**Homeowner Association Application**

Team Name: Team 2

Member Names:

Matthew Parra

Thomas Jay Lowe

Brian Spencer Hurst

Revanth Sai Matha

CSC480A Computer Science Capstone Project I

Project Proposal

National University, San Diego, CA

12-2019 Term

Instructor: Ben Radhakrishnan

# Contents

[**Introduction**](#_heading=h.1dgurwfm7w2) **4**

[**Background and the Need for the Project**](#_heading=h.30j0zll) **4**

[**Project Objectives and Scope**](#_heading=h.1fob9te) **5**

[**Customers and Stakeholders**](#_heading=h.3znysh7) **7**

[**Project Requirements**](#_heading=h.jo08r2q5c4cv) **8**

[**Functional Requirements**](#_heading=h.j1dbw0vwj9xz) **8**

[**Non-Functional Requirements**](#_heading=h.tyjcwt) **11**

[**High Level Use Case Scenarios**](#_heading=h.3dy6vkm) **12**

[**Project Assumptions and Constraints**](#_heading=h.ukftaigxpyw8) **27**

[**Project Delivery Methods**](#_heading=h.2s8eyo1) **30**

[**Project Issues and Risks**](#_heading=h.17dp8vu) **31**

[**Evaluation, Selection of Technology and Tools for Project Implementation.**](#_heading=h.gee0kuuv2jrl) **36**

[**Product Backlog: Prioritized Feature List and Effort Estimates (hrs.)**](#_heading=h.yiwqhm3pnfmj) **38**

[**Initial Architecture Design**](#_heading=h.fxcwjrhg875c) **38**

[**Agile Team Formation, Responsibilities and Implementation Iterations (schedule).**](#_heading=h.edcvzqo7cez6) **45**

[**Team Responsibilities:**](#_heading=h.1ksv4uv) **45**

[**Iteration Plan:**](#_heading=h.44sinio) **46**

[**Conclusion**](#_heading=h.cc4bxwatzgn2) **48**

[**References**](#_heading=h.z337ya) **48**

[**Appendix 1 - Proposal Plan Responsibility List**](#_heading=h.3j2qqm3) **49**

[**Attachments 5**](#_heading=h.25kxl3v6la6d)**0**

# 

# Introduction

Our project consists of a mobile application that allows its users to submit work requests, such as exterior and interior home augmentation, to their Homeowners Associations for approval. An application like this has several benefits in the Homeowners Association business environment. First, we believe it will improve communication between homeowners and the Homeowners Association, along with expediting the approval process for the desired work. Second, we believe the application will offer a better way for the Homeowners Association to track projects and homeowner requests within the community.

# Background and the Need for the Project

It is important for us to discover the needs of the Homeowners Association before any development can be done. The intent is to understand stakeholders’ objectives for the project and to gather requirements that help define software features and functions (Pressman and Maxim, 2015). A key part of Homeowner Associations has been the communication between homeowner’s and the association’s board of directors. The process of renovating can be a stressful and time-consuming project, but it can become even more complicated when you have to get permission for a renovation in a property that you own (Falk and Falk, 2019). Traditionally homeowners use phone communication and paper forms that need to be submitted for processing by the board of directors. The board members may also need to communicate new policies to the Homeowners Association members, which is typically done through a notification placed in the homeowner’s mailbox or taped to their door.

The paper process is inefficient and error prone, physical copies can be lost and the homeowner has no record of the progress of the approval for their request. The Homeowners Association must spend time to keep track of all requests and any physical notifications to the homeowners may also be lost. Implementing an application to handle communication between homeowners and the board of directors will provide a digital record that can be tracked by all parties. It will nearly eliminate the need to maintain and track any physical paperwork and allow all homeowners access to the most current policies.

# Project Objectives and Scope

Now that we have identified the needs, we need to also identify the project objectives and scope of the application. Since we are in the planning stage, we must gather the requirements for this application. The objective of this Homeowners Association application, from the board of directors and homeowners perspective, will be to provide a streamlined ability to submit and view work requests.

* The work requests submitted by homeowners might include, but are not limited to, changes in yard configuration, replacement of outdoor light fixtures, and the repair or replacement of electrical or plumbing components.
* Work requests can be made by providing homeowner contact information along with unit number and a description of the desired work, attaching images to their request as needed.
* The work requests submitted will be reviewed by the board of directors for pre-approval and final approval.
* Upon pre-approval, homeowners must provide contractor information, if necessary, along with contact information and appropriate licenses of said contractor. Homeowners will also need to attach a quote from the contractor, if necessary, and date(s) of intended work.
* Once the request is approved by the board of directors, the homeowner can set their appointments and begin work.
* The board of directors will also be provided with a means of viewing requests, communicating with the homeowner, adjusting approval levels, and monitoring the progress.

The scope of the Homeowners Association application is driven by the homeowners need to perform the fundamental task of submitting work requests to the association and the associations need to review requests and provide a response. The application will require the user, homeowner or board member, to login to the application. The login will establish the identity of the user and their residence or if they are a board member. Once logged in, the homeowner will be able to not only create new work requests but also review and edit their requests, as well as provide and receive feedback and documentation about the job performed. Additionally, the homeowner will be able to cancel their requests at any time during the process. Not only will the Homeowners Association application provide homeowners with a portal for making requests, but the board of directors will also be able to login and review requests, provide feedback to homeowners, and ultimately approve or deny requests.

# Customers and Stakeholders

It is important for this project to identify the customers and stakeholders. A stakeholder is anyone who has a stake in the successful outcome of the project—business managers, end users, software engineers, and support people (Pressman and Maxim, 2015). For the Homeowners Association application, the stakeholders of the project are:

* The Homeowners Association board of directors
* The homeowners
* The team developing the application

Communication will be maintained with homeowners on issues of concern or type of problems they want to commonly report to the Homeowners Association board of directors. In addition, the homeowners can correspond with us to discuss the best way to serve their needs of communication with the board of directors. The Homeowners Association board of directors will be provided a portal that allows the tracking of work orders and its progress, as well as functionality to provide feedback. Ultimately, we expect the board of directors to be the end customer and financially support the development and maintenance of this application. Having these stakeholders interact with their portion of the system will help us define if there are any overlapping portions between the groups that we as agile developers must account for during planning.

