10 Applications and their stakeholders

Ameya S. Daddikar
College I.D.: 161070015
Btech. Computer Engineering, V.J.T.I.

1. Google Search

Name	Type	Role
User	External	Uses the search engine to browse the web
Testers	Internal	Cross Platform Testing, Integration Testing, etc.
System Administrators	Internal	Server Management & DevOps, Content Filtering
Advertisers	External	Search based Advertisement, targeted ads, etc.
Developers	Internal	Develop and optimise search functionalities and add smart searches

2. GoDaddy Web Services

Name	Type	Role
User	External	Buys and manages domains, servers, CMS, other web services.
System Administrators	Internal	Management of server stacks, load balancers, Hypervisors, etc.
Domain Providers	External	Provides domains to sell to GoDaddy

3. Amazon Marketplace

Name	Type	Role
User	External	Searches for products and buys
Testers	Internal	Website testing
System Administrators	Internal	
Advertisers	External	Advertises based on user buying habits
Developers	Internal	
Sellers	External	Certified sellers that can post their products on Amazon
Shipping	Internal	Real time shipping updates and planning

4. Apple

Name	Type	Role
Consumer	External	Buys Apple devices and services
Testers	Internal	Hardware and Software testers
System Designers	Internal	OS/ API designers, Hardware Designers
Developers	Internal	Software developers

5. VISA Payment

Name	Type	Role
User	External	Uses VISA services for online payments
Testers	Internal	Pen testing, load testing
System Administrators	Internal	Fraud detection, devops, etc.
Businesses	External	Businesses that use VISA payments and offer discounts/ benefits for using VISA services
DevOps	Internal	Maintaining payment gateway servers
Bank	External	Enabling payment options using VISA gateways

6. BookMyShow

Name	Type	Role
User	External	View movies/plays, books tickets
Testers	Internal	Stress testing, Transaction consistency
System Administrators	Internal	Server administration
Advertisers	External	Movie ticket offers/ giveaways
Developers	Internal	Build App & Web platforms
Movie Marketers	External	Publish Trailers/ advertisements
Cinema House	External	Publishes movie show timings and lets BookMyShow handle seat bookings

7. NVDIA

Name	Type	Role
User	External	Uses Graphics cards in server stacks/ gaming rigs
Testers	Internal	Hardware benchmarking, firmware updates, etc.
Hardware Manufacturers	Internal	Graphics card manufacturing
Game developers	External	Designs game engines using the features of graphics drivers and hardware
Graphics Driver Developers	Internal	Optimizing

8. Visual Studio

Name	Type	Role
User	External	Uses as a code editor
Testers	Internal	Cross Platform testing of new features
Developers	Internal	Adding functionality

9. Android

Name	Type	Role
User	External	Uses a device with Android OS
Testers	External	Security Testing
App Developers	External	Uses the Android Java/Kotlin Frameworks to build applications
Developers	Internal	Build the Android OS on the existing linux kernel
Mobile Manufacturers	External	Build Hardware based on Android Specifications

10. Arduino

Name	Type	Role
Personal User	External	Uses Arduino board in personal projects
Embedded Systems Company	Internal	Incorporated Arduino as controller in IoT/embedded systems
Developers	Internal	Creates development IDE and Arduino API
Hardware Modules Manufacturer	External	Creates supporting modules/boards for the Arduino controller.

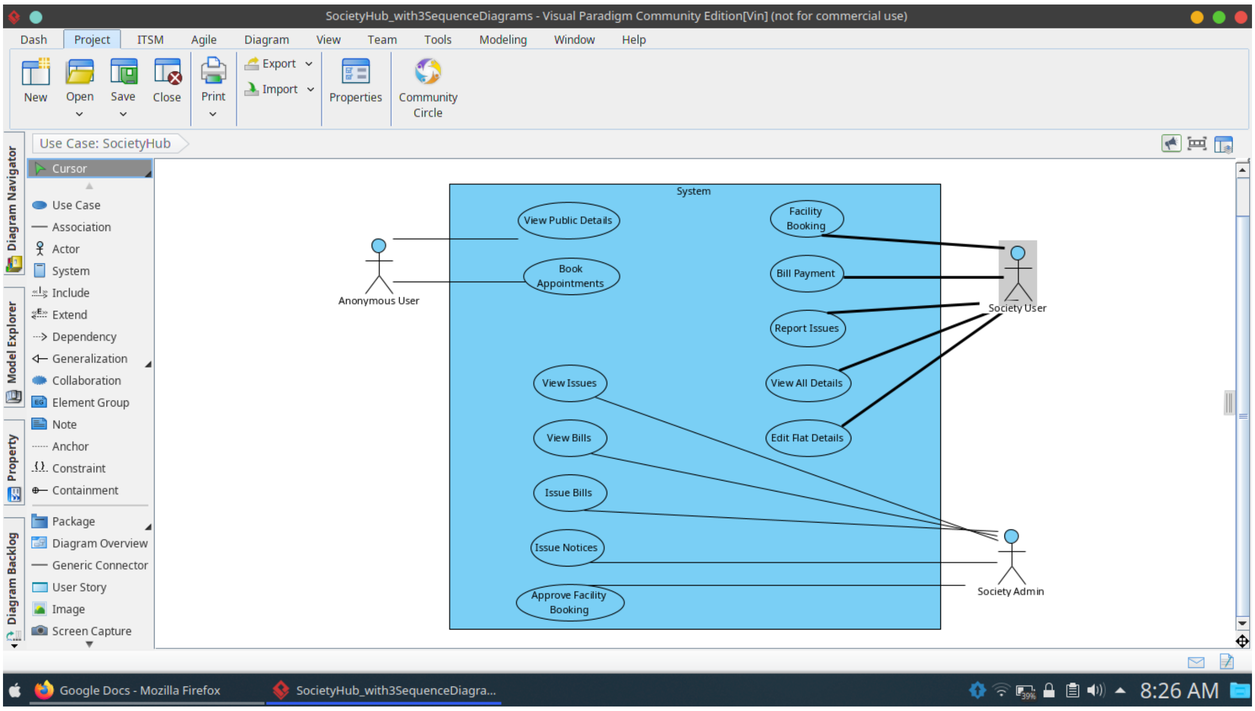
TUTORIAL 2

AIM:

To study and prepare UML diagrams for the Society Management Systerm

DIAGRAMS

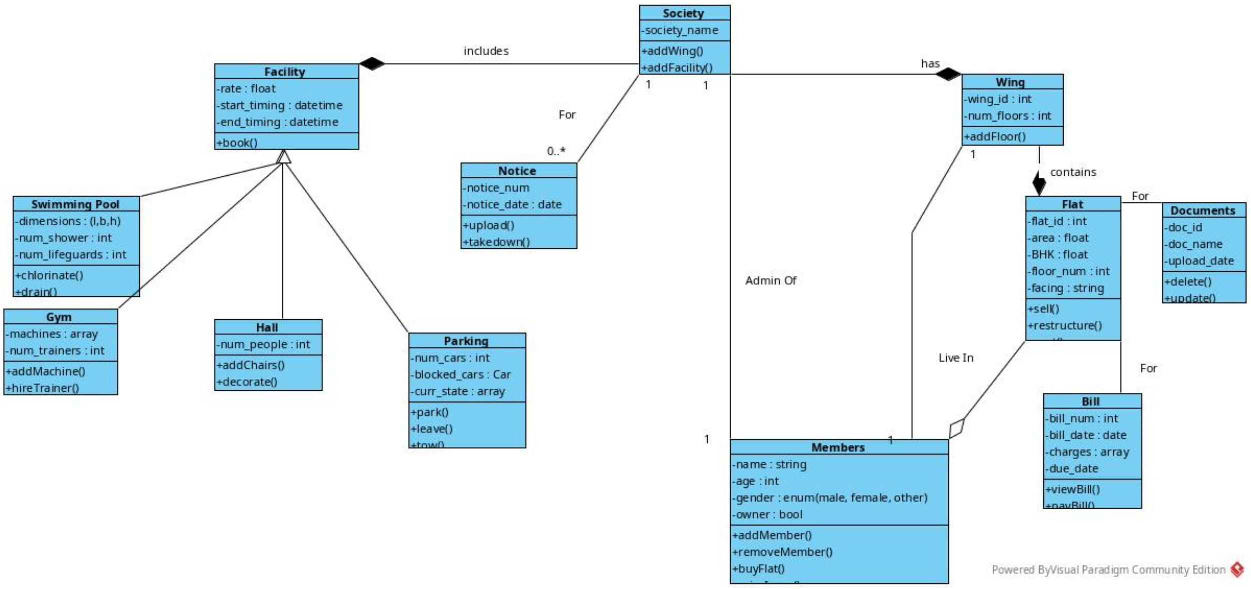
Use case diagram



Major use cases are:

- Book appointments: Flat owners and admins can have appointments with outsiders
- View Bills: Bills of maintenance, electricity, etc. are visible
- Issue notices: Admin can add notices for all or some flats
- Book facility: A facility like hall can be booked
- Report issues: A complaint manager

