *Flow* eliminates both of these requirements and their limitations.

5 Personal Investment

The issue of water and its conservation has been on every Californian's radar for quite some time. Having been raised in California, our whole team grew up being told that we need to use less water in whatever we do, whether that be showering, brushing our teeth, or watering our lawns. However, the amount of water we use has always been a mystery to us. We care about this issue because it has been with us all of our lives, even ingrained into our lifestyle. Learning to conserve water has always been a part of our lives; it was how we were raised. *Flow* will help us better quantify and reduce our water usage, helping us enact the principles we learned growing with greater efficiency, thanks to *Flow's* technology.

6 Qualifications of Team

Our team is well-equipped to create, program, and support *Flow*. Some strengths and skills that we share as a team are:

- Java
- MySQL
- Assembly
- Experience creating programs utilizing:
    - Databases
    - Graphic User Interfaces
    - Back-end Applications

Some individual skills that will help us create *Flow* are:

[7] http://www.driblet.io/
[8] https://techcrunch.com/2014/01/08/driblets-smart-water-meter-wants-to-track-your-home-water-usage/
[9] http://dropcountr.com/
[10] http://www.groundwater.org/action/home/30by30.htm
[11] http://www.meterhero.com/
[12] https://www.greenbiz.com/article/3-tools-help-track-your-business-water-usage

- Ken - Arduino, Team Leadership experience
- Jonathan - Android development, web development, cloud databases
- George - Arduino, electronics design, IoT, web development
- Justin - Marketing
- Zach - .NET, Unit testing
- Daniel - Graphic design

Furthermore, we have all experienced the need for a product like *Flow* to track our water usage and help us conserve. Our broad skillset and passion for this product make us perfect candidates to create *Flow*.

Product Requirements

1 Purpose and Scope

1.1 Business Purpose & Scope

Flow's purpose is to help people save money and water by providing users with information about their water usage. *Flow* is expected to impact a variety of people groups, such as homeowners, environmentalists, landlords, and business owners. By using *Flow*, homeowners and business owners will be able to understand exactly where they can cut down on water usage most effectively, and they will save money on their water bill by doing so. Also, users of *Flow* (especially those passionate about environmental preservation) stand to gain the comfort of doing their part to care for the environment. Lastly, landlords can benefit from *Flow* by being able to track their tenants' water usage individually, such that they can charge for water based on actual usage.

The main value of creating *Flow* is twofold: first, financial gain, and second, having an impact on the environment. *Flow*'s smart meters will be sold at a price reasonable enough to cover costs and earn capital so that the product can be continually improved. In addition, *Flow* will help its users be water-wise, which will aid the environment as a whole in the long run.

1.2 Tech Purpose & Scope

Flow provides a unique solution to the problem of water waste. *Flow* will be more than a physical water meter; it connects to a cross-platform application which provides easy access to data about not only how much water is being used, but also how best to reduce usage to save money. The most innovative element of *Flow* is the application of smart home technology to the problem of water conservation. This combination of software and hardware also makes *Flow* unique in a world where smartphone apps and real-world hardware are rarely linked. Because the scope of our business is relatively unexplored by competitors, our specific design for each water gauge may be patentable, whether it measures water using vibrations or by allowing water to pass through the device.

2 Stakeholder Identification

The support of the following stakeholders is essential to the success of *Flow*:

- Homeowners
    - Homeowners like to save money, and some like to do their part to save the environment
    - Homeowners like simple to install and easy to use products that can give them information that they understand

    - Homeowners' opinion of *Flow* is key, since they currently comprise most of the smart home market that will likely be most interested in a product like *Flow*
- Landlords
    - Landlords like to keep track of their tenants' individual utility usages, such as water, electricity, and gas
    - Landlords could properly charge their tenants for their water usage
    - Landlords also comprise a critical portion of *Flow*'s potential users, since a landlord who installs *Flow* would likely do so for an entire complex instead of for just one unit, like a homeowner would
- Business owners
    - Business owners could save thousands of dollars per year by being able to accurately track where their water is being used most and enacting policy to conserve in that area
    - Businesses could prove themselves to be more "green" to their customers, therefore bringing in a wider range of customers
    - Business owners, like landlords, are an important part of *Flow*'s market because of their tendency to buy for the entire business at once
- Universities
    - Universities could encourage and incentivize their students to cut back on water usage by tracking student and staff usage across the university, thereby saving the university thousands of dollars on water every year
- Utility companies
    - Utility companies could further encourage saving water by advertising *Flow* and partnering up to install the devices in their counties/cities

3 Persona (User Role) Identification

We have identified two types of user roles needed for *Flow* to properly handle permissions in landlord-tenant or similar scenarios. In single-user household scenarios, only the "Standard User" persona (described below) will be used, but in multiple-user scenarios, the "View-Only User" may also be used.

- Standard User
    - Standard Users are able to install and maintain flow meters and sync the meters to a router. Standard Users are able to view or remove all meters they have installed.
    - This is the default persona, and all scenarios in which an individual homeowner installs *Flow* will only use this persona.
    - In scenarios where one person has installed *Flow* and a different person wants to see the usage data, the person performing the install will be a Standard User.
- View-Only User
    - View-Only Users are given access to water consumption data by Standard Users.
    - View-Only Users can only view data from meters specifically shared with their account by a Standard User.

        - When a View-Only User is granted access to any of a Standard User's data in this fashion, the View-Only User may send a request to that Standard User to install new meters.
    - The only scenario where a View-Only User would be necessary is a landlord-tenant or similar relationship where the person who performs meter install and management for the building must be different from the person who wishes to see the data from certain meters in that building.

4 User Stories & Requirements

4.1 Functional User Stories

The following section identifies key functional requirements of the product ("any *Flow* user" refers to anyone that uses the *Flow* app, either with a Standard or a View-Only role):

- "As any *Flow* user, I want to be able to register for my own *Flow* account".
- "As any *Flow* user, I want to be able to delete my account."
- "As any *Flow* user, I want to track my water usage per outlet so that I know how to best decrease water usage to conserve and save money on my water bill"
- "As any *Flow* user, I want to receive alerts when I use an abnormally large amount of water"
- "As any *Flow* user, I want to be able to see daily, weekly, and monthly water usage data."
- "As any *Flow* user, I want to be able to compare my usage with the suggested usage put in place by local government/municipality officials."
- "As a Standard User, I want to track my View-Only Users' water usage so I can monitor/charge/reward them appropriately."
- "As a Standard User, I want to be able to easily add/remove Flow meters from my network so that I can manage my changing needs"
- "As a Standard User, I want to receive an alert when any of my View-Only Users use an abnormally high amount of water"
- "As a Standard User, I want to be able to share specific Flow meters to View-Only Users."

4.2 Technical & Usability Requirements

The following section identifies key technical requirements of the product:

- *Flow*'s application will be cross-platform, having an interface accessible through a web browser, an Android app, and an iOS app.
    - *Flow* will follow Google's Material Design
- *Flow's* app will store some user data on the user's device for (at least partial) offline usability

- *Flow* will store user data on servers in order to free up storage space on the user's device, only keeping recent portions of the data locally on the device for viewing without Internet access.
- *Flow* will implement safekeeping of user login information using industry standard password encryption.
- *Flow* will be easy to install so users without plumbing experience can install it.
- *Flow* will be priced in such a way that average users could purchase multiple meters for use on different water outlets.
- *Flow*'s user interface will be be simple and easy to learn, even for users without much experience using technology.
- *Flow* will provide visual representations of collected data, in either a graph, chart, or both.
- *Flow*'s mobile interfaces will not use more than 500MB of the phone's internal storage capacity.
- *Flow*'s mobile interfaces will only communicate meter data to or from the server while on Wi-Fi, unless data stored on the server is explicitly requested by the user.

4.3 Performance Requirements

The following section identifies key requirements of the product:

- It must not take more than 30 minutes for the average user to install a new *Flow* meter.

5 High Level Workflow

5.1 Milestones

Below is a series of milestones that *Flow* will have to pass as part of its development.

1. Basic Hardware Prototype
    a. Create a basic prototype flow meter.
    b. Purchase flow meters and an Arduino unit to test the prototype first using a shower head.
2. Basic Software Application
    a. Develop a basic prototype application using a cross-platform development framework like Cordova, React Native, Xamarin, or similar.
    b. Troubleshoot any issues transmitting data from the flow meter to the application.
3. Beta Testing / MVP
    a. Fine-tune the interaction between hardware and software and polish the visual data analytics module.
    b. MVP will consist of:
        i. Basic flow meter prototype designed to fit a common shower fixture.
        ii. Application consisting of visual analytics
            1. Bar graphs: Daily water consumption per unit
            2. Pie chart: Total daily water consumption

4. Release I
    a. Release its first evolution of the app on a website, the Google Play Store, and the Apple Store for free, while selling the smart meters online.
        i. App will contain everything in the MVP
5. Release II
    a. Release a second version including flow meters that fit appliances and faucets beyond merely showerheads
        i. Appliances ranging from dishwashers, sinks, sprinkler systems, etc.
        ii. Flow meters now sold in bundles as well as individual units
    b. App updates, bug fixes, and new features added

5.2 Timeline

Below are two timelines that very roughly outline the progress schedule for *Flow*'s development. The first timeline details the product document timeline, and the second covers the hardware and software development timeline.

10/11/2016	Product Requirements Document completed
11/1/2016	Functional Specifications Document completed
11/15/2016	Software Design Document drafted
12/1/2016	Software Project Management Plan drafted
12/8/2016	Software Design Document finalized
12/12/2016	Software Project Management Plan finalized
12/16/2016	Project Legacy Document completed

Table 1: Tentative Document Timeline Table

Feb 2017	Basic Hardware Prototype
Apr 2017	Basic Software Application
May 2017	Beta Testing / MVP
Jul 2017	Release I
Jan 2018	Release II

Table 2: Tentative Milestone Timeline Table

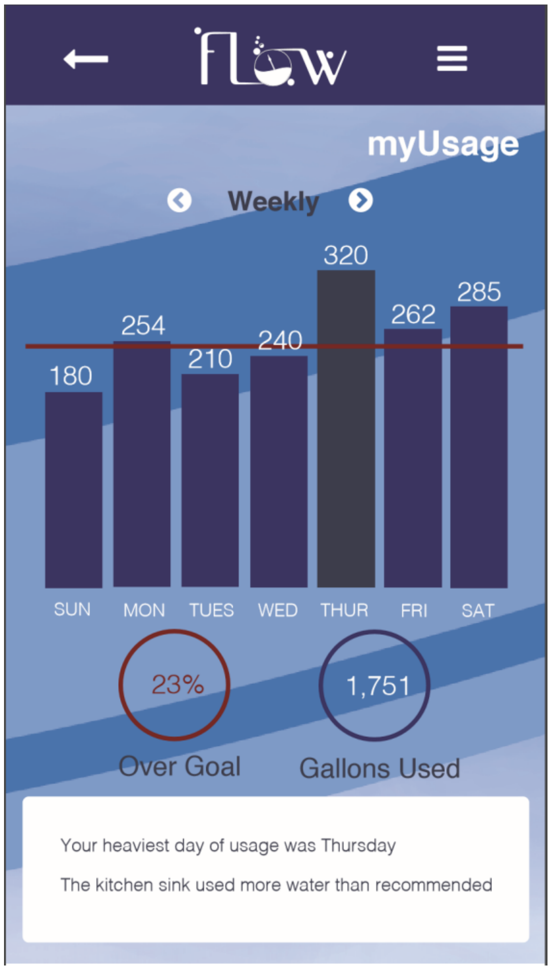
6 Acceptance Criteria

The following section details criteria that *Flow* must meet in order to be considered acceptable.

- Given that the average consumer wants their products to be easily installable, when a user installs a Flow device, it should be an easy process that takes less than 30 minutes and requires affordable, common-household tools.
- Given that small wastes of water can accumulate over time, Flow should be able to track water usage from a faucet with less than a .5 gallon/month margin of error
- Given that users will want to be able to keep track of water usage per outlet, the Flow app should be able to assign names to the different Flow devices (eg; kitchen sink, upstairs bedroom shower, etc.)
- Given that users will also want a reflection of how changing their habits changes their overall usage, the Flow app should provide an aggregate measurement from all of the devices in the network.
- Given that users will want to see how they progress in their overall water usage, Flow should keep a history of previous months and present trends in an easy-to-read graph.

7 Rapid Prototype

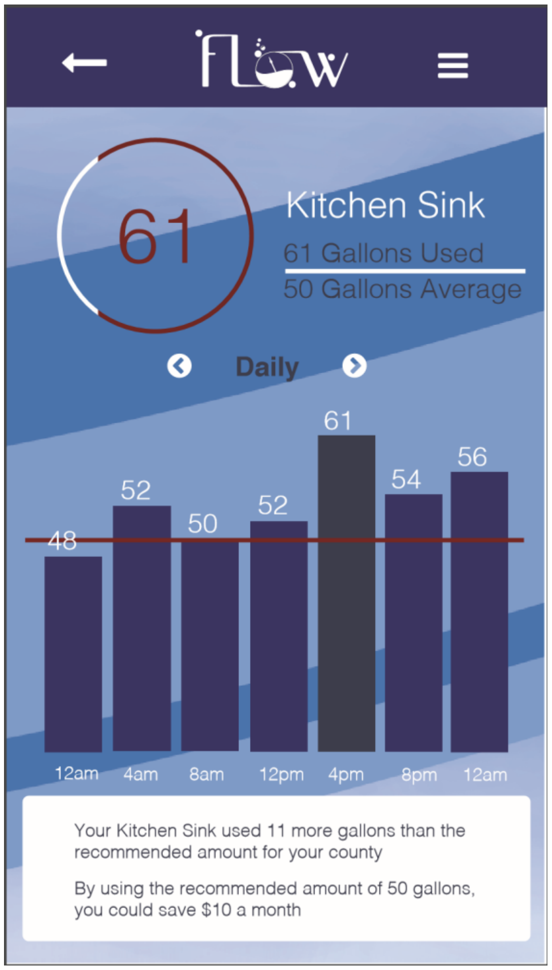
Below are some possibilities of the app design of *Flow:*



Your heaviest day of usage was Thursday

The kitchen sink used more water than recommended

Screen 1: Screen showing monthly usage data.



Your Kitchen Sink used 11 more gallons than the recommended amount for your county

By using the recommended amount of 50 gallons, you could save $10 a month

Screen 2: Screen showing individual facet usage.

Functional Specifications

1 Primary Use Cases

Flow’s most commonly occurring and most important use case scenarios are described in short in the following list, and in greater detail in the remainder of this section:

1. “As any *Flow* user, I want to be able to register for my own *Flow* account”.
2. “As any *Flow* user, I want to track my water usage per outlet so that I know how to best decrease water usage to conserve and save money on my water bill”
3. “As a Standard User, I want to be able to easily add/remove *Flow* meters from my network so that I can manage my changing needs”
4. “As any *Flow* user, I want to be able to see daily, weekly, and monthly water usage data.”
5. “As a Standard User, I want to be able to share specific *Flow* meters to View-Only Users.”
6. “As a Standard User, I want to track my View-Only Users’ water usage so I can monitor their usage”
7. “As any *Flow* user, I want to receive alerts when I use an abnormally large amount of water”
8. “As a Standard User, I want to receive an alert that tells me which of my View-Only Users are using an abnormally high amount of water”
9. “As any *Flow* user, I want to be able to compare my usage with the suggested usage put in place by local government/municipality officials.”
10. “As any *Flow* user, I want to be able to delete my account.”

1.1 UC1 - Account Registration

1.1.1 User Story Description

“As any *Flow* user, I want to be able to register for my own *Flow* account”.

1.1.2 Informal Use Case Description

Basic Course of Action:

1. User visits *Flow* website, clicks “Register” button.
2. System asks for email address, password, password verification, name, and address.
3. User input information and clicks submit.
4. System validates information and displays confirmation page.
5. System sends verification email to user’s email address.
6. User clicks verification link.
7. System confirms email verification and registration completion.
8. Use case ends.

1.1.3 Use Case *Flow* Chart

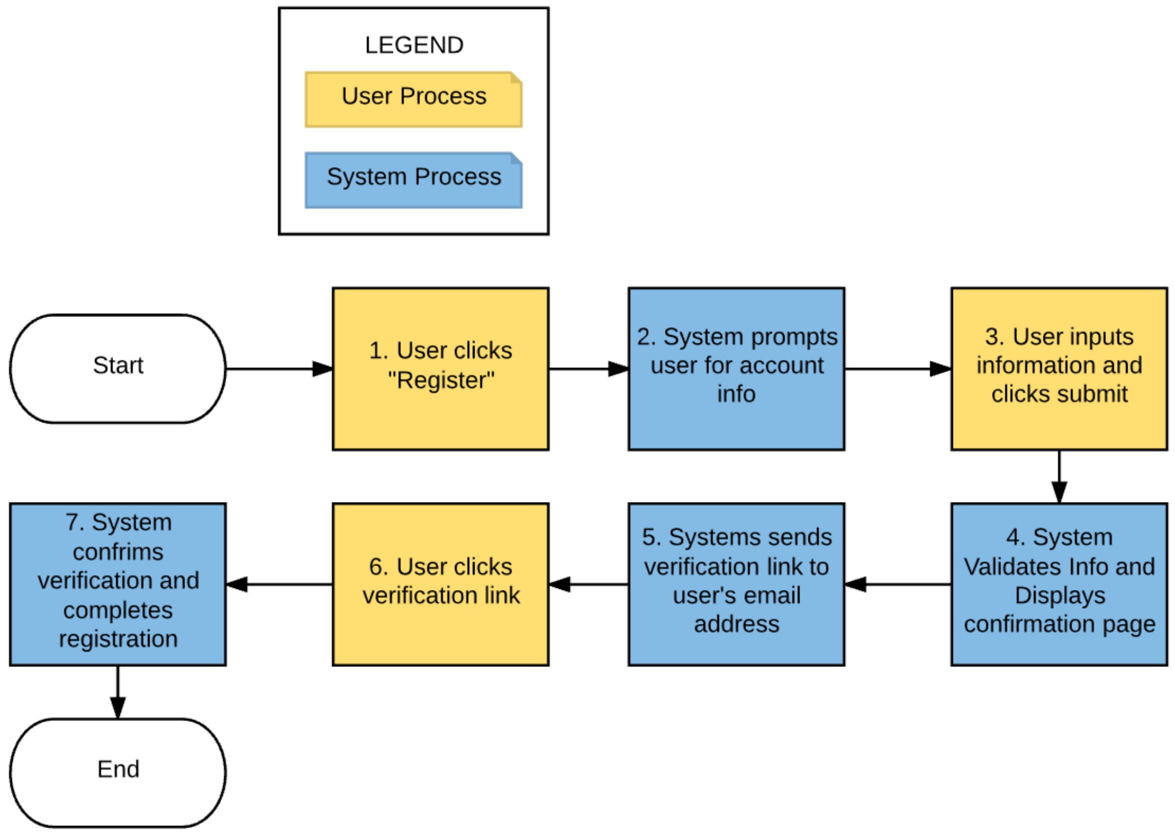

LEGEND
User Process
System Process
Start
1. User clicks "Register"
2. System prompts user for account info
3. User inputs information and clicks submit
4. System Validates Info and Displays confirmation page
5. Systems sends verification link to user's email address
6. User clicks verification link
7. System confrims verification and completes registration
End

Figure 1: Flowchart demonstrating overall flow of UC1.
Source: Lucidchart Link

1.2 UC2 - Meter-Specific Tracking

1.2.1 User Story Description

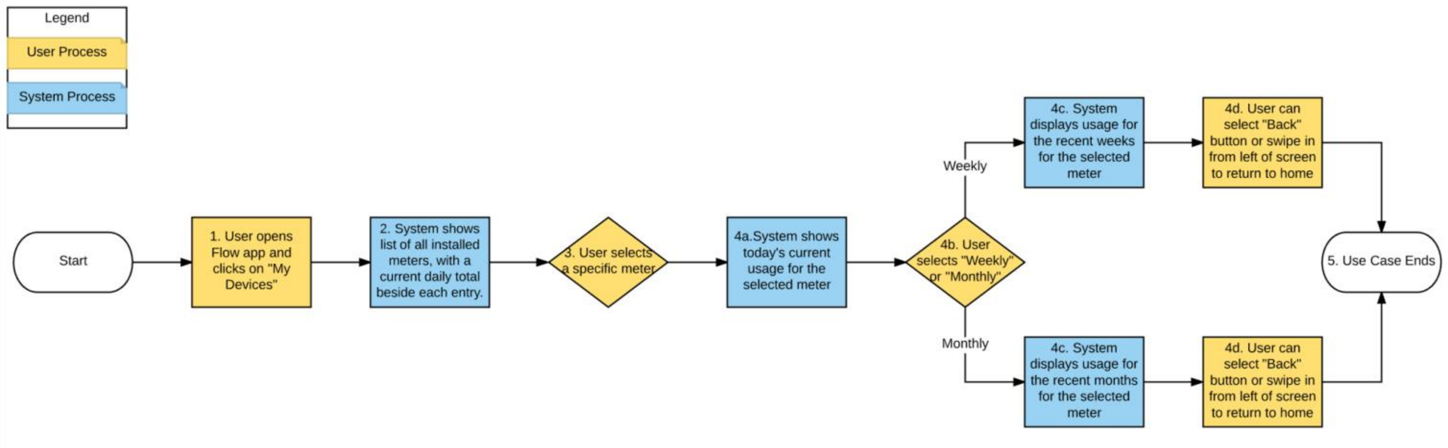
As any *Flow* user, I want to track my water usage per outlet so that I know how to best decrease water usage to conserve and save money on my water bill

1.2.2 Informal Use Case Description

Basic Course of Action:

1. User opens *Flow* application and selects “My Devices” from the menu sidebar.
2. System shows list of all installed meters, with a current daily total beside each entry.
3. User selects a specific meter from the list.
4. System shows the same data as on the “Usage” page (described in §1.4.2, copied below) limited only to data from the selected meter:
    a. System shows screen that shows today’s usage as default along with other options.

   b. User selects “Weekly” or “Monthly” option.
   c. System displays respective screen to show data for the allotted amount of time.
   d. User can select “Back” button or swipe in from the left to return to the main screen.
5. Use case ends.


Legend
User Process
System Process
Start
1. User opens Flow app and clicks on "My Devices"
2. System shows list of all installed meters, with a current daily total beside each entry.
3. User selects a specific meter
4a.System shows today's current usage for the selected meter
4b. User selects "Weekly" or "Monthly"
Weekly
Monthly
4c. System displays usage for the recent weeks for the selected meter
4c. System displays usage for the recent months for the selected meter
4d. User can select "Back" button or swipe in from left of screen to return to home
4d. User can select "Back" button or swipe in from left of screen to return to home
5. Use Case Ends

1.2.3 Use Case Flow Chart

Figure 2: Flowchart demonstrating overall flow of UC2.
Source: Lucidchart Link

1.3 UC - Adding or Removing Meters

1.3.1 User Story Description

“As a Standard User, I want to be able to easily add/drop *Flow* meters from my network so that I can manage my changing needs”

1.3.2 Informal Use Case Description

Basic Course of Action:

1. Standard User opens *Flow* application, clicks on “Settings” icon.
2. Standard User then selects “Add” button or “Drop” button.
3. IF “Add”, System will search for broadcasting *Flow* meters.
   a. User selects a meter from the list of broadcasting meters
   b. System prompts user for Wifi connection settings.
   c. User enters settings, hits “Confirm”
   d. System adds flow meter to account, use case ends
4. ELSE IF “Drop”, System will ask which units to drop and then confirmation.

5. Use case ends.

1.3.3 Use Case Flow Chart

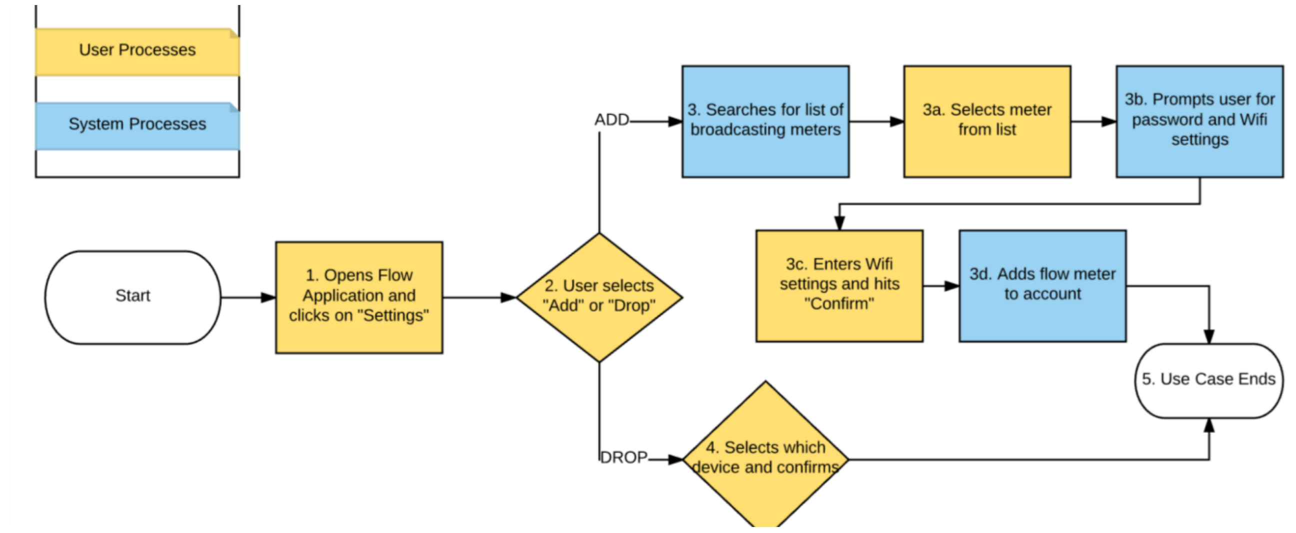

User Processes
System Processes
Start
1. Opens Flow Application and clicks on "Settings"
2. User selects "Add" or "Drop"
ADD
DROP
3. Searches for list of broadcasting meters
3a. Selects meter from list
3b. Prompts user for password and Wifi settings
3c. Enters Wifi settings and hits "Confirm"
3d. Adds flow meter to account
4. Selects which device and confirms
5. Use Case Ends

Figure 3: Flowchart demonstrating overall flow of UC3.
Source: Lucidchart Link

1.4 UC4 - Water Usage Data

1.4.1 User Story Description

As any *Flow* user, I want to be able to see daily, weekly, and monthly water usage data.

1.4.2 Informal Use Case Description

Basic Course of Action:

1. User opens *Flow* application and clicks on “My Devices” button.
2. Screen shows today’s usage as default along with other options.
3. User selects “Weekly” or “Monthly” option.
4. System displays respective screen to show data for the allotted amount of time.
5. User can select “Back” button or swipe in from the left to return to the main screen.
6. Use case ends.

1.4.3 Use Case Flow Chart

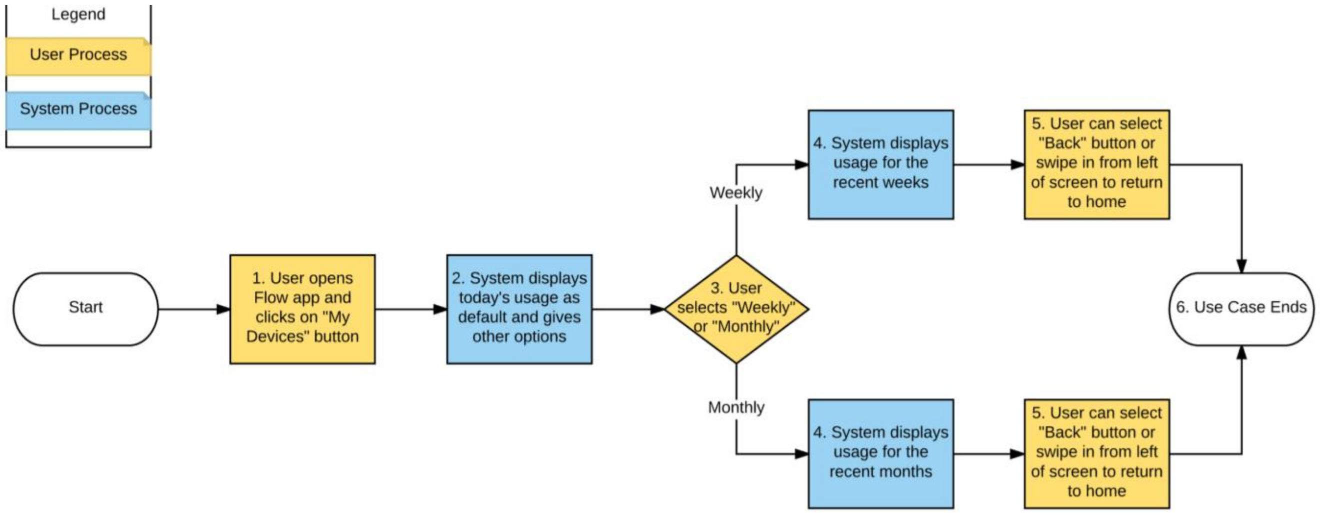

Legend
User Process
System Process
Start
1. User opens Flow app and clicks on "My Devices" button
2. System displays today's usage as default and gives other options
3. User selects "Weekly" or "Monthly"
Weekly
Monthly
4. System displays usage for the recent weeks
4. System displays usage for the recent months
5. User can select "Back" button or swipe in from left of screen to return to home
5. User can select "Back" button or swipe in from left of screen to return to home
6. Use Case Ends

Figure 4: Flowchart demonstrating overall flow of UC4.
Source: Lucidchart Link

1.5 UC5 - Sharing with View-Only Users

1.5.1 User Story Description

As a Standard User, I want to be able to share specific *Flow* meters to View-Only Users.

1.5.2 Informal Use Case Description

Basic Course of Action:

1. Standard User selects “Meter” button.
2. From overflow menu, user selects “Meter Info”, then “Add View-Only User”.
3. System displays search box.
4. User enters email address of the View-Only User they want to share with.
5. System displays the email address typed in by the user
    a. If the email address is already registered with an account, the system displays the full name associated with the account at the same time as the email address.
6. System asks for a confirmation from the user that the email address is correct.
7. Standard User confirms or denies.
8. Denial returns to the search box.
9. If the User confirms, then the system creates and/or updates the View-Only User’s account to include the new meter’s information.
10. The system sends a confirmation email to both the Standard User and the View-Only User.
11. Use case ends.

Flow 22

1.5.3 Use Case Flow Chart

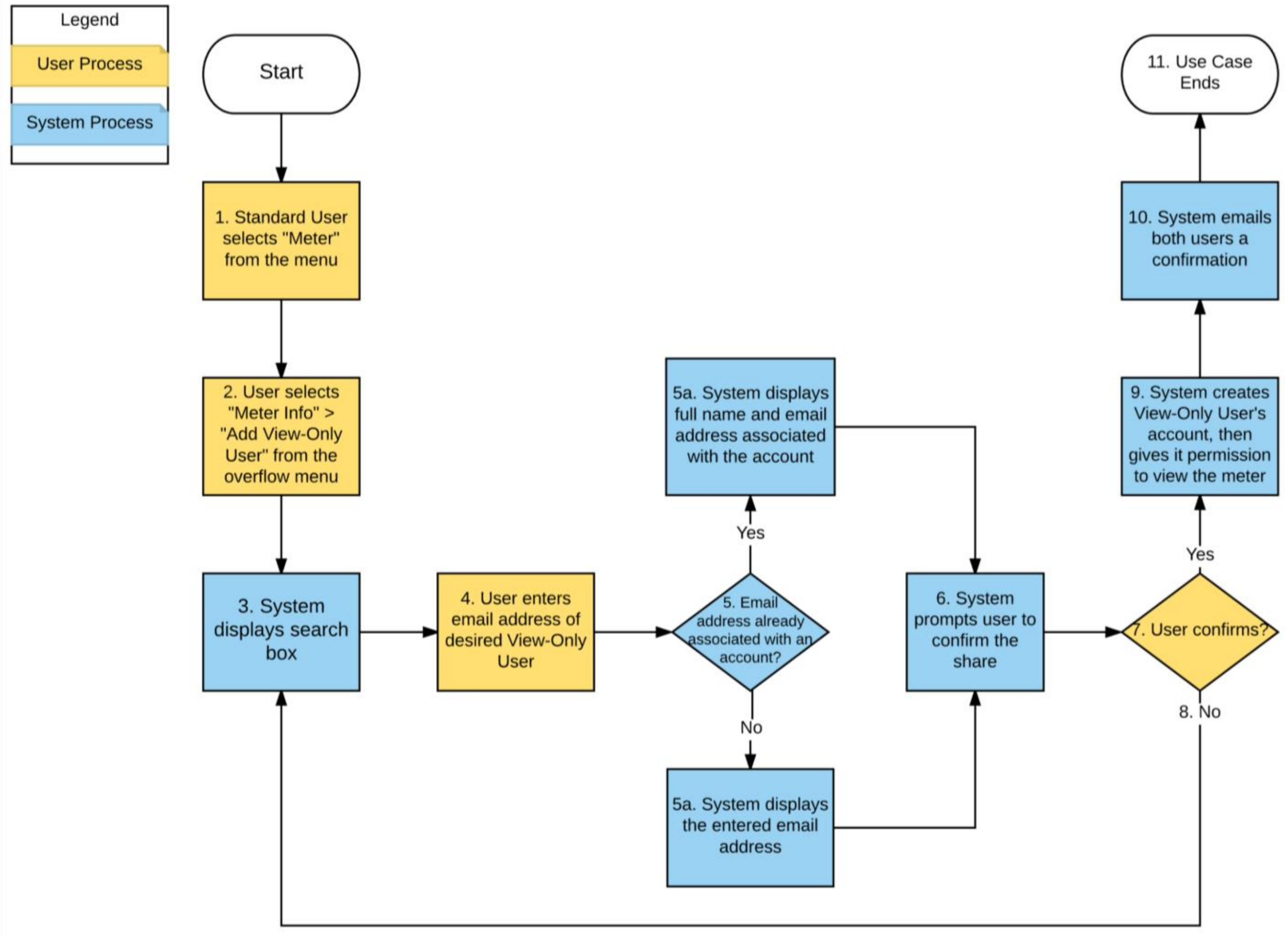

Legend
User Process
System Process
Start
1. Standard User selects "Meter" from the menu
2. User selects "Meter Info" > "Add View-Only User" from the overflow menu
3. System displays search box
4. User enters email address of desired View-Only User
5. Email address already associated with an account?
Yes
5a. System displays full name and email address associated with the account
No
5a. System displays the entered email address
6. System prompts user to confirm the share
7. User confirms?
Yes
8. No
9. System creates View-Only User's account, then gives it permission to view the meter
10. System emails both users a confirmation
11. Use Case Ends

Figure 5: Flowchart demonstrating overall flow of UC5.
Source: Lucidchart Link

1.6 UC6 - Tracking View-Only Users

1.6.1 User Story Description

As a Standard User, I want to track my View-Only Users' water usage so I can monitor their usage.

1.6.2 Informal Use Case Description

Basic Course of Action:

1. Standard User navigates to "Meter".
2. Screen shows My Devices page.
3. User selects a device to view

4. Upon selecting a device, the screen displays Daily/Monthly consumption
5. Use case ends

1.6.3 Use Case Flow Chart

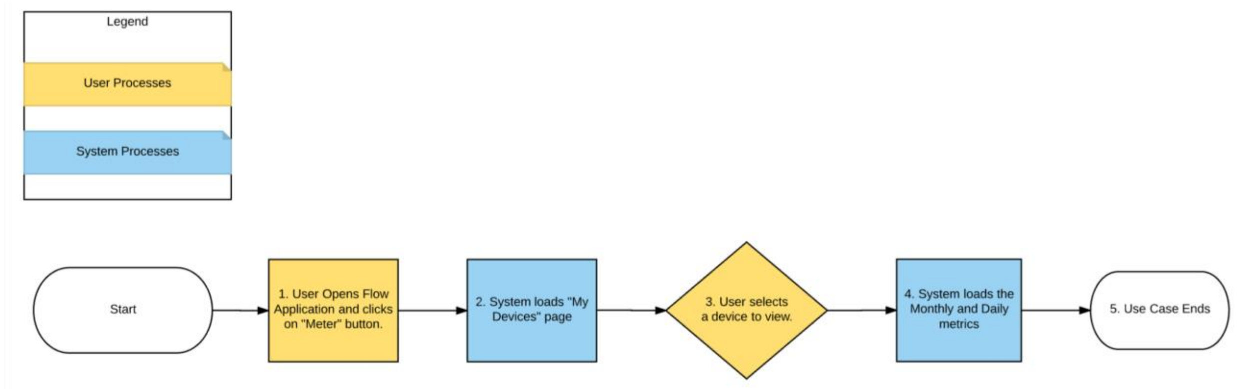

Legend
User Processes
System Processes
Start
1. User Opens Flow Application and clicks on "Meter" button.
2. System loads "My Devices" page
3. User selects a device to view.
4. System loads the Monthly and Daily metrics
5. Use Case Ends

Figure 6: Flowchart demonstrating overall flow of UC6.
Source: Lucidchart Link

1.7 UC7 - Single-User Overuse Alert

1.7.1 User Story Description

As any *Flow* user, I want to receive alerts when I use an abnormally large amount of water.

1.7.2 Informal Use Case Description

Basic Course of Action:

1. System passively evaluates total water usage, keeping track of the user's average consumption across all meters.
2. User consumes a certain (preset) percentage above their normal daily water usage in a single day.
3. System recognizes that the user's water usage exceeds the threshold.
4. System emails the user to notify them that they have overused water.
    a. If the app is in use, it displays a notification pop-up in the user's interface and adds a notification to notification stack.
    b. If the app is not in use, it adds notification to the stack, waits for app to be in use, then displays the pop-up the next time the app is opened.
5. Use case ends.

1.7.3 Use Case Flow Chart

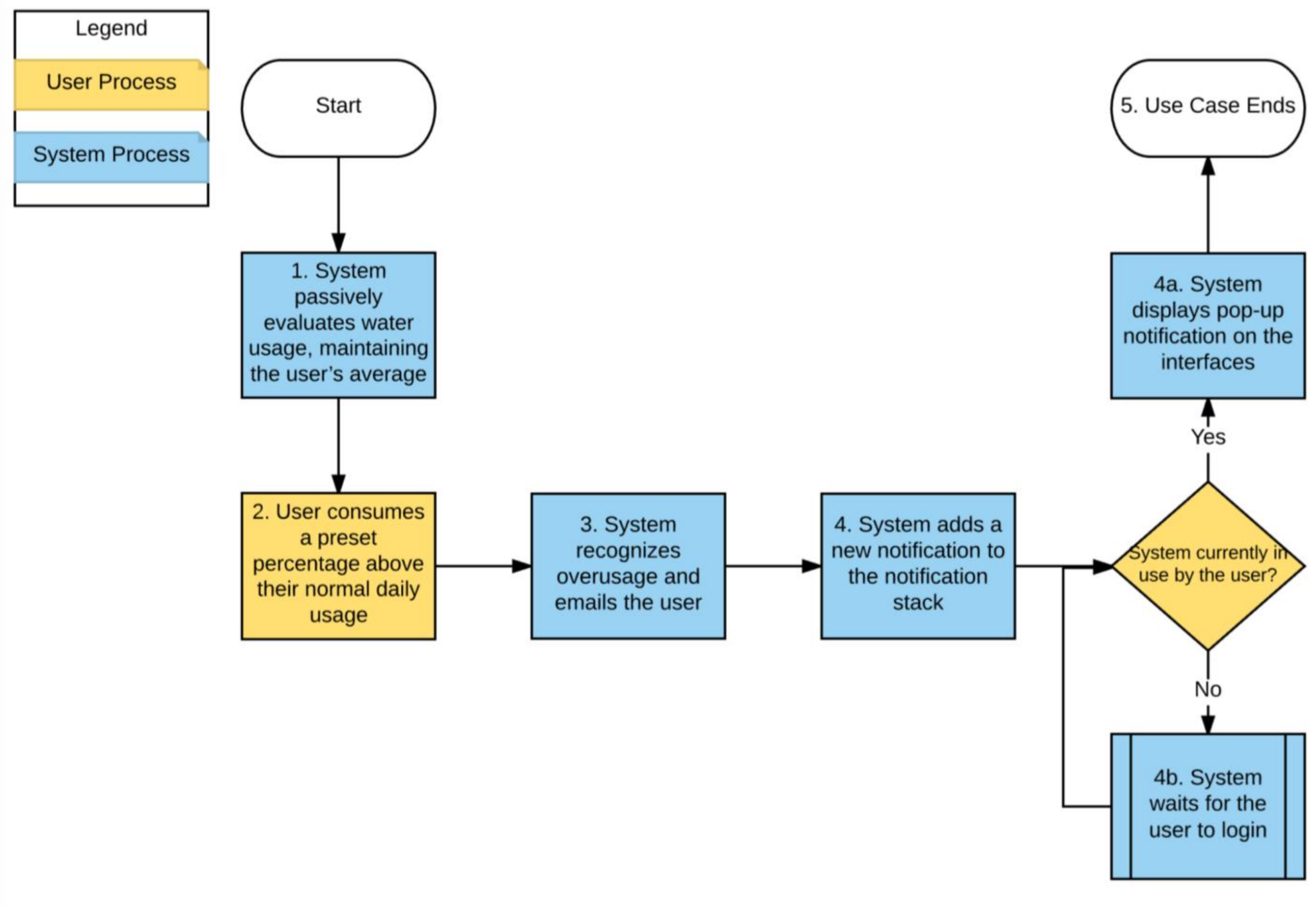

Legend
User Process
System Process
Start
1. System passively evaluates water usage, maintaining the user's average
2. User consumes a preset percentage above their normal daily usage
3. System recognizes overusage and emails the user
4. System adds a new notification to the notification stack
System currently in use by the user?
Yes
No
4a. System displays pop-up notification on the interfaces
4b. System waits for the user to login
5. Use Case Ends

Figure 7: Flowchart demonstrating overall flow of UC7.
Source: Lucidchart Link

1.8 UC8 - View-Only User Overuse Alert

1.8.1 User Story Description

"As a Standard User, I want to receive an alert that tells me which of my users are using an abnormally high amount of water"

1.8.2 Informal Use Case Description

Basic Course of Action:

1. Standard User uses Permissions (described in §1.5.2) to allow View-Only Users.
2. System passively evaluates total water usage, keeping track of the user's average consumption across all meters.
3. View-Only User consumes a certain (preset) percentage above their normal daily water usage in a single day.
4. System recognizes that the View-Only User's water usage exceeded threshold.

5. System emails both View-Only User and Standard User that View-Only User has overused water.
    a. If the app is in use, it adds the notification to notification stack and displays a notification pop-up in both View-Only User's and Standard User's interfaces.
    b. If the app is not in use, it adds a notification to the stack, waits for the app to be in use, then displays the pop-up the next time the app is opened.
6. Use case ends.

1.8.3 Use Case Flow Chart

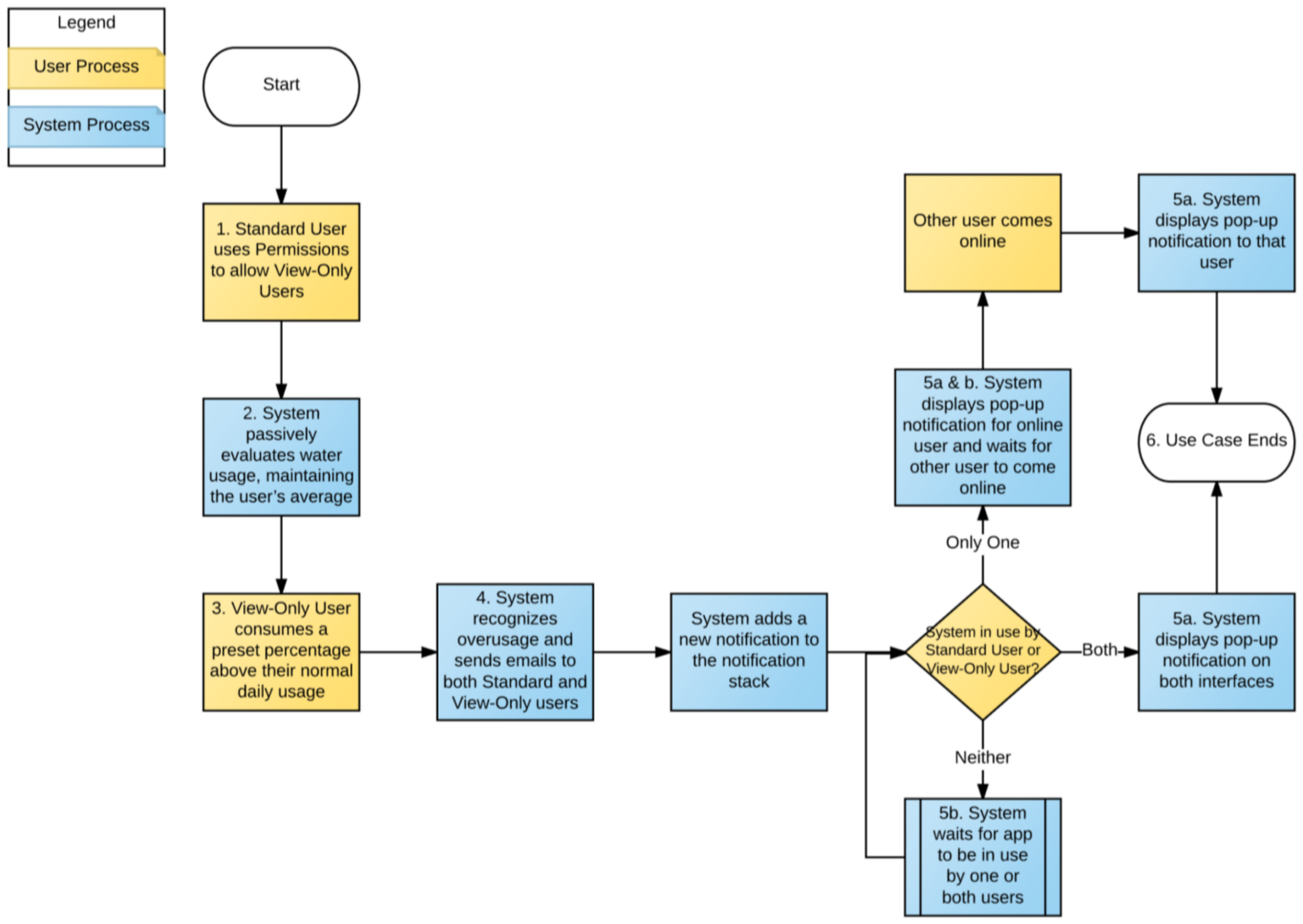

Legend
User Process
System Process
Start
1. Standard User uses Permissions to allow View-Only Users
2. System passively evaluates water usage, maintaining the user's average
3. View-Only User consumes a preset percentage above their normal daily usage
4. System recognizes overusage and sends emails to both Standard and View-Only users
System adds a new notification to the notification stack
System in use by Standard User or View-Only User?
Only One
Both
Neither
5a & b. System displays pop-up notification for online user and waits for other user to come online
Other user comes online
5a. System displays pop-up notification to that user
5a. System displays pop-up notification on both interfaces
5b. System waits for app to be in use by one or both users
6. Use Case Ends

Figure 8: Flowchart demonstrating overall flow of UC8.
Source: Lucidchart Link

1.9 UC9 - Usage Comparison

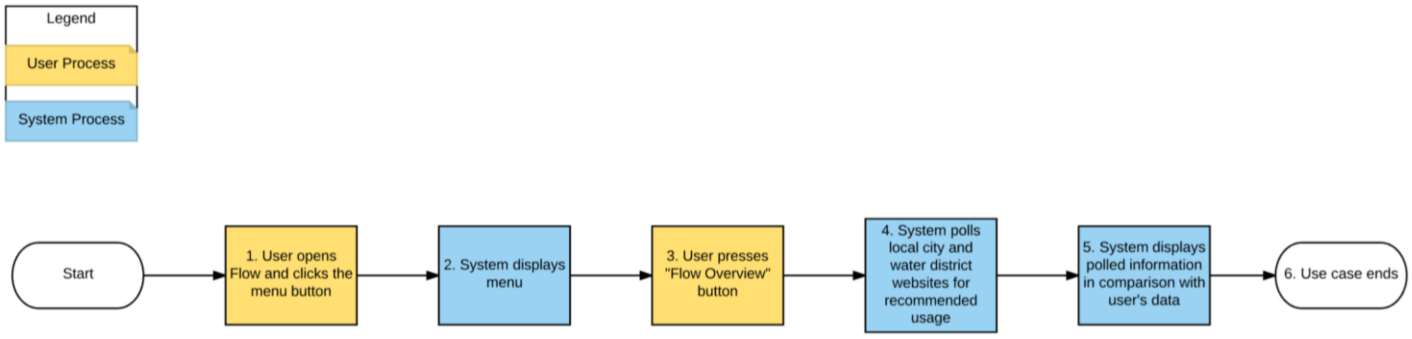
1.9.1 User Story Description

As any *Flow* user, I want to be able to compare my usage with the suggested usage put in place by local government/municipality officials

1.9.2 Informal Use Case Description

Basic Course of Action:

1. User opens up *Flow* mobile application and goes into the menu
2. System displays menu.
3. User presses “Flow Overview” button
4. System polls local city and water district websites for recommended usage.
5. System displays information regarding local usage data and recommended usage amounts for county/state in comparison with user’s data
6. Use case ends.


Legend
User Process
System Process
Start
1. User opens Flow and clicks the menu button
2. System displays menu
3. User presses "Flow Overview" button
4. System polls local city and water district websites for recommended usage
5. System displays polled information in comparison with user's data
6. Use case ends

1.9.3 Use Case Flow Chart

Figure 9: Flowchart demonstrating overall flow of UC9.
Source: [Lucidchart Link]

1.10 UC10 - Account Deletion

1.10.1 User Story Description

As any flow user, I want to be able to delete my account.

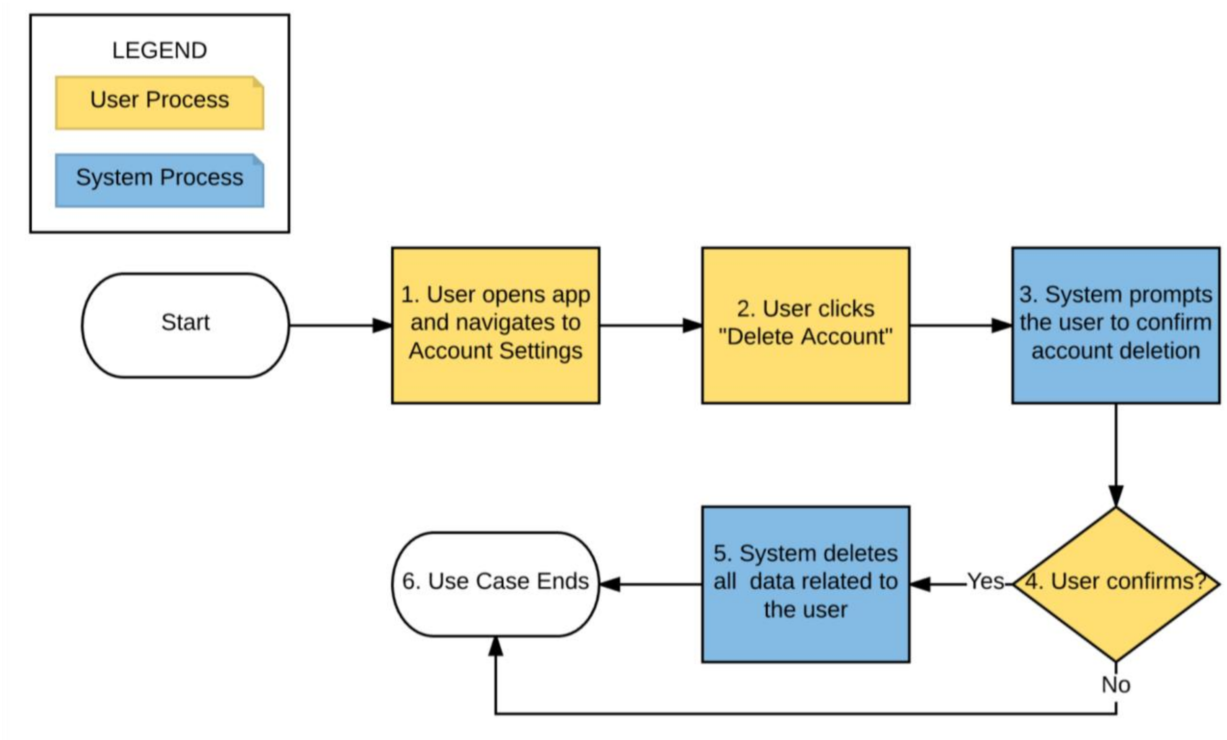
1.10.2 Informal Use Case Description

Basic Course of Action:

1. User opens *Flow* app and navigates to Account Settings
2. User clicks “Delete Account” button
3. System prompts the user to verify that they want to delete their account
4. User verifies
5. System deletes all data related to the user
6. Use case ends

1.10.3 Use Case Flow Chart

Figure 10: Flowchart demonstrating overall flow of UC10.


LEGEND
User Process
System Process
Start
1. User opens app and navigates to Account Settings
2. User clicks "Delete Account"
3. System prompts the user to confirm account deletion
4. User confirms?
Yes
No
5. System deletes all data related to the user
6. Use Case Ends

Source: Lucidchart Link

2 Application Screen Flowchart

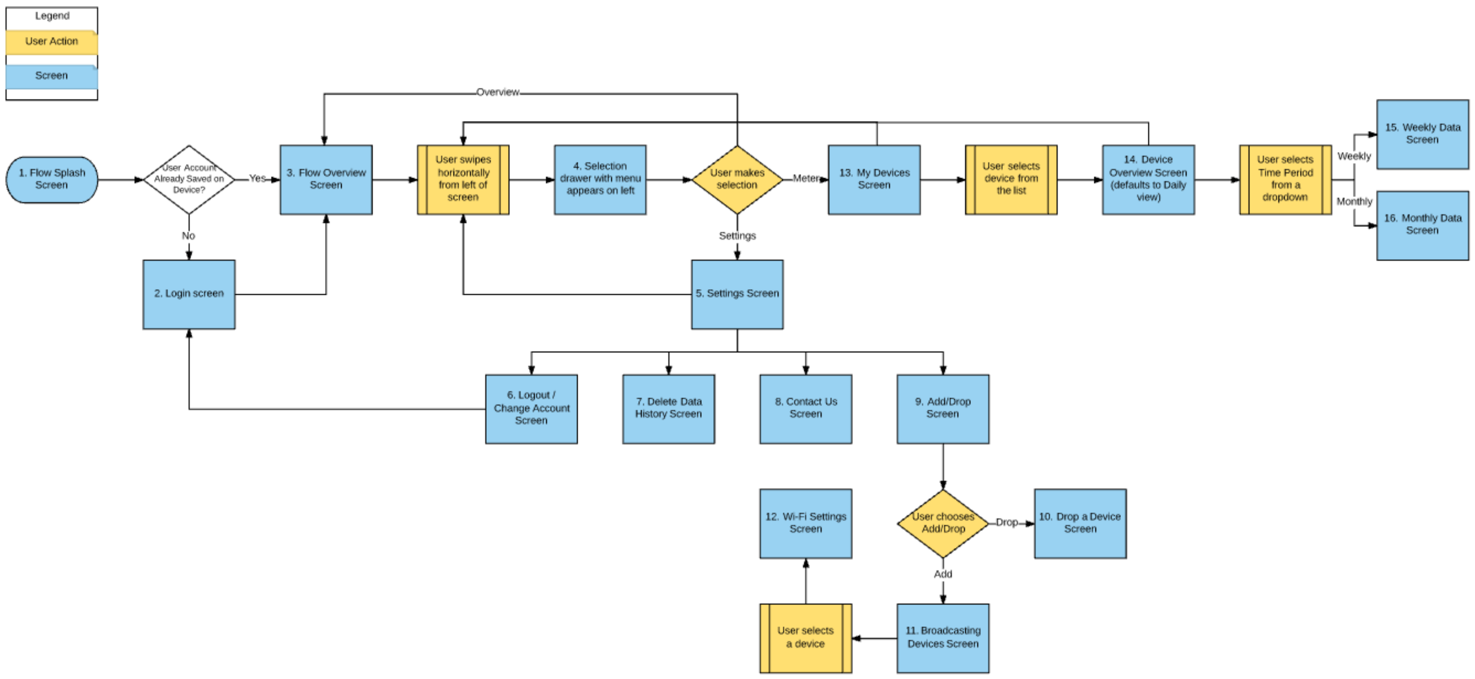

Legend
User Action
Screen
1. Flow Splash Screen
User Account Already Saved on Device?
Yes
No
2. Login screen
3. Flow Overview Screen
Overview
User swipes horizontally from left of screen
4. Selection drawer with menu appears on left
User makes selection
Meter
Settings
5. Settings Screen
6. Logout / Change Account Screen
7. Delete Data History Screen
8. Contact Us Screen
9. Add/Drop Screen
User chooses Add/Drop
Drop
Add
10. Drop a Device Screen
11. Broadcasting Devices Screen
User selects a device
12. Wi-Fi Settings Screen
13. My Devices Screen
User selects device from the list
14. Device Overview Screen (defaults to Daily view)
User selects Time Period from a dropdown
Weekly
Monthly
15. Weekly Data Screen
16. Monthly Data Screen

Figure 11: Flowchart demonstrating screens for *Flow*.
Source: Lucidchart link

3 Wireframe Layouts

3.1 Screen 1: Splash Screen

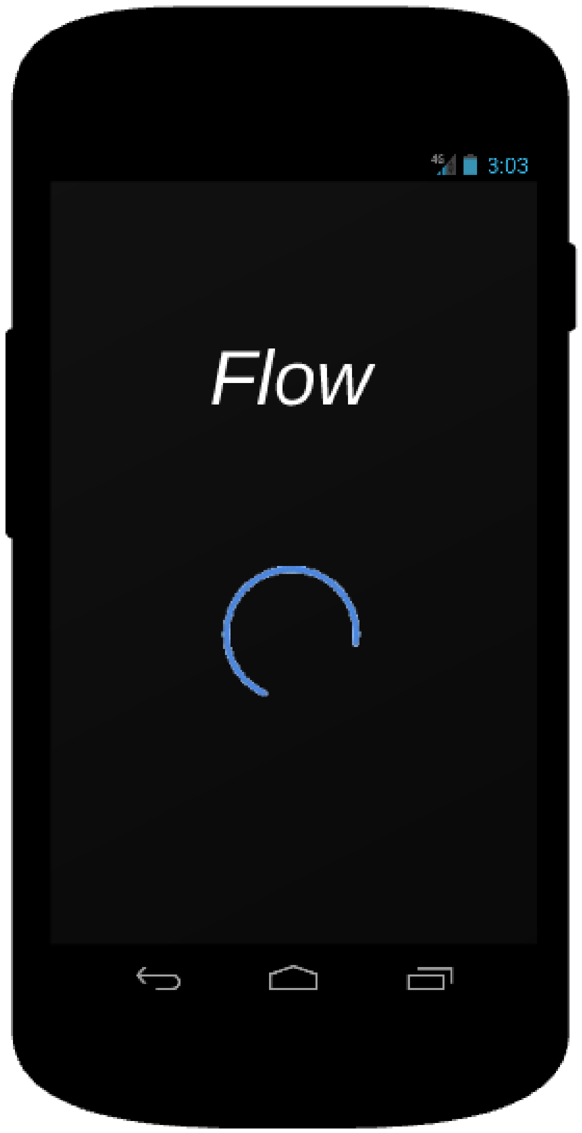


Figure 12: Wireframe demonstrating layout of Screen 1: Splash Screen
Source: [Lucidchart Link]

3.2 Screen 2: Login Page

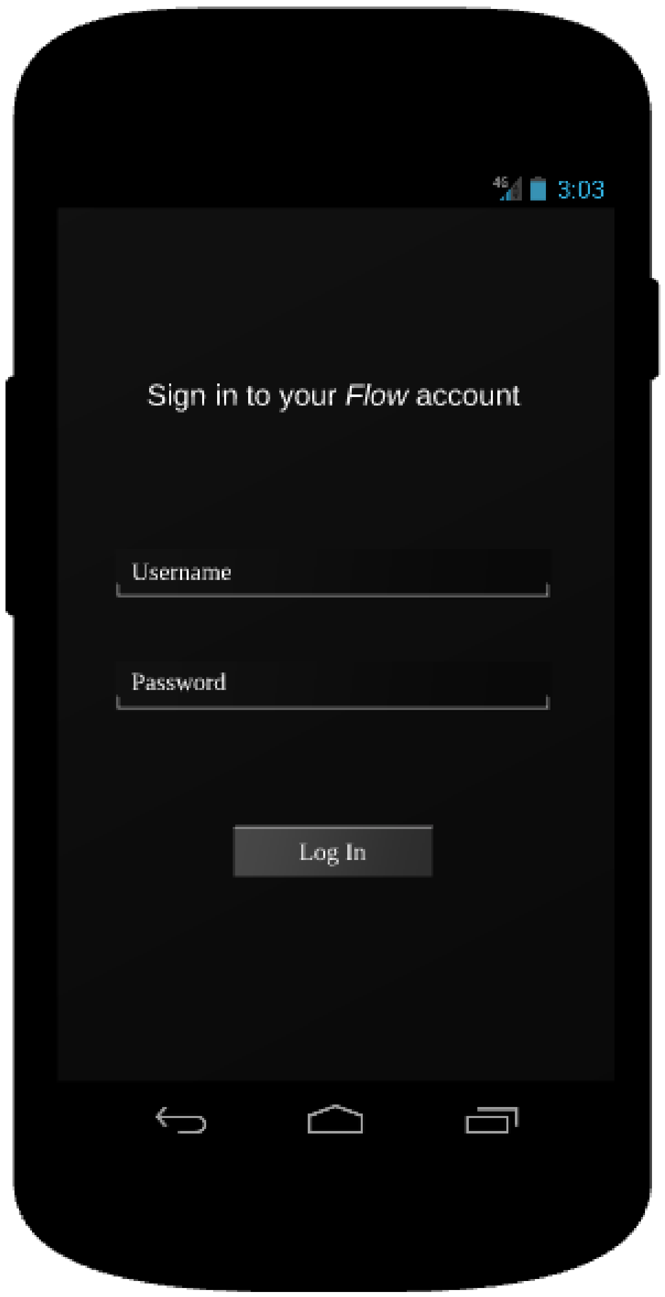


Figure 13: Wireframe demonstrating layout of Screen 2: Login Page.
Source: Lucidchart Link

3.3 Screen 3: Overview Page

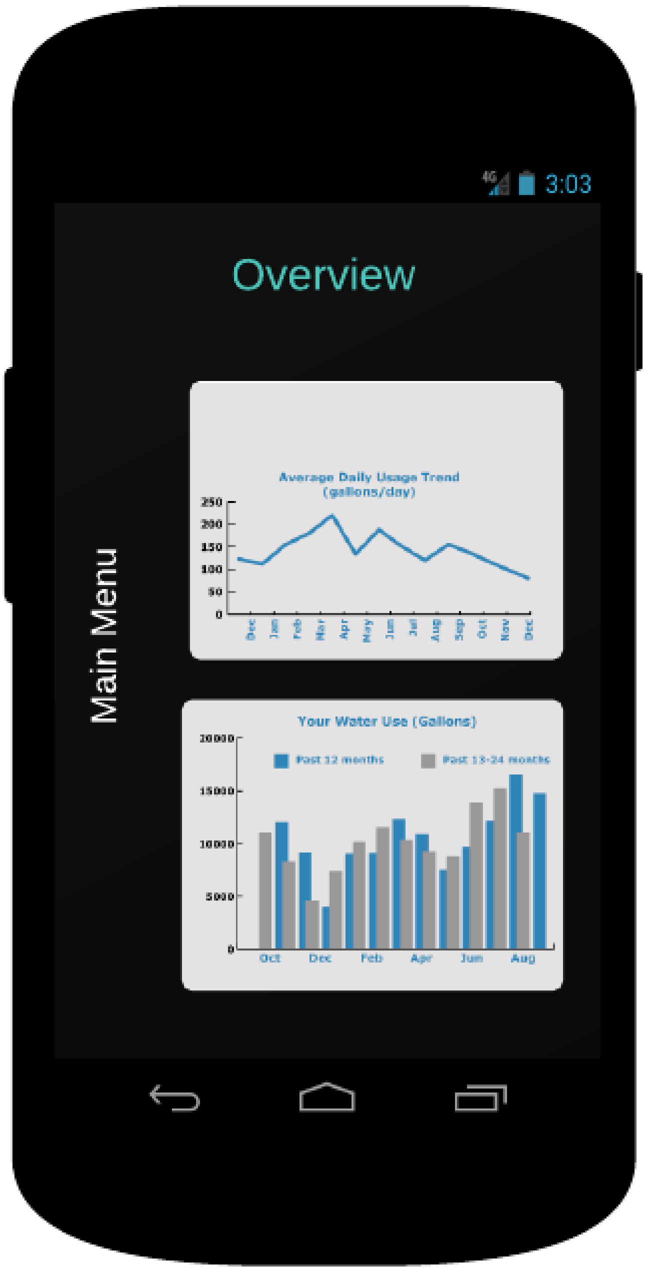


Figure 14: Wireframe demonstrating layout of Screen 3: Overview Page.
Source: Lucidchart Link

3.4 Screen 4: Main Menu Sidebar

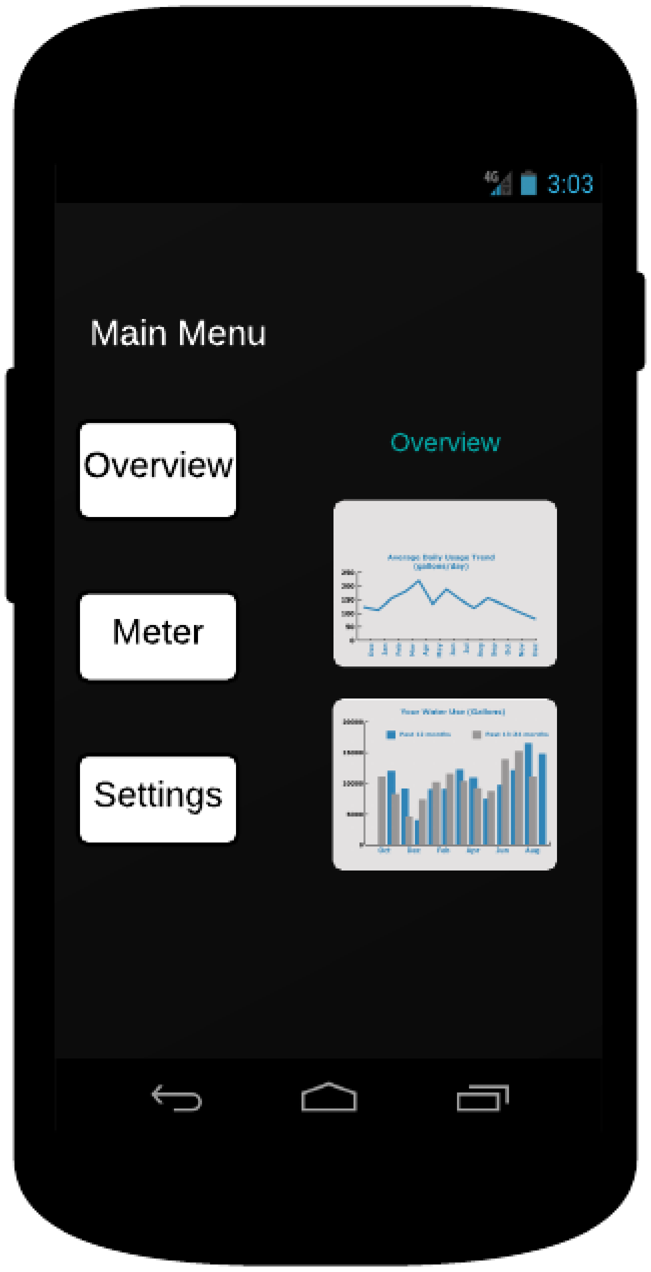


Figure 15: Wireframe demonstrating layout of Screen 4: Main Menu Sidebar.
Source: [Lucidchart Link]()

3.5 Screen 5: Settings Page

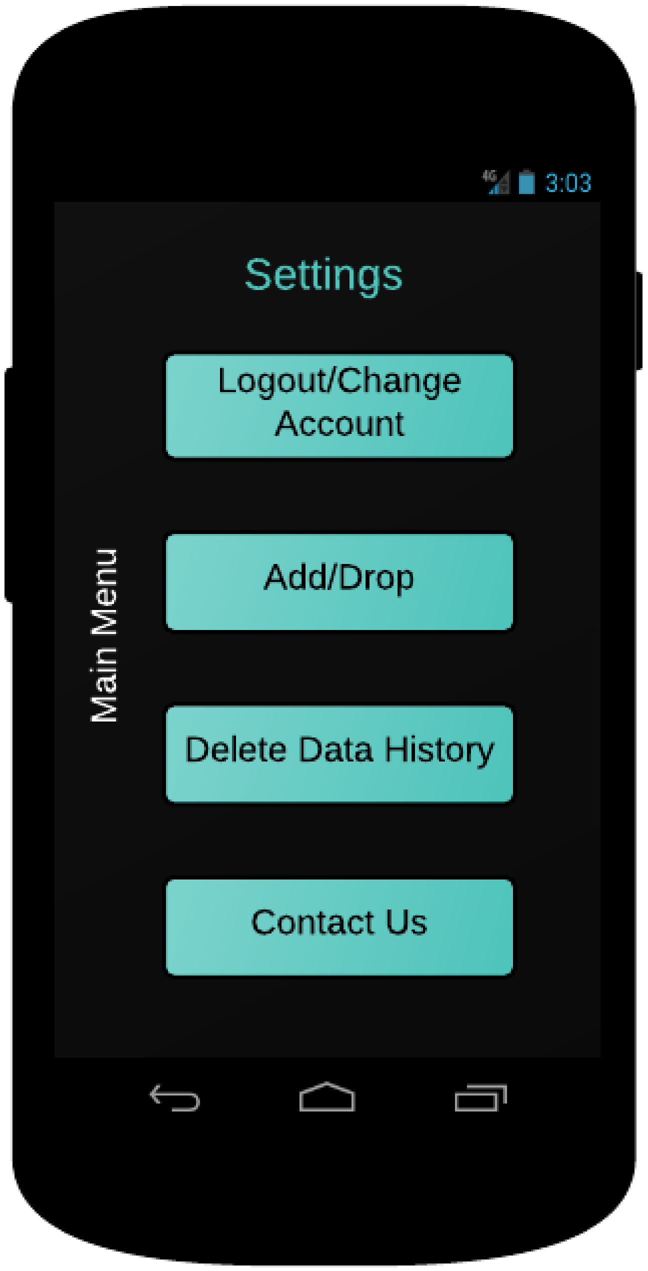
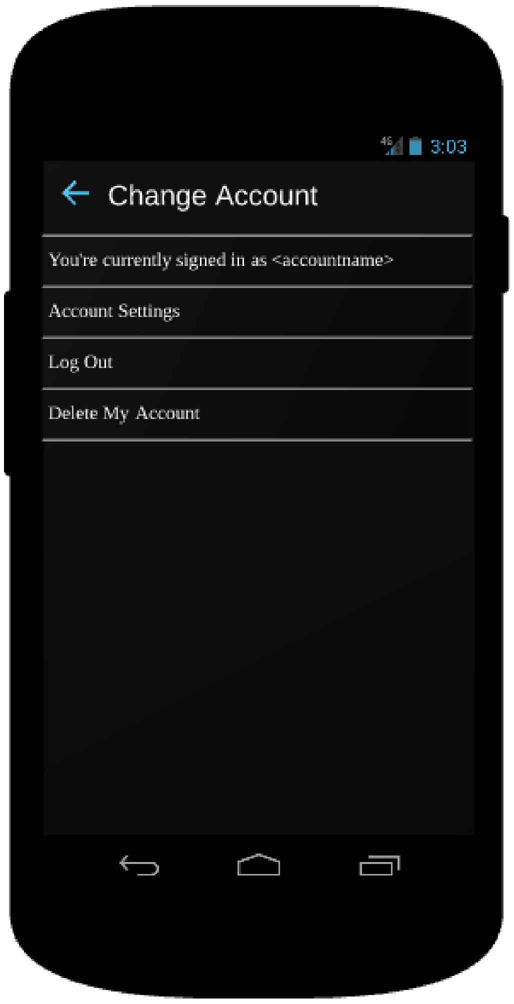


Figure 16: Wireframe demonstrating layout of Screen 5: Settings Page.
Source: [Lucidchart Link]

3.6 Screen 6: Change Account Page



You're currently signed in as <accountname>

Account Settings

Log Out

Delete My Account

Figure 17: Wireframe demonstrating layout of Screen 6: Change Account Page.
Source: [Lucidchart Link]

3.7 Screen 7: Delete History Page

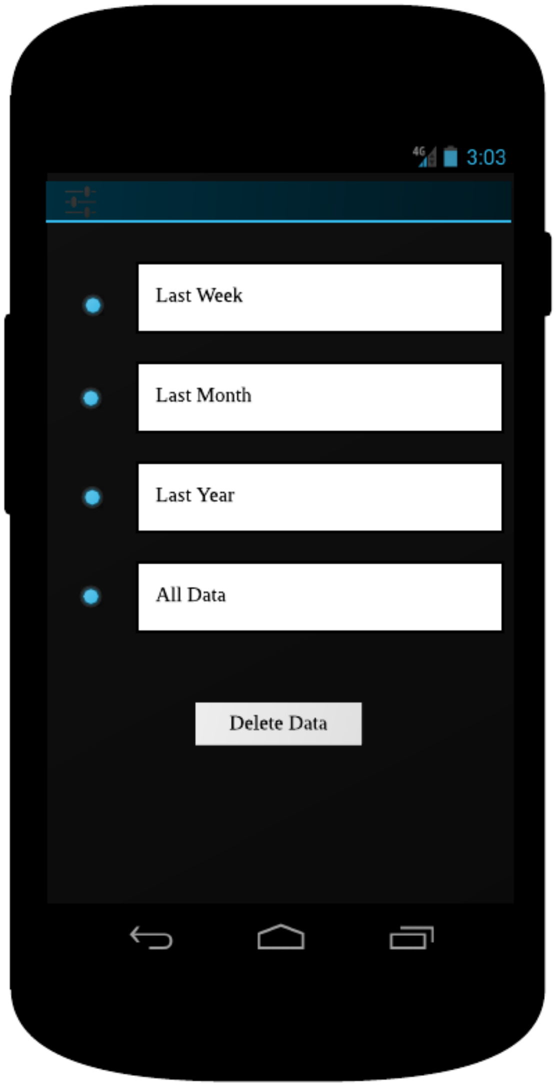


Figure 18: Wireframe demonstrating layout of Screen 7: Delete History Page.
Source: [Lucidchart Link]

3.8 Screen 8: Contact Us Page

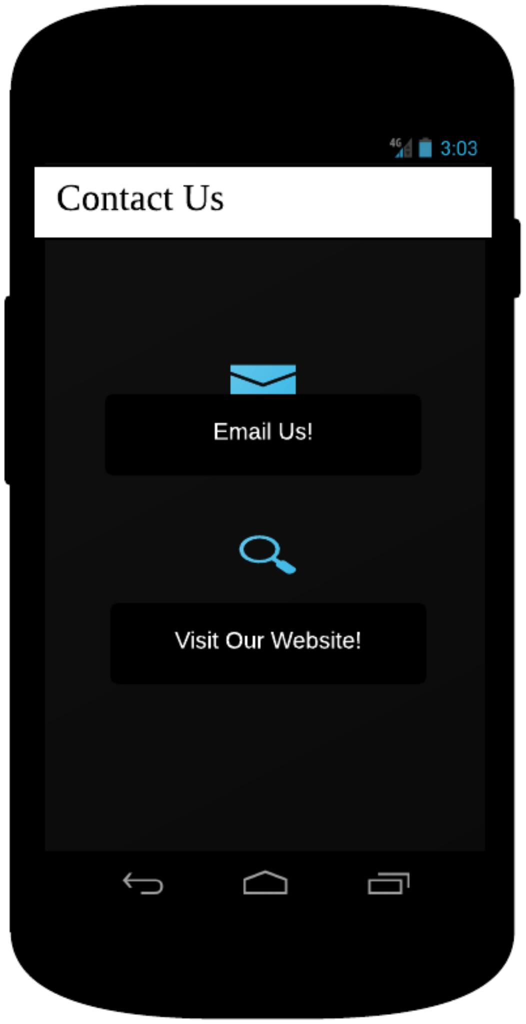


Figure 19: Wireframe demonstrating layout of Screen 8: Contact Us Page.
Source: Lucidchart Link

3.9 Screen 9: Manage Meters Page

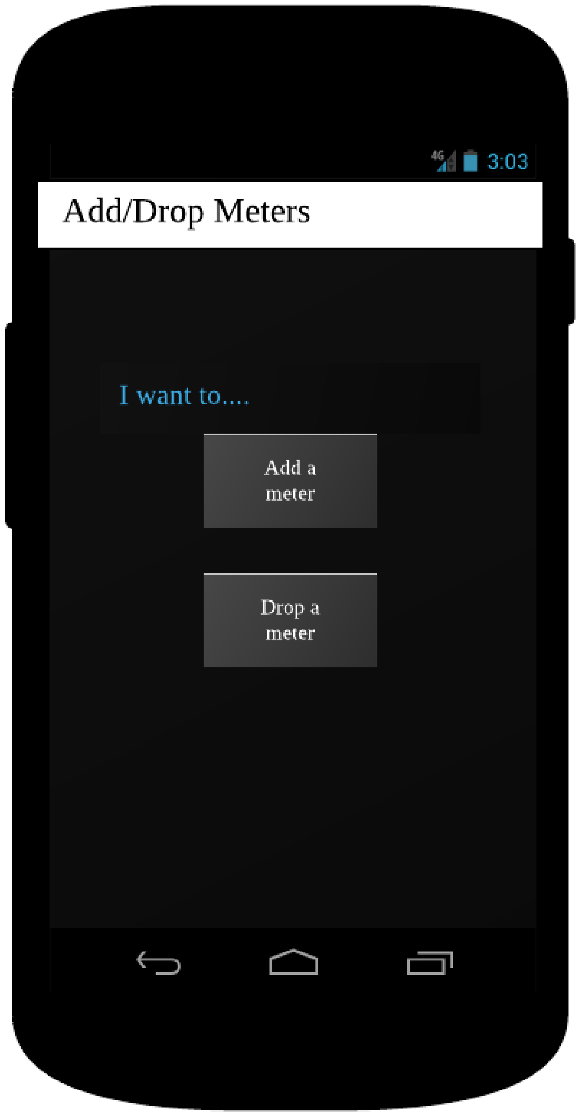


Figure 20: Wireframe demonstrating layout of Screen 9: Manage Meters Page.
Source: Lucidchart Link

3.10 Screen 10: Drop Meter Page

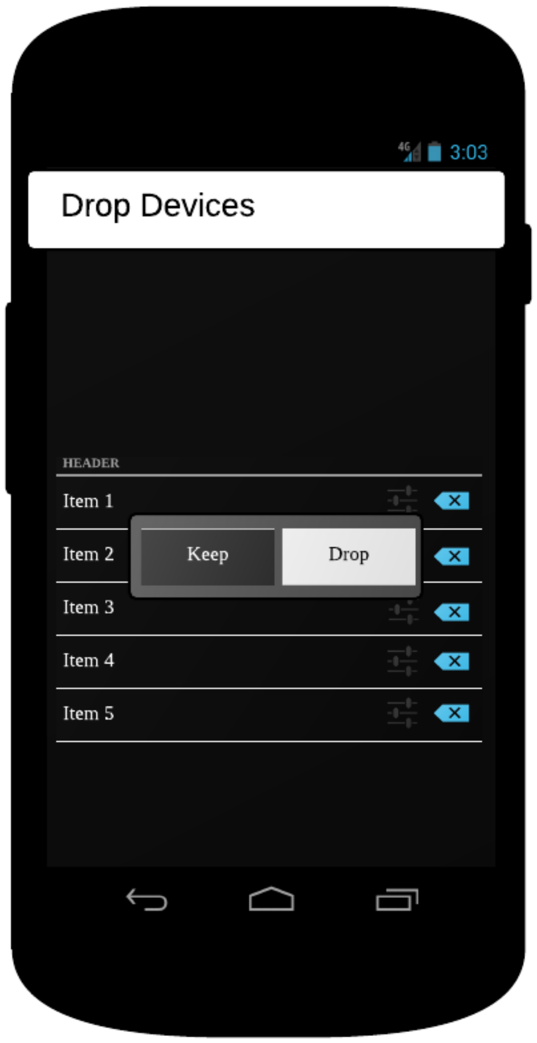


Figure 21: Wireframe demonstrating layout of Screen 10: Drop Meter Page.
Source: Lucidchart Link

3.11 Screen 11: Nearby Meters Page

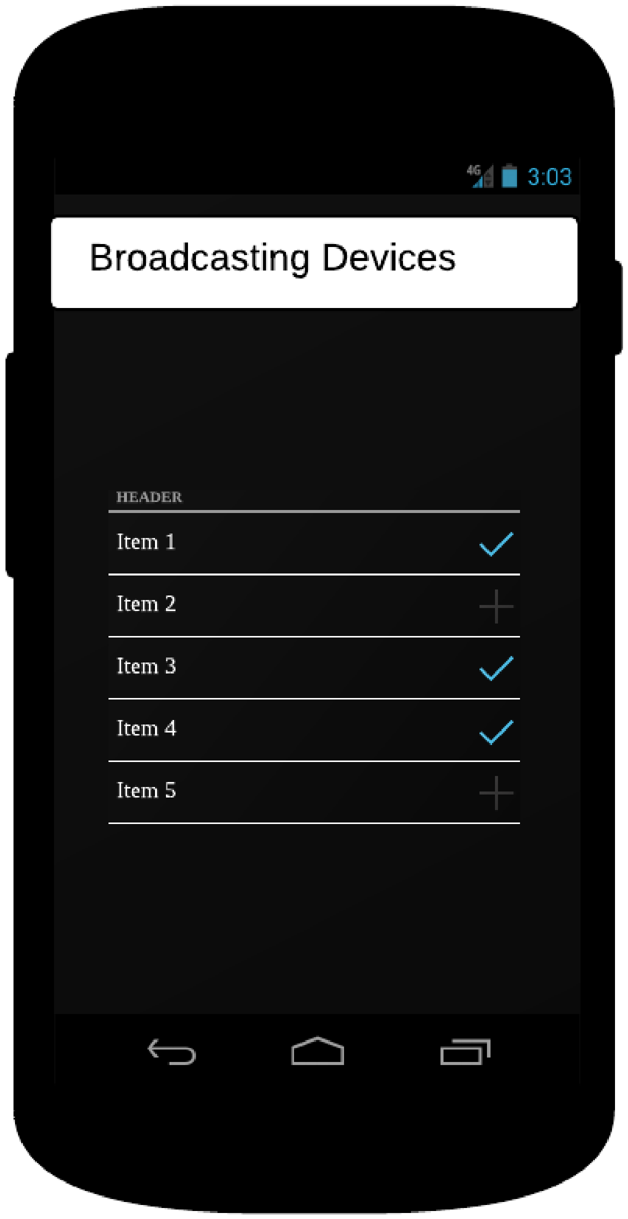


Figure 22: Wireframe demonstrating layout of Screen 11: Nearby Meters Page.
Source: Lucidchart Link

3.12 Screen 12: Wi-Fi Configuration Page

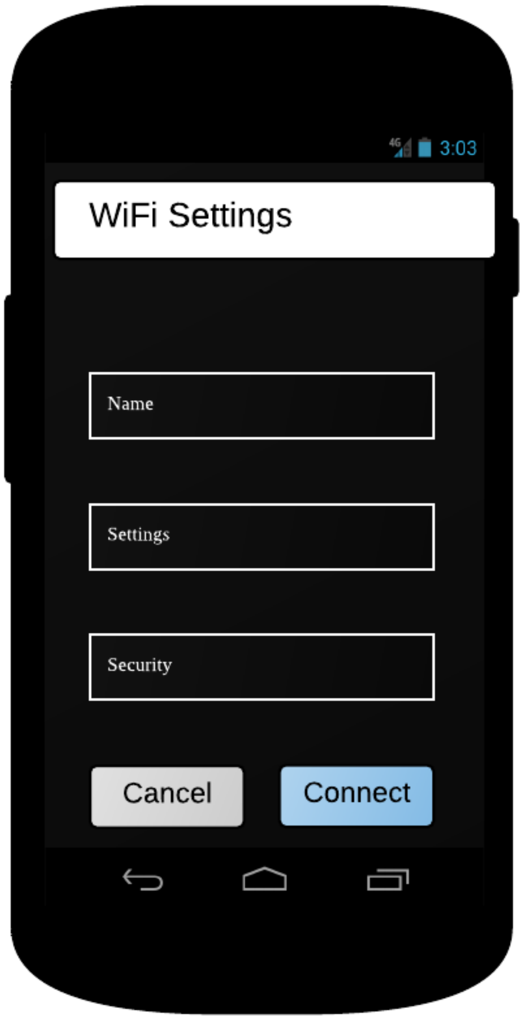


Figure 23: Wireframe demonstrating layout of Screen 12: Wi-Fi Configuration Page. Source: Lucidchart Link

3.13 Screen 13: My Devices Page

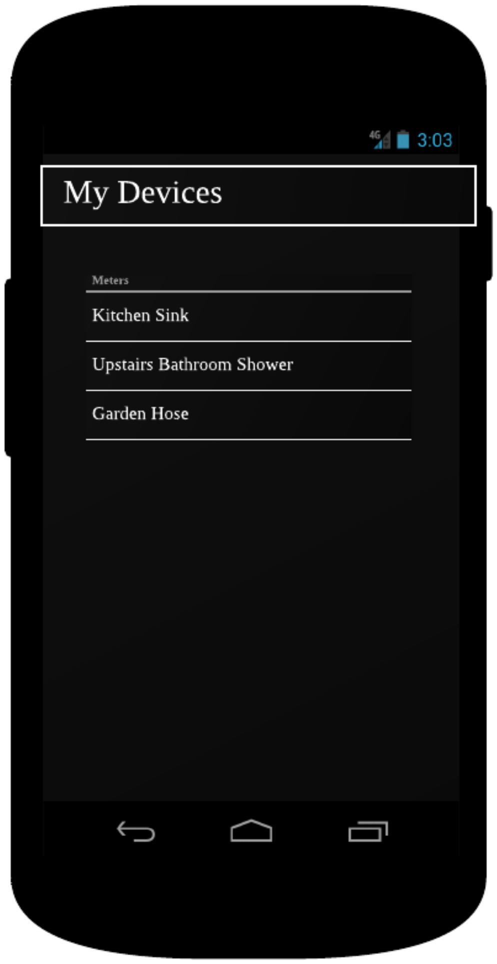


Figure 24: Wireframe demonstrating layout of Screen 13: My Devices Page.
Source: Lucidchart Link

3.14 Screen 14: Device Overview Page

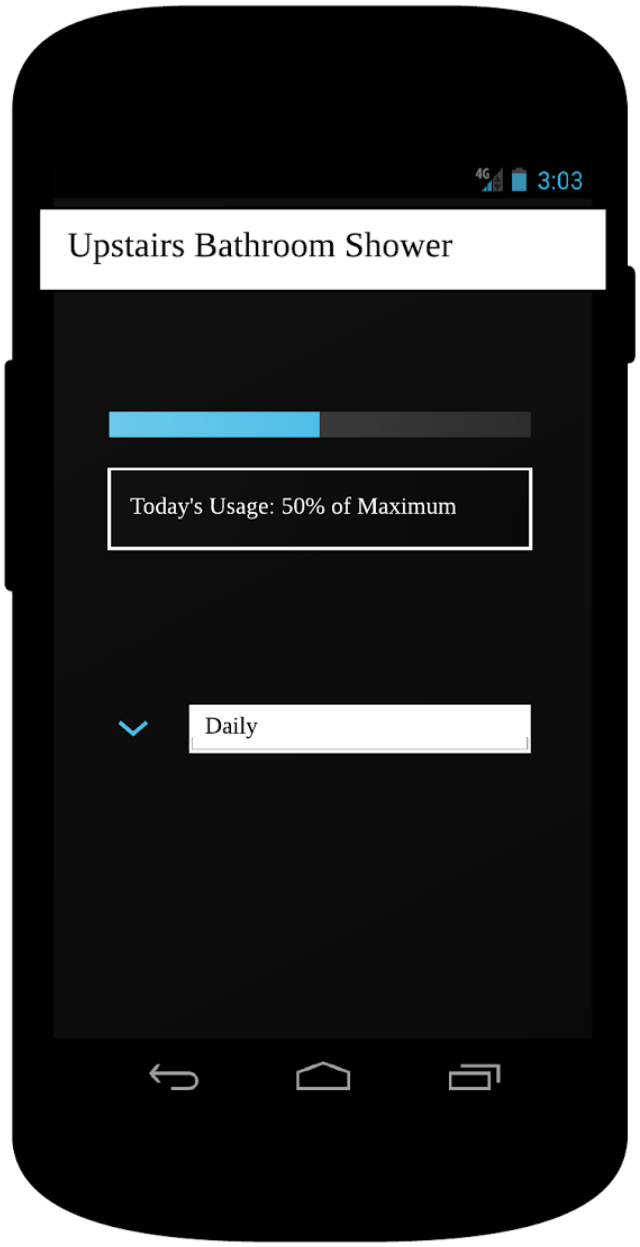
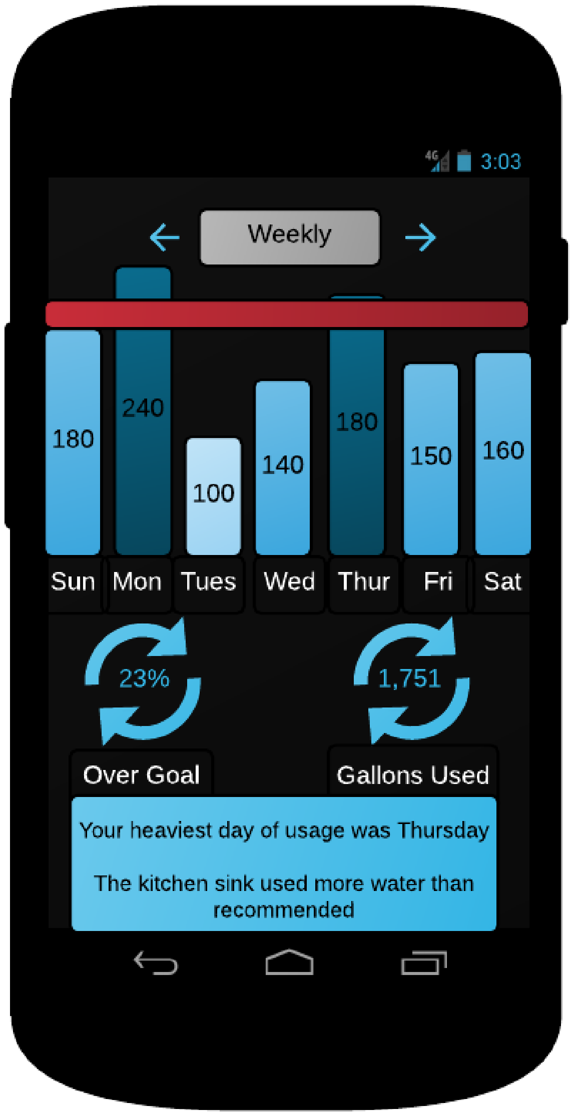


Figure 25: Wireframe demonstrating layout of Screen 14: Device Overview Page.
Source: Lucidchart Link

3.15 Screen 15: Weekly Data Page



Your heaviest day of usage was Thursday

The kitchen sink used more water than recommended

Figure 26: Wireframe demonstrating layout of Screen 15: Weekly Data Page.
Source: Lucidchart Link

3.16 Screen 16: Monthly Data Page

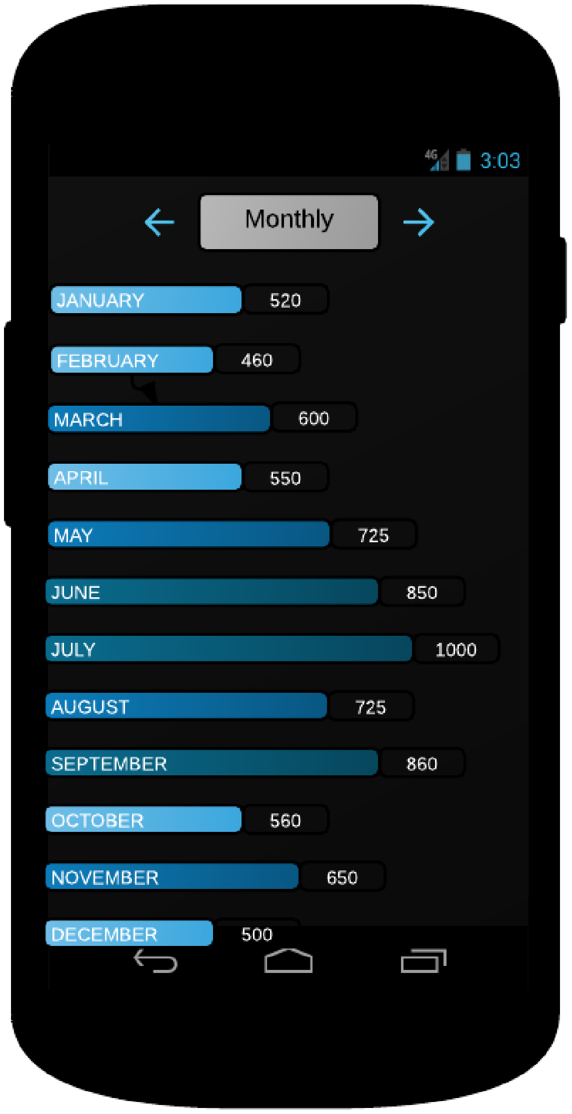


Figure 27: Wireframe demonstrating layout of Screen 16: Monthly Data Page.
Source: [Lucidchart Link]

Software Design

1 Platform Overview

1.1 Platform Requirements

Flow's architecture is comprised of three distinct entities: frontend, backend, and hardware. The mobile application will support both iOS and Android and will make use of MongoDB, Node for data storing, and an Arduino for water-volume measuring.

- Frontend:
    - React Native (User Interface)
        - Application for both iOS and Android
        - Must have interface for connecting to flow meter hardware to configure its Wi-Fi connection
- Backend:
    - MongoDB (Database)
    - Node (Backend code)
- Hardware:
    - Servers/Hosting
        - Locally host for testing
        - Use AWS (robust but more expensive) or DigitalOcean (lightweight but low-cost) for release
    - Flow Meters
        - Arduino-powered

1.2 Platform Rationale

There are an infinite number of choices for platforms to use in the construction of our various models. It is important to choose platforms that provide the optimal mix of performance, flexibility, ease of development, reliability, and cost for our particular circumstances. In the following; we will state the rationale for our platform choices.

<u>React Native</u>: (client)

- The smart home market is predominantly mobile applications and is more convenient for the user
    - Users will register accounts / track data on mobile application
    - Application will be designed on cross-platform toolchain *React Native* for both iOS and Android users
        - Cross-platform so that Flow can be targeted to both Google *Home* and Apple *HomeKit*
        - Chosen because it allows native mobile code to be used

    - Application will include a page for connecting to the Arduino (see below) to configure the Arduino's Wi-Fi connection.

Arduino: (hardware)

- Lots of support/documentation for Arduinos
- Useful for the MVP
    - Team already has experience using Arduinos and is the cheapest option for our own testing purposes (only need wifi chip and flow meter)

MongoDB: (database)

- Nonrelational database used because not much data analysis to be done besides displaying data based on time periods
    - Easier to set up than relational database
- Cheaper and easier to maintain than a relational database
- Easy to scale and add more content if needed

Node: (backend)

- Lets us have a layer of abstraction and security between the database and the outside world

2 Architecture Overview

2.1 Architecture Diagram

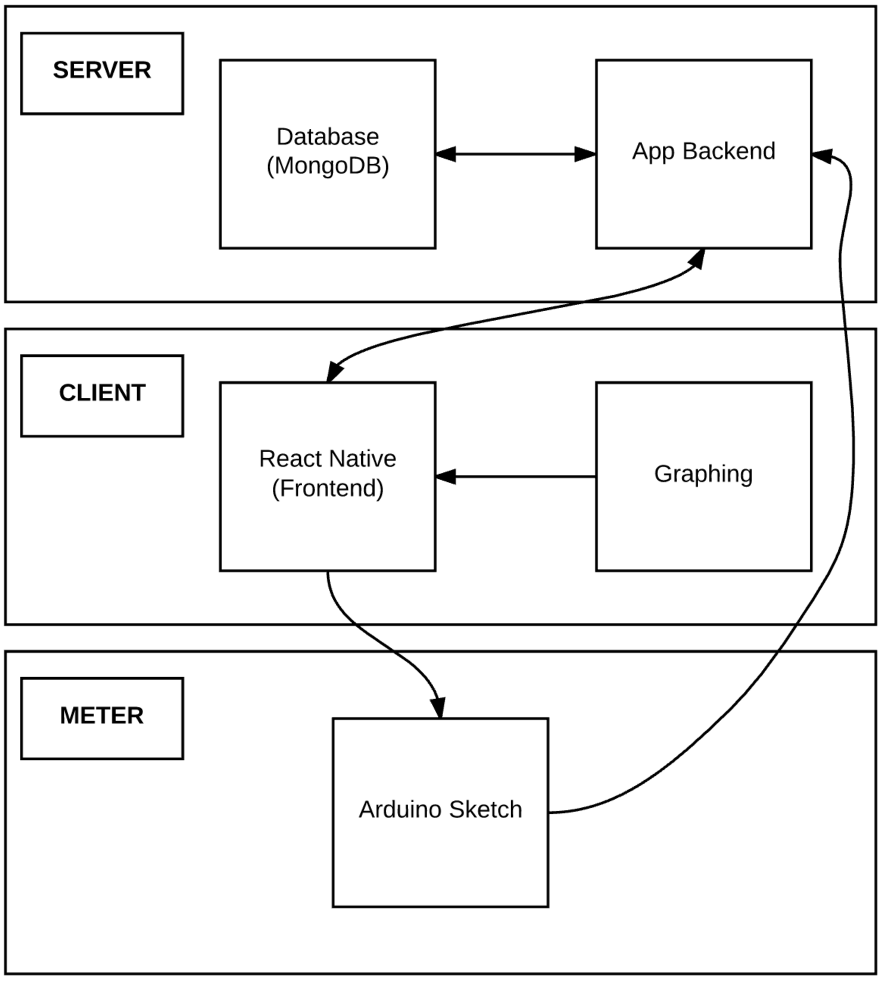


Figure 28: Flowchart demonstrating overall architecture of *Flow*
Source: Lucidchart Link

3 Object-Oriented Analysis of Use Cases

Since *Flow* has a relatively simple architecture, we initially had each team member break down their specified use case into an object-oriented schema. We went back through each use case together and identified what classes, attributes, and methods looked similar, and then consolidated those items. *Flow* will contain six major objects (User, Meter, Alert, UsageEvent, UsagePeriod, and Graph) plus two utility classes (UserService and GraphServce). Methods and attributes from the use cases were then assigned to their appropriate classes.

The following use cases are described in more detail in our functional specification document (link).

3.1 UC1 - Account Registration

“As any Flow user, I want to be able to register for my own Flow account”.

Performing object-oriented analysis on UC1 yields the following object-oriented constructs:

- **Classes:**
    - User (account)
- **Attributes/Instance Variables:**
    - Email Address
    - Password
    - Address
- **Methods:**
    - Registration - Requests and stores registration information from the user.
    - Form validation - Makes sure the data entered into the registration fields are valid

3.2 UC2 - Meter-Specific Tracking

“As any Flow user, I want to track my water usage per outlet so that I know how to best decrease water usage to conserve and save money on my water bill”

Performing object-oriented analysis on UC2 yields the following object-oriented constructs:

- **Classes:**
    - User
    - Meter (a.k.a. outlet)
    - Usage Event (e.g. flow started)
- **Attributes/Instance Variables:**
    - List of recent Usage Events
    - Water spent during the event
- **Methods:**
    - Log event start
    - Log event end

3.3 UC3 - Adding or Removing Meters

“As a Standard User, I want to be able to easily add/remove Flow meters from my network so that I can manage my changing needs”

Performing object-oriented analysis on UC3 yields the following object-oriented constructs:

- **Classes:**
    - StandardUser inherits from User
    - Meter
- **Attributes/Instance Variables:**
    - Network connectivity settings
    - User’s active meters
- **Methods:**

- Add - Adds a unit to StandardUser’s list of meters
- Drop - Deletes a unit from StandardUser’s list of meters
- Connect - Connects device to network with appropriate device settings passed in as parameters

3.4 UC4 - Water Usage Data

“As any Flow user, I want to be able to see daily, weekly, and monthly water usage data.”

Performing object-oriented analysis on UC4 yields the following object-oriented constructs:

- **Classes:**
  - User
- **Attributes/Instance Variables:**
  - Usage Amount
- **Methods:**
  - Get daily usage
  - Get weekly usage
  - Get monthly usage

3.5 UC5 - Sharing with View-Only Users

“As a Standard User, I want to be able to share specific Flow meters to View-Only Users.”

Performing object-oriented analysis on UC5 yields the following object-oriented constructs:

- **Classes:**
  - StandardUser
  - ViewOnlyUser
  - Meter
- **Attributes/Instance Variables:**
  - StandardUser’s email address (to which a confirmation will be sent)
  - ViewOnlyUser’s email address (to which a confirmation will be sent)
  - SpecificMeter
- **Methods:**
  - View visible meters
  - Share a meter

3.6 UC6 - Tracking View-Only Users

“As a Standard User, I want to track my View-Only Users’ water usage so I can monitor their consumption

Performing object-oriented analysis on UC6 yields the following object-oriented constructs:

- **Classes:**
  - Standard User inherits from User
  - View-Only User inherits from User

  - Meter
- **Attributes/Instance Variables:**
  - usageAmount
- **Methods:**
  - Share - Shares usage data to parent of View-Only User (Standard User)

3.7 UC7 - Single-User Overuse Alert

“As any Flow user, I want to receive alerts when I use an abnormally large amount of water”

Performing object-oriented analysis on UC7 yields the following object-oriented constructs:

- **Classes:**
  - User
  - Alert
- **Attributes/Instance Variables:**
  - User’s email address (to which the alert will be sent)
  - Amount of water used
  - Percent overage from regular usage
- **Methods:**
  - Pop up alert notification
  - Check for over-usage

3.8 UC8 - View-Only User Overuse Alert

“As a Standard User, I want to receive an alert that tells me which of my View-Only Users are using an abnormally high amount of water”

Performing object-oriented analysis on UC8 yields the following object-oriented constructs:

- **Classes:**
  - StandardUser
  - ViewOnlyUser
  - Alert
- **Attributes/Instance Variables:**
  - StandardUser’s email address (to which the alert will be sent)
  - Amount of water used
  - Percent overage from regular usage
- **Methods:**
  - Pop up alert notification
  - Check for over-usage

3.9 UC9 - Usage Comparison

“As any Flow user, I want to be able to compare my usage with the suggested usage put in place by local government/municipality officials.”

Performing object-oriented analysis on UC9 yields the following object-oriented constructs:

- **Classes:**
  - User
- **Attributes/Instance Variables:**
  - Usage Amount
- **Methods:**
  - Get suggested usage for the current User’s context
  - Get the current User’s usage

3.10 UC10 - Account Deletion

“As any Flow user, I want to be able to delete my account.”

Performing object-oriented analysis on UC10 yields the following object-oriented constructs:

- **Classes:**
  - User
- **Attributes/Instance Variables:**
  - *None*
- **Methods:**
  - Delete Account

4 UML-Based Software Model

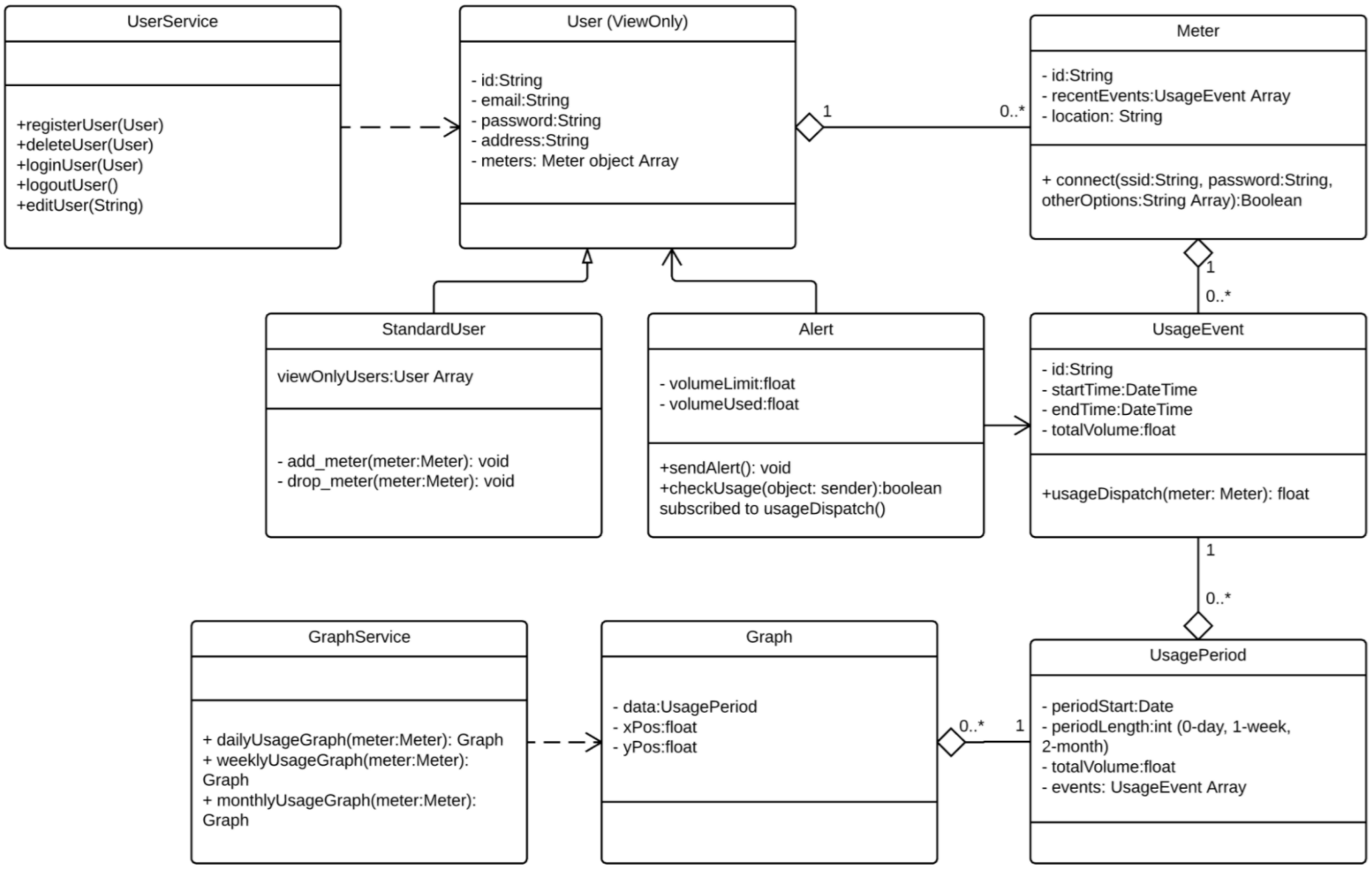


Figure 29: UML Diagram
Source: Lucidchart Link

Software Project Management Plan

1 Risk Analysis

Risk is computed as (impact × probability). Impact is defined on a scale from 1 - 5 where a 1 (5) means the risk coming true would have a minimum (maximum) impact on the project's success. Probability is also defined on a scale from 1 - 5 with the following meaning:

1. Slight possibility under unusual conditions
2. It could happen
3. 50/50 chance
4. Fair chance it will happen
5. High likelihood

1.1 Risk Analysis Matrix

The following risk analysis matrix details the most predominant risks in building *Flow*.

Risk #	Description	I	P	R
R1	Not able to create flow meter that accurately tracks data	5	3	**15**
R2	Water/Humidity proofness of hardware	5	3	**15**
R3	Time constraint possibly too difficult for current MVP	4	3	**12**
R4	Device may be expensive to produce	3	3	**9**
R5	Difficulty keeping *Flow* meters connected to the Internet	4	2	**8**
R6	Device may be difficult to install	2	4	**8**
R7	Difficulty deploying and connecting to servers	4	2	**8**
R8	Difficulty managing large amounts of data	3	2	**6**
R9	Application may be clunky, hard-to-use (failure to use framework well)	3	2	**6**
R10	May lack communication needed to successfully build project	3	2	**6**

Table 1: Risk analysis matrix listing the predominant risks and their risk score, computed as Risk = Impact * Probability (R = I * P). Risks with a risk score greater or equal to 12 are highlighted in Red to demonstrate they are high-risk and need to be TAMEd.

1.2 Risk Management Strategies

While ordinary risks can be dealt with during the development process on the fly, some severe risks ought to be dealt with before they happen. For these high-risk situations (where the calculated risk is greater than or equal to 12 in the table above), a plan to address the risk must be derived before development begins. These plans typically follow one (or more) of the following four strategies for approaching risk: Transfer, Accept, Mitigate, and Eliminate (TAME).

- **Transferring** risk involves shifting responsibility for the high-risk portion of the project to a third party.
- **Accepting** risk is simply recognizing the risk and agreeing to deal with it when it occurs.
- **Mitigating** risk reduces the impact, probability, or both of the high-risk situation, so that it becomes less risky.
- **Eliminating** risk involves doing whatever must be done to make the high-risk situation go away, whether that means dropping a feature, making a design shift, or other drastic change.

Below, we analyze our high-risk situations and provide potential plans according to these four risk management strategies.

1.2.1 Risk Response Strategy for R1

Risk R1 (Not able to create flow meter that accurately tracks data) will be TAMEd in the following ways:

- **Transfer Risk** - None
- **Accept Risk** - We will look into what our hardware options are for building a flow meter and figure out what will be the best option for us (including cost and efficiency). We will also accept that this is our biggest risk and plan our time accordingly to first solve this issue before anything else.
- **Mitigate Risk** - Look into liquid flow meters and see range of discrepancy allowed/accuracy of meters.
- **Eliminate Risk** - None

Risk strategies assigned to: **Zachary Clark**

1.2.2 Risk Response Strategy for R2

Risk R2 (Time constraint possibly too difficult for current MVP) will be TAMEd in the following ways:

- **Transfer Risk** - We will transfer as much graphic organization work to Daniel as possible.
- **Accept Risk** - If our MVP becomes truly too difficult for our time constraint, we may have to accept this and remove certain features from the MVP. According to our Product Requirements Document we will have a basic Software Application in April 2017. If this timeframe is not met, the first features to be removed would be as follows:
    - Account Deletion

    - Meter Data Deletion
    - Remove View Only User vs. Standard User and have a combined User account
    - Usage Alerts
- **Mitigate Risk** - None
- **Eliminate Risk** - None

Risk strategies assigned to: **Justin Rohweller**

1.2.3 Risk Response Strategy for R3

Risk R3 (Water/Humidity Proofness of Hardware) will be tamed in the following ways:

- **Transfer Risk** - None
- **Accept Risk** - None
- **Mitigate Risk** - We will attempt to find ways of water/humidity proofing the hardware by protecting connections using watertight cases, etc.
- **Eliminate Risk** - None

Risk strategies assigned to: **George Vine**

2 Deliverables, Components & Actions

The following structure describes the deliverables, components, actions and sub-actions of *Flow*:

- Flow meter
    - Development Environment
        - Obtain hardware prototype
            - Acquire microcontroller and flow sensor, and wire the sensor to the microcontroller's GPIO pins
            - Devise a method for running water through the sensor (simple pipe, bucket, and funnel most likely)
        - Set up a computer for developing in the language used by the microcontroller
            - Install the microcrontroller's required software (e.g. Arduino software) and libraries
            - Ensure Git is installed for project version control
    - Flow sensor
        - Measures flow volume
        - Detects flow start + stop
    - Microcontroller
        - Receives input from flow sensor
        - Parses flow event and sends it to the server

                - At the end of a flow event, aggregates an integral of the total volume of the entire flow duration, and throws out more detailed data
                - Creates event object with aggregate flow, beginning timestamp, and duration
                - Sends that event to the server backend over WiFi
            - Connects to WiFi
                - Broadcast a LiFi network that allows smartphone to connect directly for initial setup
                - Saves WiFi SSID and authentication based on input from a smartphone that connects to the setup LiFi network
                - Connects to WiFi using the saved configuration
                - When connection is lost, reopens LiFi network (in case settings have changed) and attempts on an interval to reconnect to the previously saved WiFi network (in case the settings are the same, but something else is preventing connection)
- **React Native Application**
    - Development Environment
        - Configure computer for programming in React Native
            - Ensure Git is installed for project version control
            - Install required software and libraries
                - Windows
                    - iOS development on Windows is unsupported
                    - For Android development on Windows we are required to do the following
                        - Install Node.js
                        - Install Python2
                        - Install Android Studio
                            - Make sure to choose the following upon custom install
                                - Android SDK
                                - Android SDK Platform
                                - Performance (Intel HAXM)
                                - Android Virtual Device
                        - Install Android 6.0 Marshmallow SDK
                        - Setup Android_Home environment variable
                        - Start Android virtual device
                        - Test React Native install
                - macOS
                    - iOS Development requirements
                        - Install Node
                        - Install Watchman
                        - Npm install -g react-native-cli
                        - Install Xcode from App Store

                - Test React Native install
            - Android Development requirements
                - Install Node
                - Install Watchman
                - Npm install -g react-native-cli
                - Install Android Studio
                    - Make sure to choose the following upon custom install
                        - Android SDK
                        - Android SDK Platform
                        - Performance (Intel HAXM)
                        - Android Virtual Device
                - Install Android 6.0 Marshmallow SDK
                - Setup Android_Home environment variable
                - Start Android virtual device
                - Test React Native install
        - Full requirements are listed here: React Native Requirements
        - Configure both an iOS and an Android device for testing
            - Install necessary drivers on the development computer(s) to use our own phones for testing
            - Enable app sideloading on the testing phones
    - Account Control
        - Create Login Page
            - Sends username and password back to database and is validated or not
        - Create Sign-up Page
            - Code to Create new User
                - Save/validate user-inputted data
                - Create User object
                - Write user object to database
        - Delete data associated with an account
            - Delete user object from database
    - Connect to Meter(s)
        - Services connecting to backend code
            - Requesting meter-specific, time-specific data from the database
        - Method for providing new *Flow* meters with WiFi connection settings
            - Show wifi settings page when connecting to a meter for the first time
            - Send user-inputted data to the meter
    - UI graphs that will show all the hourly/daily/monthly data
        - User control classes and UI models
            - UsageEvent objects are created and populate the graph
- **Server Backend**
    - Development Environment

        - Create and configure GitHub repository for the project
            - Make it private and add all team members as collaborators
            - Create proper ignore config, issue/PR templates, and readme
        - Install required software for a localhost test environment
            - Node.js and MongoDB
            - Ensure Git is installed for project version control
        - Set up a DigitalOcean server for testing
            - Use DigitalOcean’s Ubuntu Server starting software package
            - Install Node.js and MongoDB
            - Install Git for project version control
    - Backend code
        - Listen for incoming data from meters & apps
        - Safety checks to ensure incoming data makes sense
            - Ensure transmission matches expected format
            - Ensure transmission came from an actual meter, not spoofer
        - Store data received from meters into the database
        - Retrieve data from database and send to the app
            - Send just the data requested by app (e.g. for a certain time period)
    - Database
        - Stored procedures for aggregating data into hourly/daily/monthly (for use in the UI graphs)
            - Returns a table of the correct data subset
        - Have appropriate models for meter data
            - Ensure data models match our UsageEvent class description

3 Schedule

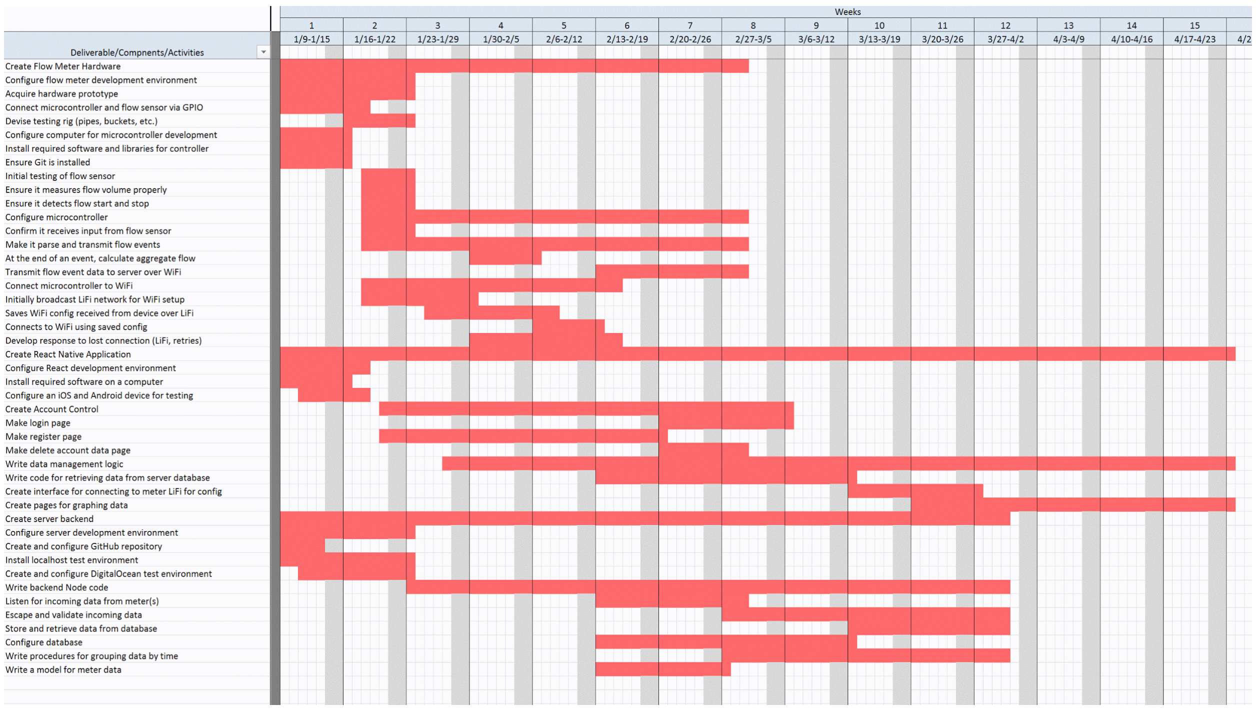


Figure 30: Screenshot of the Gantt chart schedule. For more details, click the link below.
Source: Link to Excel (via Google Docs)

Project Reflection Reports

Kennan Beard

1 Personal Reflection

1.1 On Software Development

The most important lesson I learned about software development this semester is taking things one step at a time. There comes a time during the process, mostly near the beginning, where you realize just how large of an undertaking this project will be. It can’t be done in a quick fashion, and it shouldn’t be. It should be done as a team and step by step, bite by bite. You’ll go insane very quickly if you take on too much at any one time.

1.2 On Team Work

The most important lesson I learned about working with others throughout this semester was time management. It’s difficult to keep everyone going on some of the documents, especially as Computer Science students who really don’t like writing document after document for an entire semester. All of these documents are important, and they help when taking things step by step. It can be tough to keep people motivated and doing their best work, however, but it just takes some work and reminding everyone that what we’re doing will eventually help us out in the long run.

1.3 On the Best Resources

There were several documents that I could tell would help us out a lot once we started working on them and polishing them up. On the contrary, there are some documents that just feel tedious and time consuming, but they are still of help. The documents I’ve felt will help out the most are the Software Project Management Plan, the Software Design Document, and the Functional Specifications Document. All the times in class that we had time to talk and work together as a team were incredibly helpful in terms of communication, and the first few times we pitched our project in front of the rest of the class provided valuable feedback that we had not thought of as a team.

1.4 On My Personal Strengths & Contributions

On my personal strength and contributions, I feel that I’ve done a decent job at keeping the team motivated and moving forward throughout the documents and presentations we’ve had to do. I feel that I was good at splitting up the workload somewhat evenly on most documents and that no member of the team felt too overloaded at any one time. Overall, I think that I’ve done a good job of keeping the team together and always moving forward. “Always forward, forward always.” It’s important to keep the group going and not get stuck on any one document or idea.

2 Team Reflection

2.1 Strengths

The team that I'm part of has several strengths, one being that they're great at collaboration. I've been through several group projects here at Azusa Pacific and it seems as if there's always someone slacking and not carrying their weight, but I can't say that about this team. They are great with coming up with ideas and piggybacking off of said ideas to further iterate the project and ultimately make the best MVP possible. We all think differently enough to come up with differing ideas on how to accomplish something but similarly enough to make it work and get the job done. Once the team starts on something, we finish it and we finish it well. We've taken criticism and advice with stride and done our best to keep moving the project forward into it's best possible form.

2.2 Growth Opportunities

This team, like any team, has room to grow. Our strength is our also our weakness, and we can sometimes get lost running down the rabbit hole when we get going on ideas or features we could add to *Flow*. There have often been times that we could be working on the documents that are due where we catch ourselves sitting back and talking about neat features that would be awesome to have. After a while, it takes some pulling back and realization that these features should never be part of our MVP and that we've wasted some time even just thinking or talking about them. Our team needs to be very wary of “feature creep” and keep our MVP as clear and concise as possible.

2.3 Achievement Moment

A big achievement moment that I think the team had was when we finally worked out a possible way to connect Flow to an internet connection. Ultimately the idea is very akin to how a Chromecast establishes an initial connection, but it is somewhat complex for a project such as this. It was a realm in which we were struggling with and if we didn't come up with an idea we would've been very stuck in terms of the MVP and the project as a whole. Another achievement moment was creating a successful idea and pitching it to our ZuVenturez coach who praised the idea and even stated that she would have several uses for the product herself along with knowing several other people that would love to use it.

3 Advice to Future Students

3.1 Important Courses/Content

Here is a brief list of courses that I feel have helped me in preparation for Software Engineering:
-System programming I and II: it's your entrance into something besides Java, pay attention and learn how to program in something that isn't Java

-Data Structures: The challenges in this class will really push you and test your limits, it'll help on bigger projects
-Database and Advanced Database: Every software engineering project needs a database, pay attention on how to build a good one and the fundamentals of relational vs. non-relational

3.2 General Advice

As far as general advice goes, I would say to pay attention in class and don't take team time for granted. There were times where we cut the deadline close but we really shouldn't have had to if we had only worked a bit more during team time. With that, don't forget to have fun with this process. It's a learning experience and there is a lot of room to make it your own experience depending on how you treat the class and the software engineering process. Form a team with friends that you can have fun with but make sure they'll also be willing to put in the work when things get difficult and/or tedious. I'm proud of my team and the things we've accomplished, and I'm looking forward to the implementation process come January. At the end of the day, you want to be proud of your work but be able to say you had some fun and learned a lot along the way.

Jonathan Ming

1 Personal Reflection

1.1 On Software Development

Over the course of this semester's project, I learned a great deal about the process of software development, but the lessons of good testing, risk assessment, and an appropriate idea of an MVP stood out to me as particularly important. First of all, before this project I had greatly underestimated the importance of having a methodology for testing that was planned out in detail before starting development. I didn't realize how much of an aid it can be to have a test case be almost like a goal that the project will seek to meet. Second, I learned valuable—generally, not only for software engineering—strategies for evaluating and addressing potential risk situations. Finally, I also learned by experience how important it is to work out a clear understanding of the inclusions and boundaries of the project's MVP as soon as possible. Having an ambiguous MVP leads only to confusion in writing both documentation and code.

1.2 On Team Work

Probably the most important lessons about teamwork I learned via my experience working with *Flow* is that delegation isn't something to be afraid of; it can help the whole team do better. As this semester's draft leader, I had a lot of formatting and editing to do on our documents, which meant I had less time to work on drafting the text. Since my teammates would finish their sections of text before me, I was able to ask for help on my text so that I could format graphics, flowcharts, and the text itself uniformly. It ended up working out well for our team as a whole, since I, thanks to my teammates' help, didn't end up becoming a bottleneck in our process.

1.3 On the Best Resources

The full app screen flowchart, software design document, and in class discussion of scheduling (duration/work) and risk management most helped me feel prepared to begin the development phase next semester. Creating a flowchart for every screen in the project helped keep our ideas about exactly what would appear in the application organized and well-defined, which I feel will make creating the screens much easier in the future. The software design document provided us an opportunity to define the architecture of our application, which also was encouraging to me. Finally, the lecture and subsequent discussion on scheduling and risk management also helped me feel like the task ahead of us was approachable instead of daunting.

1.4 On My Personal Strengths & Contributions

I believe that my most valuable contributions to *Flow* so far have had to do with my attention to detail. Being the draft leader, it was my job to ensure people's syntax, diction, and formatting aligned such that the documents felt uniform and professional. Between that and the role I took formatting and laying out our flowcharts, I spent a lot of time splitting hairs over spacing, sizing, and the nuances of our documentation aesthetic.

2 Team Reflection

2.1 Strengths

My team this semester showed unexpected strength in its ability to spread the workload fairly among teammates. Each person played a part in bringing us this far, and we all helped one another along the way such that no one had an unfair workload. Also, my team did a good job of paying attention to existing software processes and methodologies that we can imitate in our project, to avoid reinventing the wheel. We kept an eye out for similar projects and apps to the one we were working on, and noticed what things seemed smart and what didn't about others' work.

2.2 Growth Opportunities

As far as I've learned from this semester, our team's largest opportunities for growth are in the more practical skills needed for our project. React Native appears to be the most fitting framework for our application, however none of our team members have used even ordinary React before, much less React Native. Also, only one of our team members has experience in the hardware side of our project. It's in these practical skill areas that we need to grow the most, and I anticipate a lot of growth in that area between now and the end of next semester.

2.3 Achievement Moment

I was probably most impressed by my teammates when the devised, almost on the fly, the idea for an app backend that handles requests to and from our database. I had assumed that both the app and the meter would just have to deal with data formatting, request forming, and other low-level data management problems on their own, but the idea of a server-side application to streamline that process wouldn't have crossed my mind without my team.

3 Advice to Future Students

3.1 Important Courses/Content

Two of my previous classes at APU that prepared me most for this project were Data Structures and Database Management. What made these classes particularly suited to this semester's work were their emphasis on properly structuring an application. Whether considering the ways in which data is stored in an retrieved from a database, deciding how our application should be structured class-wise, or determining the best way to convey data between one part of our application and another, these two classes were greatly helpful.

3.2 General Advice

The best general advice I can give to future students about to engage in the software planning process is to surround oneself with teammates of various specialties. If your team has a member good at UI/UX considerations, one at databasing considerations, one at class structures, one good at diagramming and flowcharts, and one who's an especially good writer, you'll be a great deal better off than if you began with 5 masters of databasing. With a diverse skillset in the team, each member can help the others notice things they might have otherwise not considered.

George Vine

1 Personal Reflection

1.1 On Software Development

The most important thing I learned about software engineering this semester was just how much of the logical work during actual development can be eliminated during the planning phases. When going through the iterative process of identifying user stories and requirements, and creating a software specification based on these requirements allows you to do much of the conceptual design before implementation. If you can move from this stage down into class and function definitions and UML documents, you can eliminate so much of the mental heavy lifting from the implementation phase. Additionally, doing this makes it much easier to divide programming tasks, because every team member has a specification for the software, so all they must do is develop the pre-decided classes and functions.

1.2 On Team Work

The most important thing I learned about regarding team work this semester is how important it is to divide efforts into clearly defined roles for each team member. This rang true for the work that we did for the class, having a team leader, a draft leader, and a presentation leader, as well as other team members made it easy to divide and conquer when preparing our assignments. We have also begun to see how the work we have done now to plan our implementation will make it much easier to divide the implementation effort when we begin to build Flow.

1.3 On the Best Resources

The most invaluable resource that made me feel more prepared and confident was the presentations given by the other teams. These presentations helped us to see where other teams were succeeding or needing growth, which allowed us to examine our own progress under a new light. Each of the teams seemed to excel in different types of tasks, so it was great to be able to observe a team that was strong in an area that we were weaker in.

1.4 On My Personal Strengths & Contributions

As presentation leader, I believe the time I spent preparing presentations regarding our developments each week benefited the team. Additionally, because I have worked before with some of the technologies that we will be using, I believe I could provide insight during the software specification process as to what technologies would be best to meet our unique requirements. Lastly, because I have a lot of experience working on teams that deliver important products, I believe that I could contribute some knowledge regarding more obscure topics such as Agile development, requirements capturing, and risk management.

2 Team Reflection

2.1 Strengths

I believe our team's biggest strength was our analysis of requirements and use cases. I believe that we put a lot of efficient work into thinking about exactly what users will need to be able to do, and how to break these actions down into individual functional requirements and use cases. The set that we defined has served as a strong foundation for the continued conceptual development of Flow.

2.2 Growth Opportunities

I think that our team could grow in the area of pre-implementation software design. It seems like, as a team, especially while defining classes and creating the UML diagram, that we are still reluctant to fully define our system before implementation, because it seems so much more natural to design the high level logic as we are writing code. Our team needs to fully embrace the fact that designing before implementation is the most efficient way to do things.

2.3 Achievement Moment

One thing that greatly impressed me about my team was their early recognition of the fact that Flow is a perfect integration for Apple's and Google's smart home integrations. My team identified the fact that smart-home devices is a fast growing market, and that Flow fits perfectly within it. I would have not made this connection myself.

3 Advice to Future Students

3.1 Important Courses/Content

- Database Management Systems (Efficient data storage, relational vs. non-relational systems)
- Systems Programming 1 and 2 (General design and programming of systems)
- Independent Software Project with Dan (Hardware, Project design, project scheduling/management)

3.2 General Advice

Put a lot of effort and find passion in the projects that you are working on. Do research on how to fill the requirements of the project. Propose informed ideas to your team-mates, and understand why your team-mates make the decisions they do. If you are a junior or senior, the project you work on in this class can be an ace on your resume and in interviews, especially if you can talk with specific detail about the challenges that you faced and the decisions you helped to make, and if you can show passion. Additionally, pay a lot of attention to the video lectures, and the in class information. The information you learn in this class will most likely be the most valuable you learn during your education in terms of giving you the training to be a thoughtful, effective, and valuable engineer in the future.

Justin Roweller

1 Personal Reflection

1.1 On Software Development

I learned several important lessons in software development this semester. The first one was that you need good teamwork to develop large, or even small projects. Furthermore, creating your own application requires a large amount of planning, and developing concrete work to do from abstract ideas and requirements. Lastly, I learned that one of the most important skills a person can have is the ability to break down a large problem into many smaller pieces, therefore making the seemingly large projects very procedural and simple.

1.2 On Team Work

The most important lesson that I learned about working with others this semester is that you need good communication. Within our group specifically, we are constantly communicating to make sure everything is getting done. Furthermore, it is good to have a leader who keeps the group on track, and knows when we need to buckle down and produce some work. Often, in previous projects in other classes, I have simply asked to divide the work and do my part. This semester, however, even for the parts that we split up, there is a lot of review and interaction between all the team to make sure the final draft is the best that it can be, which I believe is beneficial.

1.3 On the Best Resources

Some processes, documents, lectures, and conversations that made me feel more prepared to begin implementing my product were: the ZuVenturez talks, the lectures on the different kinds of processes as well as the online lectures on product requirements, as well as in-class conversations. Some ZuVenturez talks described how to get a valid idea that could succeed, and they encouraged my group to do so. Personally, the different kinds of processes mentioned in this course also encouraged me, such as the waterfall method or the different kinds of iterative processes. Lastly, the in-class conversations that planned which platforms we will use to make our program also helped feel more prepared in a very specific way.

1.4 On My Personal Strengths & Contributions

My personal strengths and most valuable contributions were mainly my work efficiency, consistency, and cooperation with my team. Regarding work efficiency and consistency, I always did my part quickly and with precision in my projects. While a lot of the creativity came from other people in the group, I believe that, no matter the idea we chose, I did my best to make it succeed and be completed well.

2 Team Reflection

2.1 Strengths

Our group had several strengths this semester. First of all, we had good communication. We used Google Hangouts as a way to communicate and make sure people do what they need to do. Furthermore, all group members consistently show up to class, meaning we could make the most of our class work time. Also, we have had excellent leadership in that each person knows their role and does an excellent job in it.

2.2 Growth Opportunities

Our group had several growth opportunities during the semester and still has some areas where we can grow. One example of an area where we had to grow was in our distribution of work. At the beginning of the semester, when we still were in ZuVenturez, we had no communication over who and how we would do the work for ZuVenturez. This, along with other factors, ultimately lead to us dropping out of the competition. However, we quickly adapted and had much better communication for the rest of the semester. In terms of still growing, I believe we should communicate even better. Mainly what I mean by this is that if the group has found the answer to a certain question regarding what direction we are going to go in with our project, we should let the whole group know so that there is consistency in work.

2.3 Achievement Moment

I want to brag on the leadership that we have in our team. I was impressed specifically on the first project where we had assigned different people different parts to do, and was pleasantly surprised after completing my part that we had a drafts leader who had polished up our work. Along with this, we had a presentation leader who had completed a Powerpoint, and a general leader who had communicated with the graphic design artist and had come up with several images for what our logo would or could look like.

3 Advice to Future Students

3.1 Important Courses/Content

I have taken several courses that I believe have prepared me for this project so far, and several course that I believe have prepared me for next semester. This project requires a lot of teamwork, and I would recommend the early computer science courses for their projects and the teamwork that they require (cs220, cs225). Another class that I would recommend is data structures, as it seemed to round out my skills in programming, and I believe that it will be helpful next semester. Furthermore, I believe Database Management Systems has been a useful class as it adds another skill to my arsenal, and has made understanding what we will be doing next semester much easier. Lastly, this class itself has been (to me) the most useful in terms of what I might do at a real job.

3.2 General Advice

Some general advice that I would give to students is to pick your group members wisely, as you will constantly be working with them. I personally feel lucky to be in the group I am in. Furthermore, think of an idea that you actually will want to make. I also think that you should know your role in your team, and make sure to voice your opinions in your group as much as possible. Lastly, I think you should not be afraid to ask other group members or the professor for help, as they are all willing.

Zach Clark

1 Personal Reflection

1.1 On Software Development

The most important lesson I learned about software development is the amount of time that goes into just creating good documentation. I had absolutely no idea that software creation would require as much documentation as it does (especially since we are all simulating a business start-up). Not only is it crucial to plan the structure of your architecture ahead of time, but it is also necessary for when there are changes that need to be made in the future. As software developers, we must learn to budget time appropriately.

1.2 On Team Work

I learned from working with others on a project of this caliber that it is essential that everyone is vocal with one another. It is not helping whatsoever if we are not all completely transparent about how we feel. If someone has a bad idea, it is the group’s duty to either refine the idea or suggest something else. It is important to listen to each other’s ideas, but it is more important to be realistic with one another and have everyone come to an agreement.

1.3 On the Best Resources

Throughout the entire experience, it was incredibly beneficial to our team to be able to see what other teams were creating. When we saw examples from other groups, it allowed us to go back to the drawing board and to refine our product by implementing the ideas we liked from other groups and discarding the ideas that did not contribute to the project. Likewise, I’m sure that all groups benefited from this process and it was a great way for the best ideas to bubble to the surface.

1.4 On My Personal Strengths & Contributions

I believe that one of my best strengths is being able to work well with others. I am flexible to new ideas and I am always open to suggestions. I work best when I am able to get everyone on the same page and focused on the task at hand. My strongest ability is vocalizing what needs to be done and who needs to do it, because I believe it is important that we all share the load.

2 Team Reflection

2.1 Strengths

Given the complex and multifaceted nature of our idea, I think it is fantastic with how realistic our team is about our end product. We were able to get on the same page with what needs to be in the MVP and what is unnecessary because we know how difficult it will be to simply get the basics of our project working. Everyone is good at vocalizing with one another and I feel that each person brings something unique to the table.

2.2 Growth Opportunities

I think that we should focus more on the finer details of our project. Myself included, I think our group really did not want to worry about all the bits and pieces of the project because of how complex it will become.

2.3 Achievement Moment

Each member of our team is very brilliant. I think that our group is going to finish the application by the end of the year and that it will be a fully functioning product. I was surprised with how much background knowledge some of the students in our group have and it makes me happy that we are working on a project that plays to their advantage (specifically with hardware).

3 Advice to Future Students

3.1 Important Courses/Content

I think the three most important classes as a prerequisite to Software Engineering 1 are CS 225, Advanced Database, and Data Structures. In CS 225, you learn all of the very important aspects of object-oriented programming. In database, you learn the fundamentals of relational databases and SQL language, and with data structures, you learn the basics for common data structures and algorithms.

3.2 General Advice

The best advice for this class is to never get behind on the coursework. Fortunately, our groups stayed on top of everything and that really helped because each assignments builds upon the last one. They aren't always related, but if you haven't really hammered out the framework of your application, you are only hurting yourself.