Software Requirements Specification

Iteration-01
Software Engineering

<Housing Society Management>

Presented To: Ma’am Saba Kanwal

Presented By: Muhammah Azhan, Hassan Jamshaid, Leena Rizwan

(21i0425, 21i0408, 21i0430)

Table Of Contents:

1. Introduction

1.1. Purpose

This Data Specification (SRS) is intended for the development of applications designed for housing associations. This document refers to version 1.0 of the application, which is designed to facilitate interaction between managers and residents of housing associations. It includes the essential skills needed for seamless communication, management, and community collaboration.

1.2. Document Conventions

This SRS complies with legal standards for clarity and consistency in application. Unless otherwise stated, the importance of higher requirements is assumed to result from detailed information. Key points and concepts can be highlighted and kept consistent throughout the document to improve readability.

1.3. Intended Audience and Reading Suggestions

This document is intended for all parties involved in its development and use. The entire process involving developers, project managers, testers, and writers. It also meets the needs of potential customers and property managers.

It is recommended for developers and testers to start with the maintenance process to understand the purpose and scope of the software. Project managers will see details and priorities for their roles. It should focus on describing features and functions that directly affect how users and administrators interact with the application.

1.4. Product Scope

The software listed in this document is intended to be a real estate organization that enables effective communication and management between managers and insiders. Its main objectives are:

To provide residents with access to information, requests, and participation in community activities.
Simplify administrative tasks like managing requests, tracking expenses, and delivering important announcements.
Increase transparency and accountability in housing associations through easy access to information and data.
Express participation and interest of the entire community through effective communication and collaborative work.
The application helps improve social life by increasing efficiency and maintaining transparency and satisfaction in housing associations according to the company's goals.

1.5. References

2. Overall Description

2.1. Product Perspective

This product is a replacement for existing systems of its type-while housing society management apps are not a new concept, there aren't enough of them around for districts that may need them. And those that do exist, do not contain the features that our product will, making it more user-friendly for homeowners as well as management personnel.

2.2. Product Functions

This product will allow the Homeowner user class to:
1. Pay utility bills and other fees online
2. Receive notifications about power outages, or immediate danger
3. View a Community Calendar for local events
4. Call for maintenance workers
5. Submit feedback
6. Communicate with fellow users

This product will allow the Admin user class to:
1. Issue bills and fees to Homeowner(s)
2. Manage Homeowner accounts
3. Broadcast notifications about outages or advertisements
4. Add to the Community Calendar
5. Create polls to retrieve Homeowner feedback
6. Dispatch maintenance workers on Homeowner Request

2.3. User Classes and Characteristics

There are two user classes: Homeowner and Admin.
Homeowners are expected to use this product most frequently. They will use the functionalities highlighted for them in Section 2.2, and will require almost no technical expertise, as the app will come with tutorials provided on first login in each device. They will have their login credentials and user data encrypted before it is stored, thereby granting them data security. The user may need to be able to read either English or Urdu to fully utilize this product, and may require intermediate experience with web application usage.
Admin users are not expected to use this product over-frequently, only for work or managerial purposes. They will utilize the functionalities highlighted for them in Section 2.2, and will require mediocre technical expertise. Their login credentials will be encrypted before they are stored, therefore granting them data security, and they will have management/administrative privileges over the Homeowner user class. They will need to be able to read either English or Urdu to make use of the administrative functionalities, and require a mediocre amount of technical/database usage experience.
Since Homeowner user class will be using the app far more frequently and are, therefore, the target demographic, they will be important to satisfy. However, as the product owner will naturally be an Admin user class user, Admin will be more important to satisfy.

2.4. Operating Environment

As a web application, this product will not only function properly on all PC operating systems, but shall, by the end, be compatible with mobile phone operating systems as well. This is to ensure accessibility of the product,

and therefore make it more user-friendly, as not all households have laptops or computers, but nearly all citizens in housing societies have smartphones. As a web application, it will not require a large subset of the device's storage, therefore, it will be able to peacefully coexist with many other applications. It will, however, require the internet for functioning, and have to coexist with a variety of companion applications, such as the host device's own calendar application, and other similar applications that the host device may have installed.

2.5. Design and Implementation Constraints

Due to regulatory and legal privacy policies, we cannot ask the user for their address or CNIC data-as we are not affiliated with any housing society as of now, we will be held liable in the case of a data breach. The product, due to the memory constraints that come with making it primarily a mobile phone application, will need internet access for most, if not all, functionality. It will also need to be made using a flexible coding language and IDE, capable of being compiled on mobile or PC. Communication and Security of that communication over the internet wll be provided by our online database system, available to us on a subscription basis. Due to having two distinct user types with two distinct veiws, we will need to design the application with OOP (Object Oriented Programming) design principles and programming standards in mind.

2.6. User Documentation

Upon the first login in a new device, the product will launch tutorials (that can be toggled off)for basic features, such as the payment of bills and fees. In addition, a helpline will be established for additional user support, with active call center staff 24/7. User manuals will also be available online, for both user classes, on the product's installation page.

2.7. Assumptions and Dependencies

Our constraints as third-party application developers keeps us from ensuring that those making Homeowner accounts truly live in the societies where they claim to, and our staff limitations may prevent us from releasing a full product upon launch, or being able to cater to specific time constraints. Other applications on the device could also affect the performance of the product; for example, if the Calendar or date/time system is accidentally set to a different year, or month, then the user may see a different interface for the Community Calendar, or not be able to send in a request for maintenance worker due to database errors. And ofcourse, its main limitation is that of internal signal; if the user's internet is slow, the application may lag.
We must, therefore, assume that the user has a stable wifi connection, that their date/time is set to the correct values, and that they are telling the truth about which housing society they live in, and are submitting feedback for.

3. External Interface Requirements

3.1. User Interfaces

- **Layout:** There will be a main menu that will contain buttons that lead to different features of the app. There will be a separate page for every use case such as announcements, bill payment, feedback, booking, etc.
- **Responsive Design:** The application will be responsive and automatically change its layout to fit standard resolutions used by phones.
- **Documentation:** There will be a help button available on every page that will provide information about the page.

- **Importance:** Every user will be displayed a user interface that is relevant to them. The administration department will have different UI pages and residents will have separate UI pages.
- **Navigation:** The application will provide different methods of navigation based on the device being used. Mouse and keyboard for desktop and laptops and touch for phones.

3.2. Hardware Interfaces

- **Supported Devices:** The application should be compatible with mobile devices running different operating systems such as Android, IOS as well as computer systems running Windows.
- **Database Servers:** The application will be connected to external Microsoft Azure database servers to store user information.

3.3. Software Interfaces

- **Supported Operating Systems:** The application should be compatible with: Android versions 10, 11, 12, 13 and 14, IOS 15, 16 and 17, Windows 10 and Windows 11.
- **Database systems:** The application should store its database on Microsoft Azure.
- **Payment Gateways:** The application should provide payment gateways through JazzCash, Easypaisa, PayPal, NayaPay and SadaPay.

3.4. Communications Interfaces

- **Communication Protocols:** The application should use communication protocols like UDP and TCP for communication over the internet. TCP should be used for secure communication that deals with personal information of residents while UDP should be used for establishing quick connections between various application components.
- **Communication Encryption:** The application should use a combination of Secure Socket Layer (SSL) and Transport Layer Security (TLS) to encrypt its network communication.

4. System Features

First, we will cover the features from the Homeowner user class' view:

4.1. Payable Utility Bills and Miscellaneous Fees

4.1.1. Description and Priority

The Homeowner will be able to pay all due fees from home, via this application, without leaving their own home. It is a feature of High priority, with high benefits(8/9, since a user may also go and pay in the bank, and administrators may simply mail a printed bill), low cost(2/9, with the assumption that the management issuing these bills are also a user class within the application) and low risk (2/9, since the application will not retain the user's payment information).

4.1.2. Stimulus/Response Sequences

Once the Homeowner is logged in, they must select Transaction. The system will display options for which bill they may wish to pay, and once selected, the system will then return the bill details. After this, the Homeowner will input their bank details, approve the payment, and then confirm that approval once the system checks that their account has the balance to cover this transaction. The system will then save this change in the database, and remove this particular bill from the payable bill list, and display to the user the previous screen of payable bills.

4.1.3. Functional Requirements

REQ-1: The user inputs the correct bank details. If they input a bank number that is not their own, the user themself is liable. And if the user inputs the incorrect pin number, the system will prompt them to input it once again, until it is correct, or the user exits out of the interface.
REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the options screen.

REQ-3: The Admin had issued this particular Homeowner payable bills and fees. If not, they will not see any data on the options screen.
REQ-4: The user must be online, with a consistently stable internet connection. If not, the transaction may not be saved in the database, nor functionally be completed at all.

4.2. Receive Immediate Notifications

4.2.1. Description and Priority

The Homeowner will be able to see urgent notifications sent out by the Admin user class, about scheduled power/gas outages, immediate dangers in the neighborhood, or package deliveries. The priority of this feature is High, because it greatly increases the usability and convenience of Homeowners when information about immediate danger or scheduled power/gas outages is dispatched from a reliable source, instead of online group chat messages about such issues. It has high benefits (8/9, for reliable information to people who cannot access reliable information any other way), low cost (1/9, given that the Admin class actively works to spread notification on this app), and virtually zero risk (1/9, the only risk being the accidental leakage of user location information, but even so, it will only be their housing society, not their address, as our product will not store such information).

4.2.2. Stimulus/Response Sequences

Once the user is logged in, they must select Notifications, to view a backlog of important notifications sent out by the Admin user class. These notifications will be in the form of a list, and localized per housing society, so that overall traffic on the application is decreased, and relevant news is accessible to the Homeowner user.

4.2.3. Functional Requirements

REQ-1: The user must be online, with an at least moderately stable internet connection. If not, the product may not be able to access the database with the issued notifications.
REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the notifications screen.
REQ-3: An Admin user must have issued a notification for Homeowners of the user's area to be able to see. If not, the screen will appear empty.

4.3. Community Calendar

4.3.1. Description and Priority

The Homeowner will, with this feature, be able to see local events, their dates and details, as posted by the Admin user class. They will also be able to, for select events or public areas, be able to register or reserve the space for private events respectively. The priority for this feature is Medium, since the user can also go to the housing society's own office for a possibly more reliable registration. The benefits of this feature are a 5/9, the risks are also a 5/9, since the registration information may be unreliable, and the cost is 4/9, since every new event will require a new table to store user data.

4.3.2. Stimulus/Response Sequences

When the Homeowner is logged in, they must select the Community Calendar to view the month's calendar, with events input by the Admin users for the current month, and learn more about the event by selecting it. If an event is registrable, the user may select it to register and input the relevant information. If a venue is open on a date that the user selects, they may choose to reserve it for that day, for a private event. After any of these operations are complete, the user may exit back into the Homepage.

