

Software Requirements Specification

for

TBD

Version 1.0

Prepared by

Group Name: Team #4

|  |  |  |
| --- | --- | --- |
| Tyler Vanderhoef | 11107384 | tyler.vanderhoef@wsu.edu |
| Chris Willette | 11444663 | c.willette@wsu.edu |
| Matt Burris | 11416704 | matt.burris@wsu.edu |
| Wyatt McGehee | 11180080 | wyatt.mcgehee@wsu.edu |

|  |  |
| --- | --- |
|  |  |
| Date: | October 10th, 2015 |
|  |  |

Contents

Revisions iii

1 (TODO: MATT) Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 (TODO: GROUP) Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 (TODO: WYATT/TYLER) Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behaviour Requirements 6

4 (TODO: CHRIS) Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

5 Other Requirements 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Tyler Vanderhoef  Chris Willette | Initial Version | 10/9/2015 |

# Introduction

## Document Purpose

The purpose of this document is to outline the requirements and specifications for the DropIt (v1.0) application. This document will outline the requirements required for the application. The application is self contained, so all required information can be found here.

## Product Scope

This system is a Drag And Drop task collaboration tool that aims to simplifying work-flows. It will allow better communication with team members by adding the ability to delegate and comment on tasks. The application will give the users the ability to drag tasks into separate categories of work to improve organization. Further, this application can be used on multiple mobile platforms allowing for the most usability and connectivity.

## Intended Audience and Document Overview

This has been written with the client and software engineers in mind. It is laid out in sections, all of which contain sub sections for each important discussion topic. Section 2 contains a broader overview of the system as a whole. This section is more geared towards the client since it is written at a higher level. Section 3 is more geared towards a software engineer in that it contains more specific documentation than a client would care to read. It is optimal that the document is read top to bottom. This will ensure a better understanding of the system prior to getting into any of the more detailed discussions. If a software engineer is reading this document, they may wish to skip to Section 3 so they can read on the requirements from a more detailed view. For a client, it is recommended to start at Section 2 and skip Section 3 all together.

## Definitions, Acronyms and Abbreviations

* Azure Mobile Services (AMS)
* Application Programming Interface (API)
* JSON Web Token (JWT)
* Portable Class Library (PCL)
* Project Management System (PMS)
* Software as a Service (SAAS)
* Software Requirements Specification (SRS)
* Standard User (SU)
* User Interface (UI)
* Windows Phone (WP)
* Extensible Application Markup Language (XAML)

## Document Conventions

This SRS follows a pattern based on sections and sub-sections. In general, a new section will have a header a number by it. This signals which section you are in. From there, a section will have subsections that are numbered starting with the section number followed by subsection number (an example would be 2.1, which is section 2 subsection 1) In some sections, it is required to have a third level of subsection, in which case the number scheme would be 2.2.1 (section 2, subsection 2, subsection 1).

## References and Acknowledgments

Not applicable.

# Overall Description

## Product Perspective

DropIt it an extension of the web based PMS called Waffle.io. Currently Waffle.io does not have a mobile application and DropIt attempts to compliment the site. DropIt will extend the features of Waffle.io by providing a mobile platform for project work-flow.

The current version of Waffle.io offers integration with the website github.com; however, users must first have existing GitHub accounts and preexisting projects (repositories) on GitHub. DropIt, while modeled after Waffle.io’s main functionality, will not integrate solely with GitHub nor will it require pre-existing user accounts or projects. DropIt will allow for integration with various web-based services which are to be determined and added as DropIt evolves as a product.

DropIt will be primarily developed for Android and iOS operating systems with the remote possibility of being ported to WP. This design decision was primarily based on lack of market presence for WP. The overall design and feature set of DropIt is built on the assumption that it will be used by relatively small teams. Projects requiring large teams will be able to designate parent-child relationships between different DropIt projects, allowing a hierarchical organization for larger projects and teams.

## Product Functionality

**2.2.1 Project Based Functionality**

1. It should allow a user to view a list of projects
2. Users should be able to create projects
3. Users should be able to delete projects