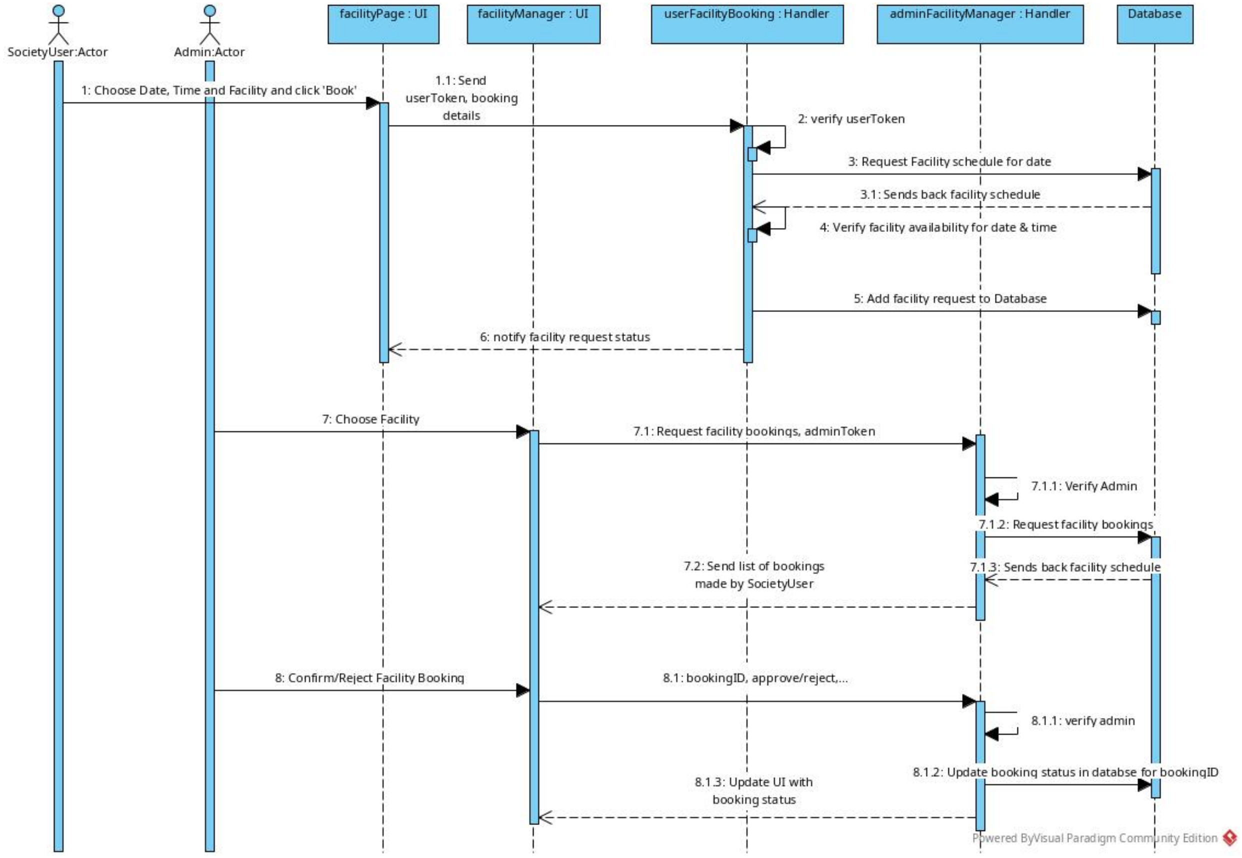
Class diagram



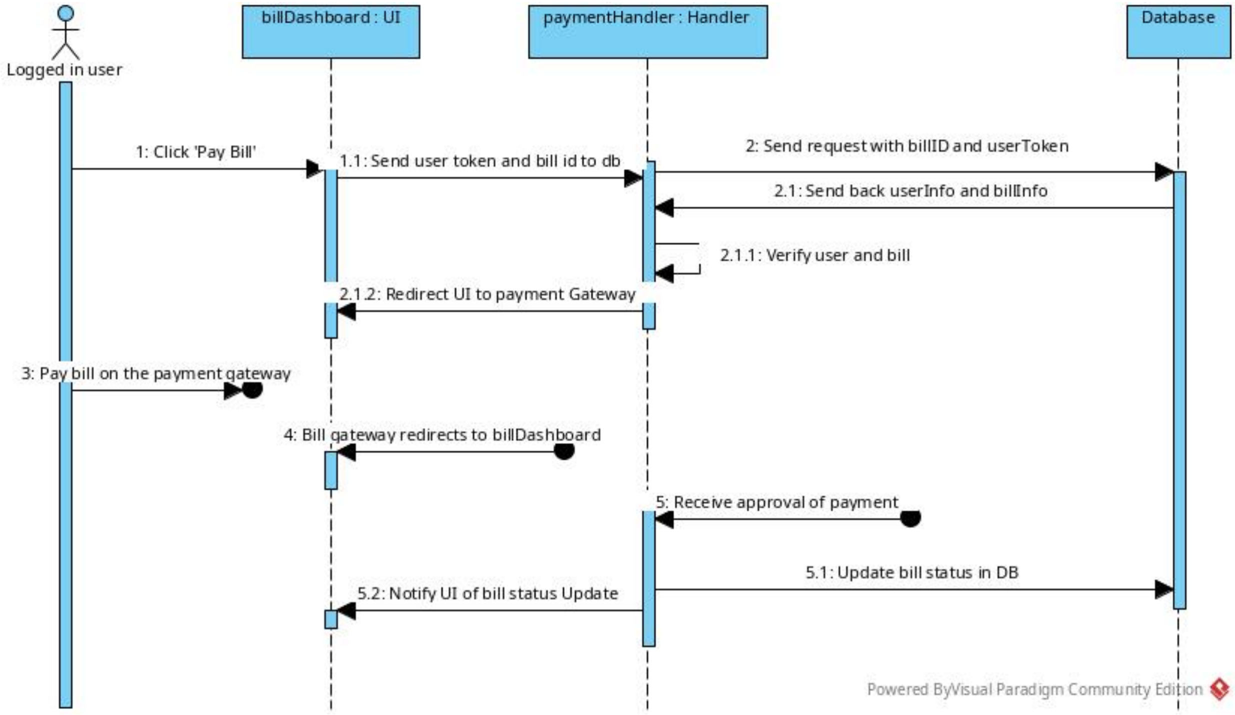
Major classes are:

- Society: Overall society
- Facility: A part of society; consists of a facility commonly available
- Notice: A notice class whch is announces
- Mamber: A person living in the society
- Wing: A single wing in the society
- Flat: A single flat belonging to a wing
- Bill: Represents a bill of a flat for a given period

Sequence diagram

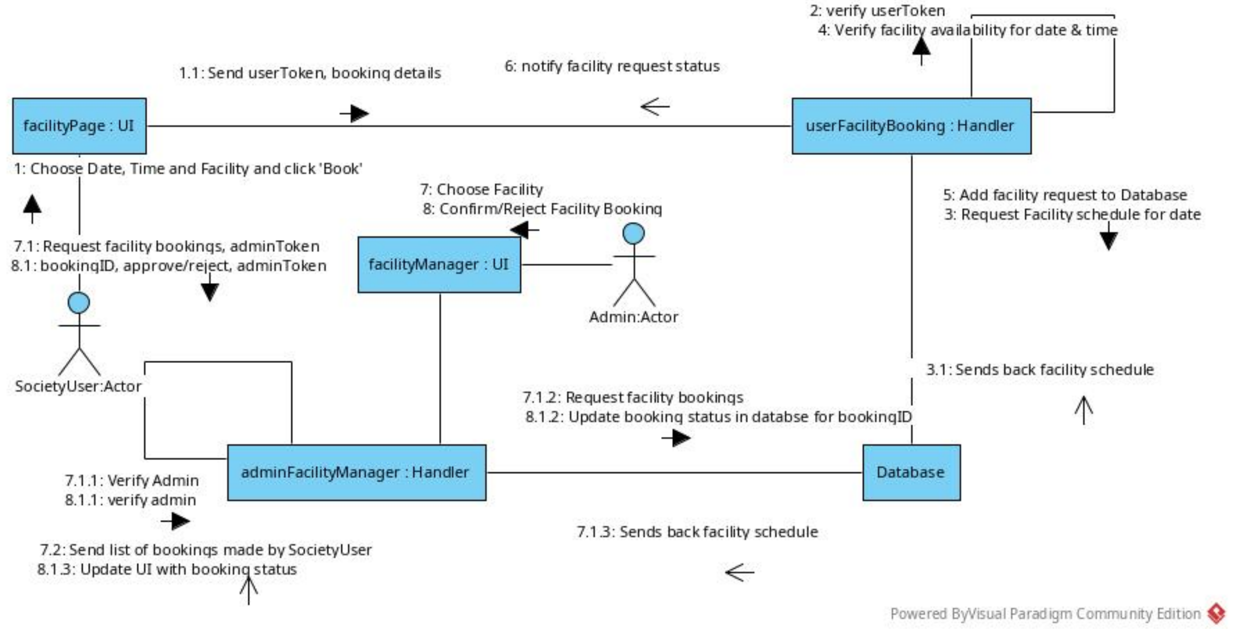

SocietyUser:Actor
Admin:Actor
facilityPage : UI
facilityManager : UI
userFacilityBooking : Handler
adminFacilityManager : Handler
Database
1: Choose Date, Time and Facility and click 'Book'
1.1: Send userToken, booking details
2: verify userToken
3: Request Facility schedule for date
3.1: Sends back facility schedule
4: Verify facility availability for date & time
5: Add facility request to Database
6: notify facility request status
7: Choose Facility
7.1: Request facility bookings, adminToken
7.1.1: Verify Admin
7.1.2: Request facility bookings
7.2: Send list of bookings made by SocietyUser
7.1.3: Sends back facility schedule
8: Confirm/Reject Facility Booking
8.1: bookingID, approve/reject,...
8.1.1: verify admin
8.1.2: Update booking status in databse for bookingID
8.1.3: Update UI with booking status
Powered ByVisual Paradigm Community Edition