4.3.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the product may not be able to access the database with the issued notifications or new changes to the calendar.

REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the Community Calendar.
REQ-3: A local Admin user must have posted an event for Homeowners of the user's area to be able to see. If not, the calendar will appear empty.

4.4. Call Maintenance Workers

4.4.1. Description and Priority

The Homeowner will be able to put in a request for maintenance workers of multiple kinds (plumbers, electricians, etc) in a waiting list that is viewable to Admin users. This is a High priority feature, because it makes this app especially useful to people who require the accessibility features of this product. the benefits are high (7/9), the cost is low(2/9), and the risk is medium, due to sensitive information about the worker or the Homeowner being stored for a variable length of time (3/9).

4.4.2. Stimulus/Response Sequences

When the Homeowner is logged in, they may select Request Maintenance, and input the relevant information on the page they see next-information such as the type of maintenance requested, the urgency of the situation, and their home address. Afterwards, the system will log this information into a database table visible to Admin users, who will henceforth work on dispatching a Maintenance Worker.

4.4.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database.
REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the Request Maintenance screen.
REQ-3: Their home address must be accurate, otherwise, it will be difficult for the maintenance worker to find their home.
REQ-4: The Admin database must have a maintenance worker of the type required and within the locale of the Homeowner requesting the maintenance.

4.5. Register Visitors

4.5.1. Description and Priority

The Homeowner will be able to Register Visitors that they expect with identifying information that will be temporarily stored in a semi-public database available to Admin users. The priority of this feature is Medium, the cost is 7/9, since it will require responsive and quick Admin users dedicated to the response of these inquiries, and the risk is 4/9, since the visitors' information will be stored in a database.

4.5.2. Stimulus/Response Sequences

When the Homeowner is logged in, they may select Register Visitor, and then input all their vistor's relevant information, such as CNIC card info, vehicle license plate number, and name. After they fill in the information and confirm, the information will be stored on a database visible to Admin users. Guards at the gate of the housing society can register an inquiry about a visitor with the Admin users, and within a reasonable time frame, the Admin users will respond with whether or not the person at the gate is a registered visitor and thus allowed inside without further questioning.

4.5.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database.
REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the Register Visitors screen.

REQ-3: The Homeowner entered the relevant information correctly. if the visitor's information is incorrect, then the visitor may not be allowed inside the housing society.

4.6. Feedback Submission

4.6.1. Description and Priority

Homeowners will be able to submit feedback on facilities or events or venues in their local area, both for the benefit of other homeowners who may plan to visit, and for the benefit of management who seek to improve the overall experience of those who visit. The cost is 3/9 at most, because while it may require extra data to store online, it will not be difficult to set up. The benefits are 4/9, and the risk is 1/9, virtually none.

4.6.2. Stimulus/Response Sequences

When the Homeowner is logged in, they may select Submit feedback, and input the name of the location visited, and their positive or negative feedback for it. Now this feedback will be visible to any homeowner who searches for the location on their map, and it will also be visible to those who own or manage these locations, as this information will be public. The Admin users may choose to take down reviews if they are irrelevant or hateful, however.

4.6.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database.
REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the Submit Feedback screen.
REQ-3: The location must be one already registered in the application. If it is not registered, the feedback may not be entered into the database, as it would never be visible, and thus be a waste of data.

4.7. Communication with Fellow Residents

4.7.1. Description and Priority

Homeowners will be able to chat in a community chat room based on their locale. Every housing society (or phase of every housing society) will have their own independent community chat. The priority of this feature is Low, because such group chats already exist, and their inclusion in this app would mostly just serve to make the database even heavier than it already will be. The benefits are 3/9, the risks are 4/9, and the cost is 7/9. Especially if users begin to send irrelevant messages.

4.7.2. Stimulus/Response Sequences

When the Homeowner is logged in, they may select Community Chat. This will open up a chatroom with every other Homeowner account in their own housing society, thus fostering inclusivity and interaction. These will automatically be made and included per housing society registered into the application. The Homeowner may then type in a text message, send it, and other Homeowners will be able to see it and respond in real time, granted they are all online to see it. If the Homeowner wishes to exit, they may simply exit back to the Homepage screen.

4.7.3. Functional Requirements

REQ-1: The user must be online, with a moderately stable internet connection. If not, the application may not be able to write into the database that the others will have to access to view the new message.
REQ-2: The user is logged in. If they are not, they will not be able to view the Main Menu screen, much less the Community Chat screen.
REQ-3: Homeowners must follow common courtesy and avoid hateful speech in these chat rooms-if three or more users report a Homeowner of hateful speech, with proof of it, and Admin may choose to delete their account.

And now, we will cover the features from the Admin user class view:

4.8. Issue Bills

4.8.1. Description and Priority

The Admin user class will be able to issue payable bills to Homeowner users, by entering the amount, the Homeowner's user ID, and other details. It is a feature of High priority, with high benefits(8/9, since administrators may simply mail a printed bill), low cost(2/9) and low risk (2/9, since the application will not retain the Homeowner's payment information).

4.8.2. Stimulus/Response Sequences

Once the Admin user logs in, they must select Issue Bill, and the system will display a screen for them to input the reason for the bill (electricity, gas, water, violation fees). They will input the amount, the homeowner's ID, other relevant details, and send the bill. Which is to say, the bill will be recorded in a database, and shown to the relevant Homeowner when they log in.

4.8.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database that the Homeowner will read from.
REQ-2: The user is logged in. If they are not, they will not be able to view the Homepage, much less the Issue Bills screen.
REQ-3: The user must input the homeowner information correctly. If the Homeowner does not exist, the bill wont be written into the database, and an error message will be displayed.

4.9. Manage Homeowner Accounts

4.9.1. Description and Priority

The Admin user class will be able to update the passwords for, read data on, and delete Homeowner accounts. This feature is of High priority, because Homeowner accounts will need database support of this type from administration. The cost is low (3/9) the benefits are extremely high (9/9), and the risk is minimal (2/9), because Admin accounts with managerial power over Homeowner accounts can only be created by other Admin accounts.

4.9.2. Stimulus/Response Sequences

Once the Admin user logs in, they must select Manage Accounts. From there, they may choose one of four options. The first is Create Admin, in which they may create a new Admin user by entering in relevant login information. The second is View All Account Details, in which they may view the login credentials and other information for both, Homeowner, and Admin users. The third option is Update Password, in which they can, upon request, update the password for Homeowner users. Fourth, they may choose Delete Account, through which an Admin account can delete any Homeowner or Admin account by entering their account ID, and then confirming that action. If they wish to back out of any of these screens, the exit will take them back to the Homepage.

4.9.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database that the Homeowners will read from.
REQ-2: The user is logged in. If they are not, they will not be able to view the Homepage, much less the Manage Accounts screen.

REQ-3: Any data entered about other accounts-password, ID, etc-must be correct. If any of it is faulty, then the liable Admin user is to blame for deleting the wrong account, or reseting the wrong account's password. If the user ID does not exist, then the action will not be completed, and an error message will be displayed.

4.10. Broadcast Immediate Notifications

4.10.1. Description and Priority

The Admin user class will be able to broadcast immediate messages for dangers within the neighborhood, scheduled power/gas outages, or even advertisements for local restaurants and brands. This is a High priority feature, as it allows homeowners to obtain information about danger and outages from reliable sources instead of group chats that may not be telling the truth.

4.10.2. Stimulus/Response Sequences

Once the Admin user logs in, they must select Broadcast Notification. After that, they may input information relevant to their housing society, for the Homeowner accounts belonging to their own housing society to be able to view. These notifications may be about scheduled gas/power outages, immediate danger such as violent stray animals, or criminals spotted nearby. After this message is complete, they may select Send, and then be redirected back to their Homepage.

4.10.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database that the Homeowner will read from.
REQ-2: The user is logged in. If they are not, they will not be able to view the Homepage, much less the Broadcast Notifications screen.
REQ-3: The Admin user must have Homeowner users in the database that belong to the same housing society. If they do not, the Notifications will be added into the database, but they will not be visible to anyone.

4.11. Add to Community Calendar

4.11.1. Description and Priority

The Admin user class will be able to add events and event details to their local Community Calendar. These events may include store openings, seasonal walks, funfairs, etc. The priority of this feature is Medium, the benefits of this feature are a 5/9, the risks are also a 5/9, since the registration information's security may be unreliable, and the cost is 4/9, since every new event will require a new table to store user data.

4.11.2. Stimulus/Response Sequences

Once the Admin user logs in, they must select Community Calendar. After that, they may select a date with no other event registered to it, and input event details, such as a short description, a Title, a venue, a time-frame, and whether or not the event is registrable. They may then enter this information, and thus record it in a database. After this, they will be taken back to the screen showing them available dates on the Community calendar.

4.11.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database that the Homeowner will read from.
REQ-2: The user is logged in. If they are not, they will not be able to view the Homepage, much less the Community Calendar screen.
REQ-3: The date that the user wishes to add an event to must be empty. If it is not empty, then the database will not write.

4.12. Create Polls

4.12.1. Description and Priority

The admin user may set up polls about certain facilities or venues to receive Homeowner feedback on their experiences, so that steps may be taken to improve mediocre or lackluster experiences. This feature is of Medium importance, and the cost is 3/9, because it may require extra data to store online, but it will not be difficult to set up. The benefits are 4/9, and the risk is 1/9, virtually none.

4.12.2. Stimulus/Response Sequences

Once the Admin user logs in, they must select Create Poll. After this, they will be redirected to a page where they can set up questions, and answers for users to vote upon, often to rate a facility or event out of 5, as well as set up optional questions with written answers for Homeowner users to fill. After this, the Admin will be redirected back to the Homepage, and if they go to the List of Polls again, they can select an old poll they created and view its results.

4.12.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into the database.
REQ-2: The user is logged in. If they are not, they will not be able to view the Homepage, much less the Create Poll screen.

4.13. Dispatch Maintenance Workers

4.13.1. Description and Priority

This feature allows Admin users to contact and dispatch maintenance workers registered with affiliation to the application in the locale of the Homeowner who is requesting the maintenance, and then record that they sent a maintenance worker to the Homeowner in a database. This is a High priority feature, because it makes this app especially useful to people who require the accessibility features of this product. the benefits are high (7/9), the cost is low(2/9), and the risk is medium, due to sensitive information about the worker or the Homeowner being stored for a variable length of time (3/9).

4.13.2. Stimulus/Response Sequences

Once the Admin user logs in, they must select Dispatch Maintainence Worker. They must then select from a list of workers to dispatch for the Homeowner who requested the maintenance, and then a message will be sent to the worker, and a record will go into the database about the Homeowner's request being catered to at the current date and time, and the Homeowner's request will be erased from the pending list of requests.