# Project Requirements

## 5.1 Functional Requirements

It is important for us to understand the requirements necessary to provide a working product. Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks (Functional and Nonfunctional Requirements: Specification and Types, 2019). The following represents the requirements for the user application, which will be utilized by the homeowner.

* **Login**

Required login will allow only authorized personnel to access the application. There will be an input field for the username and an input field for a password. A submit button will be used to process the login information.

* **Submit work request**

The user’s name, address, and contact information can be obtained from the user that is currently logged into the application. There will be an input field for the user to describe the nature of the repair that needs to be done. There will be an option to attach any supporting photos or documentation. A submit button will be used to process the request. The current date and time will be a part of the submitted information.

* **View request status**

After a successful login the view will display the most recent requests. The list will have the option to order by status (pending, pre-approved, approved, denied), or date and time of the submission.

This will also have an input field for submitting additional information. There will be an option to attach any supporting photos or documentation. An update request button will be used to process any new information.

* **View Homeowners Association Policies**

These will be read-only files maintained by the Homeowners Association board members that contain all Homeowners Association policies. A scrollable list will display all policies and rules. When the user selects one, its official documentation will be displayed.

* **Create new account**

Initial account creation will require a user’s name, address, phone number, and email. There will need to be input fields for each of these. A submit button will be used to process the account creation.

The following represents the administration application, which will be used by management members:

* **Login**

Required login will allow only authorized personnel to access the application. There will be an input field for the username and an input field for a password. A submit button will be used to process the login information.

* **View pending requests**

Upon successful login a list of all pending requests will be displayed. Selecting any request will display the information about the request. There will be an option to approve, deny, or put the request in pre-approval status. There will also be an input field that the management can use to give feedback to the requestor. There will be an option to attach any supporting photos or documentation. There will be an update request button used to submit any changes made.

* **Update Homeowners Association Policies**

This will display all current files about the Homeowners Association policies or rules. There will be an option to add a new file and delete an existing file.

* **Remove users**

A list of all users will be displayed. Selecting a user will display the user’s information and give an option to edit that information. There will be an option to delete an existing user.

* **Create new account**

Initial account creation will require a user’s name, address, phone number, and email. There will need to be input fields for each of these. A submit button will be used to process the account creation.

## 5.2 Non-Functional Requirements

Often referred to as system requirements, the non-functional requirements of this application are centered around security and reliability. Nonfunctional requirements describe how a system must behave and establish constraints of its functionality (Functional and Nonfunctional Requirements: Specification and Types, 2019).

* **Security**

The main purpose of security is to prevent unauthorized access to the application. Username and password will be used to provide security. The application will utilize an encryption algorithm to mask the password that is stored on the database.

* **Performance**

The retrieval and displaying of information will be quick and efficient. We will try for no longer than a 2 second delay from the request for information to displaying it.

* **Reliability**

The transfer of information between the user application and the admin application will be accurate.

* **Error Handling**

Proper error handling will be utilized to prevent application crashes.

## 5.3 High Level Use Case Scenarios

1. **Name of use case / title: UC1**

Login

1. **Description - some short text describing the scope.**

The login is the entrance to the application. Only after successfully logging in will the rest of the application features be made available. This will compare user input to stored information. If there is a match, then access is granted; otherwise, an error message will display notifying the user that the information input was not correct.

There will also be an option to create a new account for first time users.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The user of the application is the primary actor. It could be a homeowner or management. The application is also an actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

Only that the application is running without error.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The user enters their username and password into the appropriate input fields, and then presses a login button.
3. The application connects to the server and access the table with usernames and passwords.
4. The application searches the table for a match to the user input.
5. A match is found, and the user’s information is loaded from the database and displayed.
6. **Extensions - flow of application when it deviates from success scenario's flow:**
7. Alternate flows - other options of correct flow

The user selects the option to create a new account.

1. Exception flows - flow of events for when things go wrong
2. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
3. A match is not found. The application displays an error message letting the user know that the either the username or password was incorrect.
4. **Success guarantee (aka. Post condition) - state of application after everything is done**

The user’s main page displaying their requests is displayed, or the admin’s main page displaying all pending requests is displayed.

If the create new account option is selected, then a new account form will be displayed.

1. **Name of use case / title: UC2**

Submit repair request

1. **Description - some short text describing the scope.**

There will be an input field for the user to describe the requested repair. There will also be an additional option to attach a photo or documentation that will support the request.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The homeowner is the primary actor. The application is another actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The user has successfully logged into the application.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The user inputs data into the input field describing the repair request and presses a submit button.
3. The application places the user input and user information into the database table for requests.
4. A message lets the user know that the input was successfully submitted.
5. **Extensions - flow of application when it deviates from success scenario's flow:**
6. Alternate flows - other options of correct flow

The user could also attach a photo or document to the message prior to pressing the submit button.

1. Exception flows - flow of events for when things go wrong
2. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
3. If the user presses submit without any input in the input field, then a message telling the user that a blank request will not be sent will be displayed.
4. **Success guarantee (aka. Post condition) - state of application after everything is done**

After a successful submission the user’s main view will be displayed. The newly submitted request should be displayed with a pending status. The database table for requests will be updated with the new request in it.

1. **Name of use case / title: UC3**

Create a new account

1. **Description - some short text describing the scope.**

There will be an input fields for the user to fill in their name, address, email, phone number, and password. There will be a submit button that will begin the processing of the user input.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The user of the application is the primary actor. It could be a homeowner or management. The application is also an actor

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The new account form is properly displayed.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The user inputs data into each of the input fields and presses the submit button.
3. The application places the user input into the database table for member information.
4. A message lets the user know that the input was successfully submitted.
5. **Extensions - flow of application when it deviates from success scenario's flow:**
6. Alternate flows - other options of correct flow
7. Exception flows - flow of events for when things go wrong
8. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
9. If the user presses submit with any input field blank, then a message telling the user that all fields need to be filled will be displayed.
10. Some validation to ensure names only contain characters, phone number only contains numbers, and emails have correct format can be done. If any of these do not have appropriate input a message letting the user know to correct it can be displayed.
11. **Success guarantee (aka. Post condition) - state of application after everything is done**

After a successful submission the login view will be displayed. The database table for members information will contain a new entry with the data input by the user.

1. **Name of use case / title: UC4**

View Homeowners Association policies and rules

1. **Description - some short text describing the scope.**

This is a list of all policies and rules that the user can access for reference, but not be able to edit.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The homeowner is the primary actor. The application is another actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The user has successfully logged into the application.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The application displays a scrollable list of all policies and rules.
3. The user selects a rule or policy from the list.
4. The application displays the information for the selected rule or policy.
5. The user can use the back button to return to the policies and rules list or press the home button to return to the requests view.
6. **Extensions - flow of application when it deviates from success scenario's flow:**
7. Alternate flows - other options of correct flow
8. Exception flows - flow of events for when things go wrong
9. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
10. If the user presses submit without any input in the input field, then a message telling the user that a blank request will not be sent will be displayed.
11. **Success guarantee (aka. Post condition) - state of application after everything is done**

This does not make any changes to any database. When the user is finished with reviewing documents then the requests view will be displayed.

1. **Name of use case / title: UC5**

View request status

1. **Description - some short text describing the scope.**

If a request status is selected from the list, then all its information will be displayed. The user’s name who initiated the request, the time the request was sent, all supporting documents that were attached to the request, and the description of the work requested. The current status of the request will also be displayed. If there has been a review by the management, there could potentially be feedback in the form of additional documentation, or just a message. There will also be an input field to allow the user to add any additional information to the request. This will include the ability to attach photos or documentation. There will be a button to update the request.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The homeowner is the primary actor. The application is another actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The user has successfully logged into the application and has selected one of the requests in their requests list.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The application displays request information.
3. The user can fill in the input field as well as attach photos or documentation to the request and then press the update request button.
4. The application will update the request in the database table for requests with the additional information provided.
5. **Extensions - flow of application when it deviates from success scenario's flow:**
6. Alternate flows - other options of correct flow

The user can press the back button to return to their list of requests.

1. Exception flows - flow of events for when things go wrong
2. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
3. If the user presses update request without any input in the input field or any attached document, then a message telling the user that a blank update will not be sent will be displayed.
4. **Success guarantee (aka. Post condition) - state of application after everything is done**

If additional information was submitted, then the requests entry in the database will be updated with the additional information. The requests view will be displayed by the application.

1. **Name of use case / title: UC6**

View pending requests

1. **Description - some short text describing the scope.**

The application will display a list of all pending requests. If an admin user selects a request from the list, then all the request’s information will be displayed. The user’s name who initiated the request, the time the request was sent, all supporting documents that were attached to the request, and the description of the work requested. There will be an option to set the status of this request to either approved, denied, or pending. There will be an input field for the admin user to give additional information back to the requestor about the request. There will be an option to attach photos or documentation to the request as well. An update request button can be pressed to update any changes made to the request.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The reviewing manager is the primary actor. The application is another actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The admin user has successfully logged into the application and has selected one of the requests in the list of all pending requests.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The application displays requests information.
3. The admin user can fill in the input field, attach photos or documentation to the request, as well as change the status of the request and then press the update request button.
4. The application will update the request in the database table for requests with the additional information provided.
5. **Extensions - flow of application when it deviates from success scenario's flow:**
6. Alternate flows - other options of correct flow

The admin user can press the back button to return to the list of pending requests.

1. Exception flows - flow of events for when things go wrong
2. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
3. If the admin user presses update request without any input in the input field or any attached document, then a message telling the admin user that a blank update will not be sent will be displayed.
4. **Success guarantee (aka. Post condition) - state of application after everything is done**

If additional information was submitted, then the requests entry in the database will be updated with the additional information. The pending requests view will be displayed by the application.

1. **Name of use case / title: UC7**

Update policies and rules

1. **Description - some short text describing the scope.**

The application will display the current list of policies and rules. Each item on the list will have an option to remove it from the list. There will also be a button for adding additional items to the list. In order to update a rule or policy, it will need to be deleted, and then the new rule or policy added. There will be an update button that will commit any changes made to the database.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The manager is the primary actor. The application is another actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The admin user has successfully logged into the application.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The application displays all policies and rules.
3. The admin user can delete a policy or rule document.
4. The admin user can add a new policy or rule document.
5. The application will update the database with any changes.
6. **Extensions - flow of application when it deviates from success scenario's flow:**
7. Alternate flows - other options of correct flow
8. The admin user can press the back button to return to the list of pending requests.
9. The admin user can select a policy or rule and its information will be displayed.
10. Exception flows - flow of events for when things go wrong
11. If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.
12. If the user presses update request without any input in the input field or any attached document, then a message telling the user that a blank update will not be sent will be displayed.
13. **Success guarantee (aka. Post condition) - state of application after everything is done**

If any documents were added or deleted, then the database will be updated with the changes that were made. The application will display the new list of policies and rules after the update button is pressed.

1. **Name of use case / title: UC8**

Remove users

1. **Description - some short text describing the scope.**

A list of all current member users are displayed with the option to remove any of them.

1. **Actor(s) / Primary actor - person(s) who interact with this particular use case.**

The manager is the primary actor. The application is another actor.

1. **Precondition - anything that this use case can assume to be true prior to beginning its life cycle.**

The admin user has successfully logged into the application.

1. **Success scenario - a sequence of steps describing the correct flow of events that take place.**
2. The application displays a scrollable list of all current users.
3. The admin user can select any user for removal.
4. The application will display a dialog box for the admin user to confirm that they want to remove a user.
5. The admin user will confirm the selection.
6. The application will remove the selected user from the database.
7. **Extensions - flow of application when it deviates from success scenario's flow:**
8. Alternate flows - other options of correct flow
9. The admin user can press the back button to return to the list of all pending requests.
10. The admin user can press cancel in the dialog box and return to the list of all current users.
11. Exception flows - flow of events for when things go wrong

If the application cannot connect to the database, then an error message telling the user to contact technical support will be displayed.

1. **Success guarantee (aka. Post condition) - state of application after everything is done**

After each deletion the database will be updated by removing the selected user. The application will display the list of current users.