Facility booking

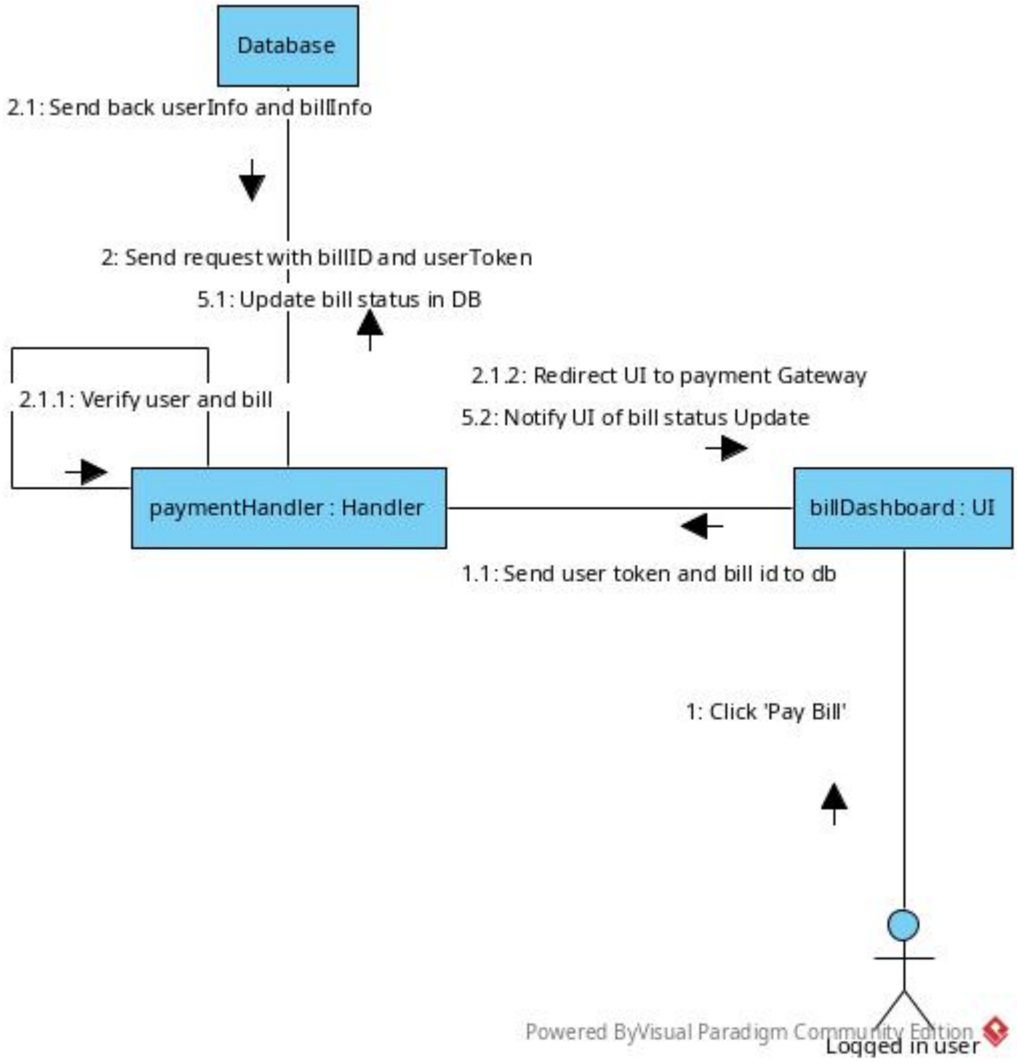

Logged in user
billDashboard : UI
paymentHandler : Handler
Database
1: Click 'Pay Bill'
1.1: Send user token and bill id to db
2: Send request with billID and userToken
2.1: Send back userInfo and billInfo
2.1.1: Verify user and bill
2.1.2: Redirect UI to payment Gateway
3: Pay bill on the payment gateway
4: Bill gateway redirects to billDashboard
5: Receive approval of payment
5.1: Update bill status in DB
5.2: Notify UI of bill status Update
Powered ByVisual Paradigm Community Edition

Bill payment

Collaboration

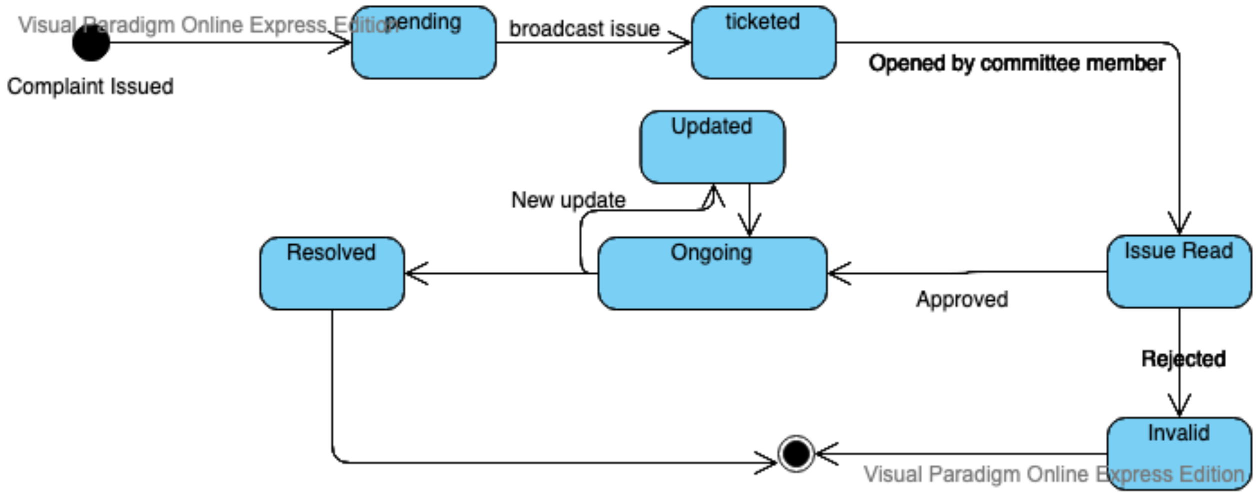

2: verify userToken
4: Verify facility availability for date & time
1.1: Send userToken, booking details
6: notify facility request status
facilityPage : UI
userFacilityBooking : Handler
1: Choose Date, Time and Facility and click 'Book'
7: Choose Facility
8: Confirm/Reject Facility Booking
5: Add facility request to Database
3: Request Facility schedule for date
7.1: Request facility bookings, adminToken
8.1: bookingID, approve/reject, adminToken
facilityManager : UI
Admin:Actor
SocietyUser:Actor
3.1: Sends back facility schedule
7.1.2: Request facility bookings
8.1.2: Update booking status in databse for bookingID
adminFacilityManager : Handler
Database
7.1.1: Verify Admin
8.1.1: verify admin
7.1.3: Sends back facility schedule
7.2: Send list of bookings made by SocietyUser
8.1.3: Update UI with booking status
Powered ByVisual Paradigm Community Edition

Facility booking


Database
2.1: Send back userInfo and billInfo
2: Send request with billID and userToken
5.1: Update bill status in DB
2.1.2: Redirect UI to payment Gateway
5.2: Notify UI of bill status Update
2.1.1: Verify user and bill
paymentHandler : Handler
billDashboard : UI
1.1: Send user token and bill id to db
1: Click 'Pay Bill'
Powered ByVisual Paradigm Community Edition
Logged in user

Bill payment

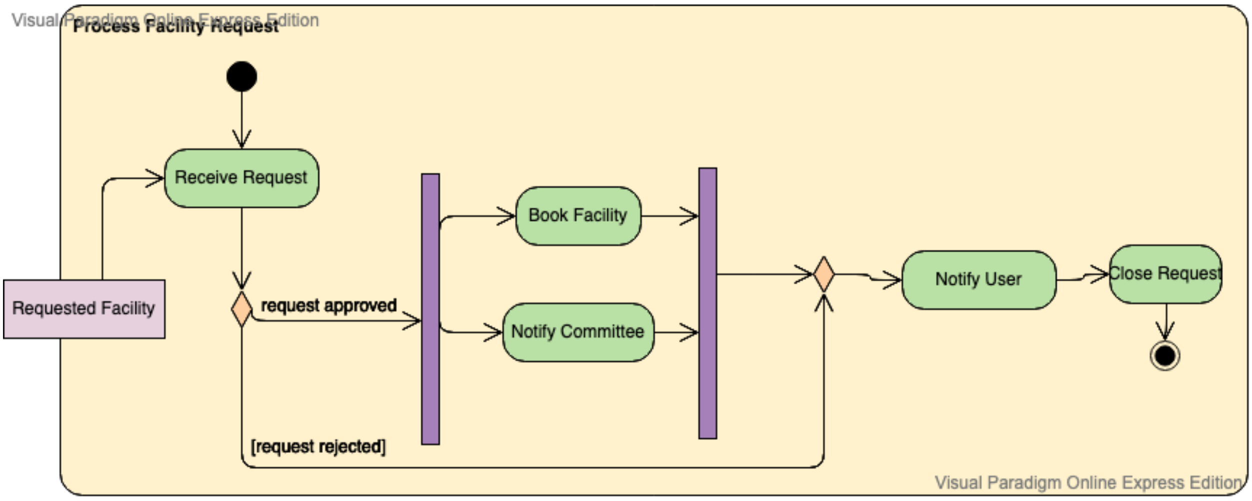
State transition



State transition diagram is used to show the movement of system through various states
The given diagram is for complaint states. It moves as:

- User issues a complaint
- The complaint is in a pending state until it is in the system as a ticket
- When opened by a committee member, it becomes a read issue
- Committee member may accept or reject the issue
- If accepted, it must be updated
- Finally it is resolved