4.13.3. Functional Requirements

REQ-1: The user must be online, with a stable internet connection. If not, the application may not be able to write into nor read from the database.
REQ-2: The user is logged in. If they are not, they will not be able to view the Homepage, much less the Dispatch Maintenance Worker screen.
REQ-3: The Admin database must have a maintenance worker of the type required and within the locale of the Homeowner requesting the maintenance.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

- The application should respond to user actions such login, register, transactions, data retrieval and data storage within a specified time frame, ideally within 2 seconds of user input.
- Loading time of various features of the application should not exceed 3 seconds.
- The application should have the capabilities to store large databases ranging in tens of millions entries and it should be able to expand the database to cater data should the need arise in the future.
- The application should be able to support multiple concurrent users' ideally a million users without shutting down or experiencing crashes.
- Rigorous tests should be conducted to test the application under load of multiple concurrent users and then optimize the performance to negate any problems during peak usage times.
- In the event of a system crash, system reboot should not take more than 24 hours.
- Database operations and queries should be optimized so they do not take more than 3 seconds to process.
- The database should be able to handle multiple transactions in parallel.
- The application should be able to function properly even under poor network conditions with low bandwidth, high ping.
- The application should be reliable with minimal crashes and performance lags.
- The application should be scalable in terms of database capacity, server capacity and concurrent users.
- The application should be highly optimize to reduce battery consumption of the devices.
- The application should not cause the device it runs on to experience crashes and unexpected shutdowns.
- The application should use the resources of the device (CPU, memory, storage) efficiently.
- The application should be able to connect to third party applications such as banking services and the communications between these should be quick and efficient. Third party communications shouldn't take more than 3 seconds.
- There should be backup servers ready to go up incase main server experiences shutdowns or crashes.
- Any issues or faults detected in the application should be fixed within a week of discovery.

5.2. Security Requirements

- 2FA should be implemented to ensure the safety of user accounts.
- Captcha should be implemented to prevent the application from being attacked by bots.
- Measures should be taken to protect the application from DDOS attacks.
- Database security should be robust enough to deal with SQL injection attacks.
- User data should be encrypted using RSA encryption in compliance with modern standards.
- Extensive tests should be performed regularly to identify cracks in application security which should be patched out quickly.
- Security features should be implemented with minimal (less than 2%) performance loss to the functionality of the application.

5.3. Safety Requirements

- Resident data used by the application such as names, addresses, phone number, other personal information should be kept private and never shared to any third parties without user's consent.
- The application should be transparent in telling the user what data it collects and for what purpose it is used.

- The application should follow the regulations set by General Data Protection Regulation (GDPR).

5.4. Software Quality Attributes

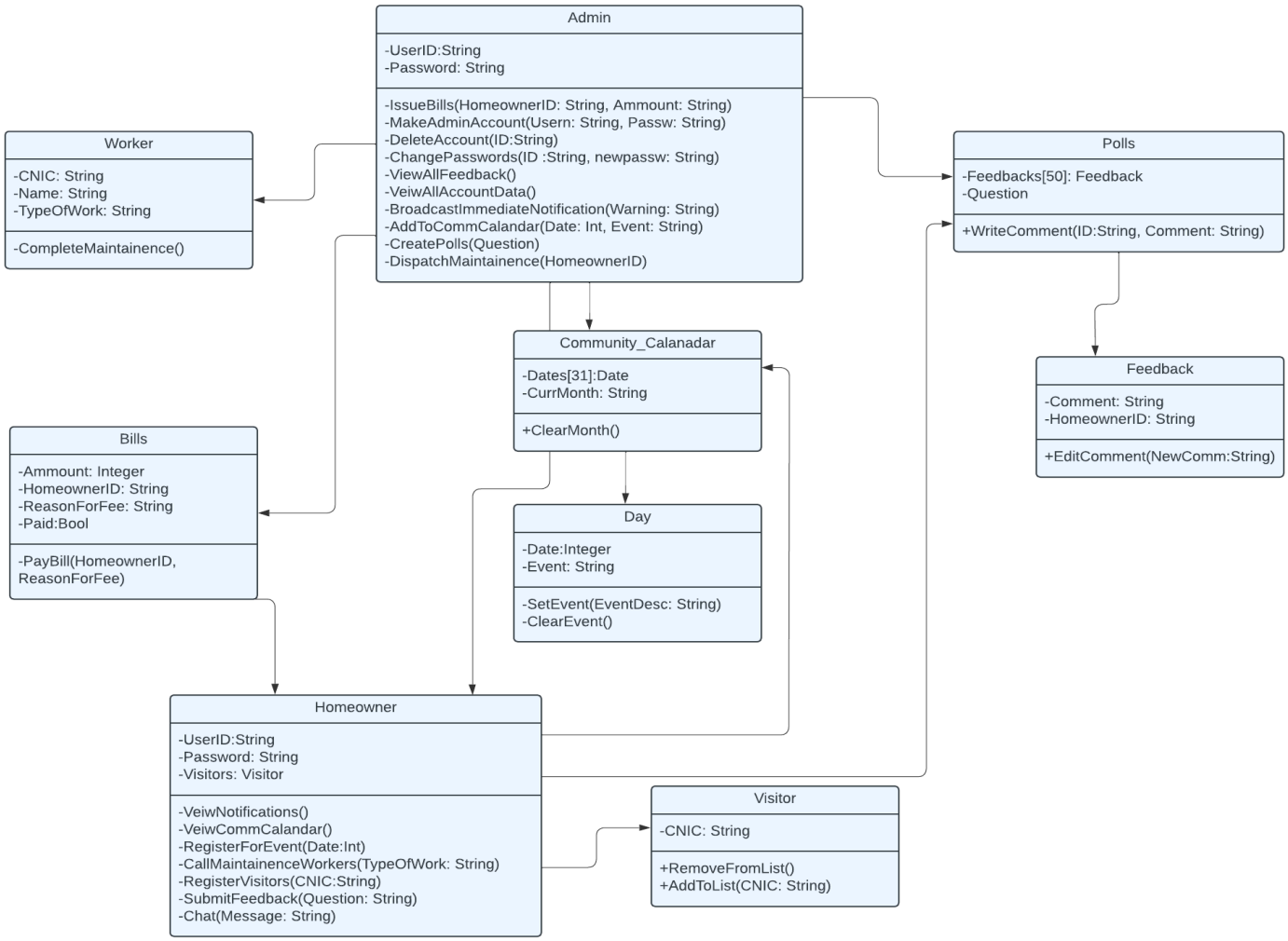
- **Maintainability:** The application should be maintained to ensure proper working and bug fixes. The application should be updated every two months.
- **Accessibility:** The application should have accessibility features for disabled people.
- **Interoperability:** The application should be integrated with a database service and payment gateways.
- **Scalability:** The application should be scalable enough to accommodate more users in the future.
- **Compatibility:** The application should be compatible with phones and computer systems running operating systems such as Android, IOS and Windows.
- **Performance:** The application should respond to user requests quickly, within a second.
- **Usability:** The application should have a simple minimalistic interface that is easy to understand and use.

5.5. Business Rules

- The application should allow admins to view resident database and make changes to it.
- The application should not allow residents to be able to view user information other than their own.
- Only registered members are allowed to access the features of the application.
- Residents are must not be allowed to double book areas for events.
- The application should apply late payment fines for bills and maintenance services.
- The application should share important announcements regarding policies or events happening in the society.
- The residents will provide feedback and vote on polls through the application.