# Project Assumptions and Constraints

Assumptions:

1. The functional requirements and non-functional requirements can be refined as the application is built.
2. The application will be used by homeowners and the Homeowners Association board members.
3. All key members of the project team have a good understanding and grasp of application development, agile processes, and the goals of what the Homeowners Association application is trying to achieve for its customers.
4. Each member of the team understands their role and works to their skill, from managing, developing, working with the customers on functional requirements, and addressing the security and quality of the application.
5. The application will be developed with a ‘Quality’ software concept in mind and work to get the application completed in the time required and refined during production.
6. All key players and team members are available for the next two months to collaborate on the development of the application.
7. The application will be a tool for the Homeowners Association to expedite their processes.
8. The Homeowners Association members will be able to conduct their business, such as approving or denying work requests without the application, but still use the application to notify homeowners of status.
9. Refinements will be made to this project proposal as well as to the Homeowners Association application as we progress through development.

**Constraints**:

1. Having four team members to conduct all of the development for this project.
   1. If a member has to drop a class or has an emergency then other team members will be expected to cover down on the deliverables.
2. Time is a constraint by having to complete all of the project’s development and deliver a product on or before the designated delivery date.
3. Developer skill level. Many of our team members are entry level software developers who are not currently in a mainstream work environment. Therefore, other team members, who are in the workforce, may have to provide development experience for any complex development issues.
4. Technology, using open source Integrated Development Environment, free collaboration software, and the use of our personal computers.
5. Customer requirements; a couple of our group members are in a Homeowners Association as homeowners but we have not been able to solicit input from a working Homeowners Association.
6. Lack of budget; the team is building this project with no monetary support.

**Project Priorities:**

1. Deliver a project proposal for the Homeowners Association application by the end of the first four weeks.
2. Deliver the Homeowners Association application, based on the ‘Quality’ concept on or before the designated delivery date.
3. Maintain constant communication amongst the team and with the product owner on project status.
4. The functional and non-functional requirements need to be developed with quality and security in mind.

# Project Delivery Methods

There are three major components of the application, mobile application(s), web services and database. The mobile application is android based and requires android 8.0 or higher to operate. In addition, the applications are built for phone sized screens versus tablets. This choice was made to simplify the maintenance and testing schedule of devices. There are two applications, one for tenants that will reside in a Homeowners Association and another for the Homeowners Association Board Members. The applications will be available to download on the google play store. In addition, the customer (Homeowners Association Administration) will be given an email with access to the .apk files to sideload applications without the google play store, along with instructions and training documentation on how to do this.

The actual android studio code will exist in a GitHub repository. The repository will be private and once the project is complete & payment received the credentials and ownership transferred to the Homeowners Association Administration. On this repository, there will also exist the code for the python web services and a document relating to the database model.

Both the Database & Web Services will be hosted on Amazon Web Service (AWS) RDS and Lambda respectively. The AWS account along with and access credentials to the database will be handed over to the client. The infrastructure will be built in such a way that automatic backups and scaling will be taken care of to ensure stability and reliability of the platform regardless of potential for any disaster. The database will be a MySQL server and will require MySQL Workbench to access.

In terms of maintenance, the company will guarantee that the systems will run for 4 years after the receipt of delivery of the software. Because FaaS (function as a service) and RDS cannot fail and the android project is backwards compatible, this will ensure maintenance is non-existent.

# Project Issues and Risks

Taking a proactive strategy will allow us to preemptively alleviate risk. For the Homeowners Association application, in the chart below, we will identify the potential risks that may prevent us, the agile team, from fulfilling the intended delivery of this application. Potential risks are identified, their probability and impact are assessed, and they are ranked by importance (Pressman and Maxim, 2015). Each given scenario is ranked by level of importance as either High risk, Medium risk, or Low risk. By systematically considering the projects schedule, support, cost, and performance estimations, we can designate a plan of mitigation. Each plan will be carried out by one of us through documentation and project planning and be tracked by designating the task as either Open, a Work In Progress (WIP), or Closed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Issue or Risk Brief Detail | Risk Probability:  High, Medium, low | Mitigation Plan | Responsibility | Status: Open, WIP, closed |
| 1 | Due Date: The given time to properly develop the application features is too short. | High | Create a through time table for each feature and its creation | Matthew Parra | WIP |
| 2 | Feature Abstraction: There are too many features requested to include in the application | Low | Review and identify the key requirements that are necessary to create an application that fills the needs of the Homeowners Association and the Homeowners | Brian Hurst | WIP |
| 3 | Adoption: The application is too difficult to deploy, ie. database deployment | Medium | Provide key documents necessary to deploy the application within the Homeowners Association’s abilities | Thomas Lowe | Open |
| 4 | Hardware limitations: Not all Homeowners and board members have smart phones/data plans to accommodate application use | Medium | If the homeowner or board member is unable to facilitate the application, templates with similar forms will be provided | Revanth Matha | WIP |
| 5 | Understanding: Board members and Homeowners do not understand how to use the application | Low | Provide documentation that carefully illustrates the use of the application, from initial download to the various features within the application | Thomas Lowe | WIP |
| 6 | Management Changes: New members to the board of supervisors who require new features | High | Provide new board member or members with documentation that highlights current features and how they may facilitate their requests already | Matthew Parra | Open |
| 7 | Performance: The application does not perform as intended, ie. slow or doesn’t produce correct results | Medium | Run as many unit tests within means to prevent any unforeseen issues | Revanth Matha | WIP |

# Evaluation, Selection of Technology and Tools for Project Implementation.

The end user will be utilizing an Android mobile application. The team will build this application using Android Studio 3.2 or greater. Android Studio supports 3 main development languages and the team will be utilizing Java over Kotlin & C++ as the team has the most experience with Java.

The Android application will need to make network calls. In order to make network calls more efficiently, two dependencies, Retrofit & OKHTTP, will be utilized. Both frameworks are MIT licensed which means they are free to use for personal & commercial use. Retrofit is a solution that makes parsing JSON into structs/models easier. OKHTTP provides prebuilt functions relating to get, post, update, fetch etc which are all major types of networking calls for the reading/writing of data. Using these libraries will make it easy to interact with our API endpoints hosted on Amazon Web Services (AWS).

The rest of the technology stack will be leveraging AWS to host our domain, create endpoints for web services & maintain a database. AWS is a great choice because they are an Infrastructure as a Service (IaaS). This means that they will provision all the hardware needed and provide developer tools to access said hardware via a console GUI or CLI.

The first service on AWS that will be utilized will be Route 53 which can be used to purchase a domain name ending in .com, .org .io and many others. Route 53 also enables various DNS requirements such as managing records and setting up a mail address.

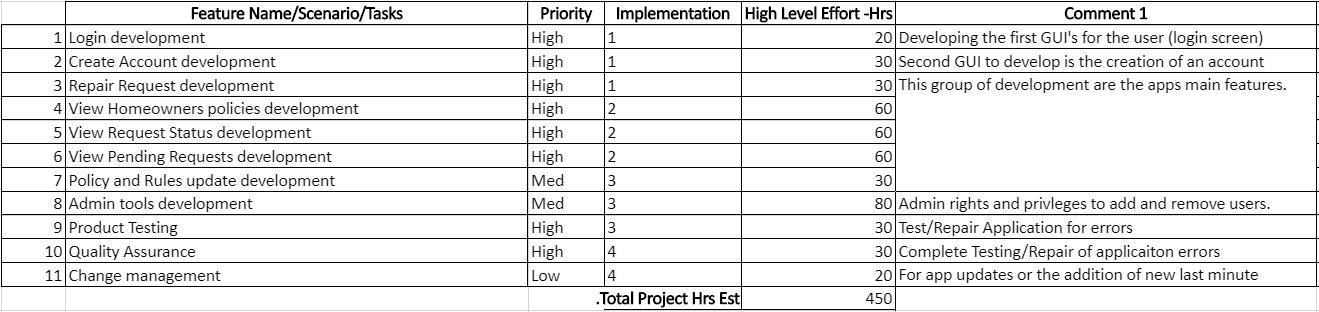
Once route 53 is designed, a web service will be connected using AWS Lambda. The Lambda function will be written in Python 3.5 or higher and is dependent on the runtime environment that Lambda supports. Certain packages or imports will be needed for Lambda such as pymysql to make the CRUD operations easier. Lambda function can be automatically to AWS API Gateway which can be used to generate endpoints that can be accessed. AWS will automatically manage the internet gateway needed which is one of the major benefits of an IaaS. Lastly, Lambda is a Function as a Service (FaaS) and there is no server associated with it. Rather, a docker image of the runtime image and code are loaded into a user defined cpu/ram amount and are replicated as needed depending on number of networking calls made. This means that Lambda is infinitely scalable.

Third our Lambda function will connect to a MySQL database. MySQL will require the team to use MySQL Workbench to connect and maintain said database. The main language of MySQL is SQL but automation scripts can also be built in Python (outside of scope) to help maintain the database. The database will be built depending on the data structures representing tables and fields within the application. The team thinks that one database with separate tables for admins versus users will be needed in order to support two separate applications (homeowner & administrator). The rest of the AWS infrastructure will not need this distinction as Lambda functions can be separated and use the same domain since users within the android app will not be able to see said domain.

# Product Backlog: Prioritized Feature List and Effort Estimates (hrs.)

Keeping track of our priorities will help us stay on track with the development of the Homeowners Association Application. A prioritized list of project requirements or features that provide business value for the customer (Pressman and Maxim, 2015). Currently the backlog of priority features will include not only the proposal, but also the development of each class and its methods required. Currently there are a projected 450 hours of work estimated to complete this project.

* Login/Logout​
* Administrator and User Account Creation​
* Work Request Creation/Editing​
* View Homeowners policies​
* View Request Status​
* View Pending Requests​
* Update Work Requests​
* View/Update Policy and Rules

Figure 1: Backlog with estimated hours

# Initial Architecture Design

In the Initial Architecture Design we have identified multiple classes, along with variables that must be maintained for the Homeowners Association Application. First, the use case diagram (Figure 2) will illustrate the basic use cases that will be facilitated by the application, followed by an activity diagram (Figure 3) that will demonstrate how the users and administrators will conduct basic tasks. The class relation diagram (Figure 4) will show how each class within the application relates in their function. Finally, the classes are broken down in the class diagram (Figure 5,6,7) and displays the variables and methods intended.

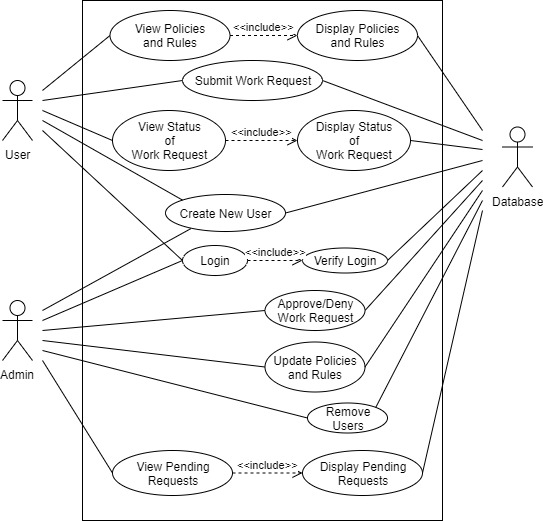
Figure 2: Use Case Diagram

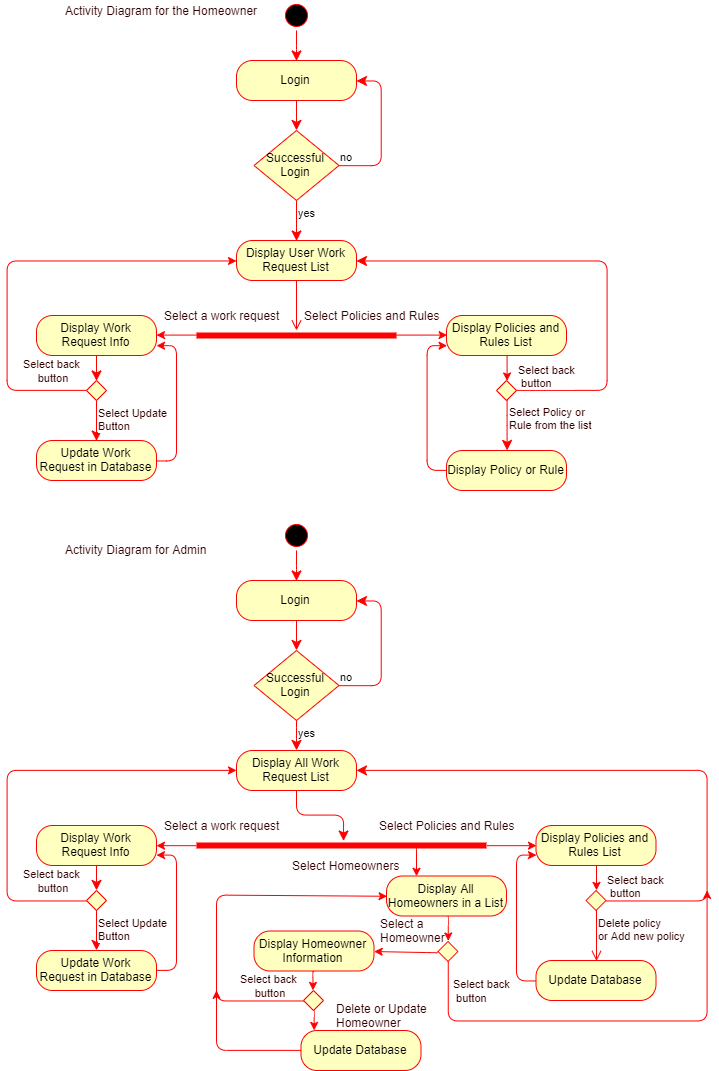
Figure 3: Activity Diagram

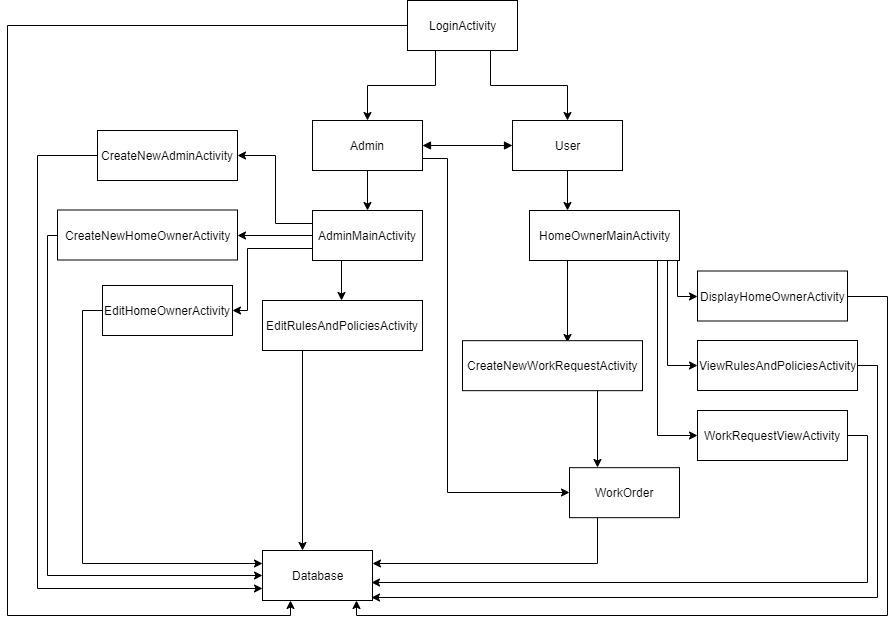
Figure 4: Class Relation Diagram

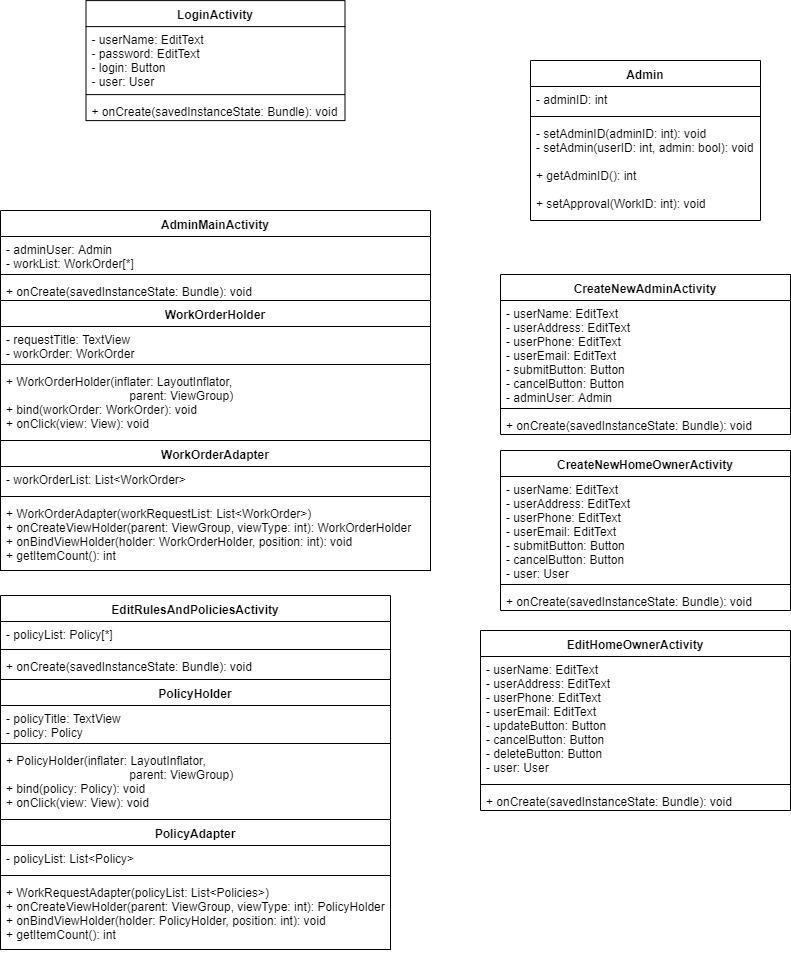
Figure 5: Class Diagram 1: Login, Admin and Admin activities.

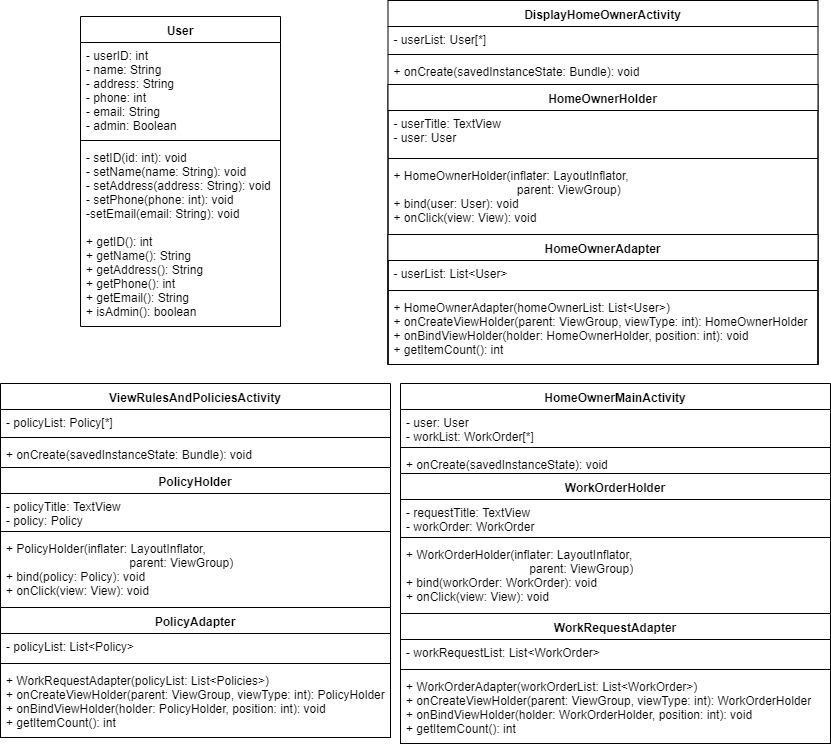
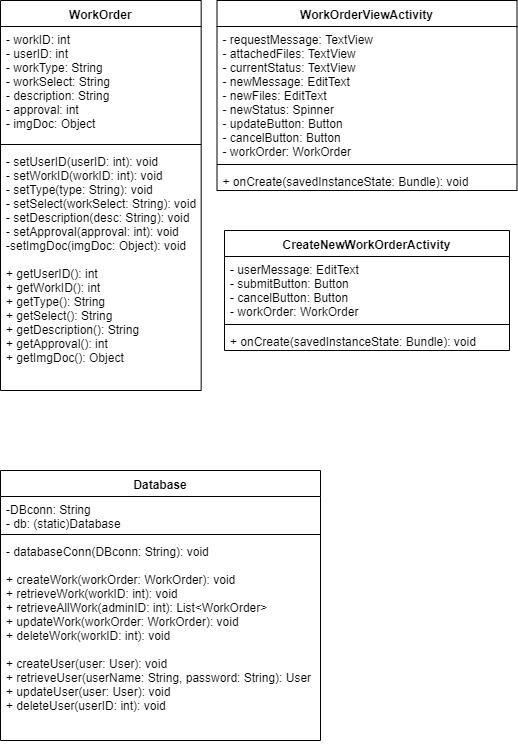
Figure 6: Class Diagram 2: User and User Activities.

Figure 7: Class Diagram 3: Work Order, Work Order Activities and Database.

# Agile Team Formation, Responsibilities and Implementation Iterations (schedule).

## 12.1 Team Responsibilities:

For our team, we have two professional software engineers and one person with an engineering background and another with a mix of software and product owner experience. All team members have experience with java and three members have experience with Android programming. This ensures that the team has the ability to create the mobile application. Figure 8 shows a list of each member and their expected roll.

In terms of individual experience, Revanth has the most experienced as he is employed as a mobile application developer. In addition he is a solution architect and is comfortable coding the entire software stack. His primary responsibility will be to work on the mobile application and cloud infrastructure as well as system design.

Matthew is the next skilled developer on the team. He also acts as a beacon since he is the team leader and makes sure everyone is on task and work is getting done as per schedule. Mattews expertise in software development is in the creation of python web services leveraging Django. Matt’s leadership will be critical in connecting all the pieces together.

Thomas came up with the product idea. He is the user and product owner as well as a software engineer himself. Thomas will be responsible for translating his vision into a design through attainable goals that reflect business value. Thomas is also a capable Android developer and will assist on that front. Lastly Thomas will be the main tester of the application, he is a end-user and thus things must pass his certification.

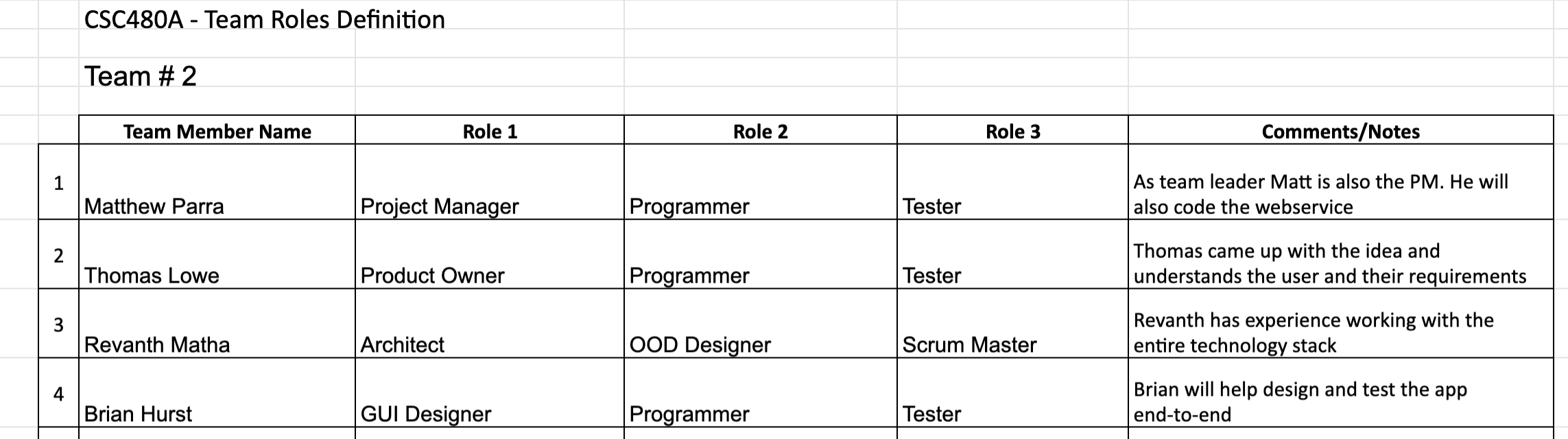
Lastly, Brian does not have much experience with software development. Instead, he possesses a keen eye for graphic design which also extends into modeling systems. Brian can assist with android development and will spend the majority of his time on the database portion as well as testing.

Figure 8: Team roles.

## 12.2 Iteration Plan:

Tracking each iteration of the project will be important to completing the project and delivering a working product at the expected delivery date. The proposed Homeowners Association application requires us to break the application into multiple iterations, prioritizing, and assessing their expected implementation. The scope and complexity of such systems must be modeled so that all constituencies can better understand what needs to be accomplished, the problem can be partitioned effectively among the people who must solve it, quality can be assessed as the system is being engineered and built (Pressman and Maxim, 2015).

The following list will define each expected feature, Figure 8 will show each feature and its priority along with the features expected iteration implementation.

* Login development
* Create Account development
* Repair Request development
* View Homeowners policies development
* View Request Status development
* View Pending Requests development
* Policy and Rules update development
* Admin tools development
* Quality Assurance and refinement

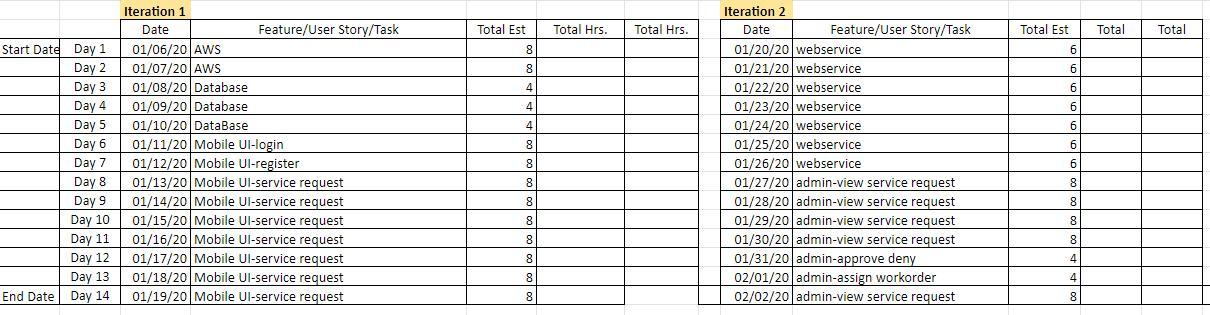
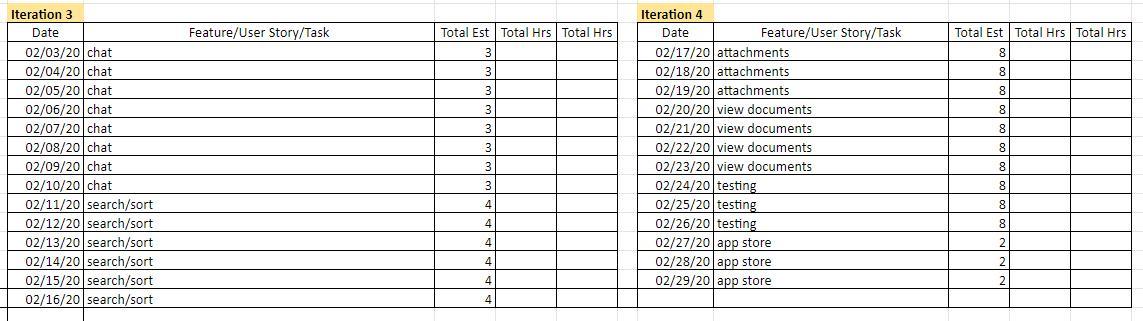
Figure 9: Project Iterations one through two.

Figure 10: Project Iterations three through four.

# Conclusion

In conclusion, this application facilitates a long standing need to maintain communication between both Homeowners and the Board of Directors. We propose a mobile application that provides Homeowners the ability to correspond with the Homeowners Association Board of Directors about home augmentations while being able to keep apprised of new Association policies. Additionally we offer the Board of Directors the capacity to maintain Association policies, provide feedback to homeowners about requested work, and establish approval or denial to homeowners requests. The proposed application will be carried out using the Android mobile platform and supported through Amazon Web Services. Harnessing each team members unique skill set and experience, the projected timeframe will take approximately eight weeks to complete.

# References

Pressman, R. S., & Maxim, B. R. (2015). *Software engineering: a practitioner's approach*. New York: McGraw-Hill Education.

Functional and Nonfunctional Requirements: Specification and Types. (2019). Retrieved 19 December 2019, from<https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/>

Falk, T., & Falk, T. (2019, November 20). Renovations with an HOA: How to get approval. Retrieved from<https://www.finder.com/what-you-need-to-know-about-hoa-approval-for-renovations>

# Appendix 1 - Proposal Plan Responsibility List

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CSC480A Project I - Proposal Plan and Responsibility List** |  |  |
| **Project Section** | **Section Title** | **Author(s)** | **Reviewer(s)** |
| **1** | Introduction | Thomas J Lowe | Revanth Sai Matha |
| **2** | Background and Need for the Project | Brian Spencer Hurst | Matthew Parra |
| **3** | Project Objectives and Scope | Matthew Parra | Brian Spencer Hurst |
| **4** | Customers and Stakeholders | Revanth Sai Matha | Thomas J Lowe |
| **5** | Project Requirements | Brian Spencer Hurst | Thomas J Lowe |
| **6** | Project Assumptions and Constraints | Thomas J Lowe | Matthew Parra |
| **7** | Project Delivery Methods | Revanth Sai Matha | Brian Spencer Hurst |
| **8** | Project Issues and Risks | Matthew Parra | Revanth Sai Matha |
| **9** | Evaluation, Selection of Technology and Tools for Project Implementation | Revanth Sai Matha | Brian Spencer Hurst |
| **10** | Prioritized Feature List and Estimates | Thomas J Lowe | Matthew Parra |
| **11** | Initial Architecture Design | Brian Spencer Hurst / Matthew Parra | Thomas J Lowe / Revanth Sai Matha |
| **12** | Agile Team Formation, Responsibilities and Implementation Iterations (Schedule) | Revanth Matha/ Matthew Parra | Brian Hurst |
| **13** | Conclusion | Matthew Parra | Thomas Lowe |

# Attachments

SCRUM Plan - Team2\_CSC480A\_SCRUM-Estimation (Excel)

PowerPoint Presentation - Team2\_CSC480A\_Dec19\_Proposal (PowerPoint)