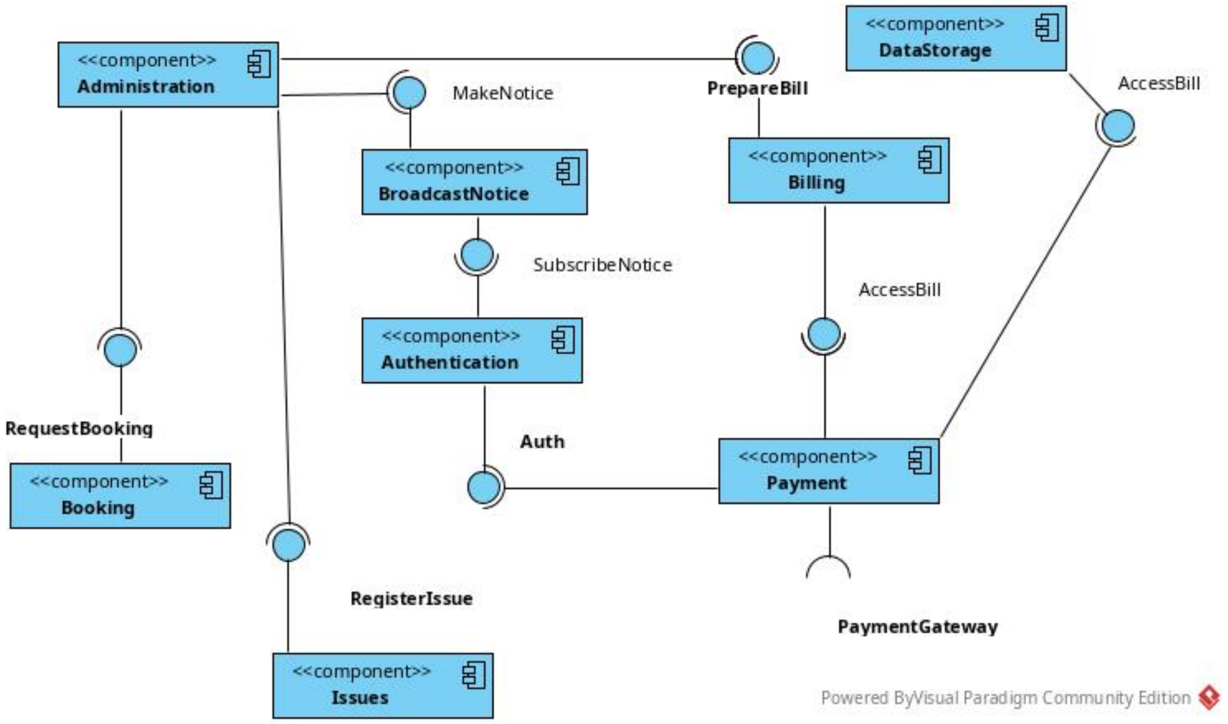
Activity



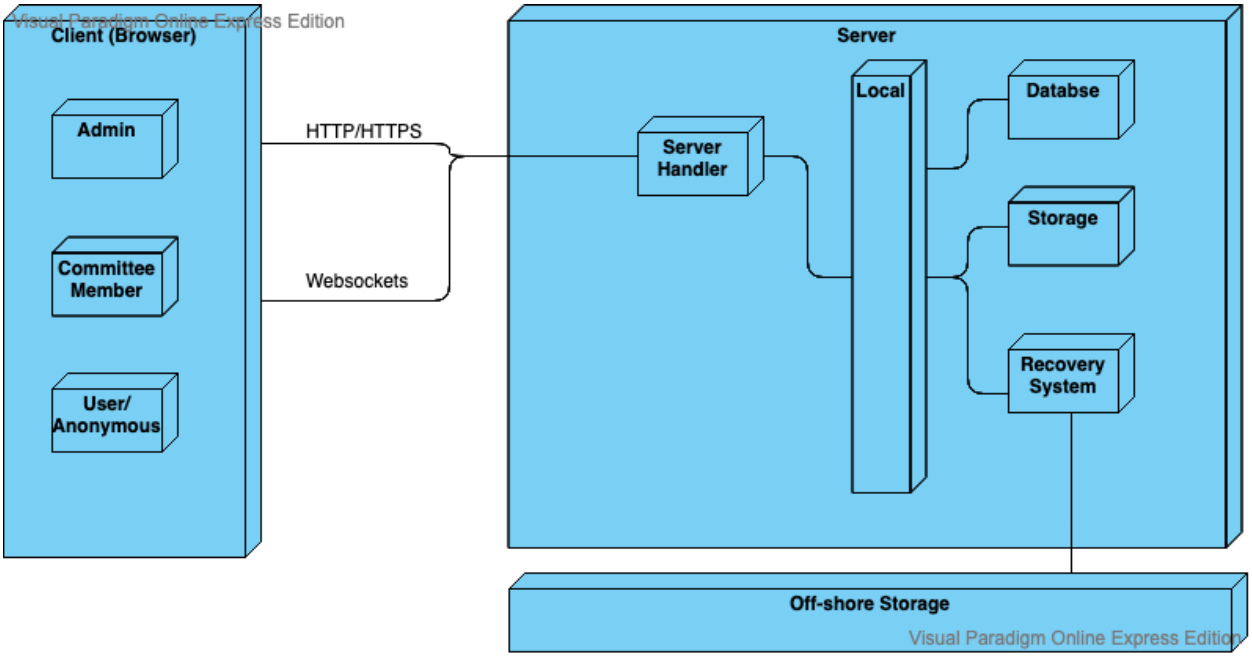
The given activity diagram is for facility request

- The user requests the facility
- The system receives the request
- If accepted:
  - Facility is marked as booked
  - Committee is notified
- User is notified of result
- Request is closed

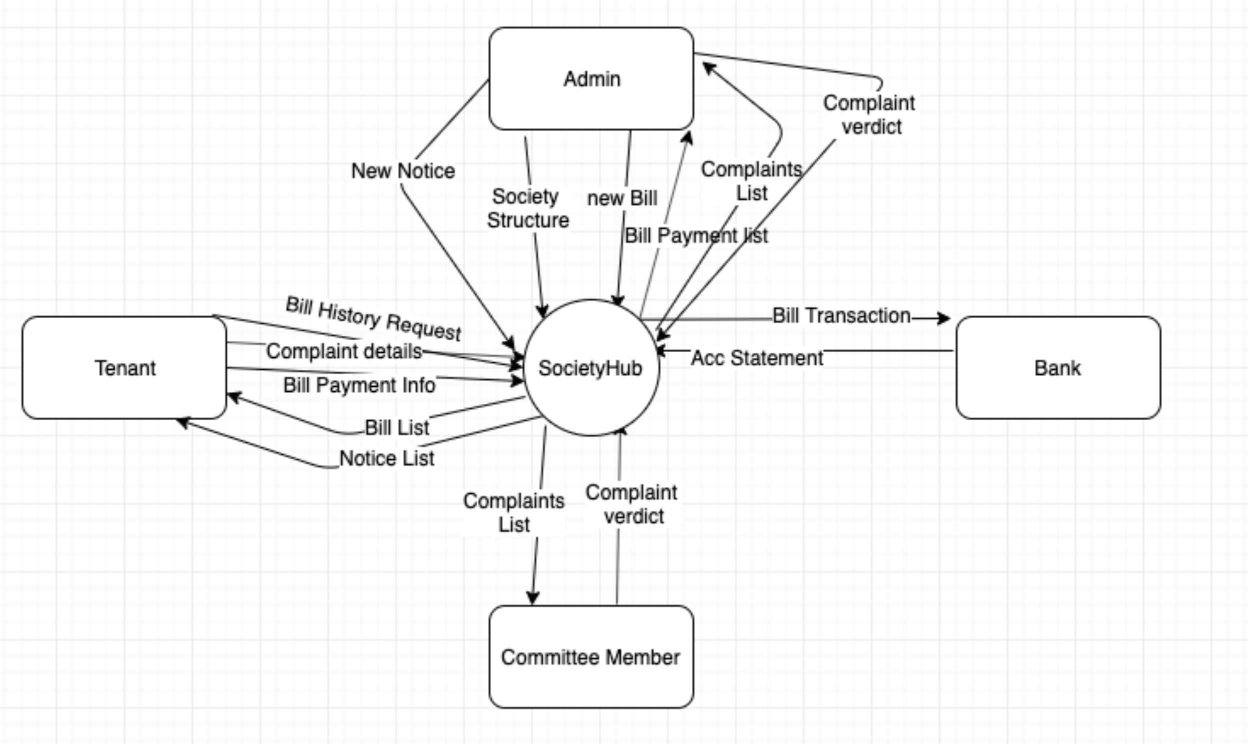
Component

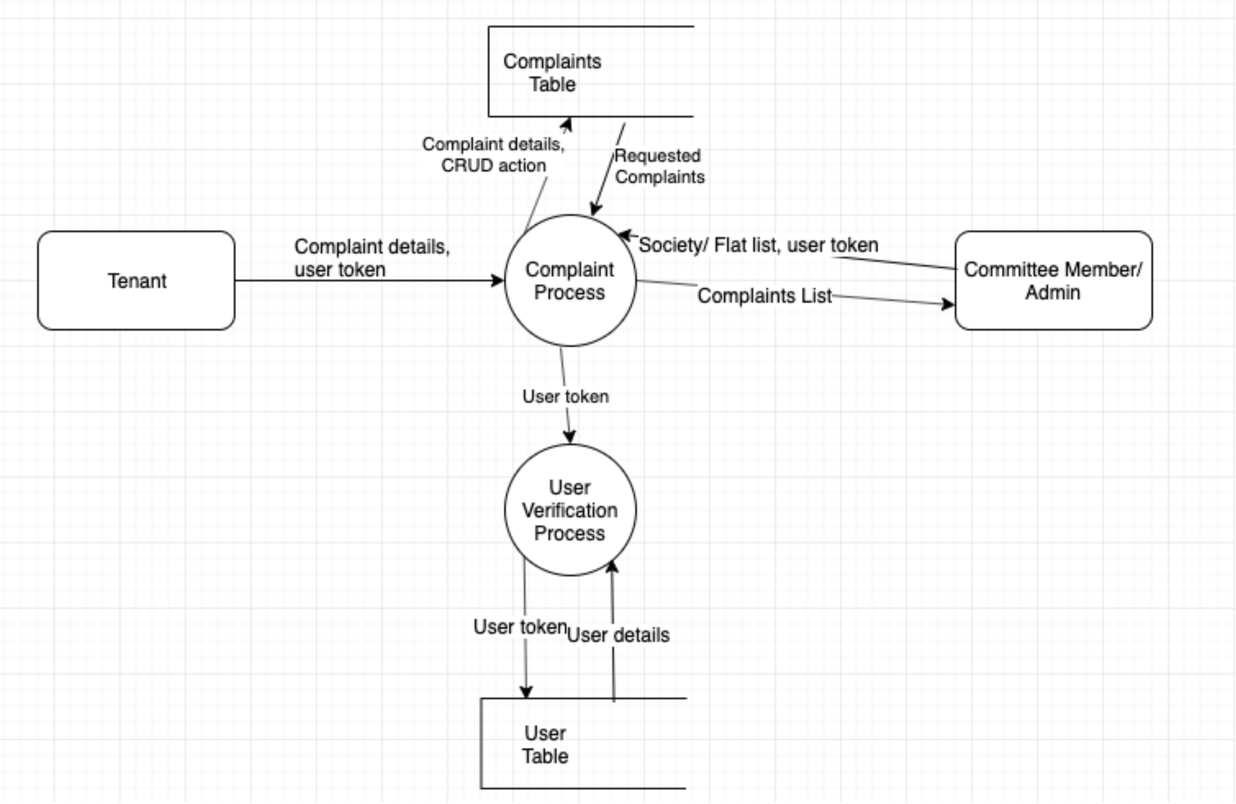


Deployment



DFDS:





CONCLUSION:

All interfaces are prepared for society management system

TUTORIAL 3

AIM:

To design logical view of Society Management System.

THEORY:

4+1 is a view model for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views". The four views of the model are logical, development, process and physical view. In addition selected use cases or scenarios are used to illustrate the architecture serving as the 'plus one' view.

ogical view: The logical view is concerned with the functionality that the system provides to end-users. UML diagrams are used to represent the logical view, and include class diagrams, and state diagrams

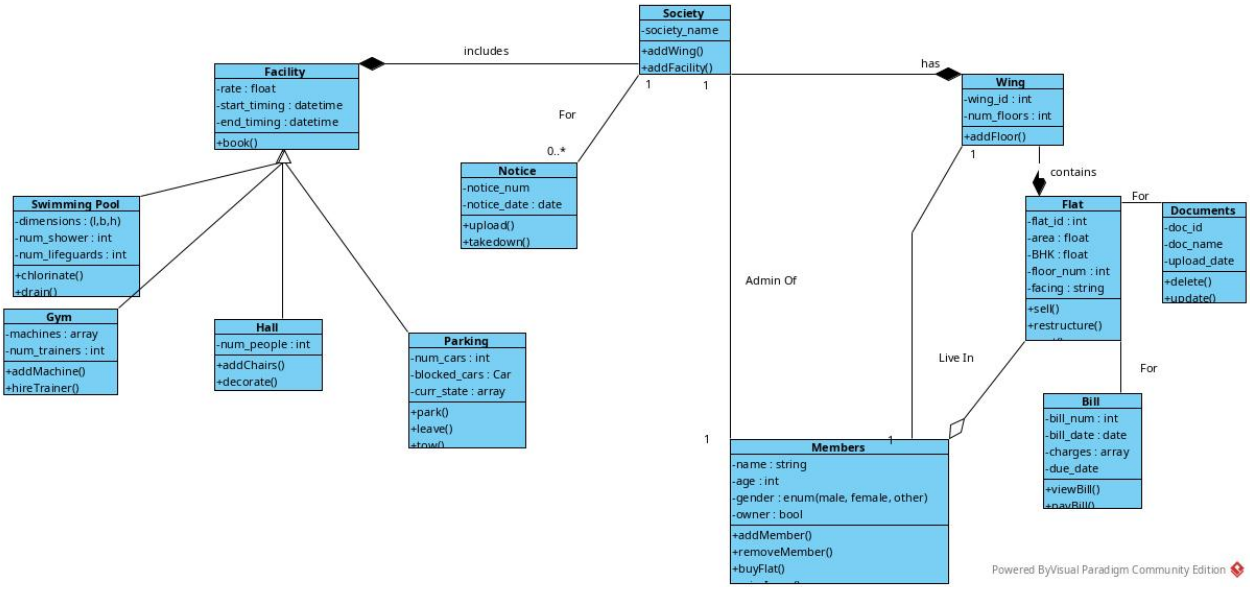
DIAGRAMS:

Class diagram:
A class diagram is the most basic static (or structural) view of the system. It represents all the types of objects (having common attributes and behaviors) as classes, and shows the relationships between these classes.

A class is represented as a rectangular box, divided into three parts. The first part contains the class name, with any constraints. The second part contains the attributes, and the third part contains the behaviors/methods.

Every attribute/behavior has a certain visibility. It may be private to the class (i.e., inaccessible directly to any other objects of any other class). It may be protected, i.e. private to the class except inheriting classes can also access it. It may be public, in which case any class can use it. The prefix notations are:

- private
+ public
# protected



Statechart diagram:
A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams −

1. To model the dynamic aspect of a system.
2. To model the life time of a reactive system.
3. To describe different states of an object during its life time.
4. Define a state machine to model the states of an object.

Statechart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

Statechart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a Statechart diagram we should clarify the following points −

1. Identify the important objects to be analyzed.
2. Identify the states.

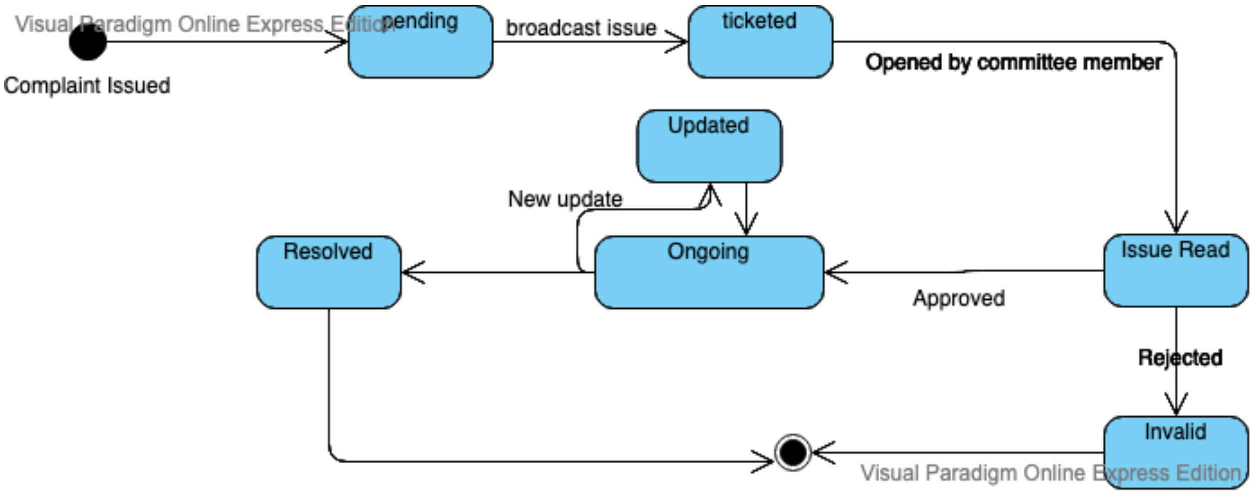
3. Identify the events.

Statechart diagrams are used to model the states and also the events operating on the system. When implementing a system, it is very important to clarify different states of an object during its life time and Statechart diagrams are used for this purpose. When these states and events are identified, they are used to model it and these models are used during the implementation of the system.

If we look into the practical implementation of Statechart diagram, then it is mainly used to analyze the object states influenced by events. This analysis is helpful to understand the system behavior during its execution.

The main usage can be described as −

- To model the object states of a system.
- To model the reactive system. Reactive system consists of reactive objects.
- To identify the events responsible for state changes.
- Forward and reverse engineering.



CONCLUSION:

Thus, logical view of society management system is made, with class diagram and statechar

TUTORIAL 4

AIM:

To create interfaces and deployment diagram for society management system

THEORY:

Deployment diagrams are used to describe the static deployment view of a system. It helps visualize the physical topology of the system, as the interconnection between physical nodes. The diagram consists of nodes and relationships between the nodes.

Deployment diagrams depict the hardware components of the system. In this, it is related to component diagrams, which similarly depict the software components of the system.

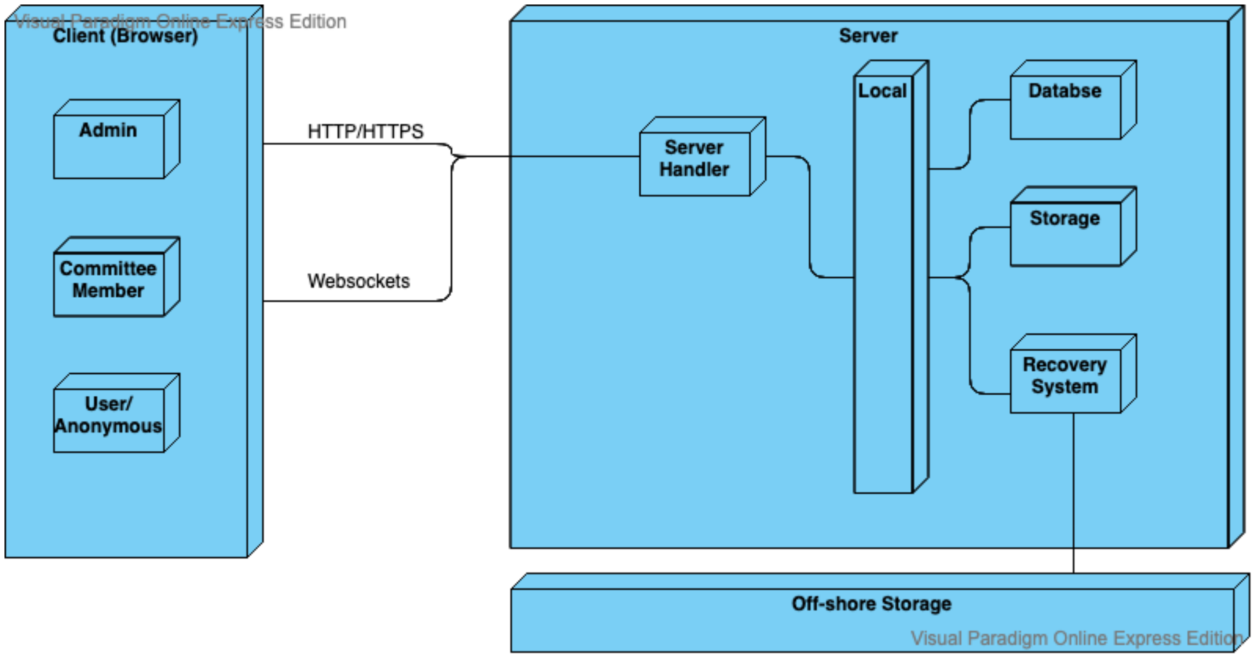
Notations are:

- Nodes: A node is a particular physical device, or type of device. It may be represented by a custom notation, or a generic cube, as shown below.

- There are two types of nodes: Device nodes, representing a device, and EENs (Execution Environment nodes), representing a system within a device.
- Databases are used to represent a common data store

- Relationships are similar to other diagrams:
    - Association
    - Aggregation
    - Composition
    - Dependency
    - Interface

DIAGRAM:



CONCLUSION:

Thus, the deployment diagram for society management system

TUTORIAL 5

AIM:

To design implementation , process and deployment views for the mini project . Also include sequence and component diagram

THEORY:

Interaction diagrams are used to show the interaction between various entities of the system. There are two types of interaction diagrams:

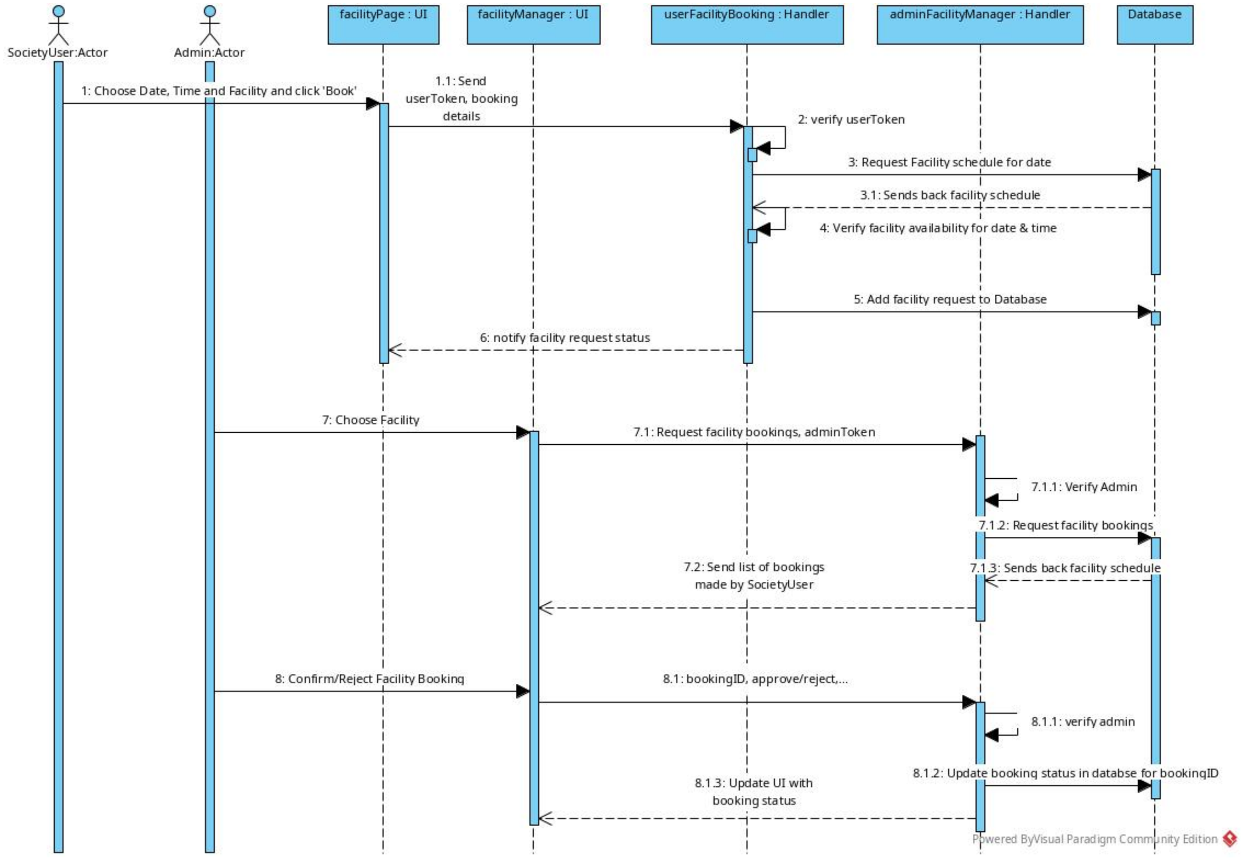
- Sequence diagrams
- Collaboration diagrams

A sequence diagram depicts interactions in a sequential order, based on time. The notations include:

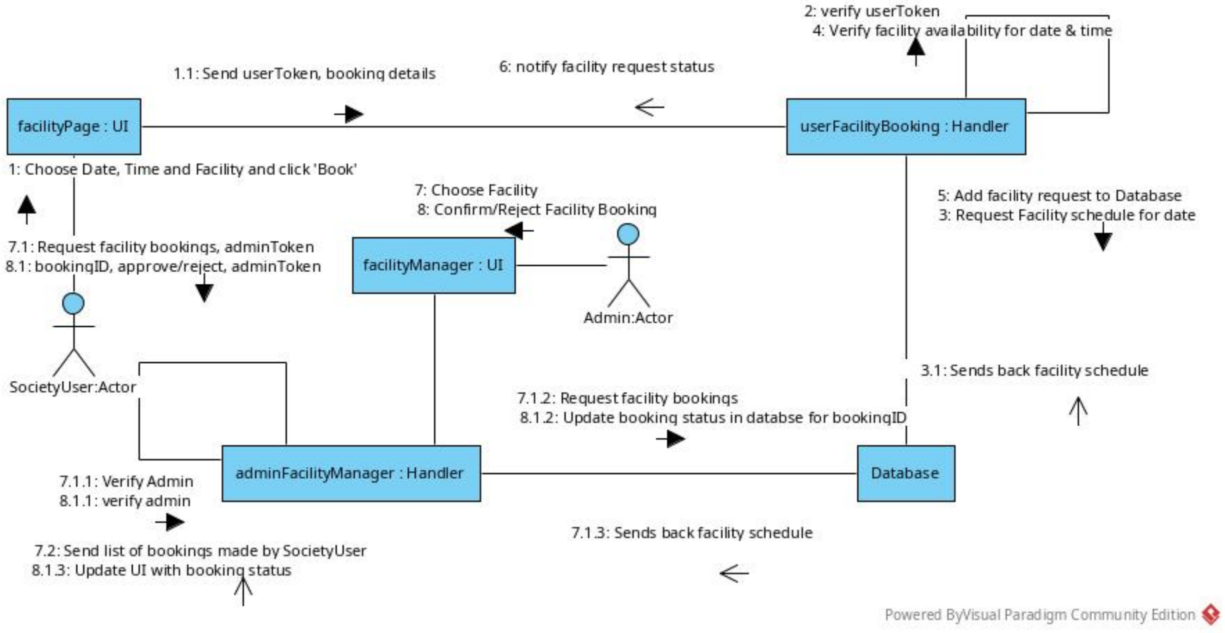
- Actor:
  Similar to use case diagram, it depicts a role.
- Lifeline:
  A lifeline depicts an individual participant in the diagram. It consists of a box with object name : class name. The box has a dashed line to show the lifetime of the object
- Messages:
  Communication happens with messages, whih are depicted as function calls over an arrow showing the direction of communication. There are different types of messages:
  - Synchronous messages: A synchronous message is represented with a filled arrow head. It depicts a message that requires a reply before any other messages.
  - Asynchronous messages: An asynchronous message is represented with an empty arrowhead. It does not require a reply.
  - Create message: It is used to create an object
  - Delete message: It is used to destroy an object
  - Self message: Is a loop back to the object itself
  - Found message: Depicted as message from a circle (unknown sender)
  - Lost message: Depicted as message to unknown recipient (circle)

- Loops and Conditions: These are depicted as frame elements (do/for loop, alt (if else), opt (if))

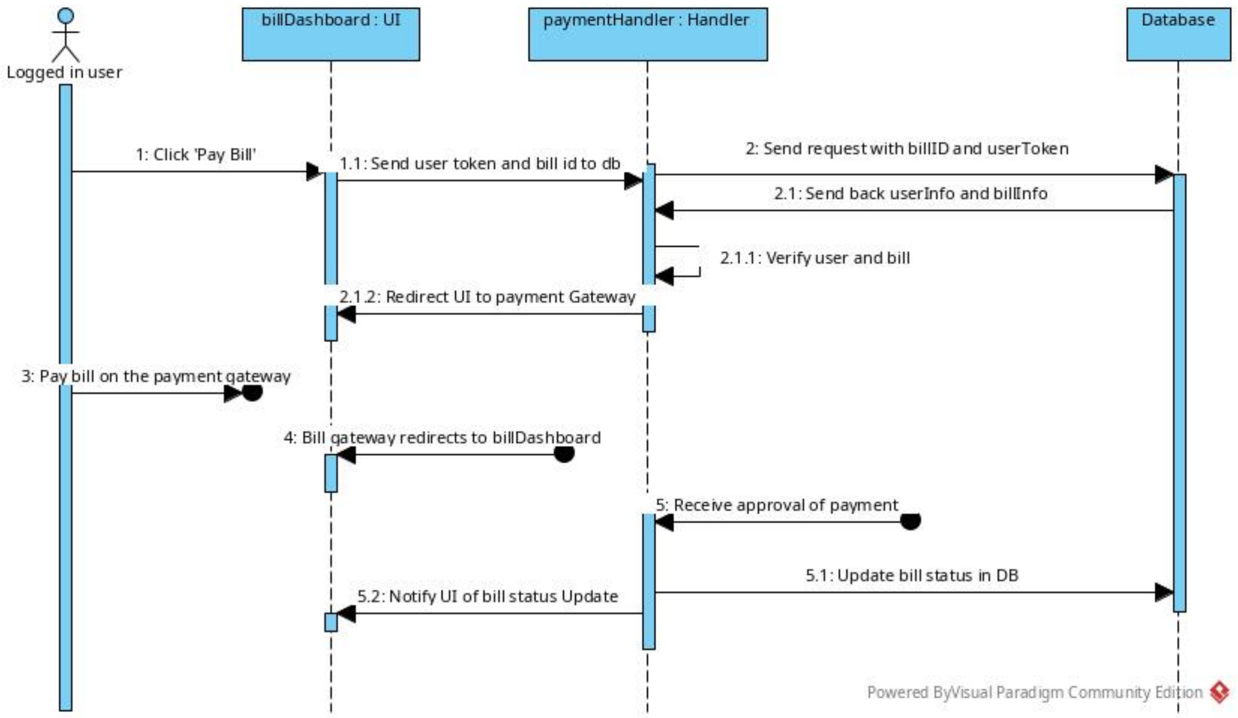
DIAGRAMS:


SocietyUser:Actor
Admin:Actor
facilityPage : UI
facilityManager : UI
userFacilityBooking : Handler
adminFacilityManager : Handler
Database
1: Choose Date, Time and Facility and click 'Book'
1.1: Send userToken, booking details
2: verify userToken
3: Request Facility schedule for date
3.1: Sends back facility schedule
4: Verify facility availability for date & time
5: Add facility request to Database
6: notify facility request status
7: Choose Facility
7.1: Request facility bookings, adminToken
7.1.1: Verify Admin
7.1.2: Request facility bookings
7.1.3: Sends back facility schedule
7.2: Send list of bookings made by SocietyUser
8: Confirm/Reject Facility Booking
8.1: bookingID, approve/reject,...
8.1.1: verify admin
8.1.2: Update booking status in databse for bookingID
8.1.3: Update UI with booking status
Powered By Visual Paradigm Community Edition

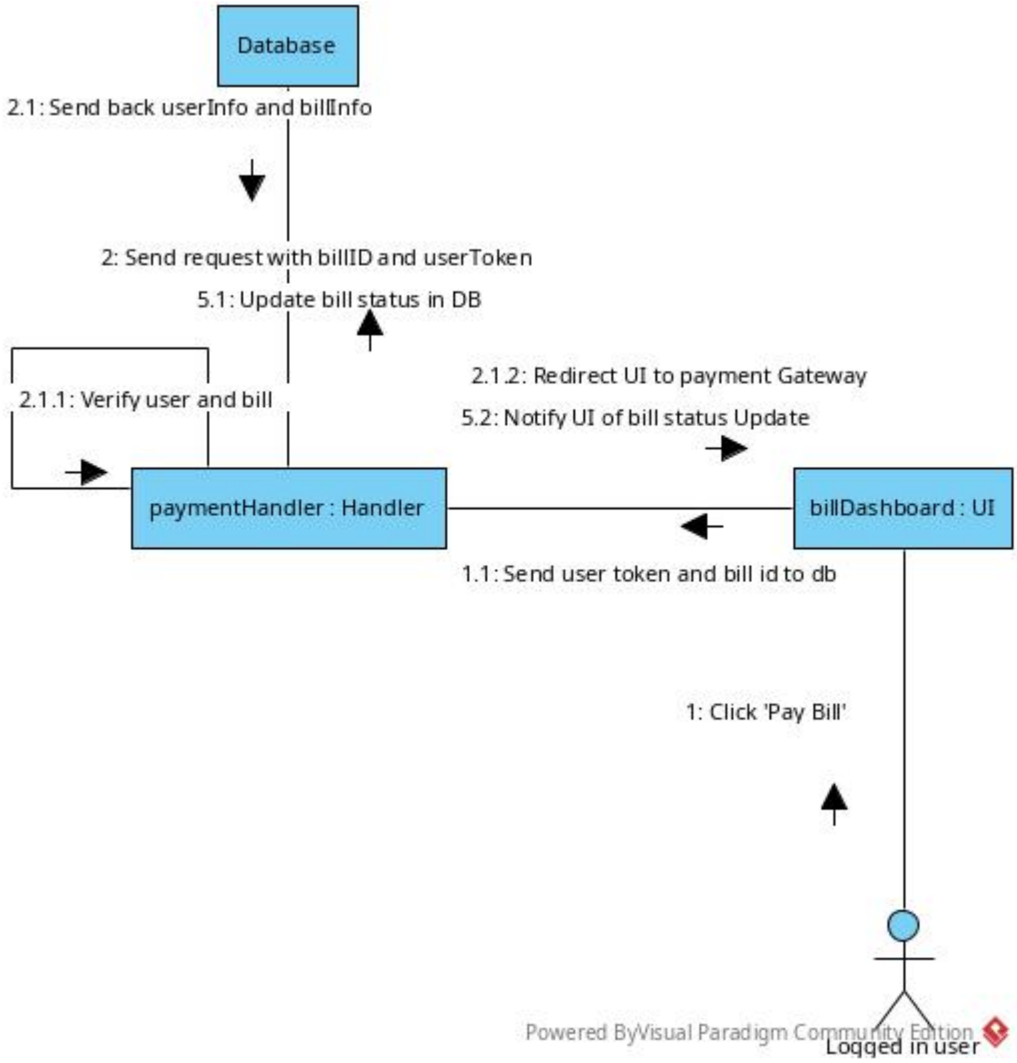
Facility Sequence


2: verify userToken
4: Verify facility availability for date & time
1.1: Send userToken, booking details
6: notify facility request status
facilityPage : UI
userFacilityBooking : Handler
1: Choose Date, Time and Facility and click 'Book'
7: Choose Facility
8: Confirm/Reject Facility Booking
5: Add facility request to Database
3: Request Facility schedule for date
7.1: Request facility bookings, adminToken
8.1: bookingID, approve/reject, adminToken
facilityManager : UI
Admin:Actor
SocietyUser:Actor
3.1: Sends back facility schedule
7.1.2: Request facility bookings
8.1.2: Update booking status in databse for bookingID
7.1.1: Verify Admin
8.1.1: verify admin
adminFacilityManager : Handler
Database
7.1.3: Sends back facility schedule
7.2: Send list of bookings made by SocietyUser
8.1.3: Update UI with booking status
Powered By Visual Paradigm Community Edition

Facility Communication


Logged in user
billDashboard : UI
paymentHandler : Handler
Database
1: Click 'Pay Bill'
1.1: Send user token and bill id to db
2: Send request with billID and userToken
2.1: Send back userInfo and billInfo
2.1.1: Verify user and bill
2.1.2: Redirect UI to payment Gateway
3: Pay bill on the payment gateway
4: Bill gateway redirects to billDashboard
5: Receive approval of payment
5.1: Update bill status in DB
5.2: Notify UI of bill status Update
Powered ByVisual Paradigm Community Edition

Bill Payment Sequence


Database
2.1: Send back userInfo and billInfo
2: Send request with billID and userToken
5.1: Update bill status in DB
2.1.1: Verify user and bill
2.1.2: Redirect UI to payment Gateway
5.2: Notify UI of bill status Update
paymentHandler : Handler
billDashboard : UI
1.1: Send user token and bill id to db
1: Click 'Pay Bill'
Logged in user
Powered ByVisual Paradigm Community Edition

Bill Payment Communication

CONCLUSION:

Thus, implementation , process and deployment views for the mini project were designed .

TUTORIAL 6

AIM:

To provide implementation of Society Management System along with user Interface

THEORY:

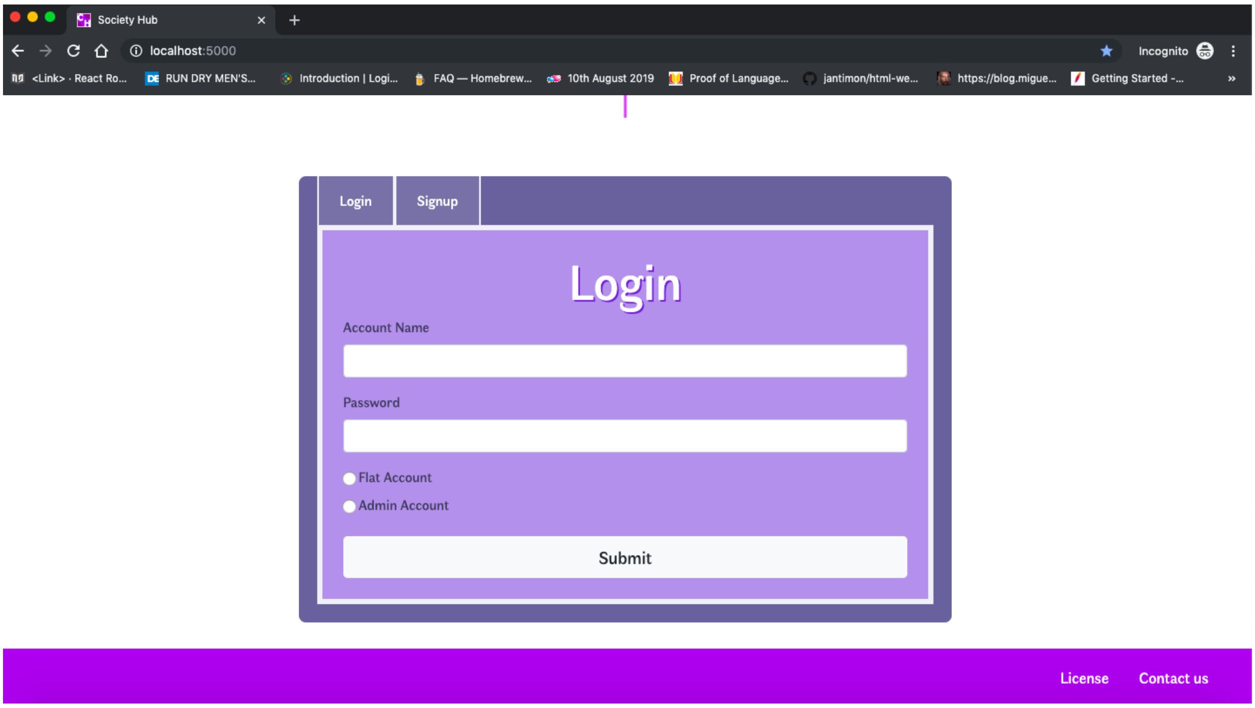
Society Hub is a web platform for housing societies to delegate a number of management responsibilities to. It aids in easy storage and retrieval of information related to various parts of the society, along with an account system for added security.

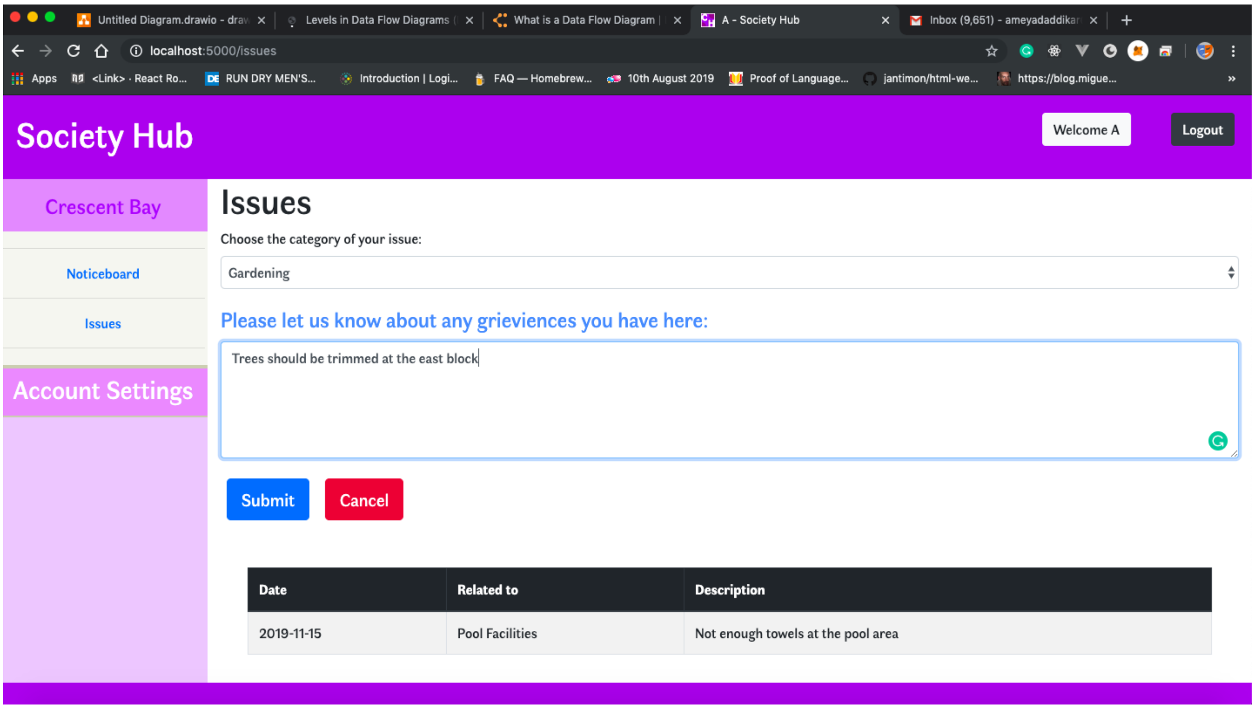
The housing societies in the country today are getting larger and more comfortable, as well as more cooperative and community-based. There are large numbers of facilities, and, as with any system, a large number of complaints related to various issues. To aid in the management of this vast community, we have developed the Society Hub. The Society Hub has an account-based system, which allows one to see details of the society, issues, facilities, bills, etc. Once verified, the user gets easy access to this system. Admin facilities are also provided to update the data as per real-world changes.

As Society Hub is a web platform, it uses database connectivity along with web layout and scripting technologies. These are listed below:

1. MySQL for the database server
2. Python for the back-end scripting
3. Flask framework for translating the python scripts to webpage layouts, and to provide database connectivity.
4. Bootstrap for styling
5. HTML5 for the main layout
6. Javascript with Ajax for the front-end scripting

USER INTERFACE:





Issues

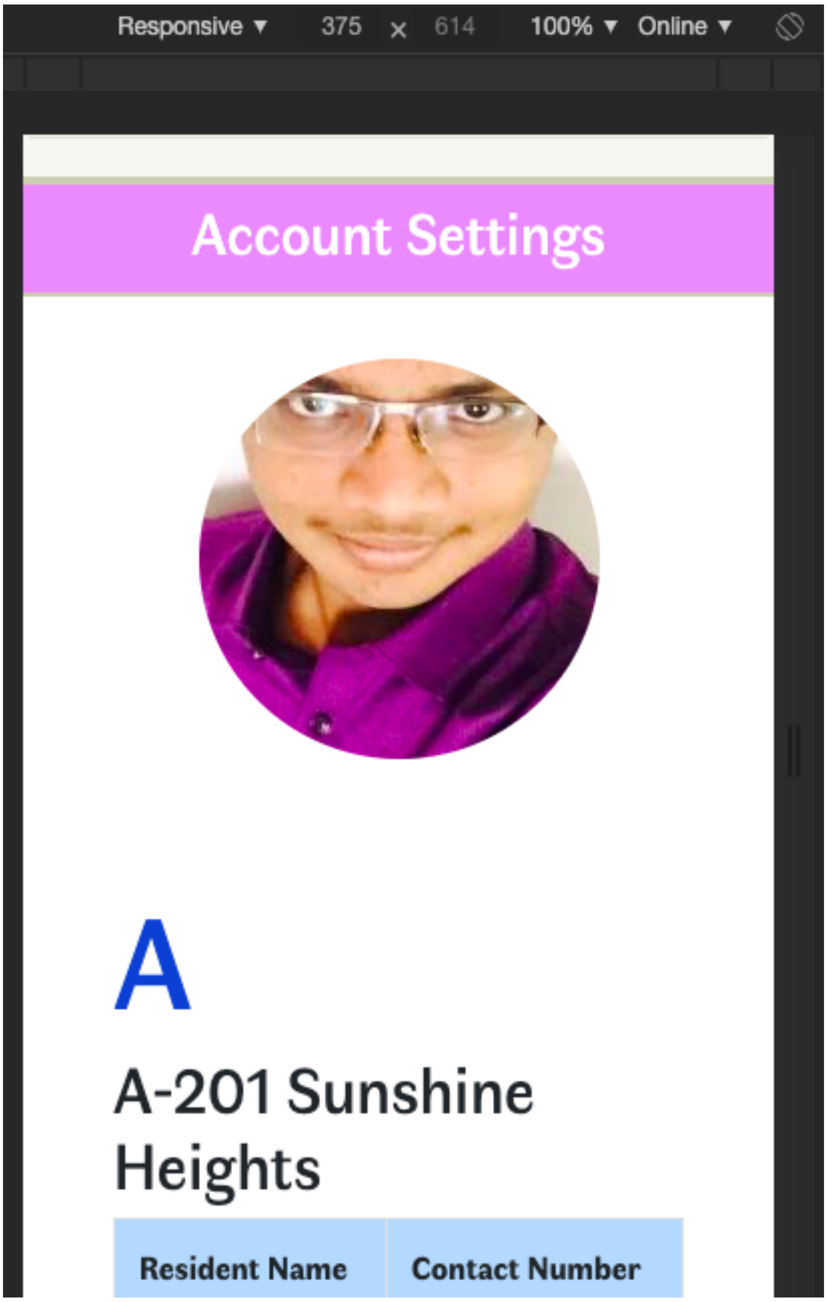
Choose the category of your issue:

Gardening

Please let us know about any grieviences you have here:

Trees should be trimmed at the east block

Date	Related to	Description
2019-11-15	Pool Facilities	Not enough towels at the pool area

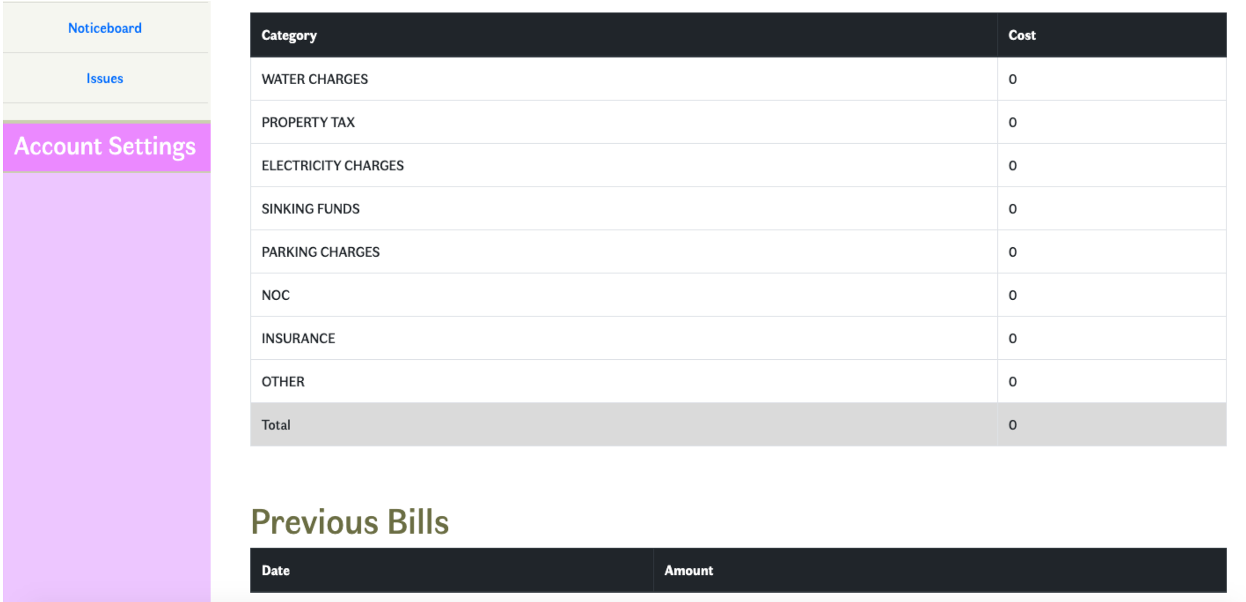


Account Settings

A

A-201 Sunshine Heights

Resident Name	Contact Number



Category	Cost
WATER CHARGES	0
PROPERTY TAX	0
ELECTRICITY CHARGES	0
SINKING FUNDS	0
PARKING CHARGES	0
NOC	0
INSURANCE	0
OTHER	0
Total	0

Previous Bills

Date	Amount

CONCLUSION:

Thus, the Society Management System is implemented with Flask, Bootstrap, JQuery