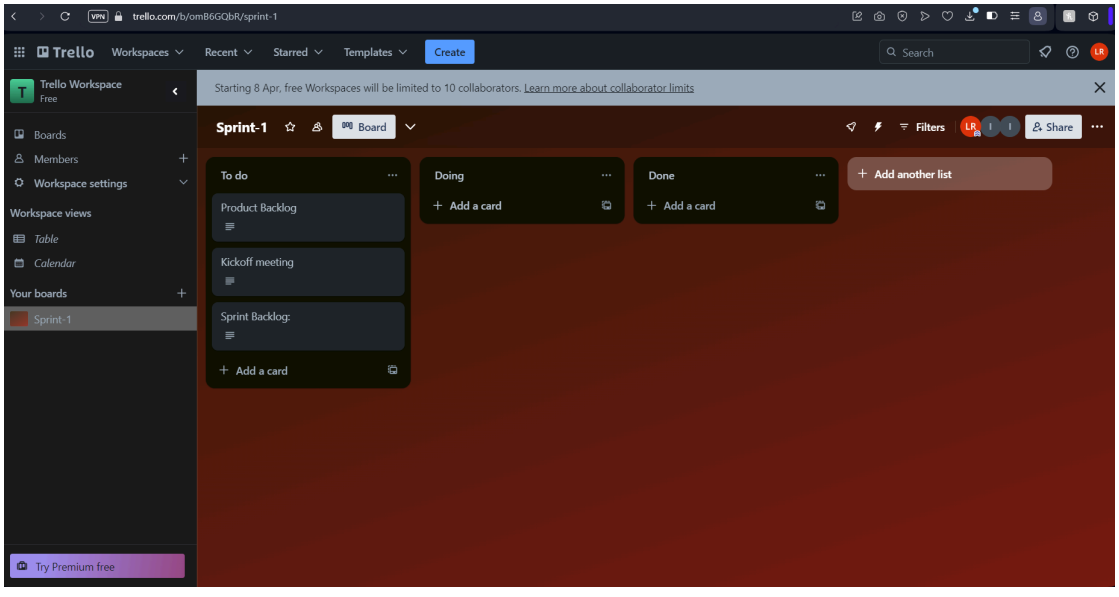
6. Diagrams

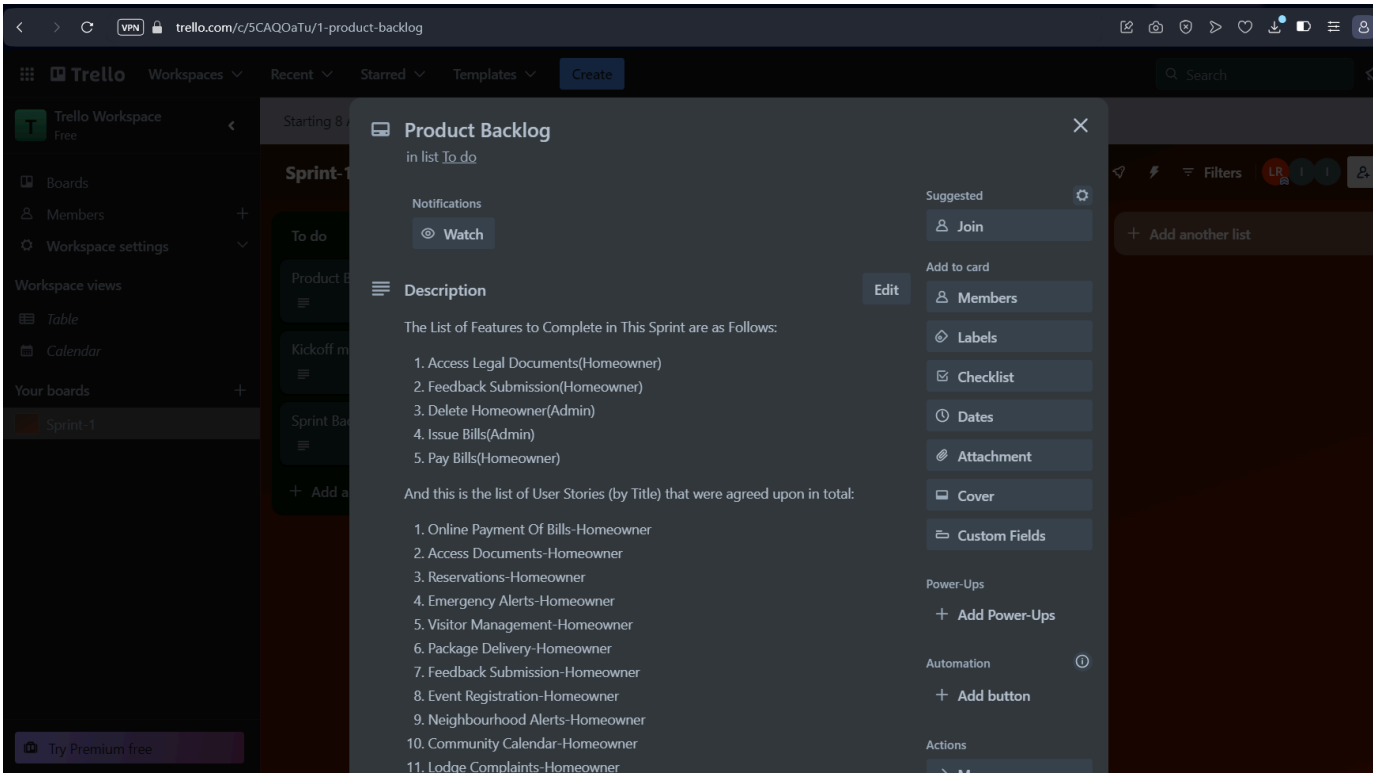
Class Diagram:

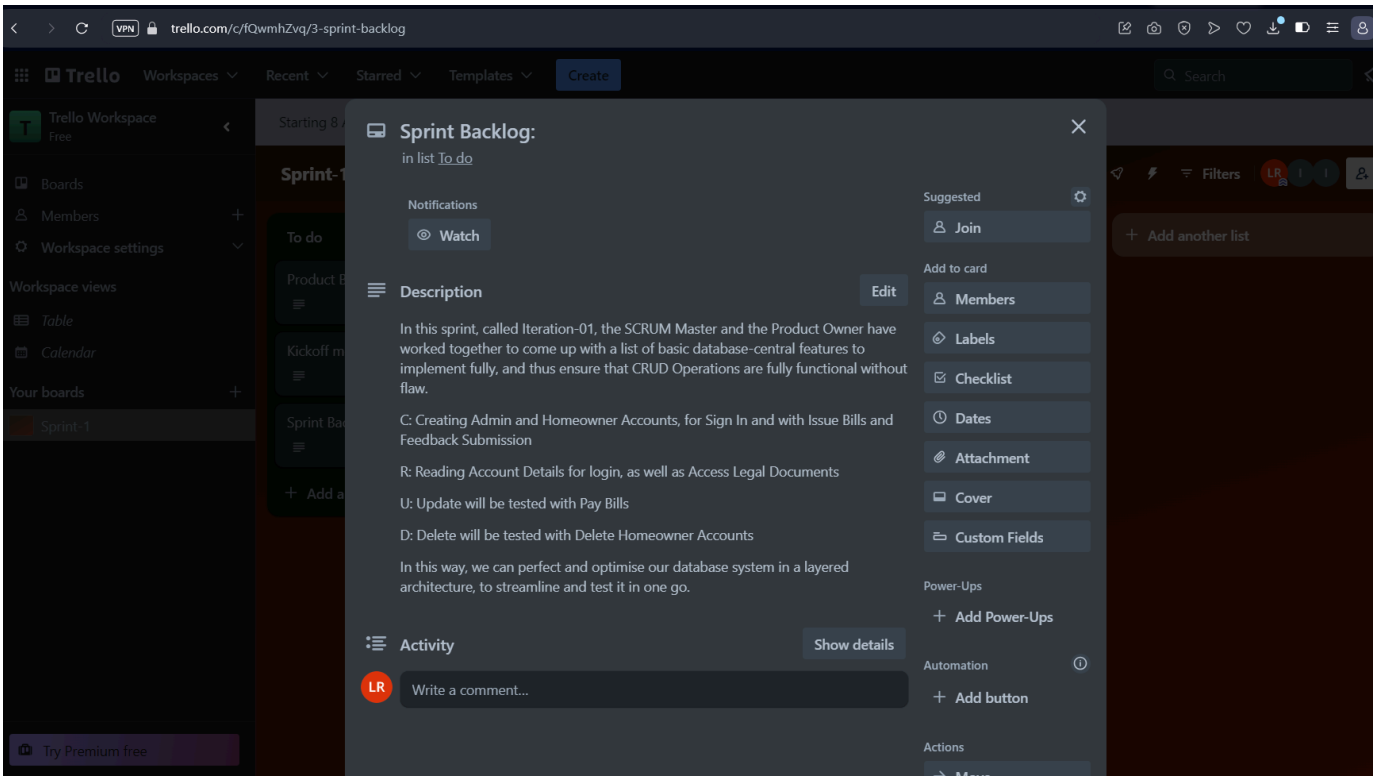


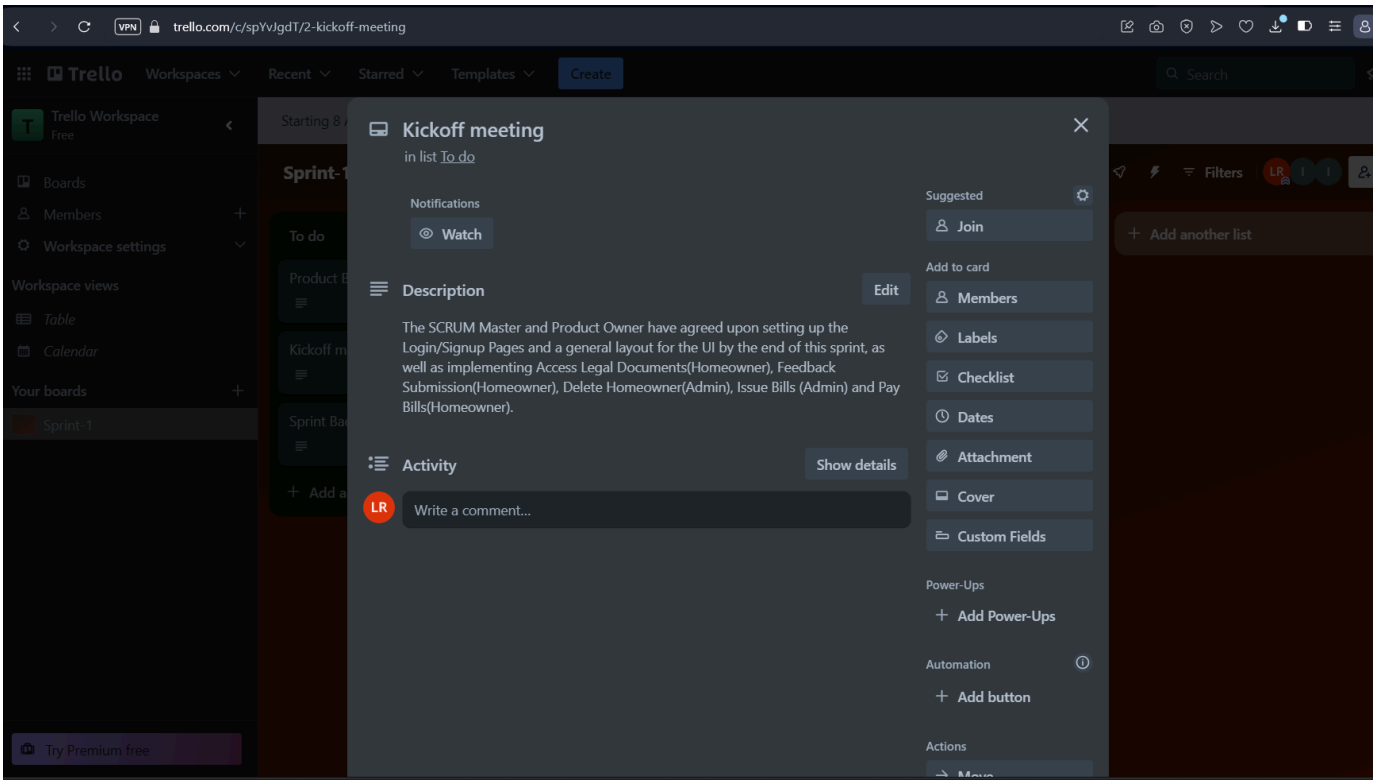
Trello Screenshots:

Snapshot 1:

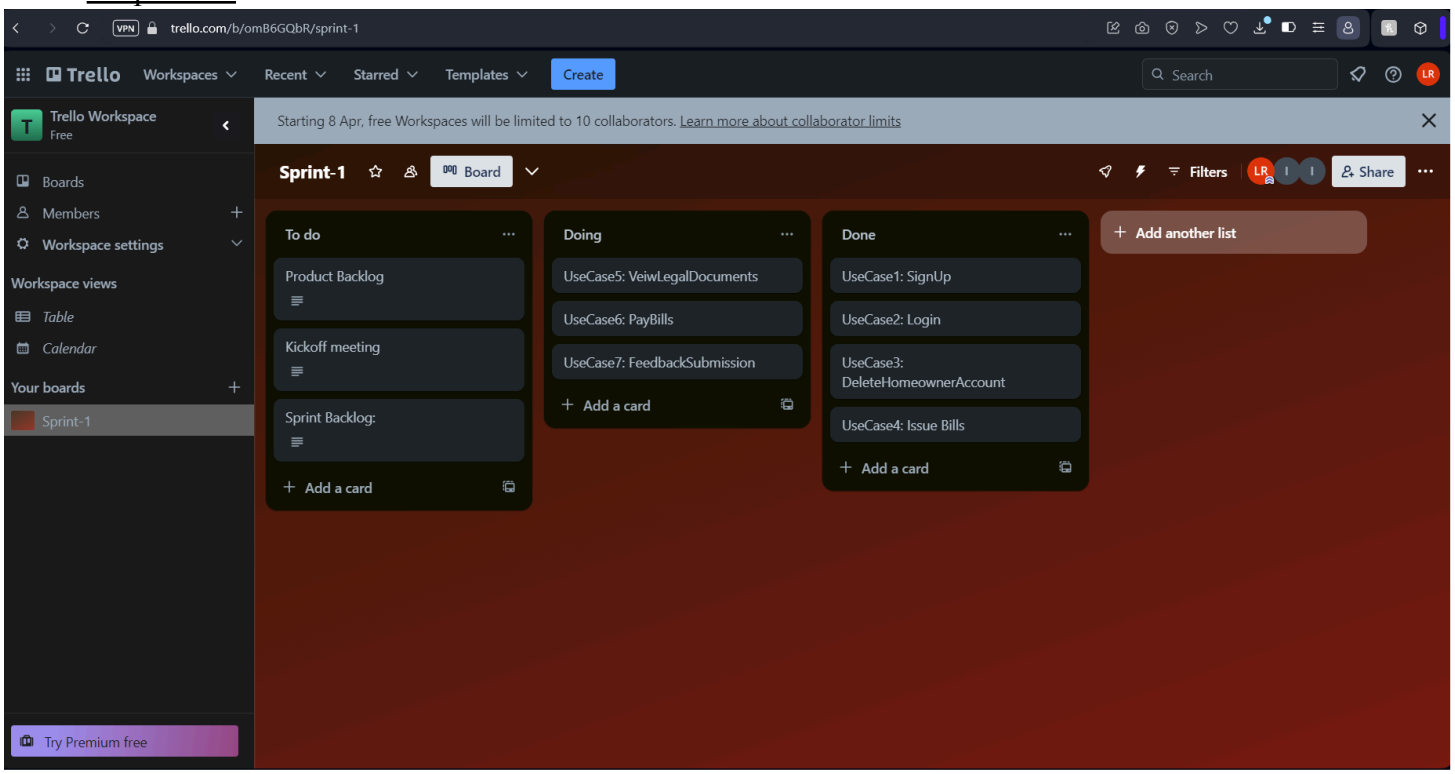



trello.com/c/5CAQOaTu/1-product-backlog
Trello
Workspaces
Recent
Starred
Templates
Create
Search
Trello Workspace
Free
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Sprint-1
Try Premium free
Filters
Add another list
Product Backlog
in list To do
Notifications
Watch
Description
Edit
The List of Features to Complete in This Sprint are as Follows:
1. Access Legal Documents(Homeowner)
2. Feedback Submission(Homeowner)
3. Delete Homeowner(Admin)
4. Issue Bills(Admin)
5. Pay Bills(Homeowner)
And this is the list of User Stories (by Title) that were agreed upon in total:
1. Online Payment Of Bills-Homeowner
2. Access Documents-Homeowner
3. Reservations-Homeowner
4. Emergency Alerts-Homeowner
5. Visitor Management-Homeowner
6. Package Delivery-Homeowner
7. Feedback Submission-Homeowner
8. Event Registration-Homeowner
9. Neighbourhood Alerts-Homeowner
10. Community Calendar-Homeowner
11. Lodge Complaints-Homeowner
Suggested
Join
Add to card
Members
Labels
Checklist
Dates
Attachment
Cover
Custom Fields
Power-Ups
Add Power-Ups
Automation
Add button
Actions

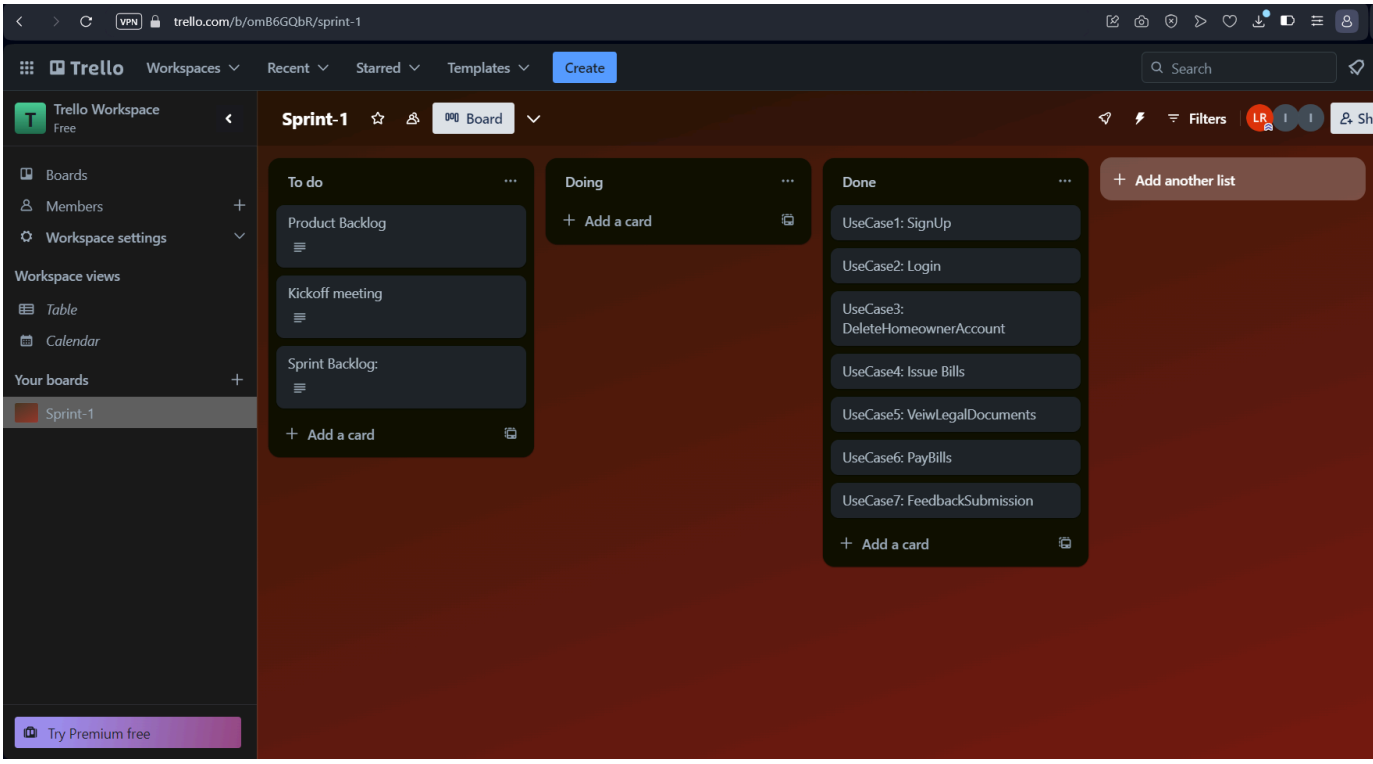

trello.com/c/fQwmhZvq/3-sprint-backlog
Trello
Workspaces
Recent
Starred
Templates
Create
Search
Trello Workspace
Free
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Sprint-1
Try Premium free
Filters
Add another list
Sprint Backlog:
in list To do
Notifications
Watch
Description
Edit
In this sprint, called Iteration-01, the SCRUM Master and the Product Owner have worked together to come up with a list of basic database-central features to implement fully, and thus ensure that CRUD Operations are fully functional without flaw.
C: Creating Admin and Homeowner Accounts, for Sign In and with Issue Bills and Feedback Submission
R: Reading Account Details for login, as well as Access Legal Documents
U: Update will be tested with Pay Bills
D: Delete will be tested with Delete Homeowner Accounts
In this way, we can perfect and optimise our database system in a layered architecture, to streamline and test it in one go.
Activity
Show details
LR
Write a comment...
Suggested
Join
Add to card
Members
Labels
Checklist
Dates
Attachment
Cover
Custom Fields
Power-Ups
Add Power-Ups
Automation
Add button
Actions


trello.com/c/spYvJgdT/2-kickoff-meeting
Trello
Workspaces
Recent
Starred
Templates
Create
Search
Trello Workspace
Free
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Sprint-1
Try Premium free
Starting 8
Sprint-1
To do
Filters
Add another list
Kickoff meeting
in list To do
Notifications
Watch
Description
Edit
The SCRUM Master and Product Owner have agreed upon setting up the Login/Signup Pages and a general layout for the UI by the end of this sprint, as well as implementing Access Legal Documents(Homeowner), Feedback Submission(Homeowner), Delete Homeowner(Admin), Issue Bills (Admin) and Pay Bills(Homeowner).
Activity
Show details
LR
Write a comment...
Suggested
Join
Add to card
Members
Labels
Checklist
Dates
Attachment
Cover
Custom Fields
Power-Ups
Add Power-Ups
Automation
Add button
Actions

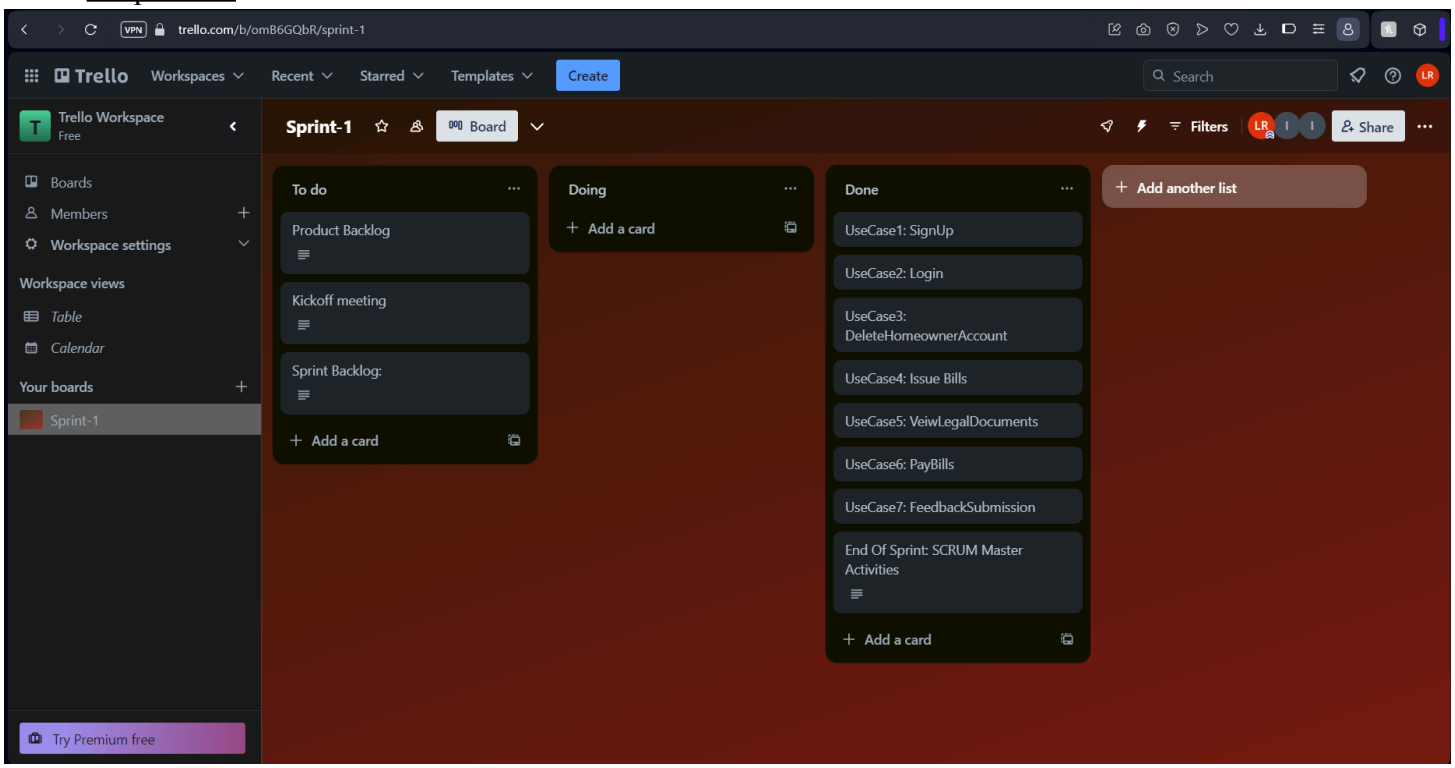
Snapshot 2:

