**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

[Contents 2](#_Toc26521037)

[1.0 Introduction 3](#_Toc26521038)

[2.0 Background and the Need for the Project 3](#_Toc26521039)

[3.0 Project Objectives and Scope 4](#_Toc26521040)

[4.0 Customers and Stakeholders 5](#_Toc26521041)

[5.0 Project Requirements 5](#_Toc26521042)

[5.1 Functional Requirements 5](#_Toc26521043)

[5.1 Non-Functional Requirements 8](#_Toc26521044)

[5.2 High Level Use Case Scenarios 8](#_Toc26521045)

[6.0 Project Assumptions and Constraints 24](#_Toc26521046)

[7.0 Project Delivery Methods 24](#_Toc26521047)

[8.0 Project Issues and Risks 24](#_Toc26521048)

[9.0 Evaluation, Selection of Technology and Tools for Project Implementation. 24](#_Toc26521049)

[10. Product Backlog: Prioritized Feature List and Effort Estimates (hrs.) 25](#_Toc26521050)

[11.0 Initial Architecture Design 25](#_Toc26521051)

[12.0 Agile Team Formation, Responsibilities and Implementation Iterations (schedule). 25](#_Toc26521052)

[12.1. Team Responsibilities: 25](#_Toc26521053)

[12.2 Iteration Plan 25](#_Toc26521054)

[13.0 Conclusion 25](#_Toc26521055)

[References 26](#_Toc26521056)

[Appendix 1 - Proposal Plan Responsibility List 27](#_Toc26521057)

[Appendix X 27](#_Toc26521058)

[Attachments, if any. 27](#_Toc26521059)

[List of Abbreviations 27](#_Toc26521060)

[Definitions 27](#_Toc26521061)

# 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 today’s 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

A key part of Homeowner Associations has been the communication between homeowners and the association’s board of directors. Excluding financial exchange for membership, there exists a need for members to communicate with the Homeowners Association to obtain approval for changes to the exterior and interior of their homes. This is traditionally done using 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 Covenants, Conditions and Restrictions to the Homeowners Association members, which is typically done through a notification placed in the homeowner’s mailbox or taped to their door.

This 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 Covenants, Conditions and Restrictions.

# Project Objectives and Scope

The objective of this Homeowners Association application, from the board of directors and homeowners perspective, will provide a streamline 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 repairing 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 board of directors will also be given the ability to maintain a community calendar as to monitor the work that is in progress or is set for a future date.

The scope of the Homeowners Association application is relatively straight forward. 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 view, the community calendar, create or edit their requests and its status, 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 log in and review requests, provide feedback to homeowners, maintain a community calendar, and ultimately approve or deny requests.

# Customers and Stakeholders

For the Homeowners Association application, the stakeholders of the project are the Homeowners Association board of directors, the homeowners and our 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 serve their needs of communication with 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 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**

## Functional Requirements

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 app. 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 app. 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 photo 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, 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 photo or documentation.

* **View HOA policies and rules**

These will be read only files maintained by the HOA board members that contain all HOA policies and rules. 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, 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 admin application, which will be used by management members

* **Login**

Required login will allow only authorized personnel to access the app. 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 pending 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.

* **Update HOA policies and rules**

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

* **Remove users**

A list of all admin users will be displayed. There will be an option to add a new admin user and delete an existing admin user. Selecting a user will display their HOA board position and give an option to edit that information. There will also be a list of homeowners with an option to remove them from the application.

* **Create new account**

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

## Non-Functional Requirements

* **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.

* **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.

## High Level Use Case Scenarios

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

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 step describing 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**

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 step describing 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**

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 step describing 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**

View HOA 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 step describing 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 a 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**

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 where 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 step describing correct flow of events that take place.**
2. The application displays the requests 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**

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 where 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 list of all pending requests.

1. **Success scenario - a sequence of step describing correct flow of events that take place.**
2. The application displays the 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**

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 step describing 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**

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 step describing 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.

# 6.0 **Project Assumptions and Constraints**

List all the project assumptions as you start the project

Constraints could be budget, time, staff, technology, etc.

List any project priorities as a part of this section

# Project Delivery Methods

Clearly state how the customer is expecting the system to be delivered – website, mobile or both

What are requirements from the customer’s perspective required on their system to run, if any. This would apply to enterprise systems.

# Project Issues and Risks

List all the items that are obstacles now (issues) and in the future (risks) that would prevent the team from accomplishing their project objectives. Categorize the risks as low, medium and high on their impact to the project to complete.

Also, propose mitigations for the risks.

Simple and effective approach for this is to create a table (see sample below). Add additional columns as needed (e.g. due date) for tracking during implementation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Issue or Risk Brief Detail | Risk Probability:  High, Medium, low | Mitigation Plan | Responsibility | Status: Open, WIP, closed |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |

(Instructor’s Note – all above 8 sections are due at the end of second week).

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

The teams would need to do some research for the appropriate technology and tools for the project. Implementation/development requirements: Software, hardware, network, database, platform, etc. Although the goal is not to do any implementation in this first course, but identification of technology and tools are needed at this stage.

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

Breakdown the project main requirements in to prioritized feature list for implementation by the Agile team with estimates for each. This would be the first pass at this stage.

# 11.0 Initial Architecture Design

This will be the initial design aspect when the team designs the high-level architecture using Object Oriented design principles: students should identify the top-level subsystem and components and how they are interconnected. UML activity and class diagrams, objects, etc.

Include a brief description of why the team chose this approach.

(Instructor’s Note – all the above 11 sections are due at the end of 3rd week).

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

12.1. Team Responsibilities: Individual team member assignment will be listed per the Agile process requirements. Project Manager/Scrum Master, Architect, coder, tester, etc. In a small team, one might wear more than one hat. But the responsibilities must be clear. Also, the specific times for meeting each day (as required by the Agile team – need to be agreed up at this stage which guarantees everyone’s participation and contribution. The project manage/scrum master should be given authority to call upon other as required to help meet the iteration goals.

12.2 Iteration Plan: The team should decide the implementation of the prioritized feature list within the 4 iterations for the rest of the eight weeks with each iteration being 2 weeks.

The team will list here the feature(s) that will be completed and working in each iterations and demonstrable. Again, simply create a table for each iterations.

Part of the last iteration will include a final project presentation during the last session of the class (CSC480C) to the Chair and other School Faculty and submitting a final project documentation as required by the Instructor.

# 13.0 Conclusion

Brief summary of this project proposal and the plan to implement it using Agile methodology in iterations/sprints.

# References

# 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 |  |
| **6** | Project Assumptions and Constraints |  |  |
| **7** | Project Delivery Methods |  |  |
| **8** | Project Issues and Risks |  |  |
| **9** | Evaluation, Selection of Technology and Tools for Project Implementation |  |  |
| **10** | Prioritized Feature List and Estimates |  |  |
| **11** | Initial Architecture Design |  |  |
| **12** | Agile Team Formation, Responsibilities and Implementation Iterations (Schedule) |  |  |
| **13** | Conclusion |  |  |
|  |  |  |  |

# Appendix X

# Attachments, if any.

# List of Abbreviations

# Definitions

(Instructor’s Note – the full document is due at the end of 4 weeks).