**2.2.2 Admin Based Functionality**

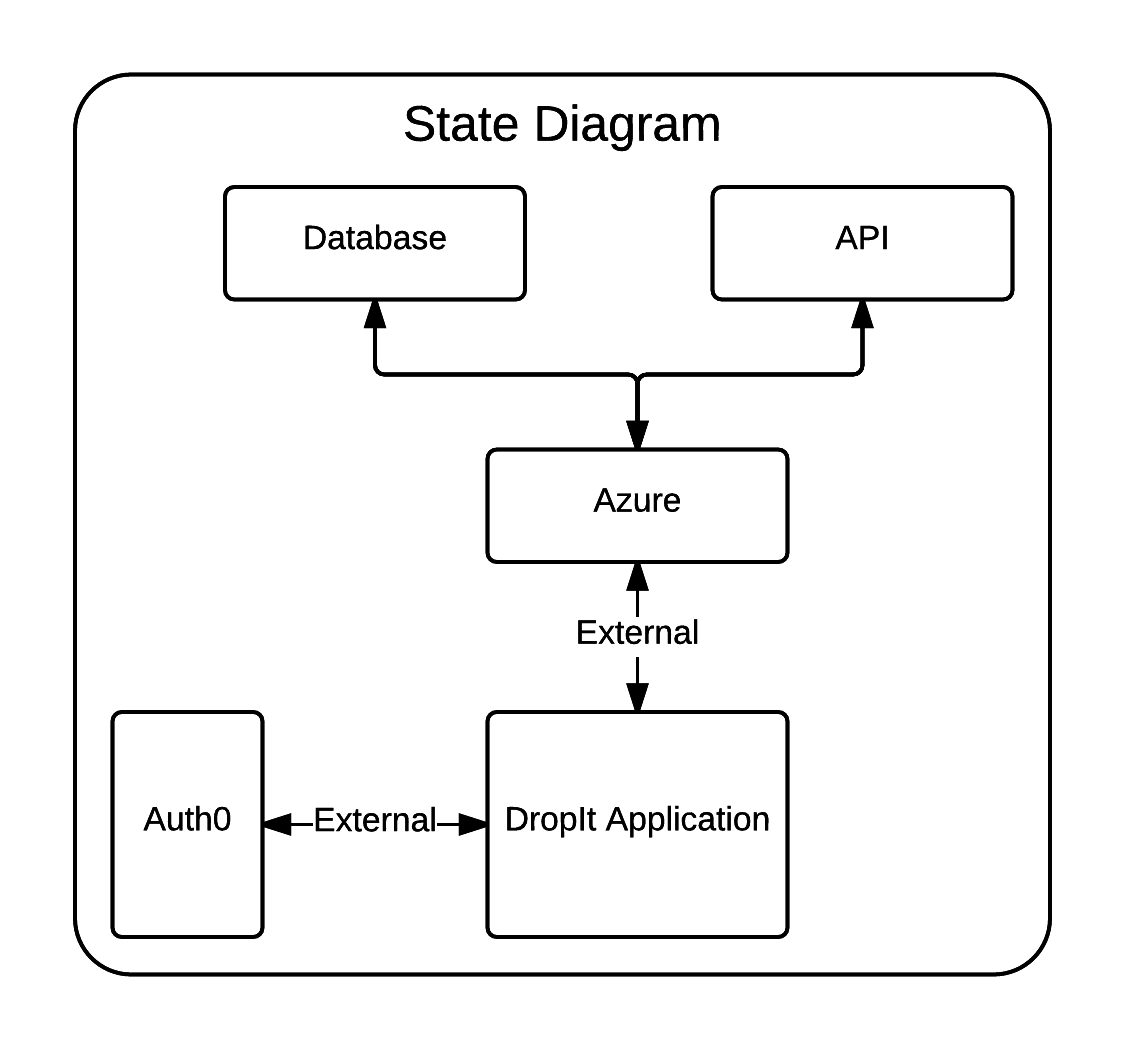
1. Must be able to add new users to a project
2. Must be able to remove users from a project
3. Must be able to create new project categories

**2.2.3 Task Based Functionality**

1. Users should have the ability to create new tasks
2. They should be able to delete existing tasks
3. They should be able to add comments to an existing task
4. They should be able to assign a task to another user in the project
5. They should be able to remove an assigned user from a task
6. They should be able to update the title and descriptions of a task
7. They must be able to drag a task into a different category

**2.2.4 Log-in Based Functionality**

1. The user must be able to create a new account
2. They must be able to sign in with an existing account
3. They should be able to request a password reset



## Users and Characteristics

All of the users of DropIt are expected to be fairly well educated as the product is geared toward increasing productivity on coordinated projects conducted in office-type settings. Despite expected educational level, the vast majority of the user base is expected to be SUs who are of limited technical or product expertise. They will primarily view new changes or postings in projects made by project team leads, but will also have the ability to make changes or post if they so desire.

Administrators are expected to be in the extreme minority, because by design DropIt will only allow a project to have one Administrator at any given time based on expected project team sizes. Once a project has been created, its initial creator is the Administrator by default until that user transfers ownership to another user, making them the new Administrator. The only except to this rule is that any member of a currently existing project can create a child project of their own. That member will be the Administrator of their child project; however, the Administrator of the originating project will also be listed as an Administrator for the child project.

## Operating Environment

DropIt will be ran on devices that support iOS 8.0 and greater, as well as Android 5.0 and greater. The application will support both phone and tablet device. Mobile platforms is the farthest reach of this application. However, it should be noted that this will not be built to run on Windows Phone.

## Design and Implementation Constraints

1. Xamarin.Forms is the API that must be used to develop and compile the cross-platform application.
2. All shared code must be written inside of a PCL.
3. The NuGet package WindowsAzure.MobileServices must be used for communication with AMS.
4. Auth0’s Xamarin.Auth0Client component (for both iOS and Android) must be used to handle user authentication and creation.
5. All UI components must be built using XAML.
6. All data should be shown and modified by using XAML Data Bindings.
7. Dependency Injection must be used to add native capabilities to the shared PCL.

## User Documentation

There will be no user manuals; however, the project’s main GitHub repository will contain tips for advanced use. The documents will be created using a combination of readme files and Wiki files on GitHub. To find information on current bugs and requests, GitHub issues will be used and open to the public.

## Assumptions and Dependencies

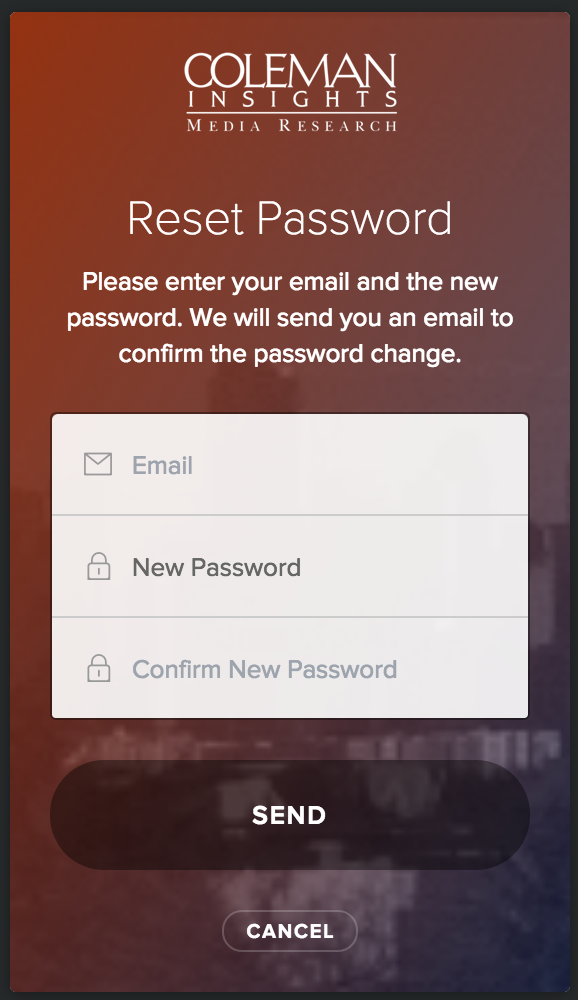
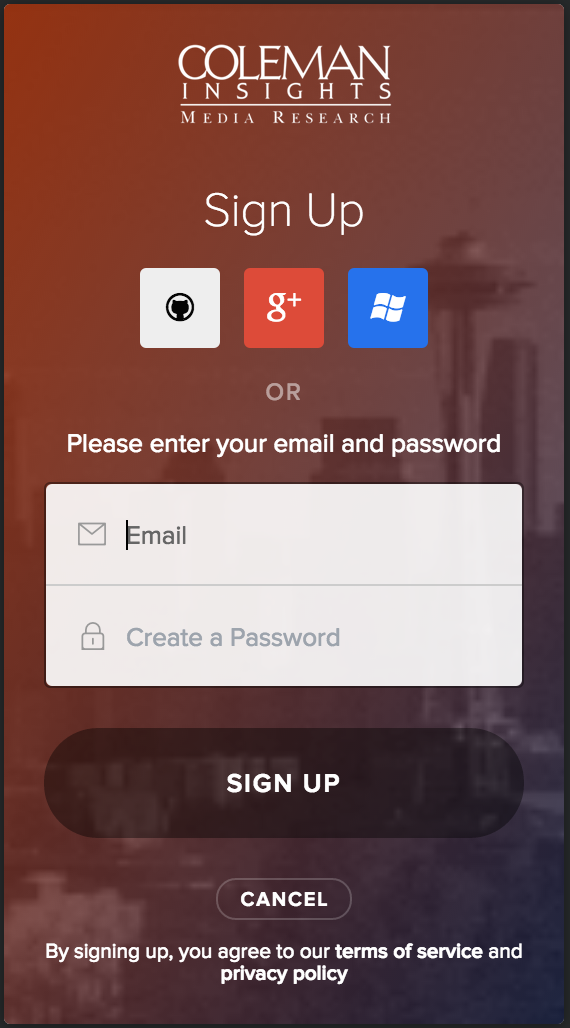
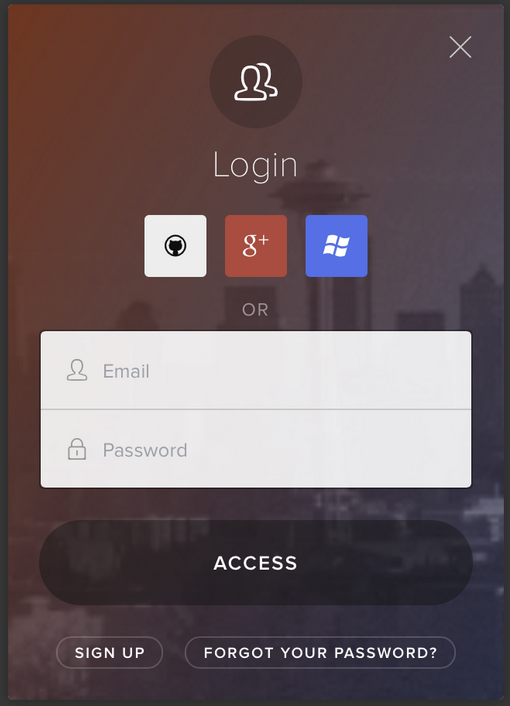
It is assumed that Azure will continue to exits for the lifetime of the project.  Additionally, it is expected that iOS and Android will not change their required development languages in the future. It will be assumed over the lifetime of the project that users will only be installing the application in the most or second-most recent versions of iOS or Android, as those are the only versions that will be officially supported at any given time. For authentication, it is assumed that the Auth0 company will continue development on their Xamarin components. And finally, the Xamarin.Forms product is assumed to continue development and support throughout the lifetime of the project.

# Specific Requirements

## External Interface Requirements

### User Interfaces

The first screen will be the log-in screen. The log-in screen (screen #1) will present the user with multiple log-in options. It will contain buttons to signify the different log-in methods. The sign-up option (screen #2) is shown by the second screenshot below. And lastly, the forgot password option (screen #3) is shown next to the sign-up screen.



Screen 1

Screen 2

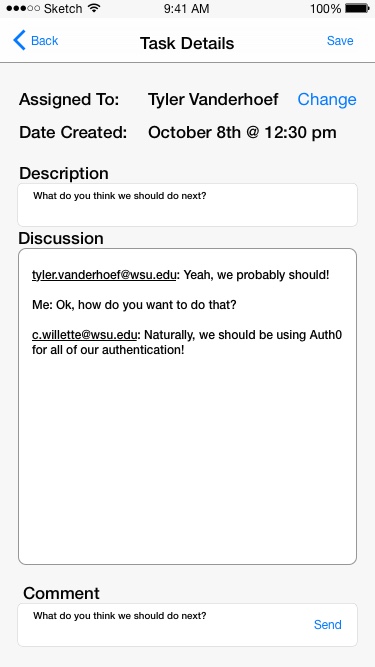
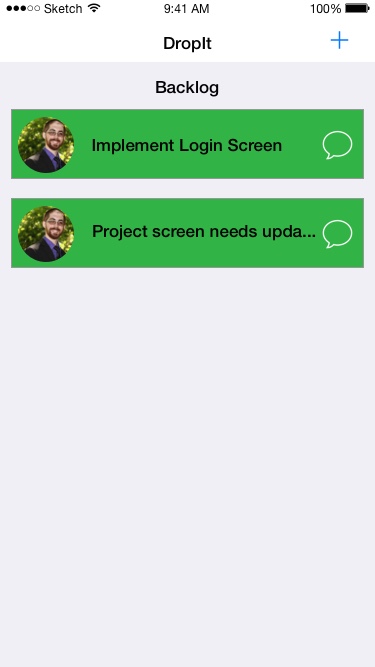
Screen 3

Once logged in, the user will be faced with the Projects screen (screen #3). The Projects screen contains a search-able list of rows that shows the projects name, as well as the number of tasks left for the project. Each project row will be tap-able, and when tapped will take the user to the Categories/Tasks page (screen #4).

The Categories page will contain a side scrolling list of Categories. Each category will have a vertical sliding list of the tasks in the current category. The tasks in a category can be re-arranged by simply dragging and dropping them on the screen. It will also allow a Task to be dragged into a different category. By tapping the plus sign in the top right corner, it will open a new Task Details (screen #6) screen to be added to the current category. Once saved, the new task will be visible. Now, to open an existing task, simply tapping on a given task will open up the Task Details page.

The Task Details page will then list out all the relevant details for the given Task. It will allow the user to change the assigned user, the description, and it will feature a discussion log.

At any point in time, if a person is an AU, then they will have the option at the bottom of the screen to view the list of users similar to the projects list. They will also have the option to add new categories to a specific project. This will be another list view like the Projects, only this time they will be grouped by Project name. Any new Category added will then be added to the list under the Projects categories slider.



Screen 4

Screen 5

Screen 6

### Hardware Interfaces

DropIt will only be interfacing with the touch screen of the phone or tablet. It will not be using any other physical interfaces to do its processing. The touch screen is required as a typical necessity of any mobile application.

### Software Interfaces

DropIt will be utilizing AMS to handle both the web API and database. The database used within AMS is a SAAS so implementation details are not known. The API of DropIt will also be using a variation of Node.js. The application will be utilizing native iOS and Android operating systems to handle network communication with AMS. For log-in and user creation operations, Auth0 (SAAS) will be used to handle authentication standards. Since the application requires touch based services, the native touch API’s will be used by the application.

### Communications Interfaces

For DropIt, web API transactions will all be encrypted over HTTPS. While logging in a user, the data will be transmitted over HTTPS and will use encrypted JWT’s sent from the Auth0 company and will be decrypted locally on the device. Since most of the application happens over web API calls, the communication is mostly just the HTTPS. No other form of communication will be used.

## Functional Requirements

### 3.2.1 Project Based Functionality

All users upon logging in to the application will have the ability to either view a list of projects they have membership in or to create a new project. Creation of a (sub)project will automatically grant that user Administrator status. If the user chooses to view the list of projects, they can then either delete any project they are an Administrator of or access the project for viewing and interacting with it.

**3.2.2 Administrative Functionality**

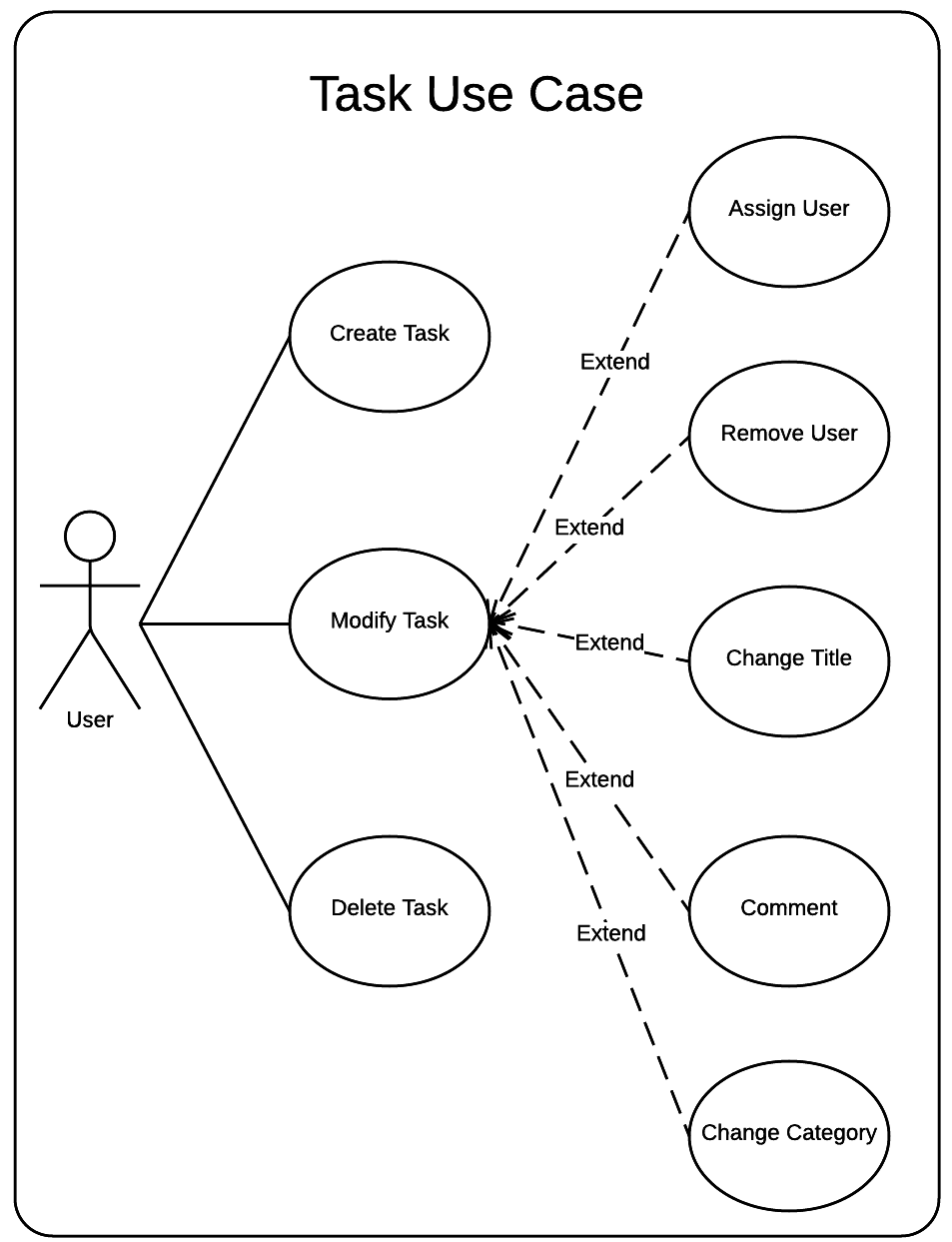
The Administrator of a project will have the ability to add or remove other users from the list of project members; transfer administrative rights to another user; and delete the project entirely. If a project has sub-projects, an Administrator of the main project may perform administrative tasks on sub-projects as well, including reassigning administrative privileges for the sub-projects.

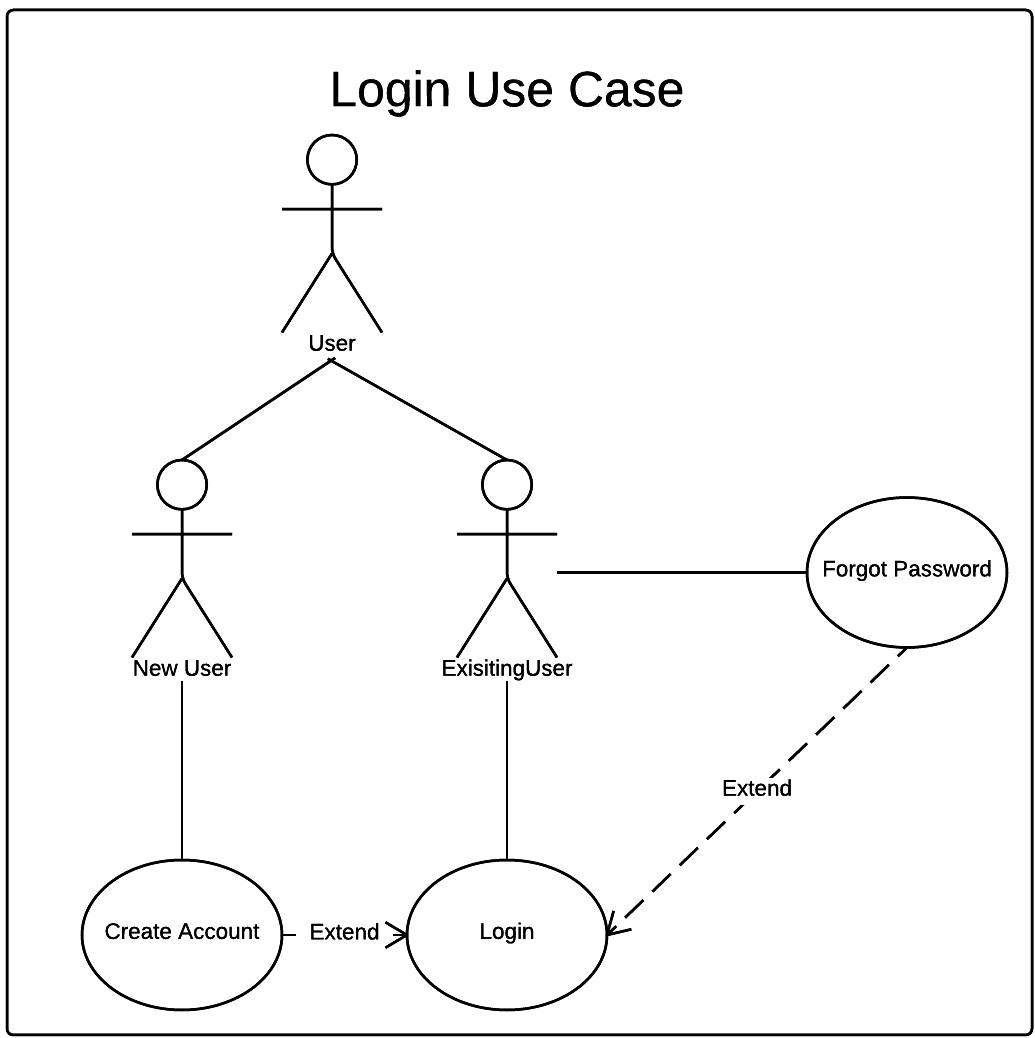
**3.2.3 Task Based Functionality**

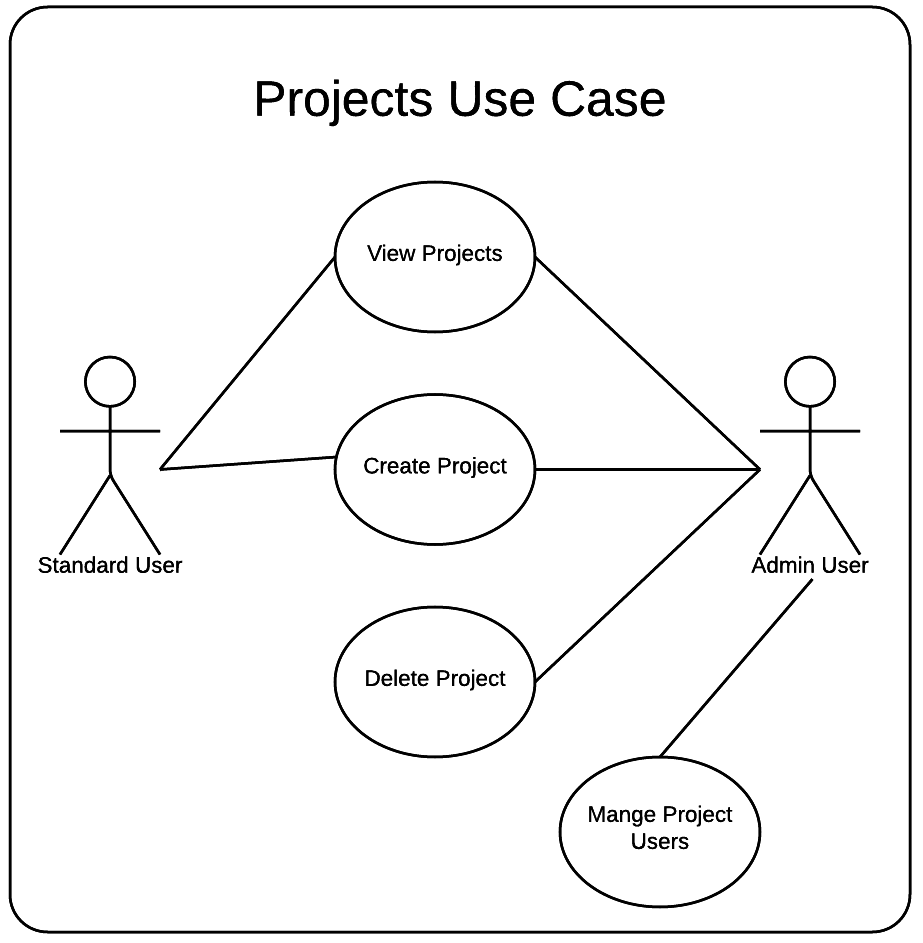
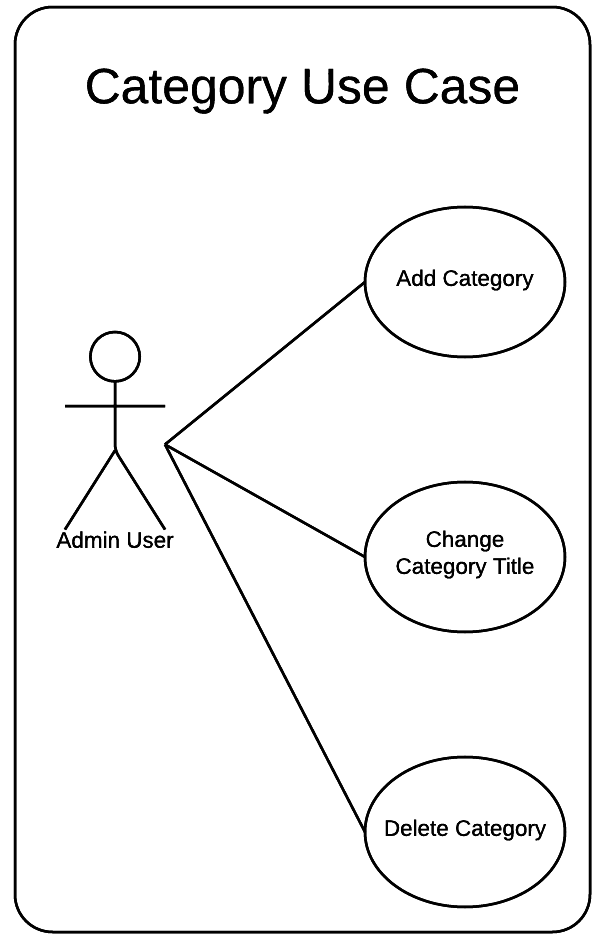
All members of a project may create, edit, view, delete, or make comments in tasks. Additionally, all users may reassign tasks to any other user including themselves or unassign the task. Editing the task may consist of changing the title or description or dragging it to another category.

## Behaviour Requirements

### Use Case View







# Other Non-functional Requirements

## Performance Requirements

1. Web requests shall not take more than 5 seconds.  This is goes toward accepted quality of service.
2. Application start-up cycle should not exceed 3 seconds (outside OS specific start-up delay). Accepted quality standard.
3. Memory consumption shall remain at minimal levels, specifically lists of projects and entries. Mobile devices are more limited in memory resources, so application must remain lean.
4. Network requests will be more conservative while using mobile networks. Mobile data for users is limited to a finite amount per month. Additionally, mobile network latencies are orders of magnitude longer than landlines, so frequent requests would time out.
5. All data operations must be asynchronous. If data operations are done on the main UI thread, the application will be slow and unresponsive.

## Safety and Security Requirements

1. Since DropIt will utilize a user log-in system, it is important not to cache user credential on the device or server. The system will be utilizing encrypted JWT’s so DropIt will need to maintain the tokens on the physical device. Because this requires storing the tokens locally, the application will encrypt the tokens prior to being saved, or it will utilize the native platform secure storage API’s.
2. The application will be needing to only allow admins to have admin abilities. Because of this, it is important that all permission granting and storing be accurate at all times so a user only has the permission they should.
3. Overall the application isn’t security critical, as it just gives access to boards for organization.

## Software Quality Attributes

**4.3.1** **Portability**

The software code base shall remain portable by using Xamarin.Forms to generate code for iOS and Android with as little effort as possible to be devoted to cross-platform development. The code will be mostly written in a PCL that allows the code to be utilized anywhere. The NuGet packaging system will allow anyone to install any client libraries that are required for the library to work. Along with using the main PCL, a separate PCL may be maintained to allow for a more generic interface into our services.

**4.3.2** **Re-usability**

The code will be written to be as reusable as possible in case other development teams would like to implement it for their own purposes. It will be written in a cross platform, well designed, and polymorphic way (base classes, interfaces, etc.) so as to encourage utilization in other systems. One library will likely need to be dedicated to use-ability standards. Certain classes and structures will be built in a generic way that allows the structure to be used in more than just one way. This encourages code re-use and allows an easy yet customizable interface into the needs of the system.

**4.3.3** **Maintainability**

DropIt will be built with maintainability in mind by designing it in a way that promotes easy development of new features and bug fixes. It will be designed in a way that allows minimal opportunity for bugs to be introduced. If bugs get introduced, the code should be easily manipulated to resolve said bug.

The software will be updated for minor fixes and will be written only to officially support latest and second-latest major iOS and Android versions. If newer versions of the two platforms necessitate updates to the code base that cause older platform versions to be deprecated, users will be required to use supported operating system versions when attempting to update DropIt.

**4.3.4**  **Feature Enhancements**

The software will accommodate possible interface or feature design changes or additions in the future. The features and layout as described in this document are subject to future change at the discretion of the development team. As mentioned in section 4.3.3, the design of the system will promote the software evolution process and should allow developers to include new features with as little friction as possible.

**4.3.5 Usability**

The software itself will be designed in such a way that there will be virtually no learning curve for users. Intuitiveness will be prioritized in the UI layout, including button and feature naming in order to reduce the need for tool tips or help documentation. The application will be built with platform idioms in mind, so the gestures that a user is used to on their device will apply to DropIt. This is key for any successful application and this application will follow those standards.

# Other Requirements



Appendix A – Data Dictionary

This section is not applicable.

Appendix B - Group Log

* Met on at 7 PM till 10:30 PM on October 8th to work on paper and revisions.
* Worked online as a group using Google Docs on October 10th starting at 8 PM.