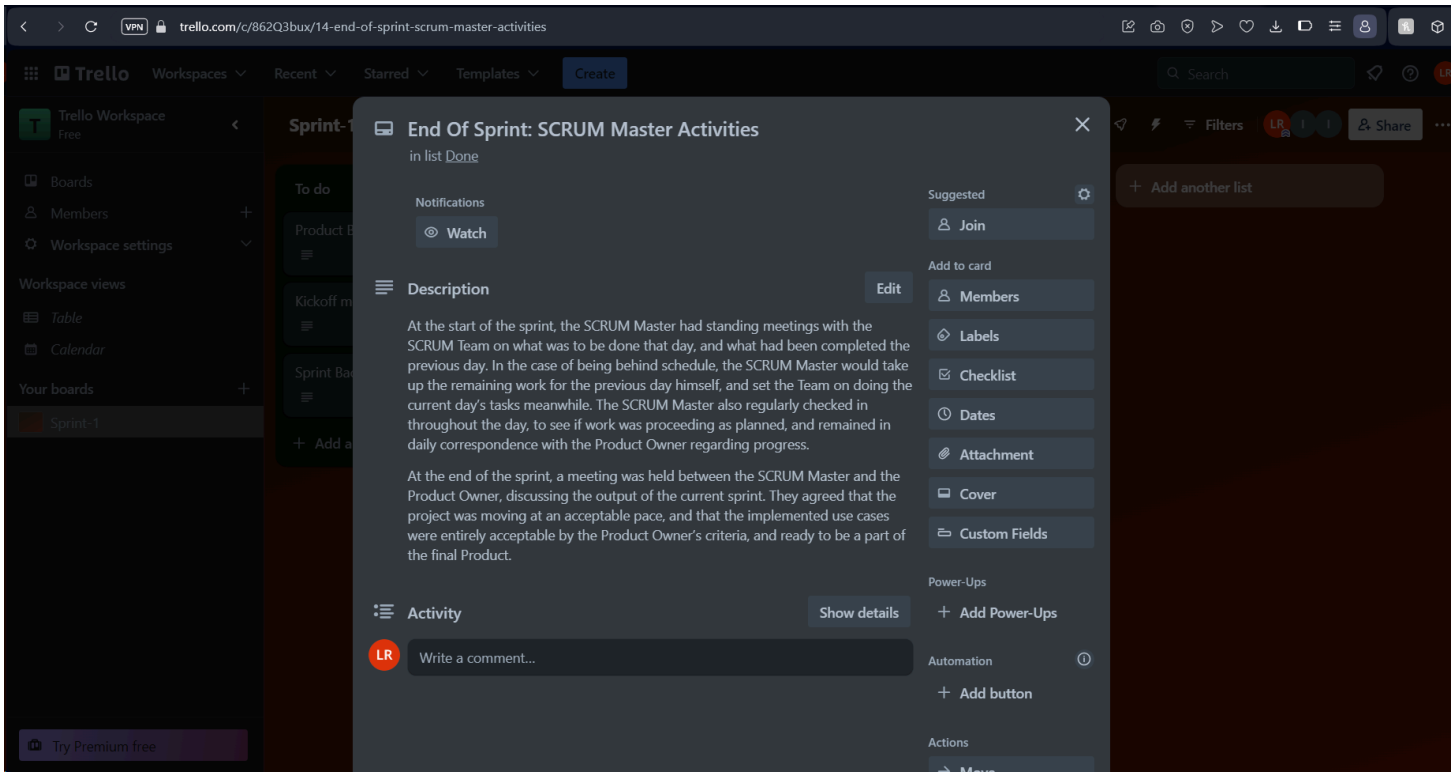

trello.com/b/omB6GQbR/sprint-1
Trello
Workspaces
Recent
Starred
Templates
Create
Search
Trello Workspace
Free
Starting 8 Apr, free Workspaces will be limited to 10 collaborators. Learn more about collaborator limits
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Sprint-1
Try Premium free
Sprint-1
Board
Filters
Share
To do
Product Backlog
Kickoff meeting
Sprint Backlog:
Add a card
Doing
UseCase5: VeiwLegalDocuments
UseCase6: PayBills
UseCase7: FeedbackSubmission
Add a card
Done
UseCase1: SignUp
UseCase2: Login
UseCase3: DeleteHomeownerAccount
UseCase4: Issue Bills
Add a card
Add another list

Snapshot 3:



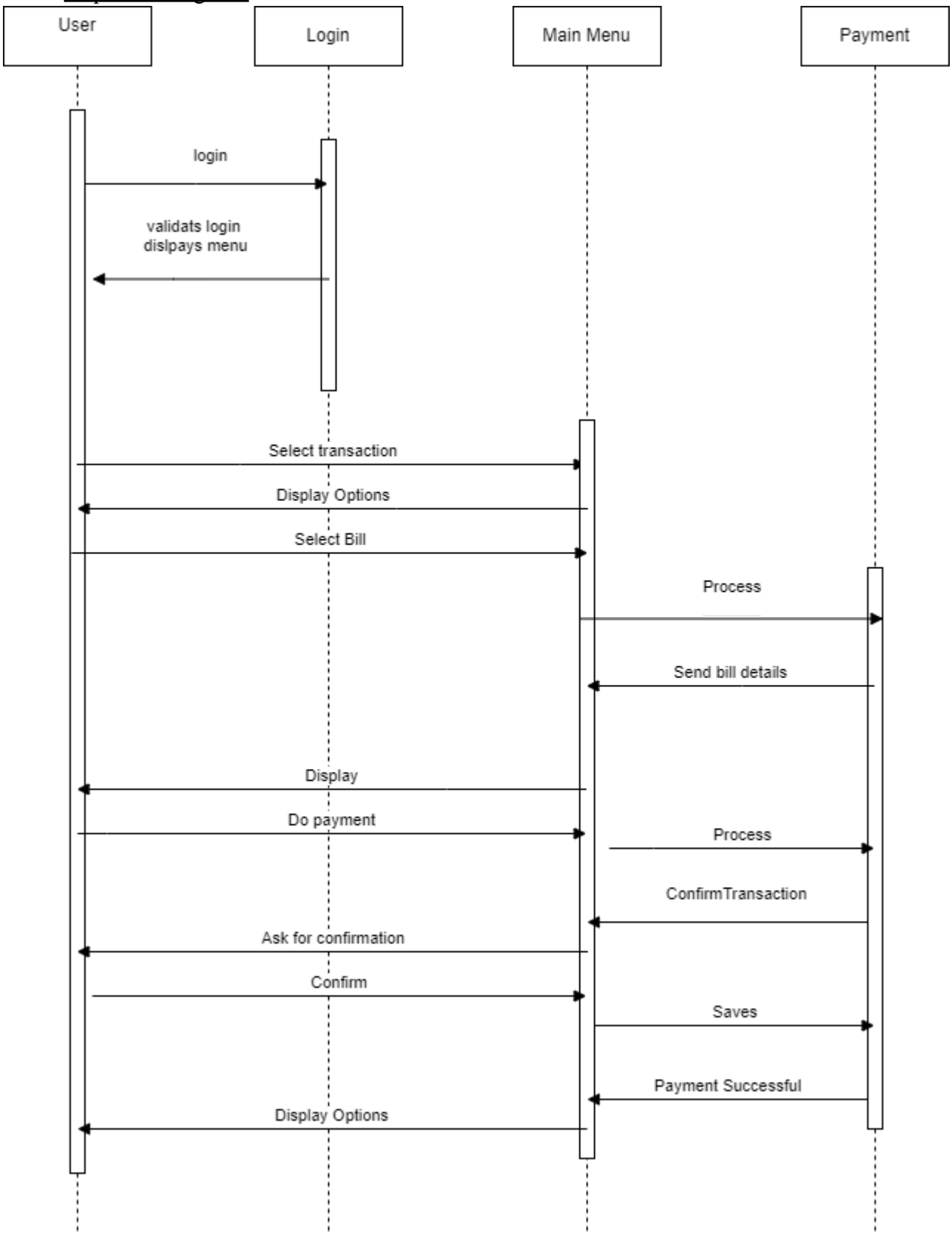
Snapshot 4:



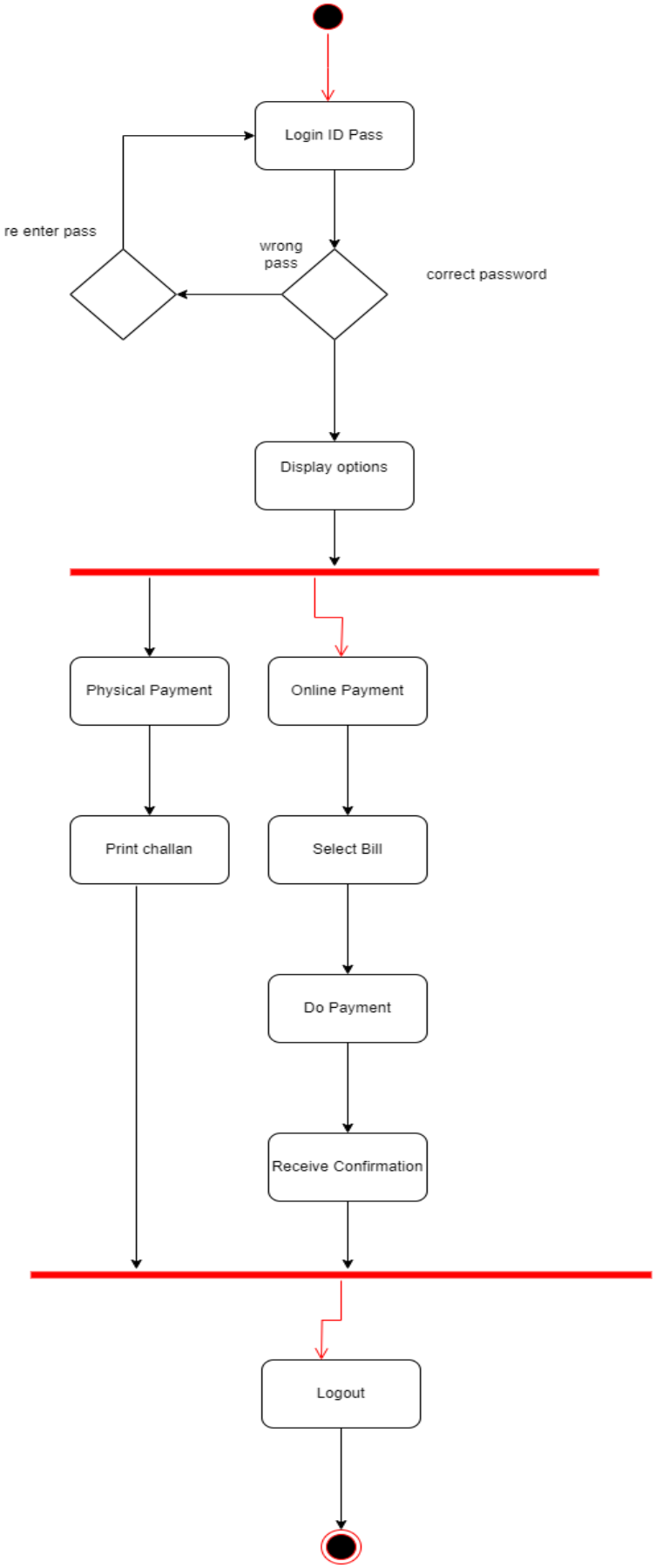

trello.com/c/862Q3bux/14-end-of-sprint-scrum-master-activities
Trello
Workspaces
Recent
Starred
Templates
Create
Search
Trello Workspace
Free
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Sprint-1
Try Premium free
To do
Add another list
Filters
Share
End Of Sprint: SCRUM Master Activities
in list Done
Notifications
Watch
Description
Edit
At the start of the sprint, the SCRUM Master had standing meetings with the SCRUM Team on what was to be done that day, and what had been completed the previous day. In the case of being behind schedule, the SCRUM Master would take up the remaining work for the previous day himself, and set the Team on doing the current day's tasks meanwhile. The SCRUM Master also regularly checked in throughout the day, to see if work was proceeding as planned, and remained in daily correspondence with the Product Owner regarding progress.
At the end of the sprint, a meeting was held between the SCRUM Master and the Product Owner, discussing the output of the current sprint. They agreed that the project was moving at an acceptable pace, and that the implemented use cases were entirely acceptable by the Product Owner's criteria, and ready to be a part of the final Product.
Activity
Show details
LR
Write a comment...
Suggested
Join
Add to card
Members
Labels
Checklist
Dates
Attachment
Cover
Custom Fields
Power-Ups
Add Power-Ups
Automation
Add button
Actions

Major Use Case 1: Pay Bills(Homeowner)

Sequence Diagram:

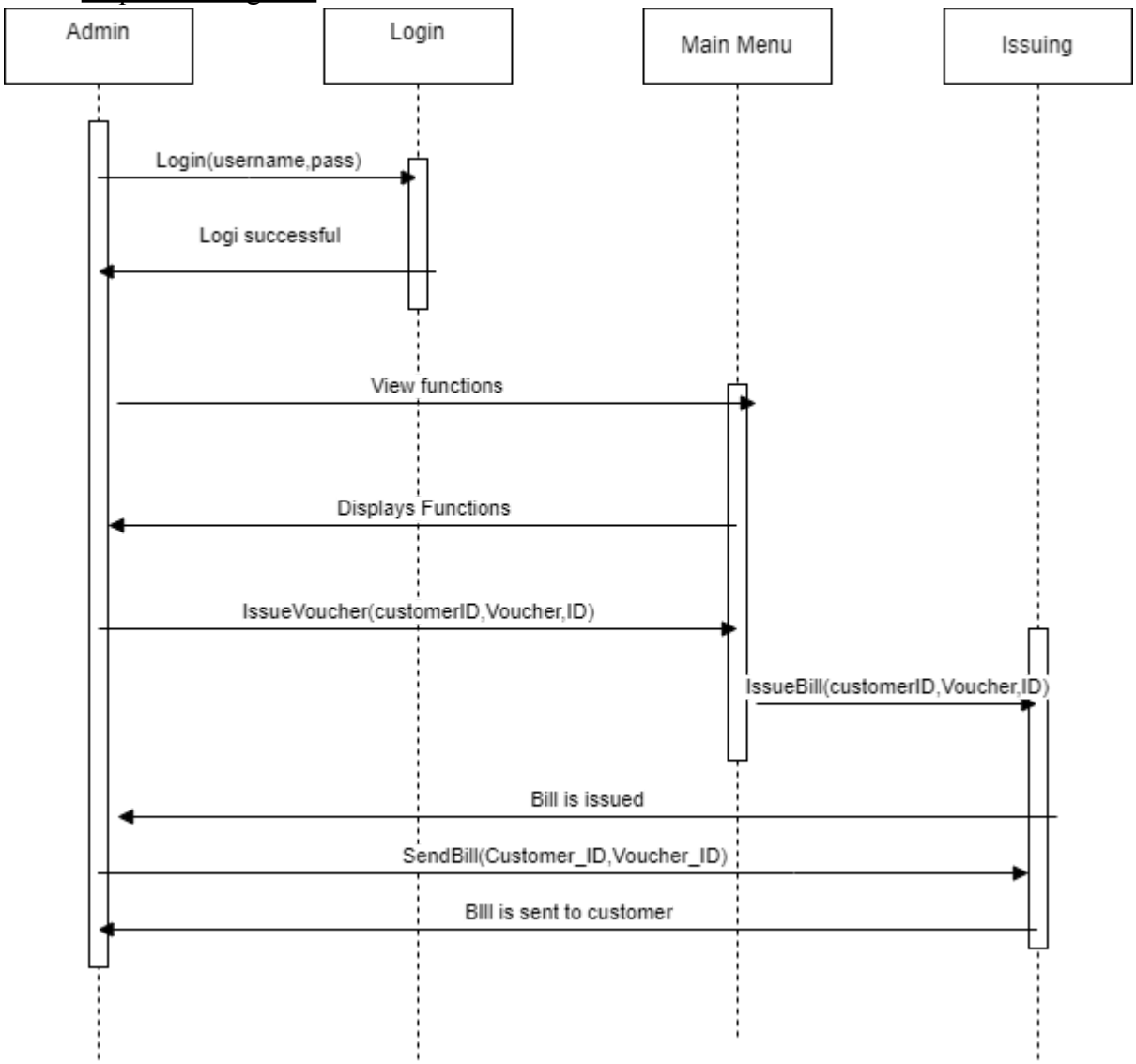


Activity Diagram:

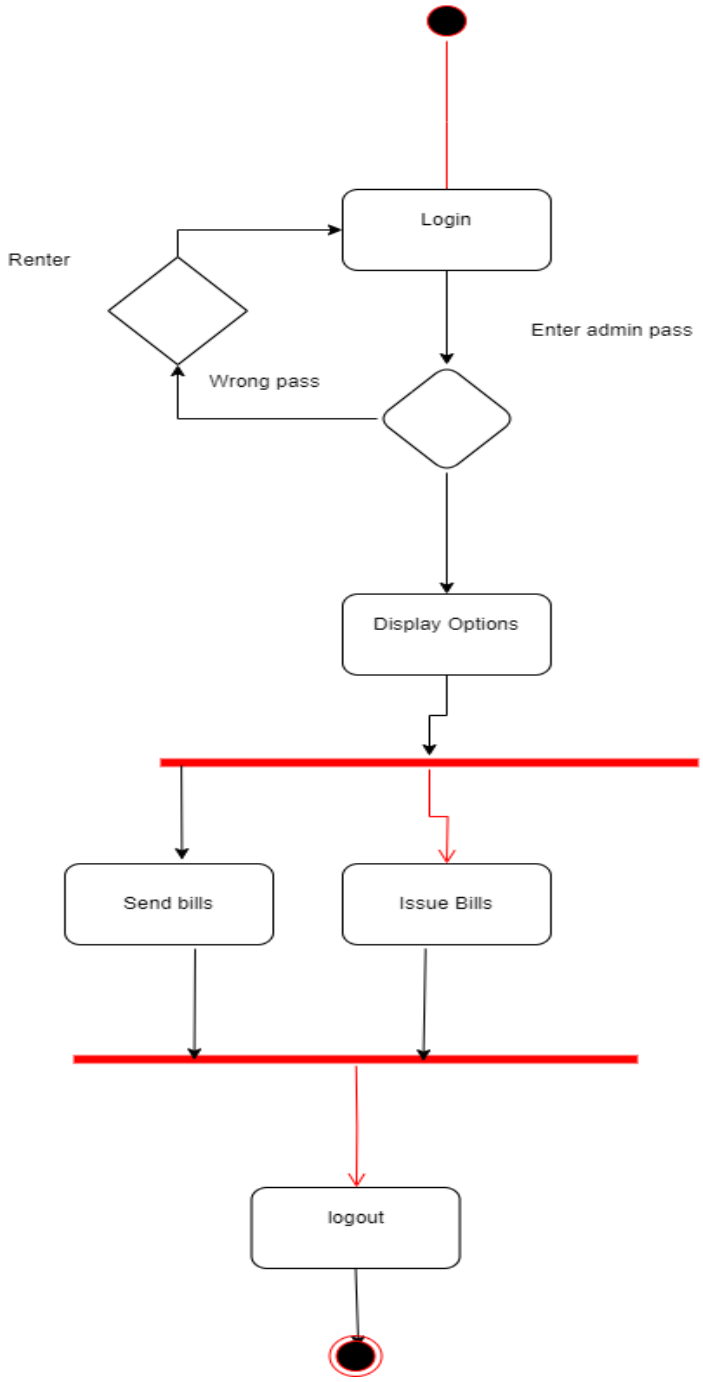

Login ID Pass
re enter pass
wrong pass
correct password
Display options
Physical Payment
Online Payment
Print challan
Select Bill
Do Payment
Receive Confirmation
Logout

Major Use Case 2: Issue Bills (Admin)

Sequence Diagram:

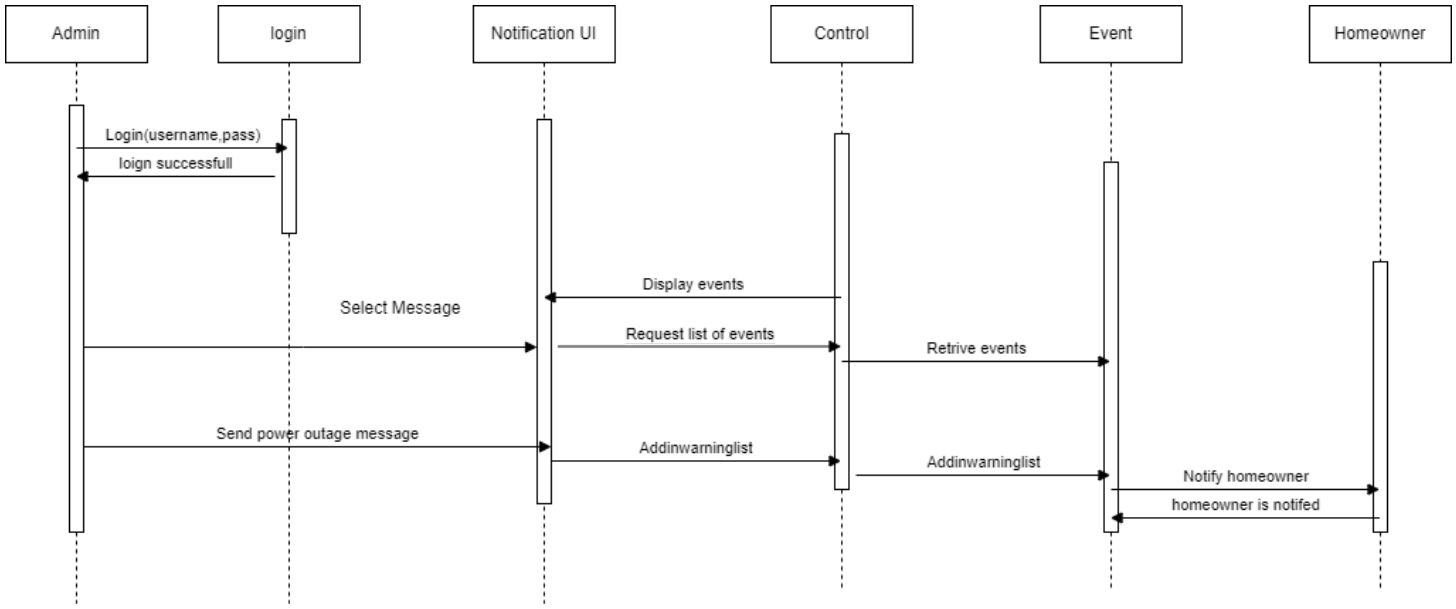


Activity Diagram:

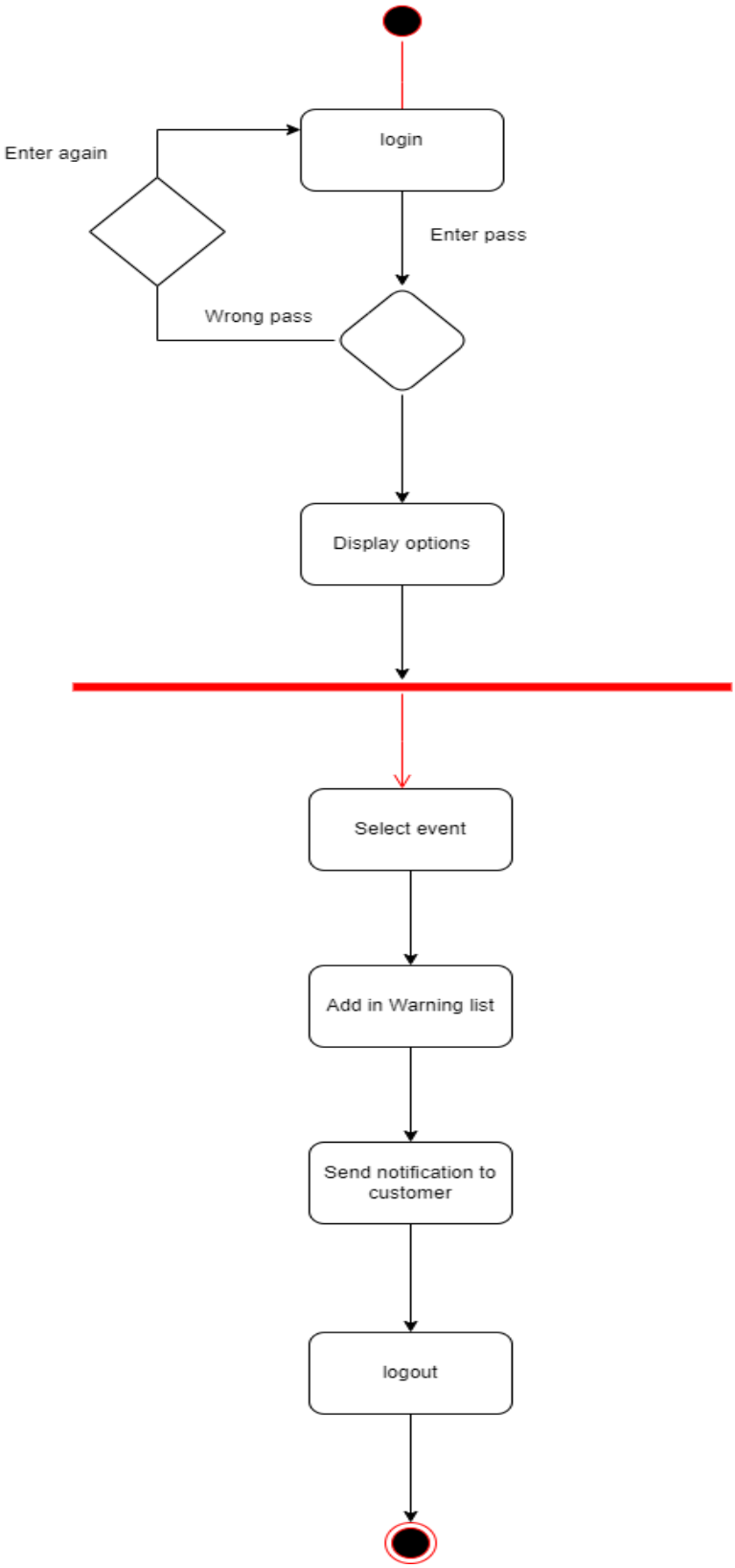


Major Use Case 3: Broadcast Warning Notifications(Admin)

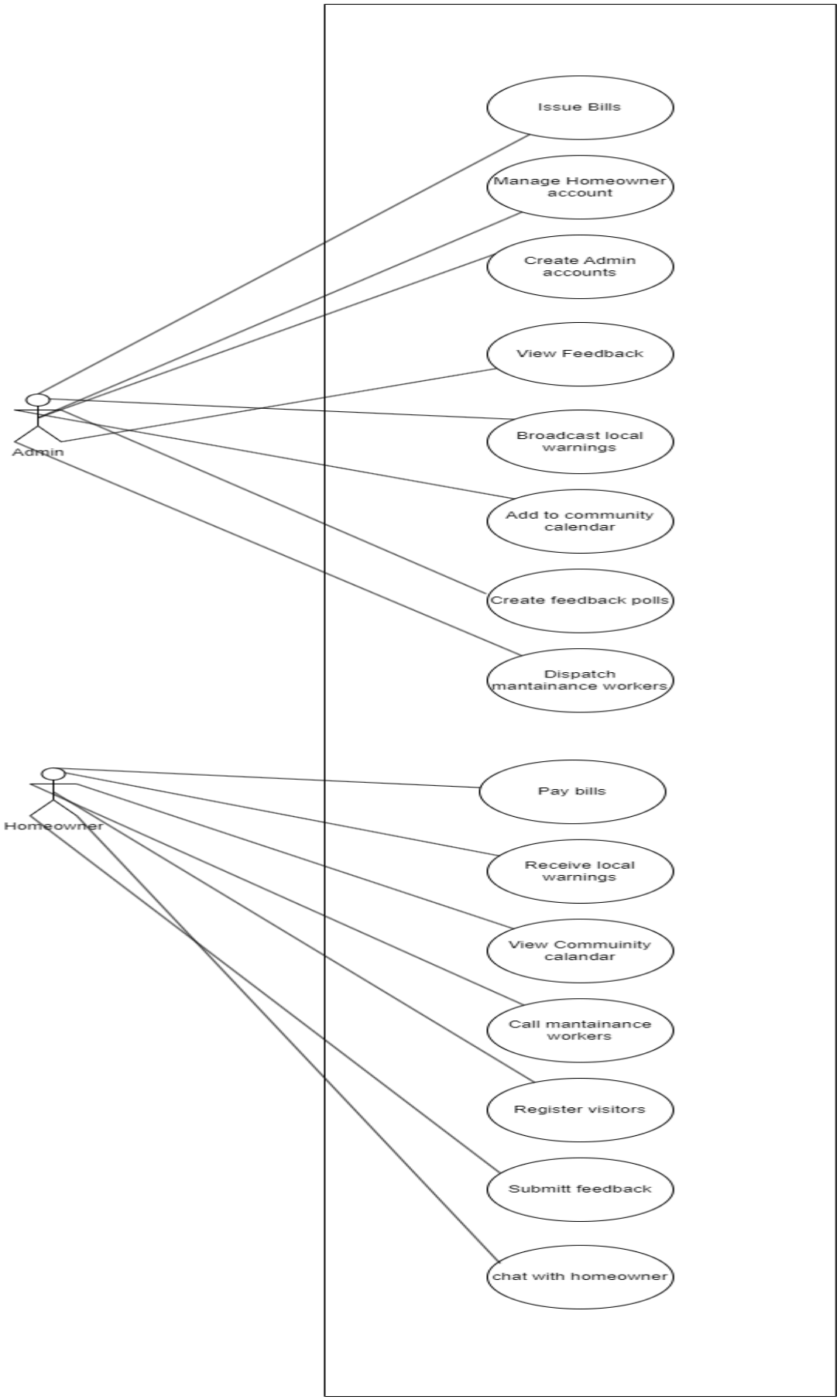
Sequence Diagram:


Admin
login
Notification UI
Control
Event
Homeowner
Login(username,pass)
loign successfull
Display events
Select Message
Request list of events
Retrive events
Send power outage message
Addinwarninglist
Addinwarninglist
Notify homeowner
homeowner is notifed

Activity Diagram:



Use Case Diagram:



Github Screenshot: