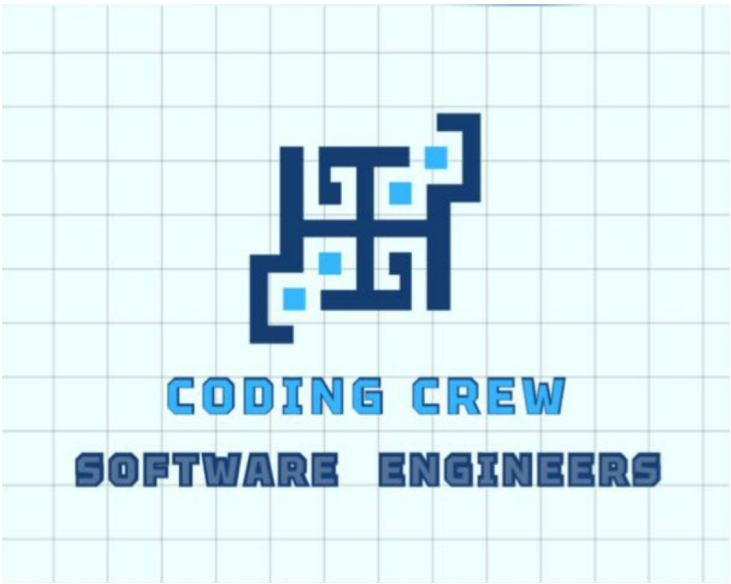
Computer Science
Software Engineering

PHASE 2:

Software Design Specification Document



Done by:

Asil Albahnasawi U21105894

Mustafa Sami U21100613

Rena Housain U21100044

Renad Faisal Ababneh U21104263

Salam Naser U21104357

Sulaf Ali Hallak U21107541

Yasmin Rajai Alshamali U21106201

1.0 Introduction

This part offers an introductory to our project, emphasizing how it works and the job of each actor and the software.

1.1 Goals and objectives

The main aim of the project is to create an application that helps people save energy from their solar panels and keep track of their energy usage. It's designed for a neighbourhood where neighbours can work together to save energy and share any extra energy they have. Additionally, the app makes it easy for residents to request inspections for their solar panels with just a button click.

According to the goals of software development, it is essential that project requirements be considered fully understood and that the application is created in a way that meets the expectations of the user. In order to make the software accurate, it is developed in a variety of scheduled stages with frequent client consultations. This paper includes many UML diagrams and different scenarios to provide users with an understanding of the software's functionality.

The main goal of “Soloro” is to conserve wasted energy; the term is derived from the word solar. We considered working it out by having neighbours trade additional energy from solar panels. Along with saving energy, this enhances the neighbourhood environment and encourages connections between people. Additionally, we developed a rewards system where if you conserve energy, you receive a certain amount of points that you can use to obtain rewards later on in order to persuade and excite people to use our application.

Residents can use the app to manage and edit their solar panels, follow up with their energy usage.

Residents can upgrade or downgrade their solar panels by submitting a request to the government employee.

Residents can buy and sell solar panel energy with their neighbours that also use the application.

Government employees can help residents fix or inspect their solar panels easily through the application.

1.2 Statement of scope:

The scope of “Soloro” is to create an easy-to-use database for solar panels. That will store information about solar panel Installation, how much energy it consumes, maintenance, and transactions like buying and selling solar energy. Different users like residents, government employees, and neighbours can access the data they need. This database will make managing solar panel info and transactions straightforward for everyone involved.

The application should be installed on the resident’s phones (apple or android) when they decide to buy a solar panel. When the resident opens the application, he or she is asked to sign up or login by the system. If they need to sign up, they are required to input their personal details such as, full name, email address, phone number, password, and date of birth. If they need to log in, they are required to type in their full name and password. However, for the government employee it’s the same procedure but since they have access to the requests of users and the awareness page where he can add or edit information being presented to the residents. The government employee should insert a special pin given to him when he first signs up, whenever they want to login. Each of the residents and the government employee has an access level.

The government employee has a higher access level than the resident as he or she is allowed to view resident's information, inspect their solar panels, check their transactions, and edit the awareness page. The employee is allowed access to such information in case an error occurs, or the system crashes. However, to be allowed to access such sensitive details the employee must enter a special pin beforehand. That way, the residents' information is in a safe place.

The residents have a lower access level than the government employee since they can only access their information like viewing account details, checking transactions history, viewing your solar panel type and specifications, in addition to viewing your payment instalments and number of points you gained throughout the whole journey of saving energy. You can also view your neighbours' details if they're planning on buying more energy from others in the neighbourhood or selling saved energy, or to chat with them about deals.

1.3 Software context

As the number of solar panels users in our community continues to grow, the need for effective management of all the related data becomes increasingly important. It's become a bit like keeping track of a bustling neighbourhood, where everyone has their own solar panels, expenses, and transactions. This is where our special software comes into play. It's like the friendly neighbourhood organizer for all things solar. This software isn't just for experts; it's designed with residents, government employees, and anyone involved in solar panels in mind. It's your digital assistant for managing solar panel information. You can effortlessly monitor expenses, record transactions, and generate reports whenever you want to check in on your solar panel system's status. It's easy to use.

No need for complicated spreadsheets or stacks of paperwork. With a few clicks, you can update your solar panel records, ensuring they're always accurate and up to date. Plus, it helps prevent errors, so you can have peace of mind knowing your solar panel data is in good hands.

1.4 Major constraints

The primary limitations when developing software for managing solar panels include ensuring that the system operates within the specified project constraints. The software's design considers how different users will interact with it. It's essential that the software is quick, user-friendly, and optimized for maximum efficiency. Moreover, the device on which the software or application is installed should meet the necessary hardware requirements to execute all tasks within a reasonable timeframe.

2.0 Data Design:

This section outlines the internal, global, and temporary data structures within the solar panels app.

2.1 Internal Software data Structure:

Upon pressing the add or update button as the user, the information entered in the respective text fields is directly transmitted to the database for addition or update.

2.2 Global Data Structure

The primary global data available system-wide is stored in the database. The database serves as an organized collection of related records. You can refer to Section 2.4 for detailed information about the structure and content of the database.

2.3 Temporary Data Structures

“Soloro” operates without the need for temporary data structures, since the data recorded in the system should be there all the time. Unless a user deletes their account, or an employee resigns.

2.4 Database Description

Outlined below are the tables, their corresponding attributes, and brief descriptions for the solar panels app.

Table Name: **Employee**

Attribute	Data Type	Constraint Type	Optional
Name	String(60)		
ID	String(15)	Primary Key	
Password	String(30)		
PIN	Long(20)	Unique Key	
EnrolmentDate	Date		yes
Role	String(10)		
BankNumber	Long(12)		

Employee is a base class for two child classes, admin and handyworker. It contains the common attributes and methods between both inherited classes.

Table Name: **Payment**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Foreign Key	
Amount	Double(15)		
Duration	int(6)		
Installment	Double(15)		
AmountRemaining	Double(15)		
Paid	Bool ()		
CardNumber	Long(12)		
CRN	Int(6)	Unique Key	
Choice	String(10)		yes
TransactionDate	Date		

Payment holds all the information related to any money entering or exiting the system, it also holds card information for residents. Payment class also ensures all payment is done on time, in an organized matter, and safely. It performs all the required operations to ensure that.

Table Name: **Resident**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Primary Key	
Name	String(60)		
Address	Location	Unique Key	
MobileNum	String(10)		
Email	String(50)		
FamilyMemberNum	Int(2)		yes
CreditCard	Payment		
Gender	Char(1)	Check Constraint values('M','F') for(Male, Female)	yes

Payment holds all the details related to a resident, and all the actions that he can perform to interact with the application, as well as other residents enrolled.

Table Name: **Account**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Primary Key	
Password	String(30)		
Balance	Double(10)		
Status	String(8)	Check Constraint Values("Active","InActive")	
TransactionDate	Date		

Residents, admins, and handy workers all have accounts. The account will allow the user to login and perform the required tasks from him. The account is the gateway for the user to access the application. Noting that each account type has different access levels.

Table Name: **System Management**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Primary Key	
NumOfResidents	Int(5)		
NumOfEmployees	Int(5)		

It contains everything that the system does such as save the id of the users and related data like number of employees and residents. In addition to some functions related to buying, signing in, choosing date and location. All these functions are connected, where the system does all things needed for the resident, admin, supplier, and handyworker.

Table Name: **House**

Attribute	Data Type	Constraint Type	Optional
Address	Location ()	Primary Key	
RoofDetails	String(30)		
Status	String(8)	Check Constraint Values(“Active”,”InActive”)	
Panel	Panel()		

Class house inherits “building” and “Villa”, these classes use the information in the main class house. They hold the details related to the solar panel installed in the house whether it’s a building or a Villa. In addition to updating any details and showing the resident any information related to their energy levels and consumption.

Table Name: **Solar Panel**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Foreign Key	
Energy	Double(3)		
Type	String(10)		yes
Length	Double(5)		
Width	Double(5)		
Quantity	Int(3)		yes
InstallationDate	Date()		
SerialNumber	Long(7)		

It holds information related to the solar panel in all aspects such as length, width, type, and things like that. It also allows the resident to request any action related to their solar panels like modifying it or updating the energy level. In addition to viewing any details about the energy in the solar panel.

Table Name: **Locations**

Attribute	Data Type	Constraint Type	Optional
Address	Location()	Foreign Key	
xCoordinate	Int(5)		
yCoordinate	Int(5)		
Nieghborhood	String(25)	Unique Key	

It holds information related to the resident's houses and the exact location using x and y coordinates. It also allows the resident to view details about their neighbors.

Table Name: **Map**

Attribute	Data Type	Constraint Type	Optional
Adress	Location()	Foreign Key	
NumofHouses	Int(5)		
House	House()	Unique Key	
Record	String(30)		

Map class contains all the details about the houses and number of houses available in a specific area, with their records and if any modifications are needed with money, payment, or solar panels generally. It also displays a map where residents can chat or visit other neighbor's houses to see their status if they're buying or selling energy.

Table Name: **Awareness**

Attribute	Data Type	Constraint Type	Optional
Statistics	String(50)		
Facts	String(100)		
Advantages	String(100)		
Rating	Double(2)		yes

It holds all the information about the awareness page in our application, where the resident can visit this page to view statistics and facts in addition to adding their own review about what they just read. However, if you're an admin you can edit the information that the resident reads through the page.

Table Name: **Reward**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Foreign Key	
NumOfPoints	Int(5)		
EnergySaved	Double(5)		

It holds the information related to points and collecting them, it also allows you as the resident to convert your energy saved into points where you can redeem rewards when your points are at a certain amount.

Table Name: **Supplier**

Attribute	Data Type	Constraint Type	Optional
ID	String(15)	Primary Key	
CompanyName	String(20)	Unique Key	
CompanyNumber	Long(7)		

The Supplier class refers to partners responsible for providing solar panels. These suppliers are associated with a company name and contact number. Their primary role is to furnish our company with solar panels available for purchase by residents registered in the app. Additionally, they can supply specific components if a solar panel is defective or has errors. The confirm function involves verifying and confirming the transaction or deal with our company.

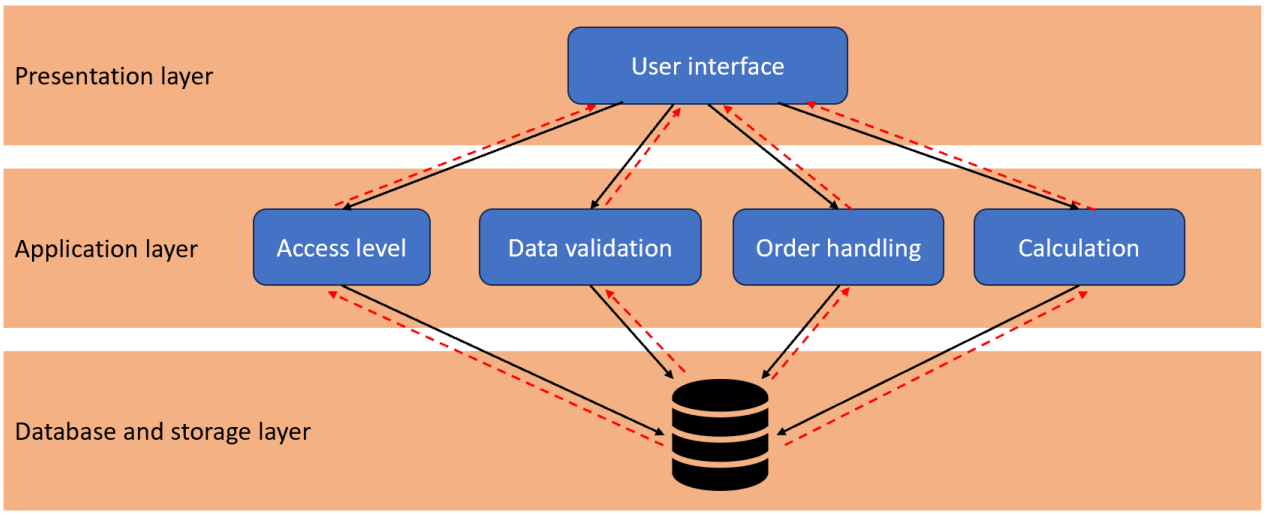
3.0 Architectural and Component-Level Design

Suitable architecture style: Layered architecture style

Reasoning: layered architecture allows a clear separation of layers, so that each layer has a distinct functionality, which will make it easier for developers to code. Additionally, layered architecture is more adaptable to change, using it makes maintaining and changing each layer individually easier, without any need to alter other layers, this is very helpful since the software engineering field has lots of changes, and continuously evolving must be part of our system.

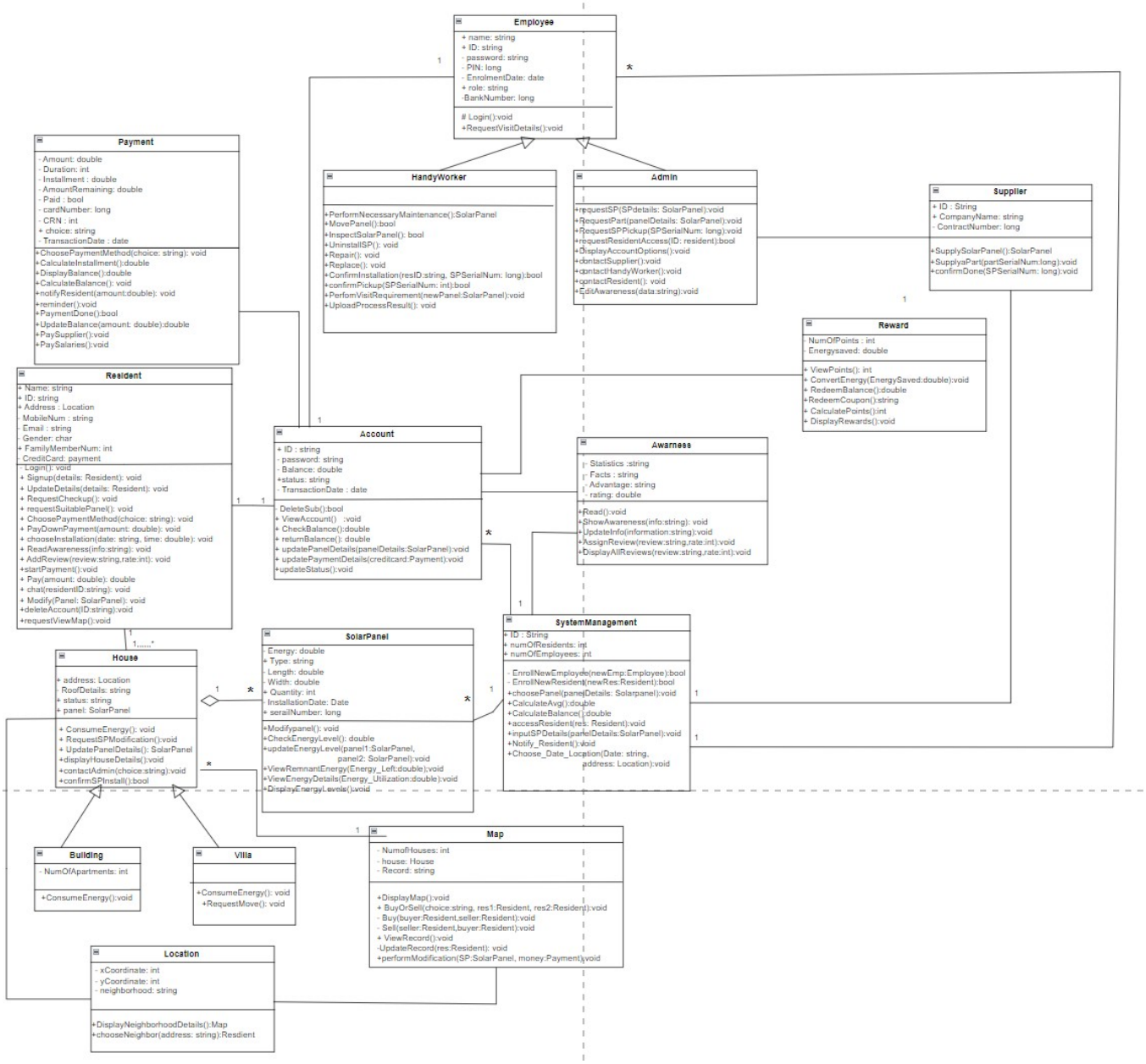
Our layered architecture will have 3 layers: **presentation layer**, **application layer**, **Database and storage layer**.

1. Presentation layer: this is the layer that will have the user interface and will handle all user interactions. It will handle all user inputs and pass it down to the application layer, as well as request data to display to the user from the application layer.
2. Application layer: will contain the main application functions. Some of these functions can be: handling levels of access between resident and employee, validating any details entered, handling orders, performing any needed calculations, handling buying and selling energy.
3. Database and storage Layer: will store all data generated from the application, including user profiles, employee profiles, solar panel performance details, locations, records, contracts, interactions, transactions, and many more. The database layer will also be responsible for data recovery if needed.



3.1 System Structure

3.1.1 Architecture diagram



3.2 Description for Components

3.2 Description for Component Payment

3.2.1 Processing narrative (PSPEC) for component Payment

The component payment consists of the class payment. It contains the attributes, which are the data needed to complete any payment procedure in the application, paired with the methods needed to perform changes and checks needed along the process. Payment object is used whenever any payment is made by the resident, or money transfer to the resident, this includes buying a solar panel, buying or selling energy, or maintenance fees. Another role of the Payment object is paying employees and suppliers. The responsibility of this component is to keep the balance up to date with all transactions, and to ensure all payment is made on time.

3.2.2 Component Payment interface description.

Payment interface will be triggered with every money transaction needed. Payment interface first starts by requesting the credit card details and authenticating it. Next interface would be anytime a resident need to perform a payment, where the payment interface will display the amount needed to pay, and resident will proceed with the payment. resident can also check his balance to see how much money he still must pay. Also, whenever a payment is required from the resident, a pop up message will be seen on the screen of the application, displaying the due date of the payment, and its amount.

3.2.3 Component Payment processing detail

Start

New resident enrolled

Payment requests credit card details

Payment displays payment choices for resident

Based on choice, payment plan will be calculated

Payment process will start by paying down payment

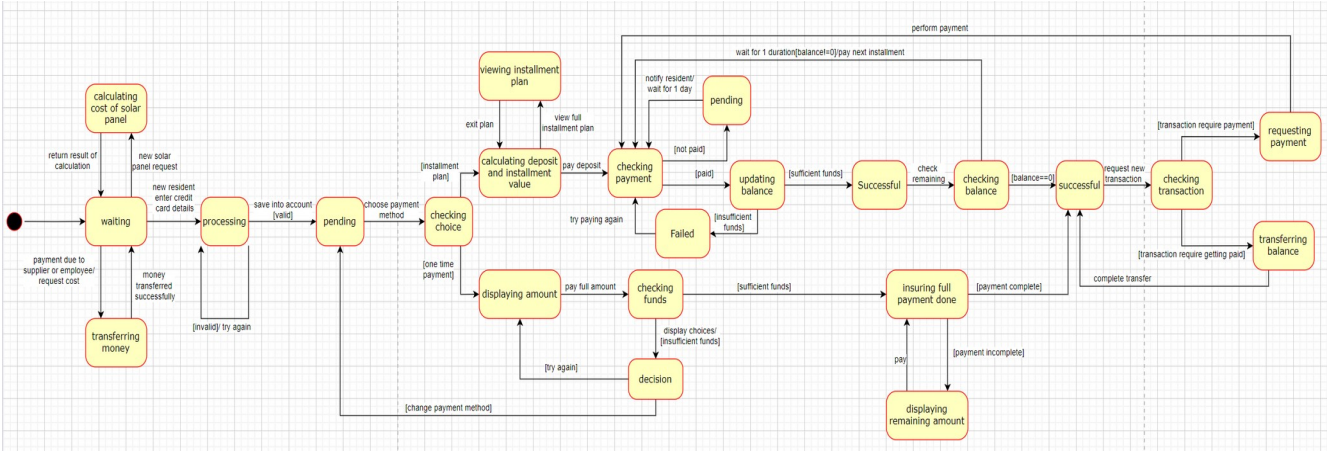
Payment will continuously check resident balance to check if he still has payment needed

Resident will continuously be notified until full payment is complete

Payment is successful as long as balance is 0

Once new transaction occur, payment process will be required from resident again

3.2.3.1 Design Class hierarchy for component Payment



When a resident is enrolled to the application, and he is required to start paying, he will first be required to enter his credit card details, and the payment class will authenticate the details. When paying for the solar panel is required from resident, he will choose a suitable payment method for him. Based on that choice, payment plan will be made by the payment class. Resident will start paying, and for each payment, resident will be notified, and the payment made must be checked to ensure its valid. Finally, when payment is successful, state will be waiting for a new transaction.

3.2.3.2 Restrictions/limitations for component Payment

If resident didn't pay for a long time, no action will be done except continuously notifying.

3.2.3.3 Performance issues for component Payment

Not Available

3.2.3.4 Design constraints for component Payment

Ensuring compatibility with payment gateways used.

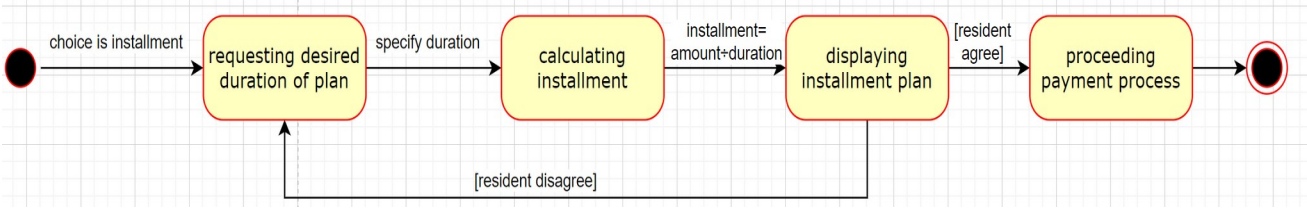
Security: insuring that the credit card details of our residents are completely secure, and that the external services used are reliable and trustworthy, to gain the trust of our customers

3.2.3.5 Processing detail for each operation of component Payment

The class Payment has a function called CalculateInstallment(), UpdateBalance(amount), notifyResident(),PaySupplier(),PaySalaries().

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. CalculateInstallment() is invoked inside another method which is ChoosePaymentMethod(), when the resident chooses instalment plan as a payment method. It requests the duration the resident want to pay over, and based on that calculates how much each instalment will be. Based on the displayed result, the resident gets to accept or deny, and in case of deny restart the process.



2. UpdateBalance(amount) is used whenever a payment is made, where we will deduct the amount from the current balance (Balance=Balance-amount)

3. notifyResident() is invoked whenever a payment is needed, so the payment class can inform the resident its time for a payment , and how much the payment amount is.

4. PaySupplier() is used whenever a part, or a full solar panel system is requested from the supplier, so that the supplier can generate needed part. Payment class must ensure safe transfer and correct amount.

5. PaySalaries() is invoked every end of the month, to pay the salaries for all employees

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. CalculateInstallment():
Start
Payment requests the duration the resident want to pay over
Calculations are made based on duration and amount.
Resident agree or reject on the instalment plan
End

2. UpdateBalance()

Start

Calculate new balance

Update balance

End

3. notifyResident()

Start

When payment is due, send notification to resident

End

4. PaySupplier()

Start

Resident needs hardware

Request sent to supplier

Supplier returns price

Payment class will transfer the cost to the supplier

End

5. PaySalaries()

Start

When end of month arrive, transfer all salaries

End

3.2 Description for Component HandyWorker

3.2.1 Processing narrative (PSPEC) for component HandyWorker

The component HandyWorker is a child class of the class Employee. It inherits attributes from class employee. These attributes are general details about the handyman. The methods describes the roles and responsibilities of the handyworker in the system. HandyWorker object is used whenever any physical role is needed, including operations on solar panels or pickups.

3.2.2 Component HandyWorker interface description.

Whenever a service from a handy worker is needed from resident, the details about the handyworker are displayed to the resident, for the resident safety. Also whenever handyworker requests suitable date to visit resident, this will also be displayed for the resident.

3.2.3 Component HandyWorker processing detail

Start

HandyWorker is hired

Handyworker login and start accepting requests

Based on received request, handy worker will contact other parties to perform task

Handyworker will notify regarding completion

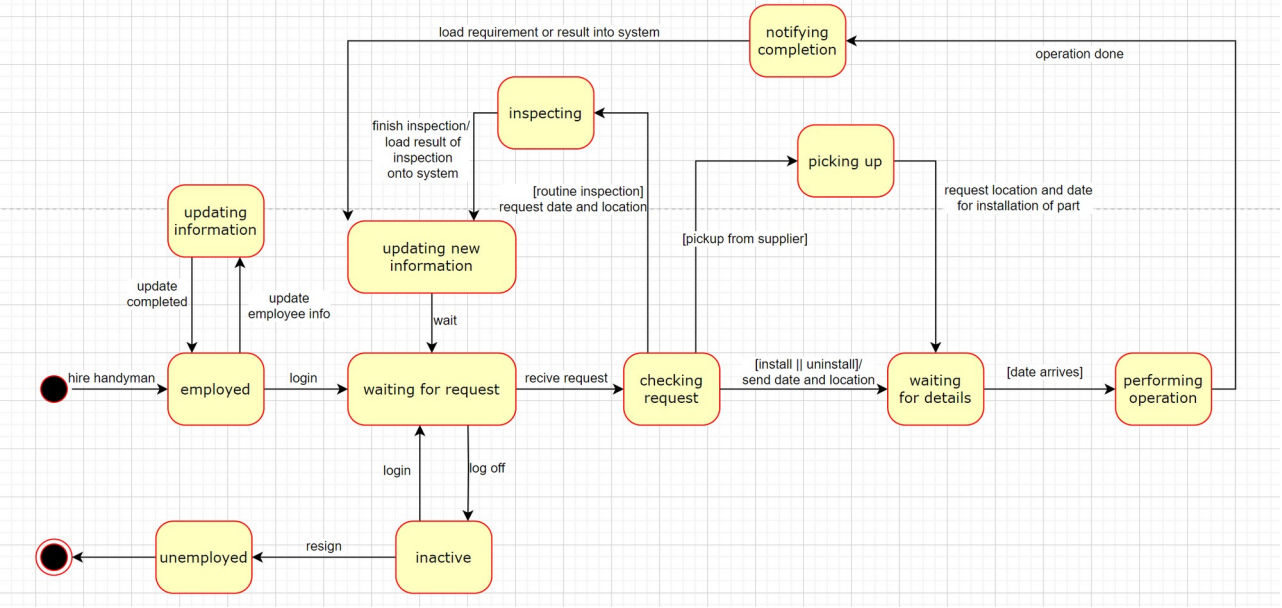
Handyworker returns to waiting state

Handy worker logs off

Handyworker resigns

End

3.2.3.1 Design Class hierarchy for component HandyWorker



When a HandyWorker is employed, his details are updated. Now, he is ready to accept requests. Requests for a handy worker are picking up, installing, uninstalling, and inspecting solar panels. Handyworker will request needed details that come with every task, and perform needed operations. After finishing task, HandyWorker will update the system with the results, then goes back to accepting requests. Handy worker can log off and resign when needed.

3.2.3.2 Restrictions/limitations for component HandyWorker

Handyworker can only perform one task at a time.

3.2.3.3 Performance issues for component HandyWorker

Handyworker can only perform one task at a time.

3.2.3.4 Design constraints for component HandyWorker

HandyWorker roles and access constrains.

3.2.3.5 Processing detail for each operation of component HandyWorker

The class HandyMan has a function called UploadProcessResult(), RequestVisitDetails().

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. UploadProcessResult(): after handy worker finish any process, handy worker must upload the result of the process into the system, so that full system is aware of result, and changes are done accordingly.
2. RequestVisitDetails(): this method allows the handyman to undirectly contact the resident, to request details regarding the visit, including date and time of visit, and location of resident.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. UploadProcessResult():

Start

Handy worker complete operation

Handy worker enter results from operation

Based on the information from handyman, new process may be invoked

End

2. RequestVisitDetails():

Start

Handy worker receive a request

Handy worker contact resident

Resident choose details suitable for him

Handy worker perform needed operation

End

3.2 Description for Component Map

3.2.1 Processing narrative (PSPEC) for component Map

The component Map has the attributes needed to display to the user regarding the map details, paired with the methods that resemble the actions the user can do using the map. Map object is accessed whenever the resident requires extra energy, or has extra energy, to swap with his neighbors that also use the application. The responsibility of this component is to make the buying and selling energy process easier for the user and more visual.

3.2.2 Component Map interface description.

Map interface will be an actual map of the neighborhood, which shows all houses. The map will also show the status of each house, status could be buyer, seller, or none. A chat pop bubble is also available with each house, to be able to chat with neighbors and agree on prices.

3.2.3 Component Map processing detail

Start

Map is displayed for resident

Neighborhood details are displayed

Resident will choose an option depending on process he wants

Map will allow resident to either buy or sell

Map will display possible sellers and buyers

Neighbors will chat to negotiate prices and amount of energy needed

Map will transfer money from resident1 to resident2 with help of payment object

Map will transfer energy from SP1 to SP2 with help of solar panel object

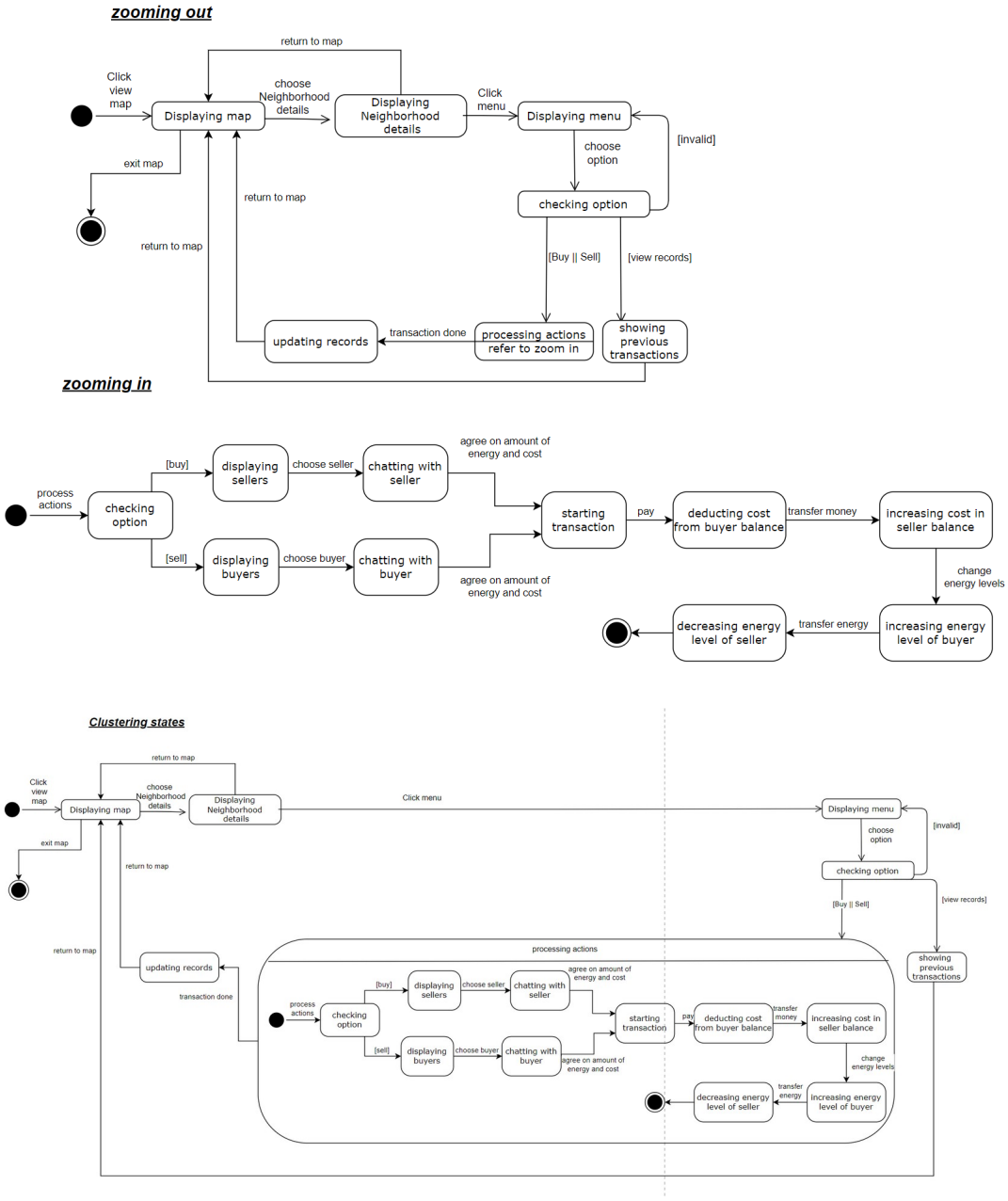
Map will update the records

Record will be displayed showing all previous transactions of resident

Resident exit map

End

3.2.3.1 Design Class hierarchy for component Map



Resident will open the map and view his neighbourhood, and choose the option he requires to perform from map. If resident requires to buy or sell energy, he will choose the neighbour to interact with. Both residents will chat and negotiate. Map will do all needed transfers after agreement is complete. Map will update the record , then display all previous transactions. Resident will finally exit the map

3.2.3.2 Restrictions/limitations for component Map

Residents cannot chat unless they require to buy or sell.

3.2.3.3 Performance issues for component Map

Not Available

3.2.3.4 Design constraints for component Map

Ensuring data availability and quality of map from map providers like google maps.

Map should be user friendly and easy to use

3.2.3.5 Processing detail for each operation of component Map

The class Payment has functions: BuyOrSell(), UpdateRecords()

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. BuyOrSell() calls the private functions Buy() or Sell() within its body. The private functions will access “Payment” and “SolarPanel” objects for transferring resources.

2. UpdateRecords() is called after finishing any buy or sell operation, and adds the new transaction to the list of previous transaction. This is used to keep track of data, and give the resident a general overview of his interactions

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. BuyOrSell()
Start
Map will display possible sellers and buyers

Neighbors will chat to negotiate prices and amount of energy needed

Map will do all needed transfers after agreement is complete.
End

2. UpdateRecords()
Start
After operation is complete, store new transaction in record
Map will display record
End

3.2 Description for Component House

3.2.1 Processing narrative (PSPEC) for component House

The component House consists of the class House. Its conations the attributes which are address, Roof Details, status and panel these attributes are the basic and unique data specify each house on another and used to complete and procedure that happen inside House class. The responsibilities of this component are to declare the variables of the house and to store information of each class details in system.

3.2.2 Component House interface description.

House interface will happen when each time a new or old resident register through the system. Each house will have unique address with containing a specific number of solar panels depend on family member and their needs. Next interface would be anytime resident wants to check his power and energy consummation or if resident wants to request a solar panel modification in case he wants to upgrade and downgrade his subscription with updating house details.

3.2.3 Component House processing detail

Start

Enter house details which is location of resident house

When resident request to see his energy of house, House will display resident energy consume and usage by calculate it.

Based on resident request if its valid and approved house can be modified and upgrade or downgrade

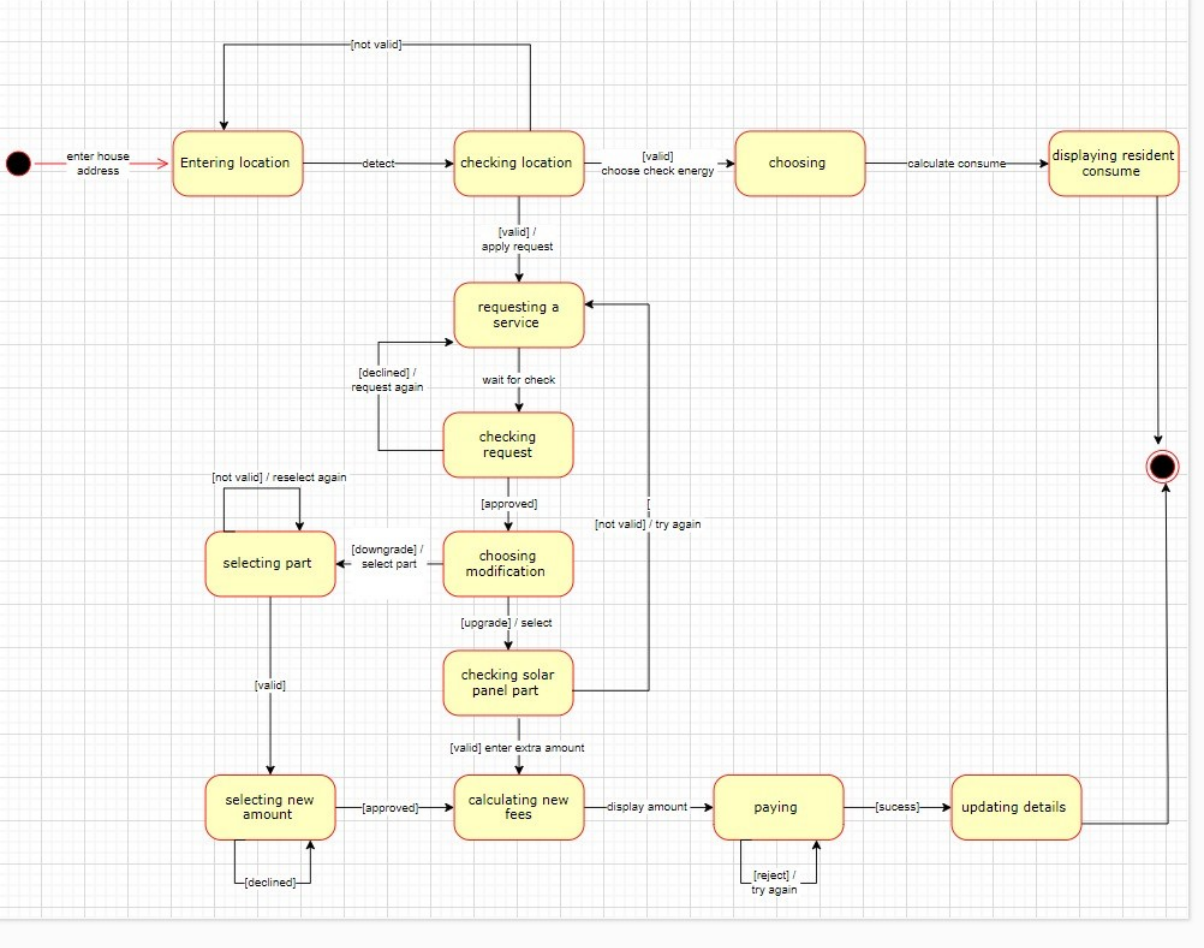
House modification will be by selecting part and amount that resident want to change in his house

House will calculate new fees of modification and do the payment transaction

House will update modifications and details

End

3.2.3.1 Design Class hierarchy for component House



When resident is enrolled to the application and wants to do know anything about his house, he should fist enter his house details, House class will authenticate the details if its valid he can check his energy consummation, as house class will calculate each resident energy consume and display it for him. House class will have other services that can be done for resident as up/down grade his solar panels this can be achieved if resident apply for a request for house modification once request is approved resident will select part and new amount of power in solar panel. By this house class will calculate cost of modification and display amount so resident can pay for it once payment is success house class will be updating its details

3.2.3.2 Restrictions/limitations for component House

Not available

3.2.3.3 Performance issues for component House

Not Available

3.2.3.4 Design constraints for component House

Ensure each house will has its own unique address and location.

All house details will be secured for each resident and limited in access by others.

3.2.3.5 Processing detail for each operation of component House

The class House has a multiple functions called RequestSPModification() and UpdatePanelDetails().

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. RequestSPModification() : this function will be called and used if residents wants to change his subscription in case he wants to upgrade or downgrade his solar panel

2. UpdatePaneldetails(): this function will be invoked after requestSPModification() is complete, to update the data related to the house to match the new data after modification

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

RequestSPModification():

Start

Apply for service request

Request will be checked if its valid

Resident must select modified part of solar panel

Calculation are made based on resident selection

Resident will pay the amount

End

UpdatePaneldetails():

Start

Modification of solar panel complete

New data loaded into system

End

3.2 Description for Component SolarPanel

3.2.1 Processing narrative (PSPEC) for component SolarPanel

The component SolarPanel consists of the class SolarPanel. It consists of multiple attributes which are the basic data of the class. These attributes are general details specify and used for each solar panel. Also methods (functions) describe what class SolarPanel can do and what is its responsibilities.

3.2.2 Component SolarPanel interface description.

Solar panel interface will happen between house and systemManagement class. as each house will have specific amount and number of solar panels and system management will be responsible to afford these solar panels to resident house.

3.2.3 Component SolarPanel processing detail

Start

SolarPanel is installed and start producing energy

SolarPanel can check its energy level

SolarPanel can calculate remaining power in each solar panel and display it to resident

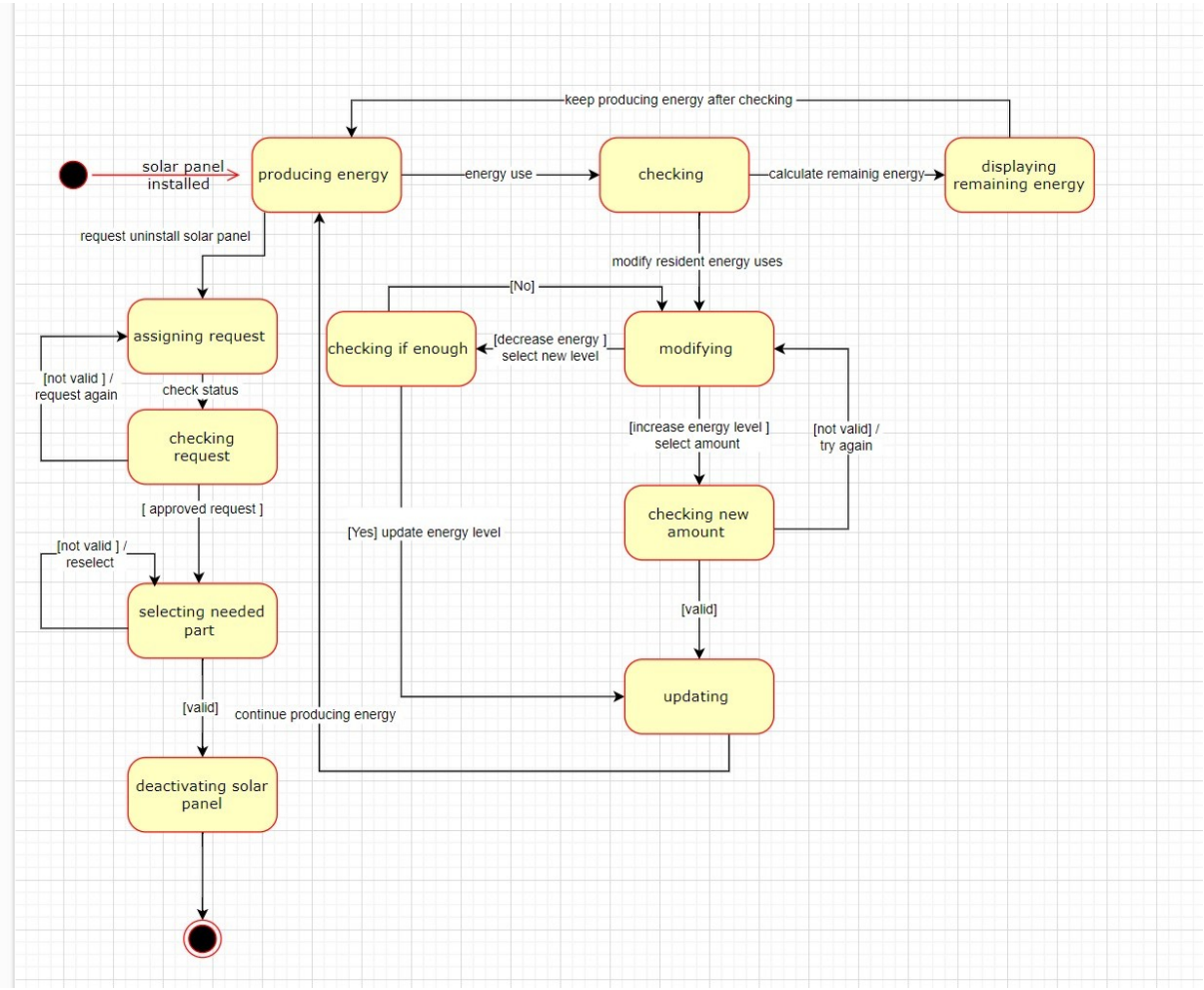
After check SolarPanel energy level we can modify it be increase or decreasing its power production

SolarPanel will be updating when its modified

SolarPanel can do a request for deactivate

End

3.2.3.1 Design Class hierarchy for component SolarPanel



When SolarPanel is activated and start producing energy, and resident start using it he can check each solar panel power and remaining energy. SolarPanel will do more services to resident as power produce by SolarPanel is not fixed we can increase and decrease it by resident choice. Also, when resident can assign a request and once request is approved we can deactivate a specific SolarPanel.

3.2.3.2 Restrictions/limitations for component SolarPanel

When a modification happens to SolarPanel it should be enough to cover resident uses.

3.2.3.3 Performance issues for component SolarPanel

Not Available

3.2.3.4 Design constraints for component SolarPanel

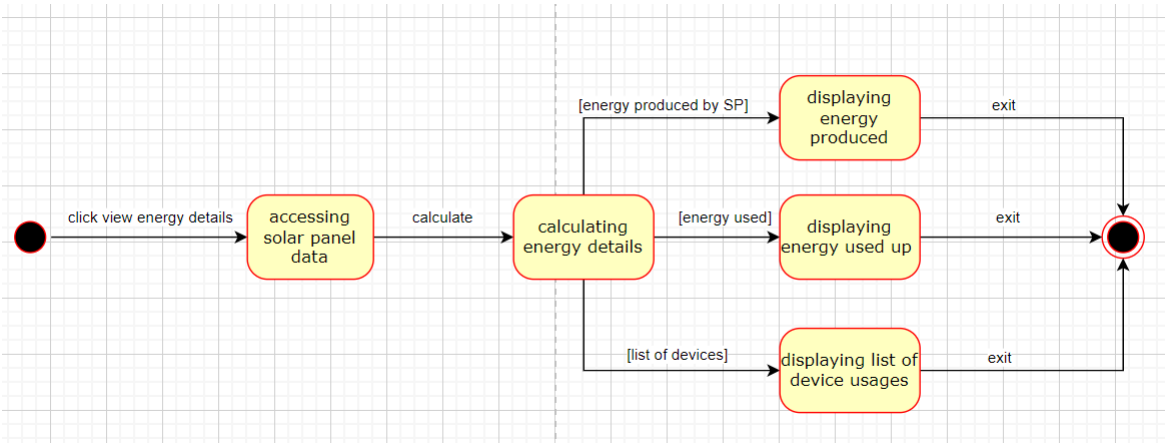
Ensure each house will have enough amount of solar panels to cover all resident consummation

3.2.3.5 Processing detail for each operation of component SolarPanel

The SolarPanel class has function called ViewEnergyDetails () , UpdateEnergyLevel()

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. ViewEnergyDetails(): this function will be called when resident wants to know all details of solar panel like what is the device or component in house that uses most of the solar panel energy or power. It also displays all SolarPanel details.



2. UpdateEnergyLevel(): has 2 parameters, both of type resident, this is invoked when a buying or selling operation is done. It update the data to match the energy transfer that occur is both residents solar panels after transaction.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

ViewEnergyDetails ():

Start

Resident apply to see solar panel details

SolarPanel will calculate and show all its details

SolarPanel display its details and information to resident

End

UpdateEnergyLevel():

Start

Transaction occurs between 2 residents

Solar panel data and attributes are updated to match energy transfer

End

3.2 Description for Component Account

3.2.1 Processing narrative (PSPEC) for component Account

The component Account consists of the class Account. It contains the attributes, which are the data needed to create any account in the application, paired with the methods needed to perform changes and checks balance in the account. Account object is used whenever any account is being created or used or with any modification in it . This includes viewing account ,checking balance, deleting subscriptions , selling energy ,Updating panel details and updating payment details . This Objects can be used for admin/resident/HandyWorker.

3.2.2 Component Account interface description.

Account interface will be triggered with every SIGN IN / LOG IN. Account interface first starts by choosing to log In or to sign In. When Sign In it will request to insert personal information along with an email. If Log In, only ID and Password are required. Next interface would be anytime a client needs to update details, view account, check balance, viewing account or delete subscription. The account interface will display all the client information. Also, whenever a payment is made the balance in the account will change.

3.2.3 Component Account processing detail

Start

Enter details to create accountn

Enter ID, verify

Enter password, verify

Log in, choose function from the list

view account, you can open panel details

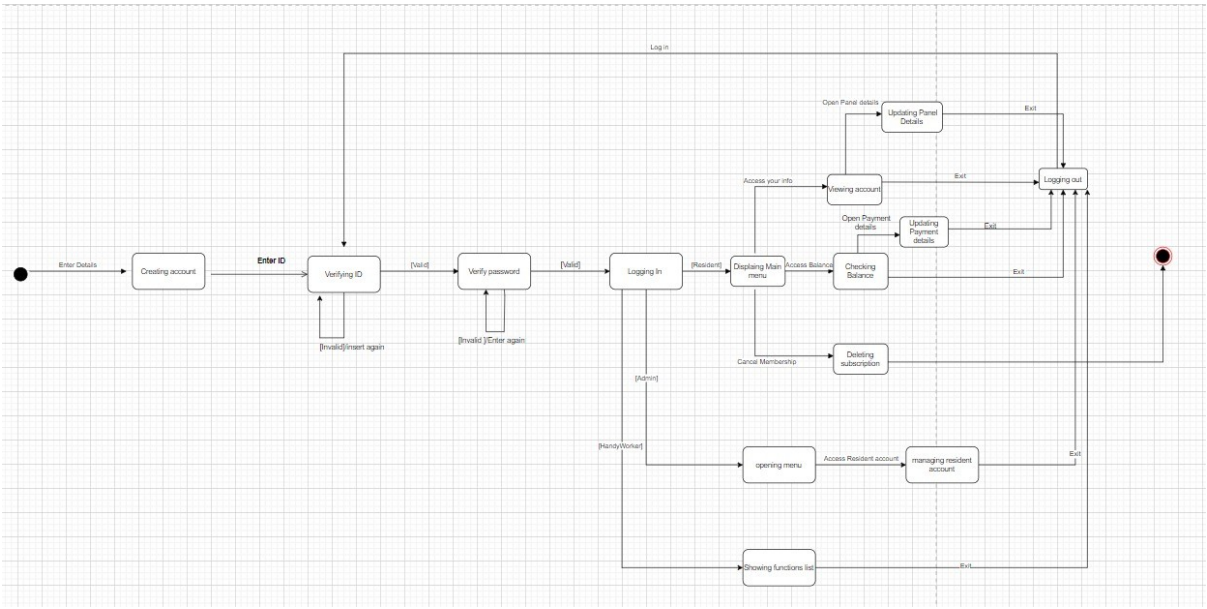
Check balance, and update payment details

Delete subscription, inactive account -> end

Log out

Log in again, enter ID

3.2.3.1 Design Class hierarchy for component Account



When a user enters his/her details the software will create an account. then it will ask him/her to insert the Id and password. if the ID or Password are not valid it will ask the user to try again. After the user successfully log in the system will open the main menu based on the ID. IF the user is a resident a whole list of functionalities will be there so He/She can choose from the list. viewing account function will show the user details. With checking balance, the resident will be able to trace their payment and see the total balance. In this function they can update their payment details. Deleting subscription will deactivate the account and it will no longer exist. If the user is an admin a menu will open, and he will have access to residents' accounts. If the user is the handyWorker a menu will open with list of available functions.

3.2.3.2 Restrictions/limitations for component Account

If the ID or Pass is not valid you cannot complete the process of logging in.

3.2.3.3 Performance issues for component Account

Not Available

3.2.3.4 Design constraints for component Account

Ensuring that accounts are separate.

Security: ensuring that Personal details are safe and can't be accessed with anyone. and the app is secured and can't be hacked.

3.2.3.5 Processing detail for each operation of component Account

The class account has a function called CheckBalance(),DeleteSub(), updatePaymentDetails(creditcard:Payment).

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. CheckBalance() is a function that shows the total balance after selling or buying energy .

2. DeleteSub() is used whenever a resident want to delete his/her subscription.

If the subscription was deleted the account will be deactivated.

3. updatePaymentDetails(creditcard:Payment). This is used when a resident wants to modify or change the payment details

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. CheckBalance() :
Start
Request to check the balance
Return the Balance
End

2. DeleteSub()

Start

Cancel Membership

Delete ID and Pass

Deactivate the account

End

3. updatePaymentDetails(creditcard:Payment).

Start

Choose what to update

Do the modification

End

3.2 Description for Component SystemManagement

3.2.1 Processing narrative (PSPEC) for component SystemManagement

The component System Management is a class that has attributes and functions. The attributes are the number of residents and number of employees. The method describes some functionality delivered by the class. System Management object is used whenever a new user is enrolled or whenever the users use the system.

3.2.2 Component SystemManagement interface description.

Whenever a new user sign up the system management will create a new account and choose panels for residents. it will display The Balance as well as the suitable number of panels. When the HandyWorker Inspect it will notify the resident and send him a notification. It will also display to the HandyWorker the date and location that the resident chosen.

3.2.3 Component SystemManagement processing detail

Start

Operating the system.

Enroll new user

Based on the new user the System management will display different functions

If user was Resident, insert details

System Management will chose suitable panel and will ask for resident approval

Based on the approval the process will continue or stop.

If approved the system management will input SP details

The payment process will start.

System Management will calculate total balance.

If user was employee/admin. they should insert details

System management will save the information.

Will go again to operating

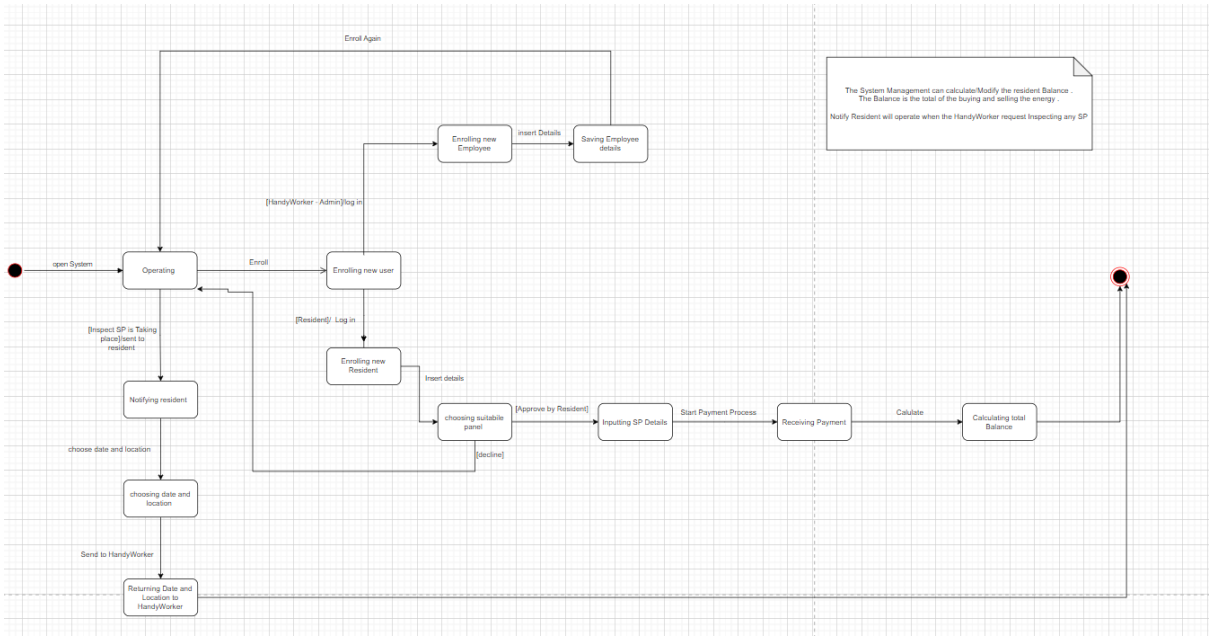
Whenever inspect SP is called by the HandyWorker the system management will notify resident

Ask resident to input date and location.

Sent date and location to HandyWorker.

End

3.2.3.1 Design Class hierarchy for component SystemManagement



When the System is operating and a new user enrolled it will see if the user is a resident or it is HandyWorker/Admin. If it is a resident the system will ask him/his to log in . After that it will choose a suitable panel depending on number of family members and other features . If the resident agrees on the deal the System will input SP details and then calculate the total cost and the payment process will start . if resident does not agree it will stop and start again .

If the new user is HandyWorker/Admin they will inset their details and it will be saved in the system and goes back to the beginning.

If “Inspect SP” function were called the SystemManegemnt will notify the resident and will ask him/her to choose date and location which will be returned to hanyWorker .

3.2.3.2 Restrictions/limitations for component SystemManegemant

SystemManegemnt can only take one new user at a time . A user cannot be an Employee and a resident with the same account .

3.2.3.3 Performance issues for component SystemManegemnt

SystemManegemnt can only take one new user each time

3.2.3.4 Design constraints for component SystemManegemnt

SystemManegemnt roles and access limitation for each user

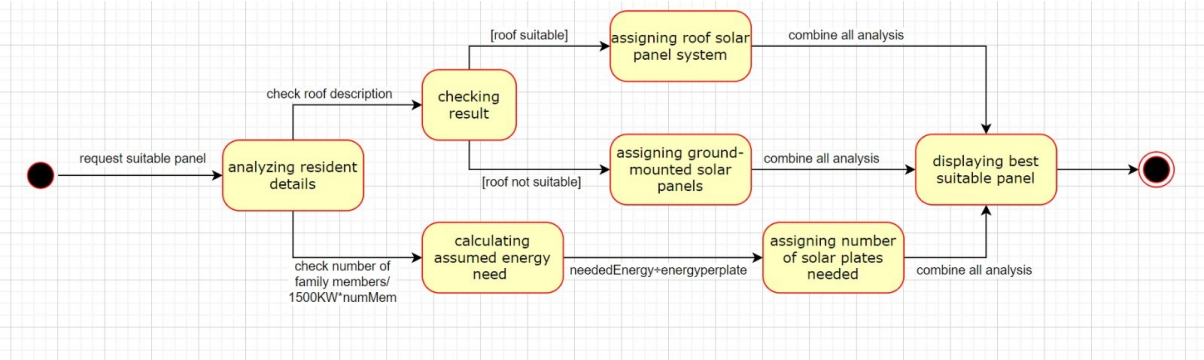
3.2.3.5 Processing detail for each operation of component SystemManagemnt

The class SystemManegemnt has a function called choosePanel(), EnrollNewEmployee(), EnrollNewResident().

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. choosePanel() after a new resident enroll in the system .it will choose a suitable package to use .The function will analyse the resident details . It will see the roof description and will calculate assumed energy needed using (number of family members * 1500KW) .Based on that it will assign number of solar plates(neededEnergy/energyperplate). After that it will display the result to the resident .

If the roof is suitable, it will assign roof solar panel. If not suitable assign ground-mounted solar panels. Finally, will sent the result to the resident.



EnrollNewEmployee() and EnrollNewResident() are similar functions, but each for different actors with different access levels.

2.EnrollNewResident(): When a new resident signs in, his detailed will be saved and a new account will be created. An ID and password will be generated for resident.

3. EnrollNewEmployee(): when a new employee id hired, his details will be saved, and his access levels will be displayed. An ID, password, and employee PIN will be generated for employee.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

choosePanel()

Start

Analyse resident details

Check roof description and check number of family

If roof suitable assign roof SP system.

If root not suitable assign ground – mounted SP.

check number of family and calculate energy needed

assign number of SP plates

display the best Suitable panel to the resident .

End

EnrollNewResident()

Start

Resident sign up

Data of resident saved

ID and password generated for resident

End

EnrollNewEmployee()

Start

Employee hired

Data of employee saved

ID, password, and PIN generated for employee

End

3.2 Description for Component Supplier

3.2.1 Processing narrative (PSPEC) for component Supplier

The component Supplier consists of the Supplier class. It contains the attributes, which are the data needed to handle requests for supplying solar panels and parts in the application, paired with the methods needed to process these requests. The Supplier object is used whenever a request for supplying solar panels and parts is made. The responsibility of this component is to manage the inventory, fulfil supply requests on time, and maintain accurate records of all transactions.

3.2.2 Component Supplier interface description.

The Supplier interface will be triggered with every request for the supply of solar panels and parts. The interface first starts by collecting the necessary details for the supply request and validating them. When a request for supply is made, the interface will display the available inventory, the required quantity, and any additional information. The requester can then confirm the order for supply. The requester can also check the status of their supply request to monitor the progress. Additionally, when a supply request is initiated, a notification message will be displayed in the application, indicating the estimated delivery date and order details.

3.2.3 Component Supplier processing details.

Start

New supplier enrolled.

Admin requests supplement process.

Based on the request, the supplier will check if it’s a panel or parts.

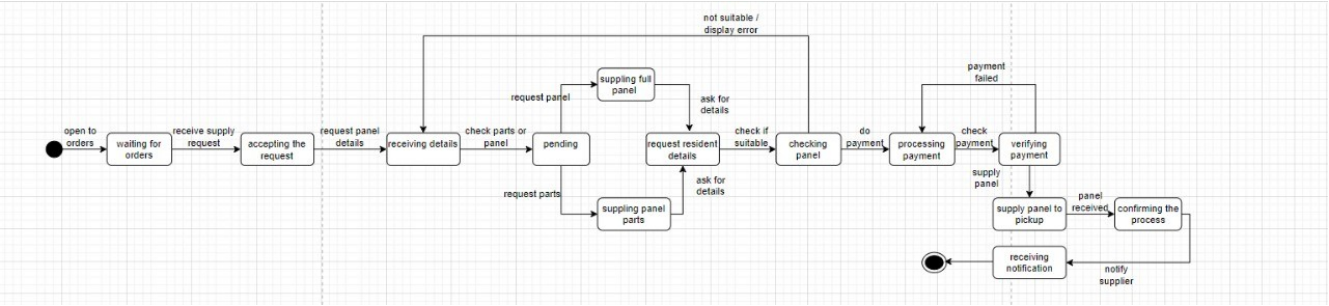
Supplier will accept the request and start processing.

Supplier will process payments.

Suppliers send the request for the admin that its ready to pick up.

supplier will receive confirmation.

3.2.3.1 Design Class hierarchy for component Supplier.



When a request for the supply of solar panels or parts is initiated by a user, the Supplier component is activated. The component will begin by verifying the credit card details of the user to ensure the payment method is secure. Depending on the user's request for solar panels or parts, they will choose a suitable payment method. the user will receive notifications, confirming the payment. Ater that the supplier will process with the supply process, the supplier will notify the admin that the request is ready for pickup, after confirming from the user the supplier will receive a notification for confirming the process.

3.2.3.2 Restrictions/limitations for component Supplier

The supplier cannot contact the resident for more details.

3.2.3.3 Performance issues for component Supplier

Not Available

3.2.3.4 Design constraints for component Supplier

The supplier can only receive the payment but cannot calculate it.

3.2.3.5 Processing detail for each operation of component Supplier

The class Payment has a function called SupplySolarPanel(), Supplyapart(partSerialNum), ConfirmDone(SPSerialNum).

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. SupplySolarPanel(): first it collects the necessary details from the admin to determine the type and specifications of the solar panel required. The method checks if the requested panel is suitable based on the details provided. If the panel is deemed suitable, the method initiates the supply process. However, if the panel is not suitable, the admin is informed, and the process is restarted to allow for adjustments in the panel specifications. This ensures that the resident receives the appropriate solar panel that meets their requirements.

2. SupplyaPart(partSerialNum): this function do the same as the supplySolarPanel(), the only difference is that it supply the parts that needed for example, if any modification is needed then this function will supply the needed parts using the serial number.

3. confirmDone(SPSerialNum): this function will confirm the supplier that the user accepted the supplement process and the payment will be processed soon.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. SupplySolarPanel():

Start

1. Collect the necessary details

2. Check if the requested panel is suitable

3. If the panel is deemed suitable:

   - Initiate the supply process.

4. If the panel is not suitable:

   - Inform the user.

   - Restart the process to allow adjustments in the panel specifications.

5. Ensure that the user receives the appropriate solar panel that meets their requirements.

End

2. SupplyaPart(partSerialNum):

Start

1. Collect the necessary

2. Check if the requested part is suitable based on the details provided and the part's serial number.

3. If the part is deemed suitable:

- Initiate the supply process for the required part.

4. If the part is not suitable:

- Inform the user.

- Restart the process.

5. Ensure that the user receives the appropriate part needed for

any modifications.

End

3. confirmDone(SPSerialNum):

Start

1. Confirm that the user has accepted the supplement (identified by SPSerialNum).

2. Notify the supplier that the user has accepted the supplement, and the payment will be processed soon.

End

3.2 Description for Component Admin

3.2.1 Processing narrative (PSPEC) for component Admin

The component Admin consists of the class Admin. It contains the attributes necessary to manage various administrative tasks in the application, paired with the methods needed to perform changes and checks required throughout the process. The admin object is used for tasks such as requesting solar panels and parts for users, contacting residents, handymen, and suppliers when necessary, and managing access details. It can also display account options for users. The primary responsibility of this component is to ensure efficient communication and operations within the application.

3.2.2 Component Admin interface description.

The admin interface is triggered whenever administrative tasks are required. It starts by requesting the necessary information or actions, including requesting solar panels and parts, contacting relevant parties, and managing user access details. When necessary, the admin interface will display account options for users, providing them with choices to configure their accounts according to their needs.

3.2.3 Component Admin processing detail

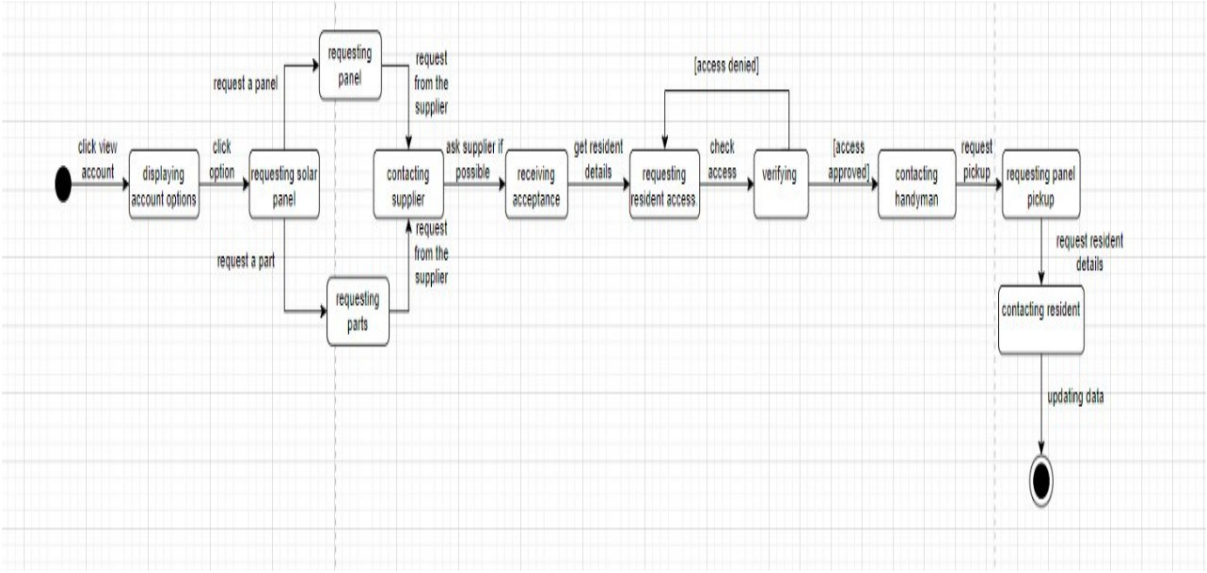
Admin can request solar panels and parts for users, ensuring that the necessary equipment is available when needed.

Admin has the capability to contact residents, handymen, and suppliers when necessary to facilitate communication and address specific requirements.

Admin can request and manage access details from users, ensuring that access permissions are correctly configured.

When needed, the admin interface can display account options for users, allowing them to customize their accounts as per their preferences.

3.2.3.1 Design Class hierarchy for component Admin



Admin can request solar panels and parts for users, ensuring that the necessary equipment is available when needed. The admin also possesses the capability to contact residents, handymen, and suppliers when necessary to facilitate communication and address specific requirements. Additionally, Admin can request and manage access details from users, ensuring that access permissions are correctly configured to maintain security and authorization. When necessary, the admin interface can display account options for users. This comprehensive set of functions ensures effective management, communication facilitation, access control, and user account customization within the application.

3.2.3.2 Restrictions/limitations for component Admin

Admin cannot track the process for any requests.

3.2.3.3 Performance issues for component Admin

Lack of accessibility to the processes and it might cause delay.

3.2.3.4 Design constraints for component Admin

Admin roles and access constraints.

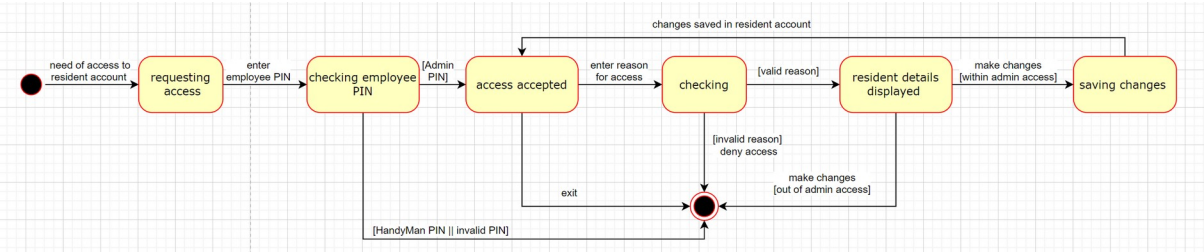
3.2.3.5 Processing detail for each operation of component Admin

The class admin has functions called:
requestResidentAccess(ResidentID),

DisplayAccountOptions(), RequestSPPickup(SPSerailNum) and more.

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. requestResidentAccess(ResidentID): this function allow the admin to request access from the user in there's any modification require to his account.



2. DisplayAccountOptions(): the admin will allow the GUI to display the account option for modify, edit, install and more options.

3. RequestSPPickup(SPSerailNum): this function is needed due to the lack of communication between the supplier and the handyman, so here the admin requires the handyman to pick up the solar panel from supplier.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. requestResidentAccess(ResidentID):

Start

Admin request access

Access approved

Give details to handyman

End

2. DisplayAccountOptions():

Start

After user click to account options

Admin display options

End

3. RequestSPPickup(SPSerailNum):

Start

Admin request the handyman to come to pick up panels.

Handyman pickup

End

3.2 Description for Component Resident

3.2.1 Processing narrative (PSPEC) for component Resident

The component Resident consists of the class resident. It contains the attributes, which are the data needed to complete any procedures related to the resident in the application, paired with the methods needed to perform changes and checks needed along the process. Resident object is used whenever the resident does any action like signing in or creating a new account or chatting with neighbors to negotiate pricings, this also includes any edits on the solar panel such as requesting a new one or asking for a checkup from the admin through the application. Another role of the Resident object is to let the resident decide how they want to pay for their solar panels. The responsibility of this component is to allow communication between the resident with admin and resident with supplier in case any changes or edits are needed to anything.

3.2.2 Component Resident interface description.

Resident interface will be triggered with every action done by the resident. Resident interface first starts by viewing the home page for the resident to decide what option they want to do from the options available. Next interface would be anytime a resident wants to update details in their account, where the resident interface will display the all the details that you can edit. Resident can also request anything related to the solar panel, which can be requesting a new one or choosing a suitable panel, in addition to requesting a check-up from the admin to make sure that the panel is working well. Also, the resident can view the awareness page to read facts and statistics about the solar panels and their effect in helping earth become a better place. The resident can also view map to check who is buying or selling energy so that they can chat with them and negotiate a

good deal. In addition to all that, if the user wants to delete or cancel their membership they are free to do so.

3.2.3 Component Resident processing detail

Start

Resident registers a new account or signs into their existing account.

The component resident displays the home page to the resident.

The home page allows the resident to update account details, modify panel, request a new one or a check up for an existing one, read awareness page, or view map.

In the component resident when the user requests a new solar panel, the system chooses the suitable one based on the house's specifications.

After the resident confirms the suitable panel, he/she settles the down payment, then they can choose the installation date.

The user can also request a checkup with the suitable date.

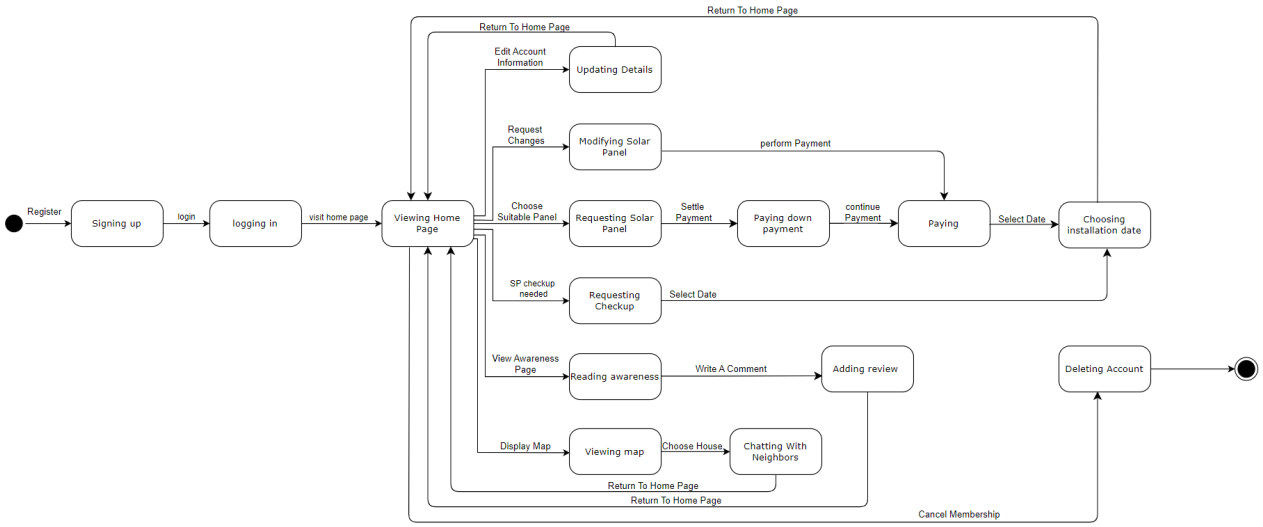
The component resident lets the user update their account information ex. username or phone number.

When the resident wants to buy or sell energy, they can access the map to chat with people that buy or sell.

In the component resident if the resident wanted to delete their account or membership, they can do that from the home page.

If the resident wanted to read information about solar panels or anything related to that, they can press on the awareness page option.

3.2.3.1 Design Class hierarchy for component Resident



When a resident signs in or creates a new account they will be faced with a few actions to choose from. When the resident chooses to update their details they can edit their personal information. When he/she is done they can return to the home page to either leave the app or choose another action. In addition to that the resident can ask for any modifications or requests regarding the solar panel, such as requesting to buy a new one or asking for a checkup on an existing one. The resident can also view the map to chat with neighbors, as well as reading the awareness page and adding a review on the app. Lastly if the resident wishes to cancel their membership they can do so through the “delete account” option.

3.2.3.2 Restrictions/limitations for component Resident

If resident didn’t pay for a long time, no action will be done except continuously notifying.

3.2.3.3 Performance issues for component Resident

Not Available

3.2.3.4 Design constraints for component Resident

Ensuring the user gets the right panel for their house.

Security: ensuring that the credit card details of our residents are completely secure, and that the external services used are reliable and trustworthy, to gain the trust of our customers

3.2.3.5 Processing detail for each operation of component Resident

The class Resident has functions called UpdateDetails(Details), PayDownPayment(), requestCheckup(), readAwareness(), Chat(ResidentID).

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. UpdateDetails(Details) is when the resident wants to edit any information related to their account, such as the phone number or card details.

2. PayDownPayment () is invoked inside another method which is requestSuitablePanel (), when the suitable panel is chosen the resident proceeds to settle down the down payment. In addition to choosing how to pay, card or cash and with installments or not.

3. requestCheckup() is used whenever the resident feels like there is a problem in their solar panel, where they can ask for the admin's help in checking up the solar panel.

4. readAwareness() is invoked whenever a the resident wants to read the awareness page or add a review.

5. Chat(ResidentID) is invoked inside another method which is requestViewMap(), that allows the user to view the neighbors details such as who is buying/selling. In addition to chatting with neighbors to discuss the pricings.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. UpdateDetails(Details):

Start

Login/sign up to account.

Choose what to edit or update.

Confirm details.

End

2. PayDownPayment():

Start

Request suitable panel.

Give confirmation to selected panel.

Choose how to pay (method).

Confirm payment.

Update balance in resident account.

End

3. requestCheckup():

Start

Contact admin for assistance

Request a checkup on solar panel.

Confirm date for checkup.

End

4. readAwareness():

Start

Visit awareness page

Read information written

Return to home page

End

5. Chat(ResidentID):

Start

View map

Visit neighbourhood houses details

Chat with selected house

Return to home page

End

3.2 Description for Component Awareness

3.2.1 Processing narrative (PSPEC) for component Awareness

The component Awareness consists of class Awareness. It contains the attributes, which are the data needed to complete any procedure related to the news and facts parts about sustainability and saving earth in the application, paired with the methods needed to perform changes and checks needed along the process. Awareness object is used whenever the resident wants to read the news or facts provided, or whenever the resident wants to write a review on what they think about the information written or anything generally. This also includes editing the page if there is anything new to add or any false information to be fixed, this action is done by the admin. The responsibility of this component is to make sure everything is up to date, and the information being written there is not outdated or false.

3.2.2 Component Awareness interface description.

Awareness interface will be triggered with every request to view the page. Awareness interface first starts by the resident requesting to view the page where all the information about solar panel and all details needed is written. Next interface would be whenever the resident wants to write or add a review. First, he or she can view the reviews submitted by other people then they can write their own review based on what they think of the app, or the information presented in the awareness page. The previous actions are done by the resident. However, the admin can update the information that's being read by the resident, in addition to saving all the reviews after a resident posts a review.

3.2.3 Component Awareness processing detail

Start

Resident or admin opens awareness page.

Resident reads information written.

Admin edits information being read by the resident.

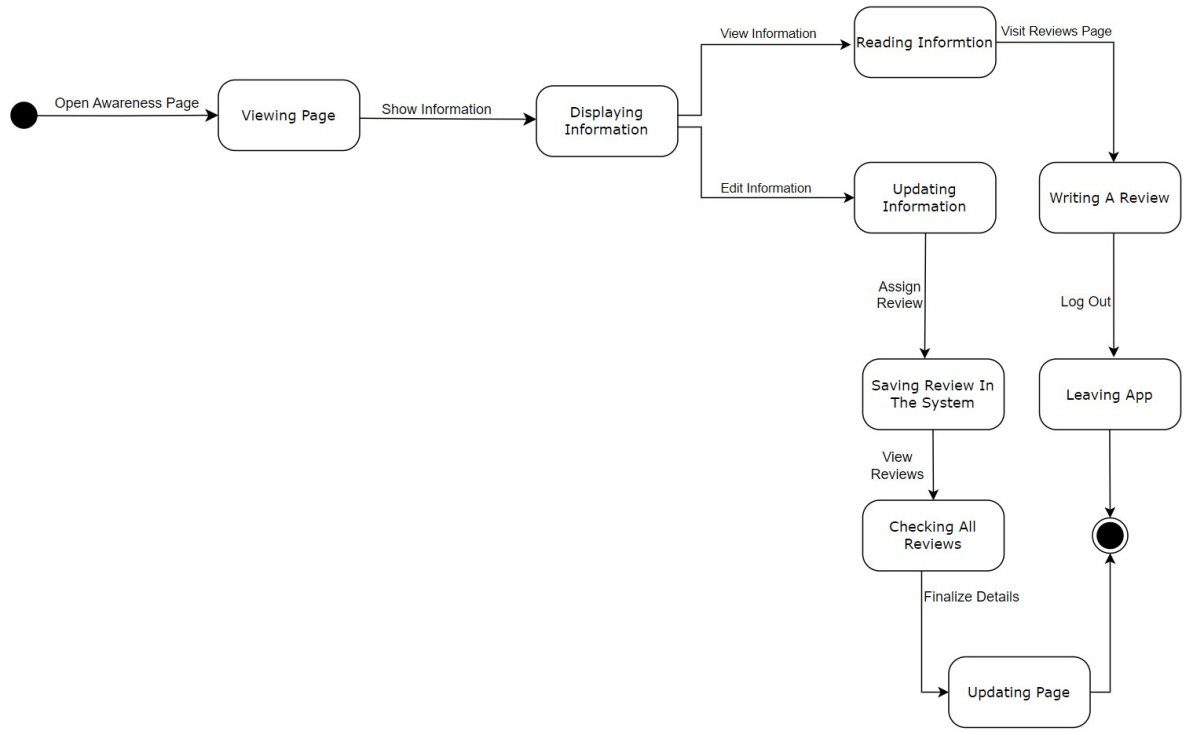
Residents have the option to write a review.

Based on choice, if the resident decides to write a review, they can view all previous reviews done by other residents, then write a review.

Admin then saves the review in the system and updates the page.

The resident can return to home page or leave the app after leaving their opinion.

3.2.3.1 Design Class hierarchy for component Awareness



When a resident signs in or creates a new account, they will be faced with a home page that gives them the a few options, one of these options is visiting awareness page. When you visit the page, you'll get to see information about the solar panel in addition to facts and statistics related to the solar panel itself. Then you can add a review about the app or the information you just read. However, the admin can also access the awareness page, but his role is to edit the information, save the new reviews in the system, and to update the page.

3.2.3.2 Restrictions/limitations for component Awareness

Not available

3.2.3.3 Performance issues for component Awareness

Not Available

3.2.3.4 Design constraints for component Awareness

Ensuring compatibility with adding and deleting reviews by the residents.

Security: ensuring that the resident's account doesn't get hacked in a way that causes the hacker to create some random reviews in the name of our residents. We should gain the trust of our customers, which means to ensure that they get the safety and security needed for that.

3.2.3.5 Processing detail for each operation of component Awareness

The class Awareness has functions called Read(), UpdateInfo(information), DisplayAllReviews(review).

3.2.3.5.1 Processing narrative (PSPEC) for each operation

1. Read() is used when the resident decides to read the information written in the information page. It will show them the facts and statistics downloaded by the admin.

2. UpdateInfo(Information) is used whenever a the admin wants to edit some type of information, when he/she is done editing they can update the page which will show the resident new details about the solar panel.

3. DisplayAllReviews(Reviews) is invoked from the AssignReviews() function where the resident writes a review then the admin assigns it to the system followed by displaying all the reviews to the resident right after the resident confirms their thoughts.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

1. Read():

Start

Visit awareness page

Read information written

Write a review

Return to home page

End

2. UpdateInfo(Information):

Start

Login with admin ID

Visit awareness page

Edit page

Save information

End

3. DisplayAllReviews(Reviews):

Start

Reviews page allows you to write your own review.

When review is written you can visit see all reviews.

Then return to home page.

End

3.2 Description for Component Reward

3.2.1 Processing narrative (PSPEC) for component Reward

The component Reward consists of class Reward. The attributes in class reward are the number of points and the energy saved. These are the only attributes needed for the methods in this class. The methods describe the roles and responsibilities of the rewards in the system and how every method functions. Reward object is used whenever the resident wants to convert their energy to points or redeem rewards based on how many points they collected.

3.2.2 Component Reward interface description.

Whenever the resident wants to check their points or calculate any of their energy into points or redeem rewards, they can visit the rewards page. When you visit the rewards page, you can view your points and all details related to that. When done you can return to home page.

3.2.3 Component Reward processing detail

Start

Resident signs in.

Resident opens points page.

Reward system displays the points the points collected by the resident to them.

User asks to calculate points; reward system calculates then displays total points to be received.

Resident selects rewards option.

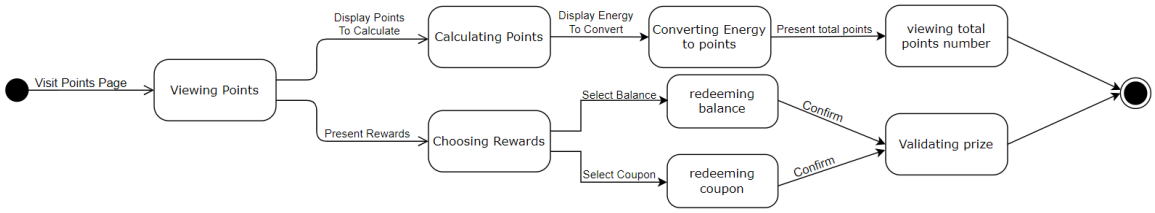
Resident chooses what reward to redeem.

Reward system displays received reward.

Resident returns to home page

End

3.2.3.1 Design Class hierarchy for component Reward



When the resident signs in, they can visit the points page. That's where they collect and check all the points they have. They have different options to choose from such as viewing points to choose either to calculate your saved points, or to choose rewards if you have enough points for that. If you as the resident choose to calculate points you can proceed with checking your energy saved and you can decide if you want to convert that energy to points or not. After that you can view your total number of points after conversion. However, if you choose the option of picking your reward you can select one of two, either to redeem balance in your account or redeem coupons for grocery stores or gym memberships and so on. Finally, after you confirm both options you can return to home page or leave the application.

3.2.3.2 Restrictions/limitations for component Reward

Reward system can only perform one task at a time.

3.2.3.3 Performance issues for component Reward

Reward system can only perform one task at a time.

3.2.3.4 Design constraints for component Reward

Reward roles and access constraints.

3.2.3.5 Processing detail for each operation of component Reward

The class reward has functions called convertEnergy(),calculatePoints(), and displayRewards().

3.2.3.5.1 Processing narrative (PSPEC) for each operation

convertEnergy() is used whenever the resident has the eligible amount of energy saved, they can convert it into points. The reason behind the converting option is to give the resident the choice of converting now or later.

calculatePoints() is used whenever you want to calculate the points you have as a resident. The system will show you how many rewards you can redeem if it's a coupon or how much of a balance you would have if you chose to redeemBalance.

displayRewards() is invoked from the redeemBalance() and redeemCoupon() functions where it shows the resident the rewards they chose after selecting and confirming.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

convertEnergy():

Start

Resident signs in

Resident visits rewards page

Then views energy saved

Rewards page gives resident the option if they want to convert energy to points or keep collecting then convert later on

If resident chooses to convert energy, reward system will display total number of points after converting energy.

End

CalculatePoints():

Start

Resident visits rewards page

After the resident converts energy to points if desired, they can calculate the points to see if redeeming any reward is possible.

Then reward system displays total number of points.

End

displayRewards():

Start

resident chooses if they wish to redeem balance or coupons.

Then they will be faced with the still available rewards after they redeemed what they want.

Confirm

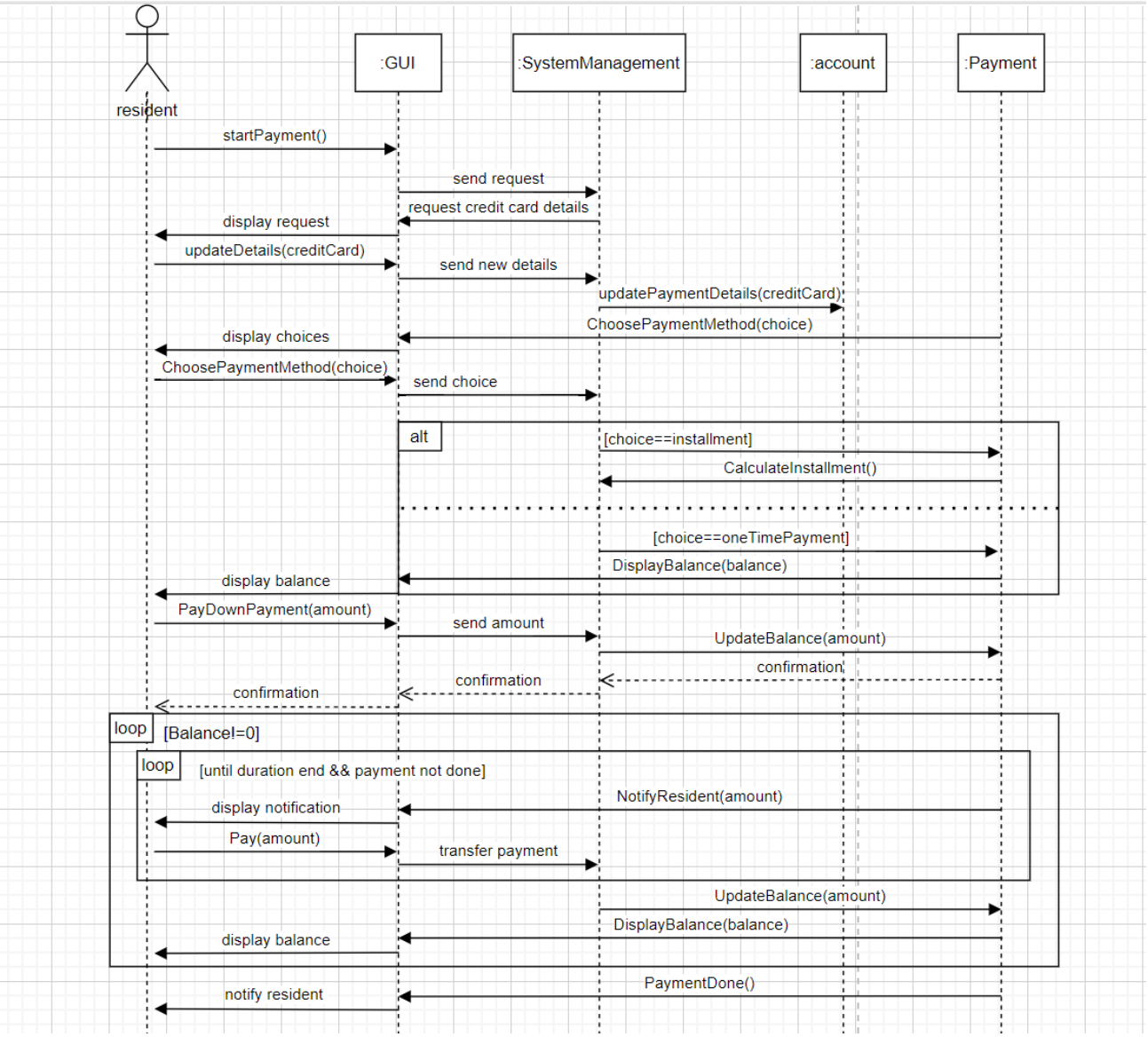
Return to homepage

End

3.3 Dynamic Behavior for Components

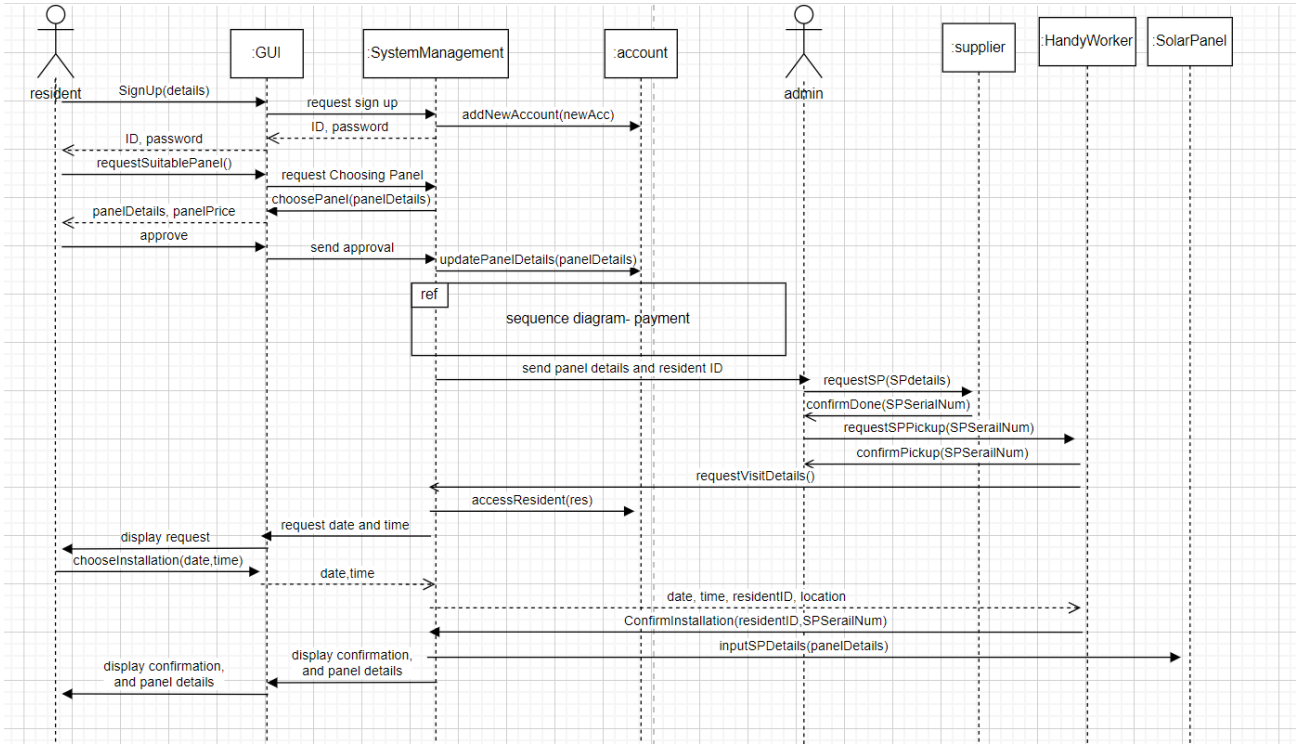
3.3.1 Interaction Diagrams

- ***Start Payment***



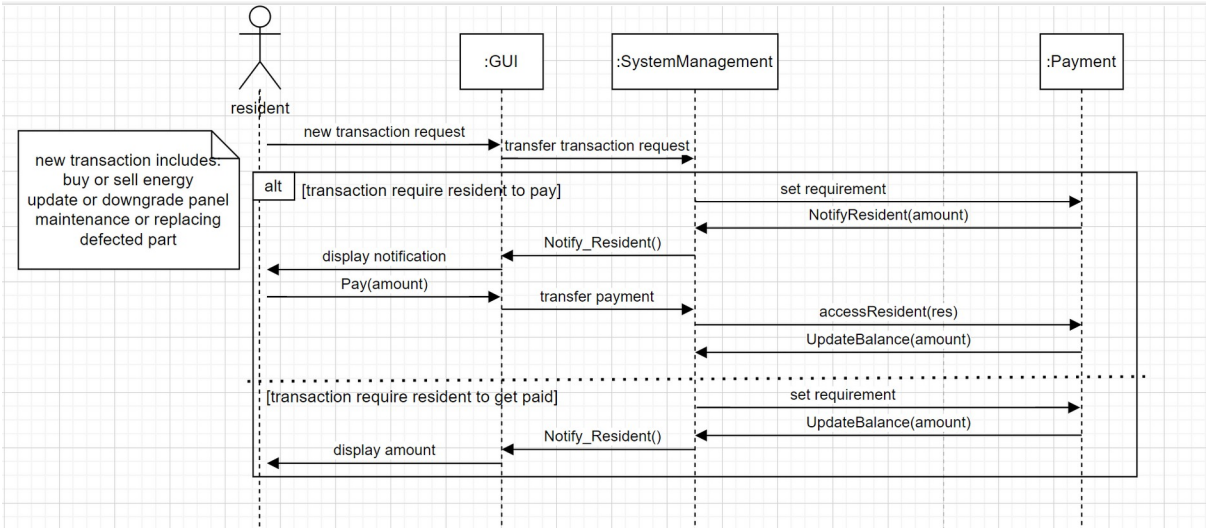
After a resident gets paired with the most suitable panel for him, the resident must pay to start the installation process. Resident will enter his credit card details, then 2 choices will be displayed for him regarding what's the payment plan he wants to follow, either installment plan, or one time payment. after resident makes choice, he will pay a down payment to start the process, and balance will be updated. Later, after a duration of 1 month has passed, the resident will have to perform the next payment required, if he still has balance in the system. This will keep repeating until all required payment from resident is complete, and balance is finally 0.

- ***Generate suitable panel***



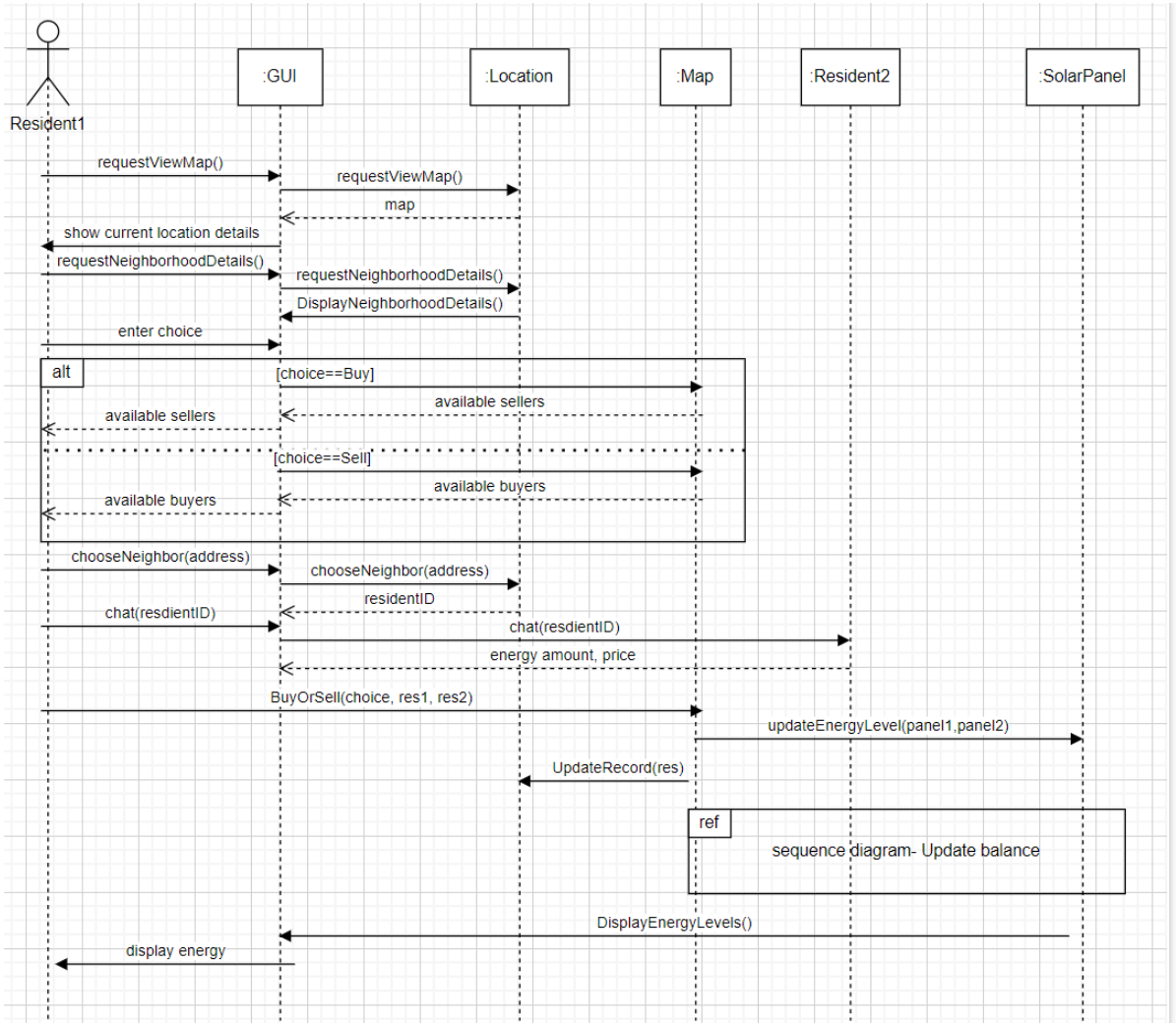
When a new resident starts up in the application, he enters his full details, system management will create an account for user and add all his details into account. solar panel will be assigned to resident based on his details. After the choice has been made, payment algorithm will apply. Once payment was made, admin will get notified with required panel details, so that he can request needed panel from supplier. Once manufacturing of panel is done, admin will be notified with the serial number of solar panel. Admin will send serial number of solar panel to handyWorker, and he will collect panel. Resident will choose the best time and date suitable for him, so handyman can install. SystemManagement will update information in resident account.

- ***Update Balance***



Many transactions in out system require the involvement of payment object to update the balance, so whenever a transaction of that type is requested by resident, the process of updating balance, whether adding money or deducting money, is performed.

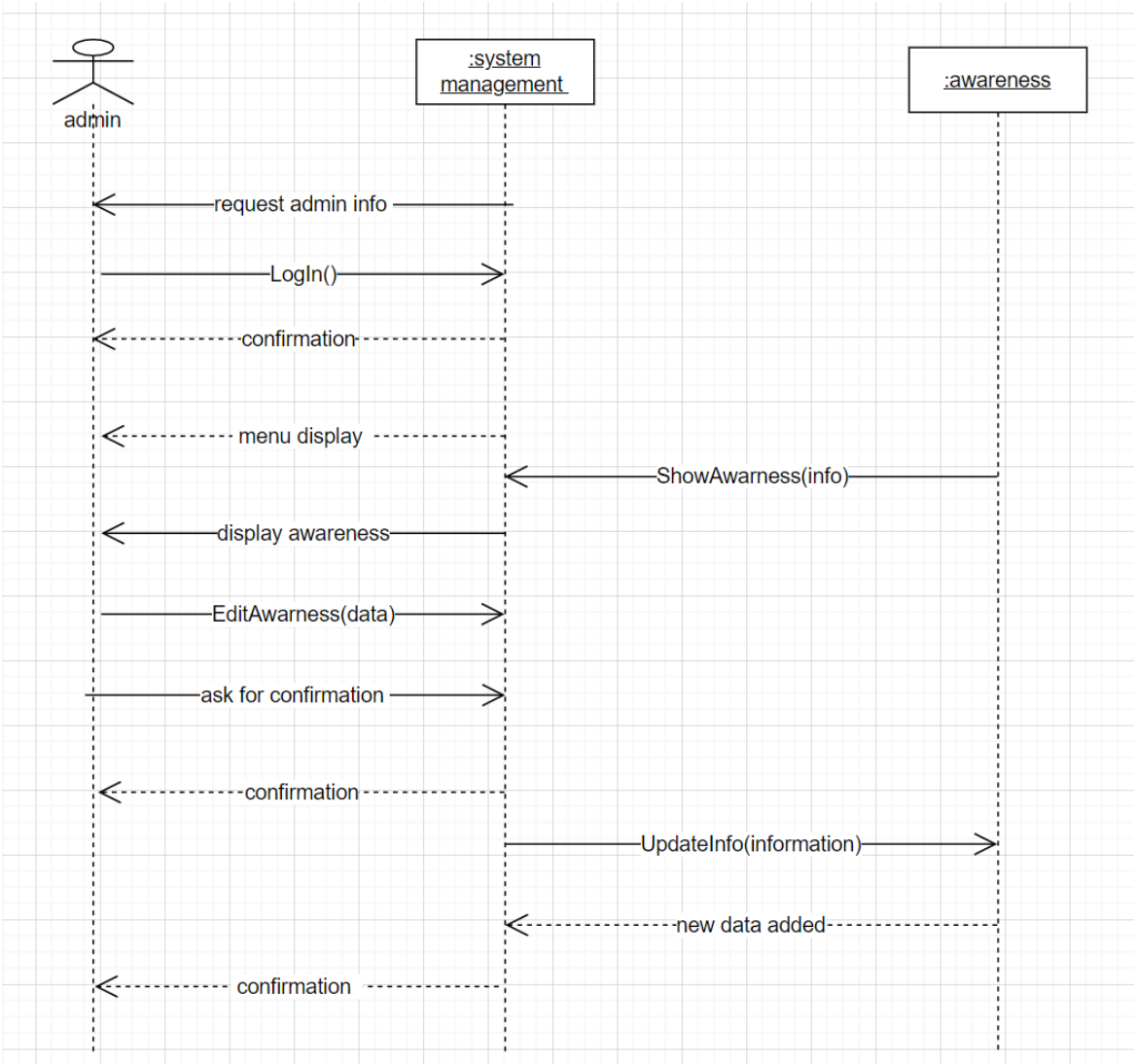
- ***View map***



When the resident1 wants to buy or sell he/she will request to view map from location, so location show resident1 the current location details. Buy or sell are done with the neighbours available so resident1 should check the neighbours by requesting location to display the neighbours' details. As the details are displayed resident1 will enter choice whether buy or sell, based on the choice the Map will show resident1 the available sellers/buyers so the resident can choose the suitable neighbour(address). Location will provide resident1 by resident2 ID to start chatting so resident2 settle the energy amount and price. One of the transmissions (buy or sell) will be done after this step, then Map will update the energy level for both solar panels (for resident1 & resident2). The new energy levels of solar panels will be returned to resident1. To check the payment process, refer to the sequence diagram "update balance".

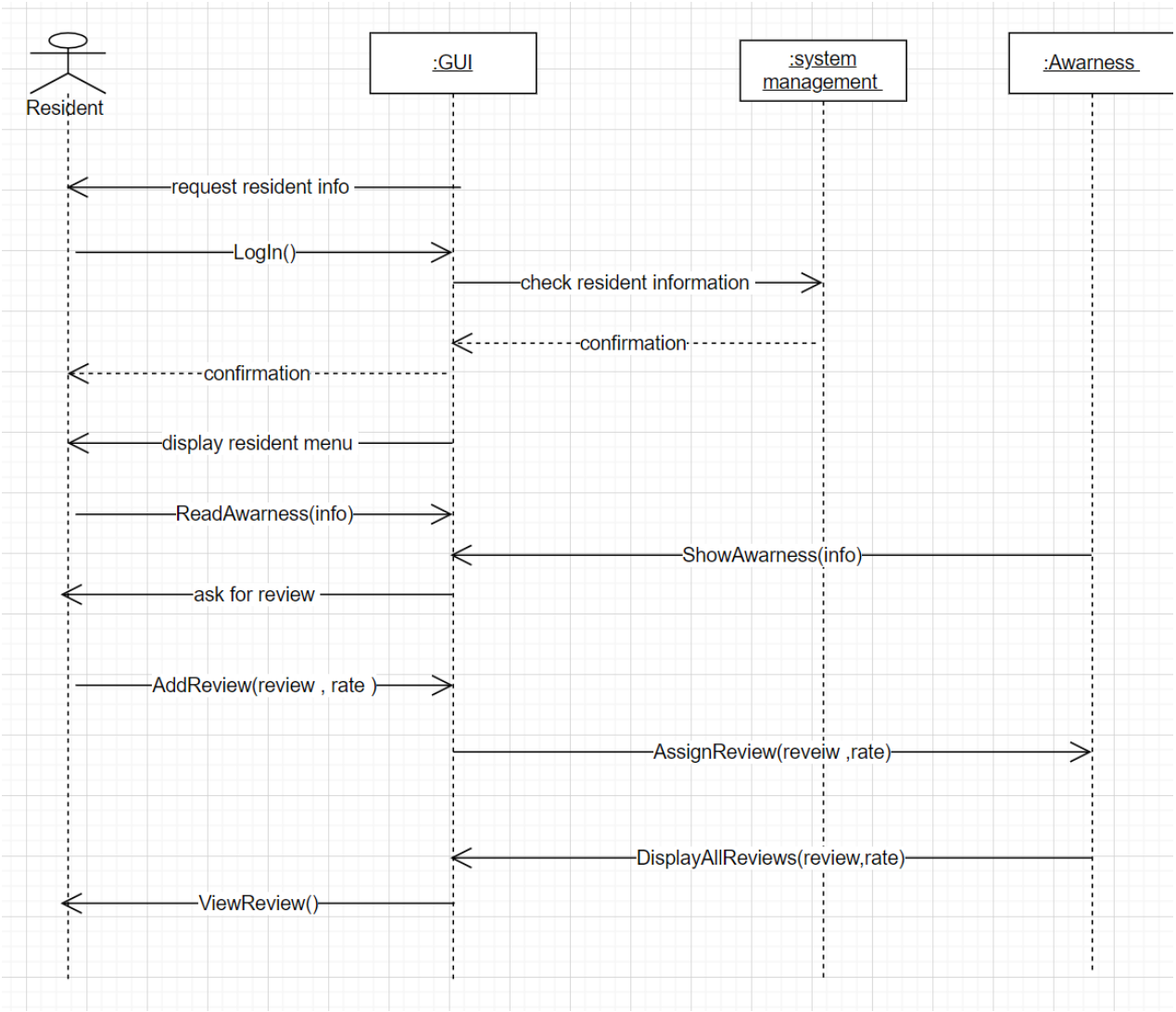
- ***Raise awareness***

- *scenario a: admin interaction*



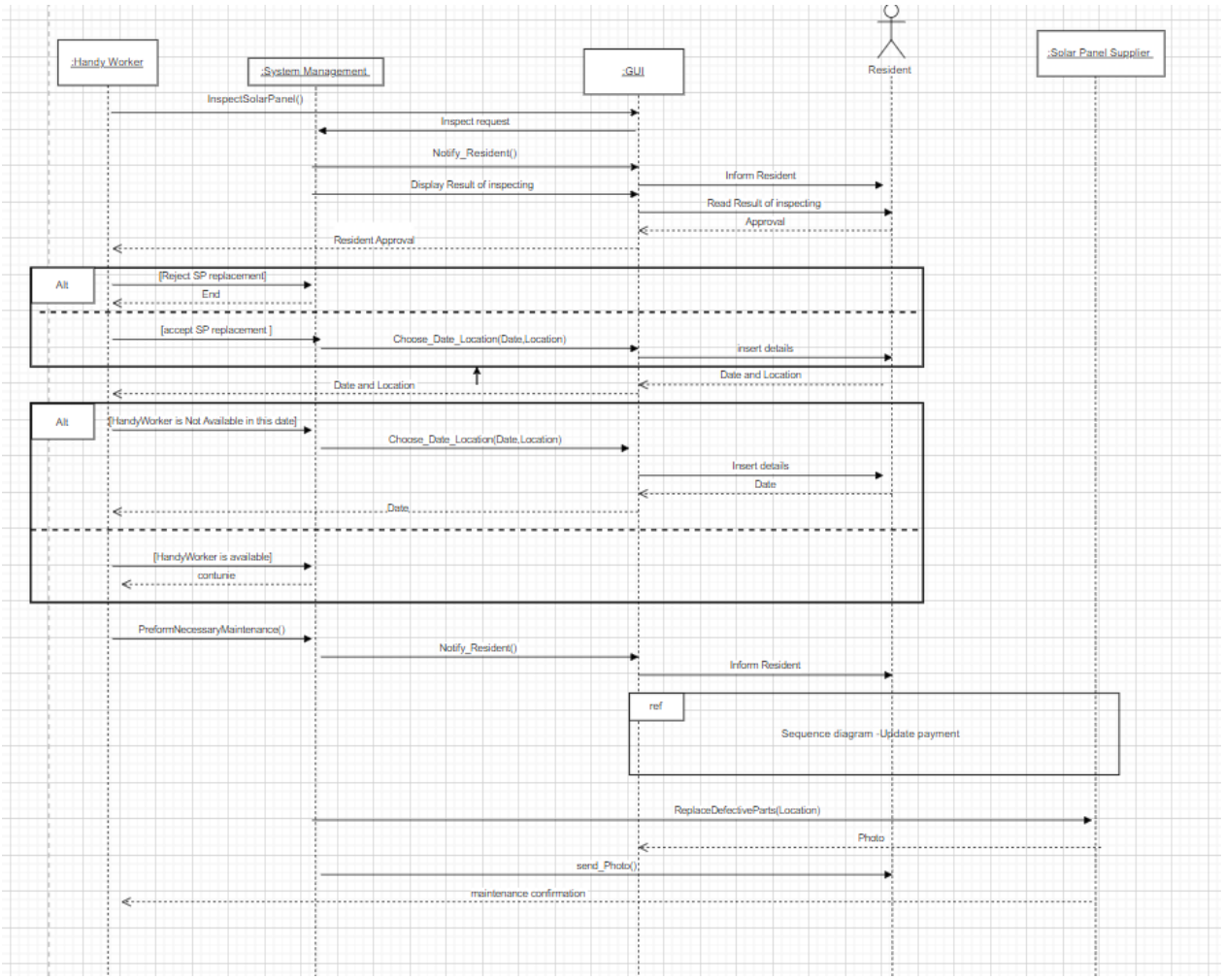
When admin wants to add or update raise awareness section this changes will be through the system management and awareness class the admin will choose Edit Awareness function after he login and enter his ID and password, once the admin details are all approved the system will confirm afterwards admin can do all his changes

- *scenario b: resident interaction*



Resident will have the ability and access to see all raise awareness section through the GUI once he login to the system through GUI, system management class will check his details ID and password and display menu for resident by Read function he can view raise awareness and read all information that was added by admin. Resident can add his review using addreview function through GUI then resident review will be assigned to the system management class also resident can see all past reviews for all users and residents.

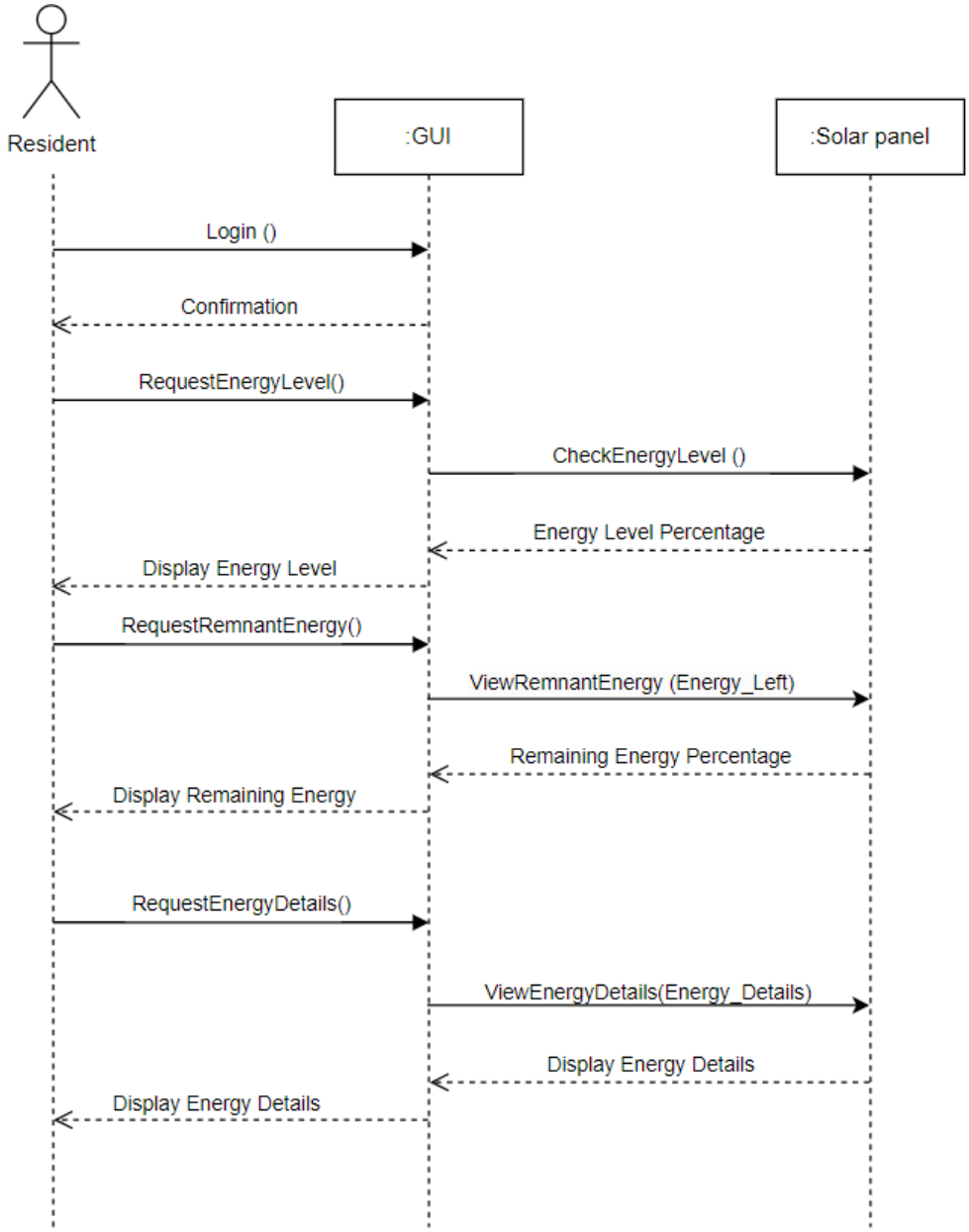
- ***Inspect Solar Panel***



When The HandyWorker Inspect the Solar panels the System Management will notify the resident Then it will send a report holding the result of inspecting . If He/She didn't approve the process will end and the HandyWorker will not do anything . However if He/She approved the The System Management will ask the resident to choose the suitable date for them and to put the location .The HandyWorker will see if he is available at the chosen date .If not The system will ask the resident to choose another date .

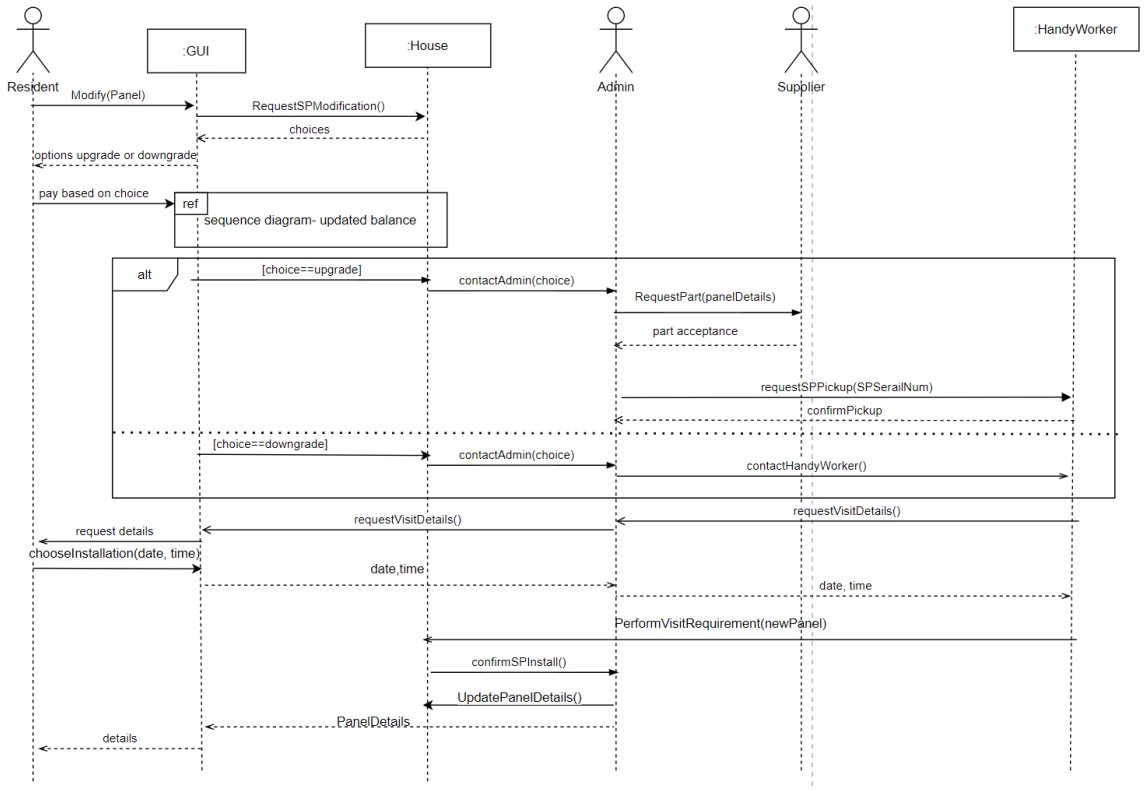
After the date is approved from the HandyWorker the process will continue and the HandyWorker will preform the necessary Maintenance . The System will notify the resident .Then The Payment will be updated So the supplier can Replace the defective parts . The Supplier will then send a photo of the work done to verify the process . Finally the process will be done.

- ***Check energy levels***



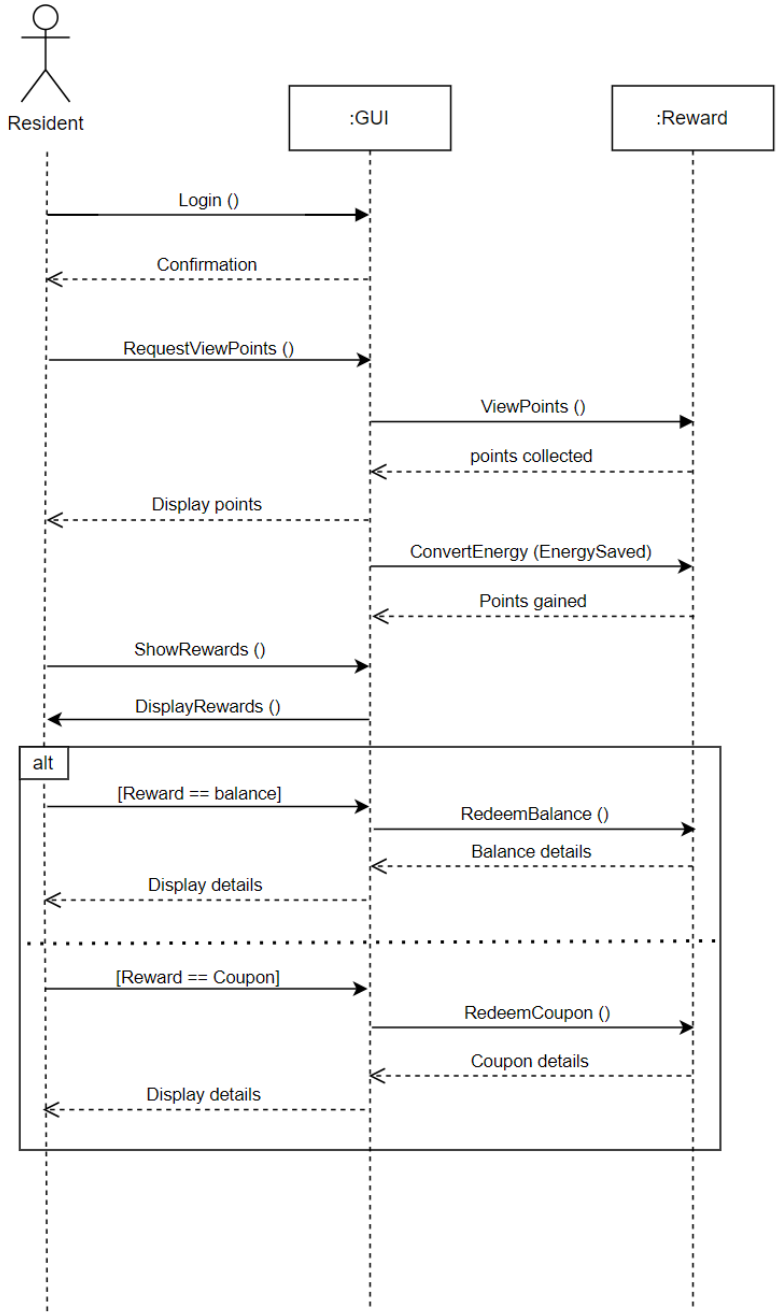
Once the resident signs in, he/she can request to check energy level. Once they check they can decide if they need to buy more energy or to sell any energy. Then the resident can also request to view the remaining energy and where all the energy is being used, as an example it can display that the light switches are consuming the energy the most.

- ***Modify my panel***



First the resident will request modify on the application after that the application will give a chance to the user to choose if he/she wants more power (upgrade) or the panel produce more power than needed (downgrade), after that the GUI will request modifications according to resident choice and the system will process the payment from the user. Regarding the choice for the upgrade the House will notify the admin to inform the supplier if there's any parts needed. Supplier will give the parts, the admin will request the handyman to pick up the parts. Handyman will request the visit requirements from the resident. Resident gives required details (available time, location). Handyman will perform the modification and the details about solar panel will be updated. For the downgrade part it will be the same, the house will inform the handyman for the downgrade, Handy man request details. Handymen perform requirements.

- ***View points***



When an already existing user signs into their account, they can view their points based on the energy they saved. When the resident saves a good amount of energy in one month, they can convert their energy into points. When these points reach the specified amount in the application, the resident can redeem rewards that are coupons for different types of things such as grocery discounts, gym memberships etc.. or he/she can add it as balance into the account since each 10 kWh saved a month is 1 point and 1 point is 0.1 dirhams.

4.0 User Interface Design

A description of the user interface design of the software is presented.

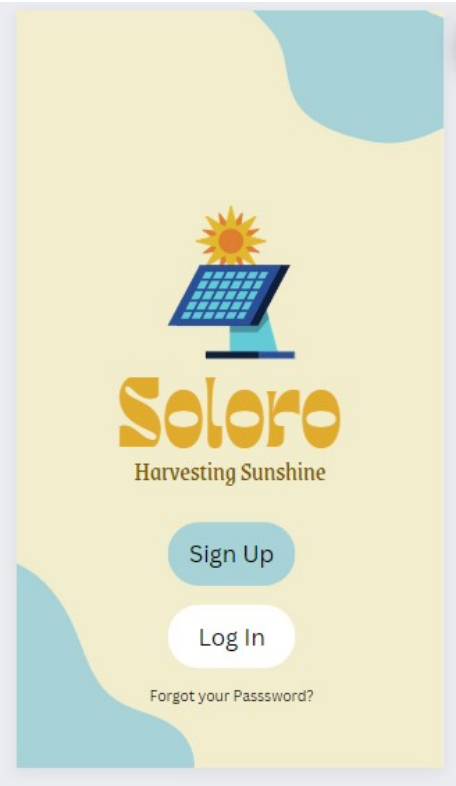
4.1 Description of the user interface

4.1.1 Screen Images

The solar panel application features a straightforward and user-friendly graphical interface, designed for easy navigation without the need for any specialized training. The interface provides a clear overview of the solar panel system's status and performance, making it accessible for users of all levels. With intuitive icons and minimalistic design, users can effortlessly monitor power generation, receive notifications, and adjust settings for optimal solar energy utilization. The application ensures a seamless experience, allowing users to make informed decisions about their solar panel system with simplicity at its core.

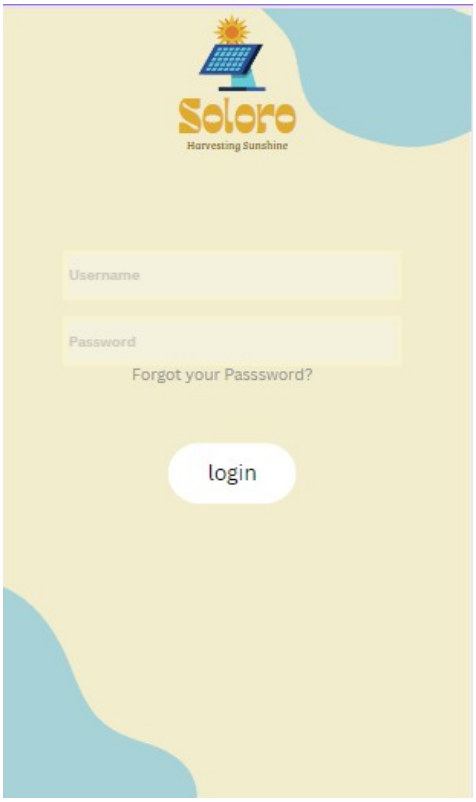
Below are detailed images and description of the GUI. The object and actions performed when you click them are also identified.

The first screen is for logging in or signing up. If you already have an account, choose the log in button, it will take you to the login page where you can enter your username and password. The system management handles the creation of usernames and passwords for security.



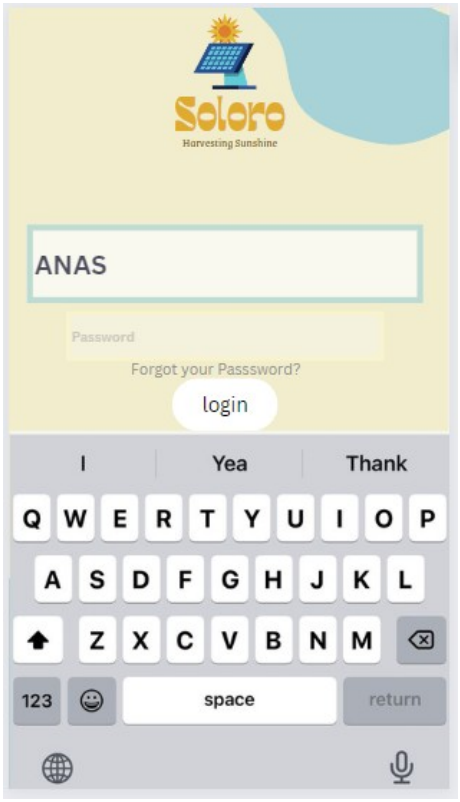
First page when you open the application.

If you already have an account , simply enter your username and password. Afterward, clicking on the login button will securely take you to your account.

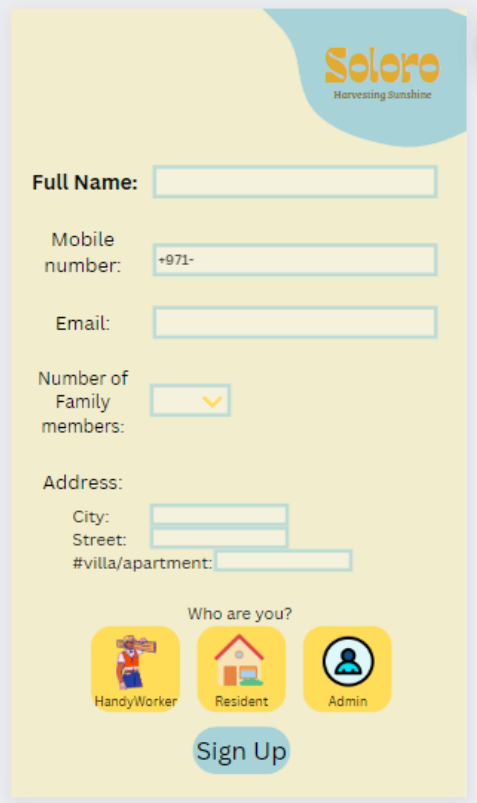


Login page

Enter username.

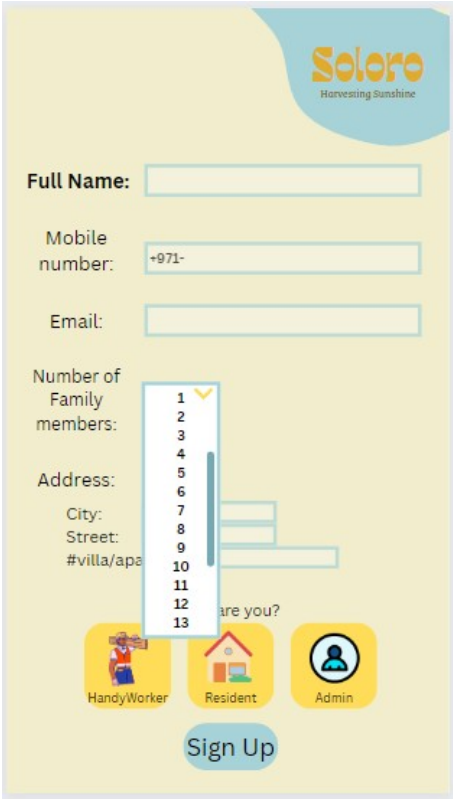


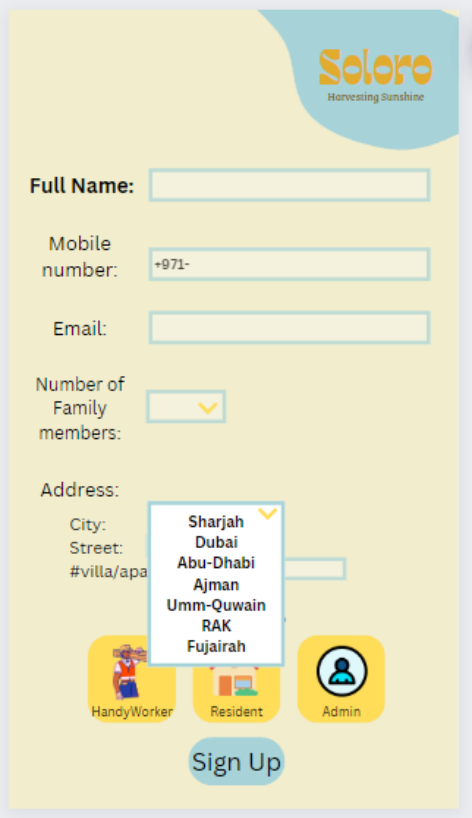
If you do not have an account , fill in your personal details like your full name, address, and gender. Next, choose your user type—handyworker, admin, or resident. After making your selection, you'll be directed to the welcome page.



Sign up details, click on your account type.

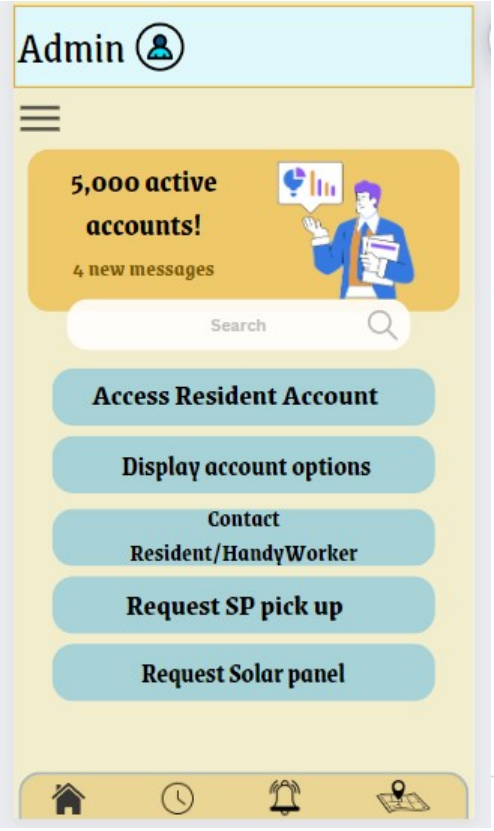
We have an option for number of family members.





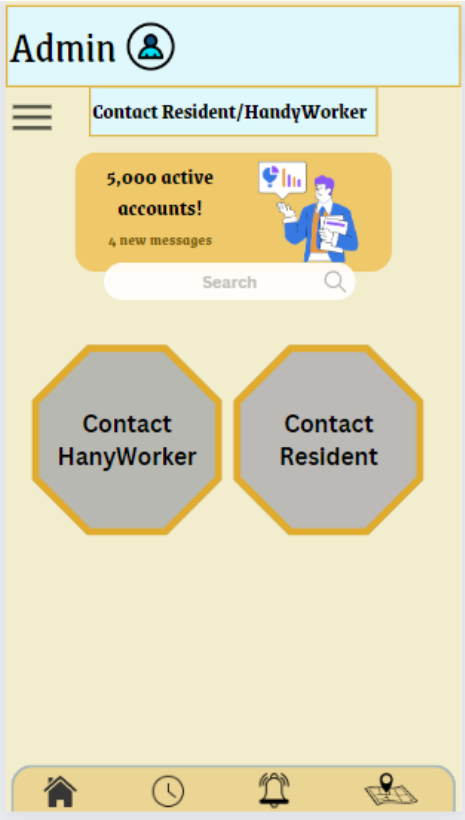
Also, for the cities.

Choosing the admin option takes you to the admin page. Here, you'll see the number of active accounts the admin is handling. The page also lists features like requesting a solar panel, displaying account options, contacting residents or handyworkers, accessing resident accounts, and requesting solar panel pick-up. Additionally, the admin page allows for easy searching of specific data when needed.



Admin's page.

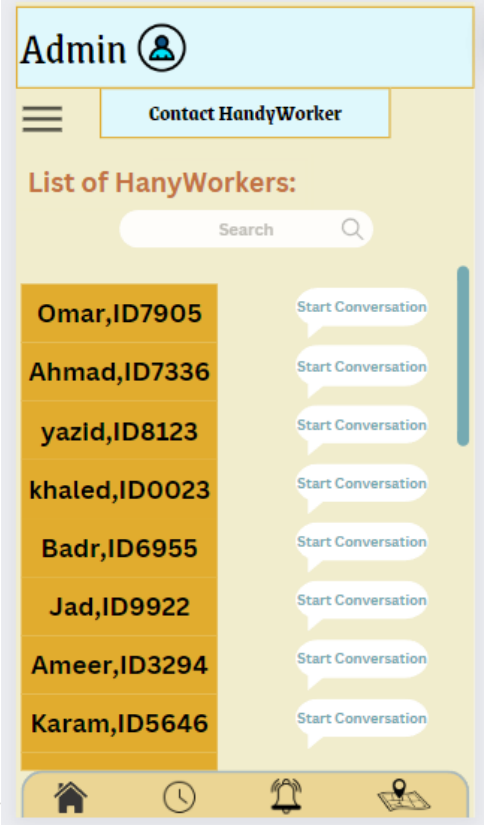
We've also included a "Contact" feature for admins, allowing them to easily get in touch with handyworkers, suppliers, and residents. This function helps the admin gather more information or request specific details effortlessly.



Contact handyworker & resident function

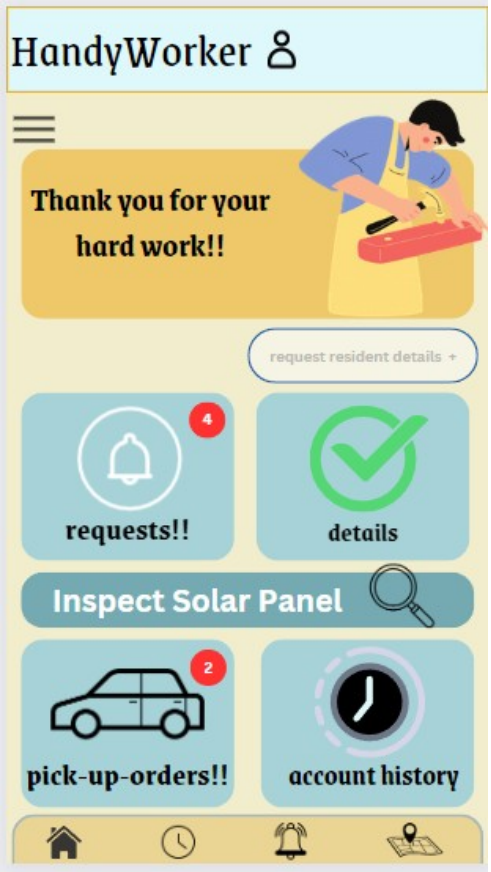
Here's the list of handyworkers available for the admin to chat with.

List of handyworkers to contact



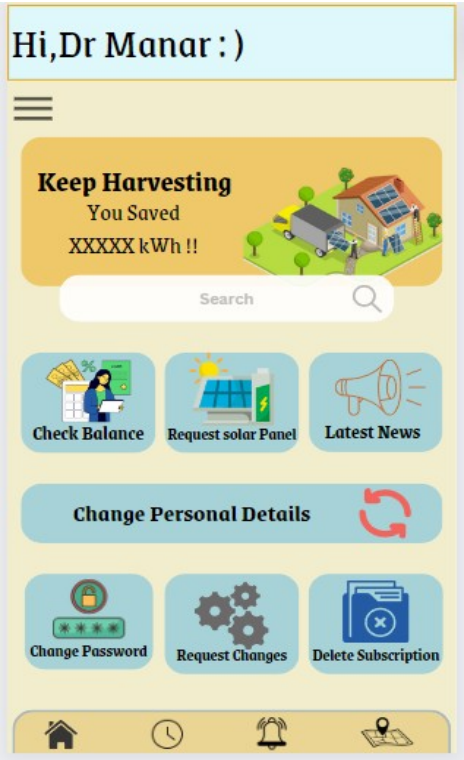
‘

As a handyworker, clicking on the handyworker button leads you to a page displaying all your tasks. This includes the requests you've received, with detailed information about resident requests. You can also see the orders ready for pick-up. The page lets you view account history and easily search for specific solar aspects you might need.



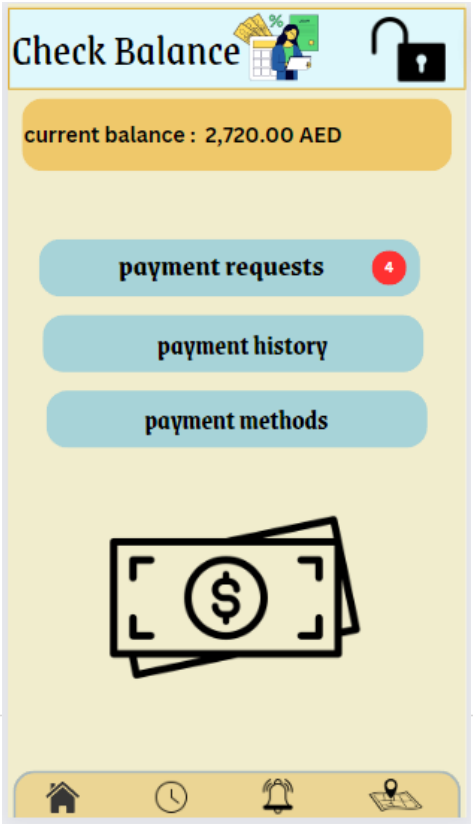
Handyworker’s page.

For residents, clicking on the resident button directs you to a page with various functions. You can check your balance, request a solar panel, stay updated with the latest news, and change personal data, among other features. It's a user-friendly space tailored for residents to manage their solar-related activities conveniently.



Resident’s page

We've added a handy "Check Balance" feature—it lets you see your balance, payment methods, and payment history in a simple and easy-to-use manner.

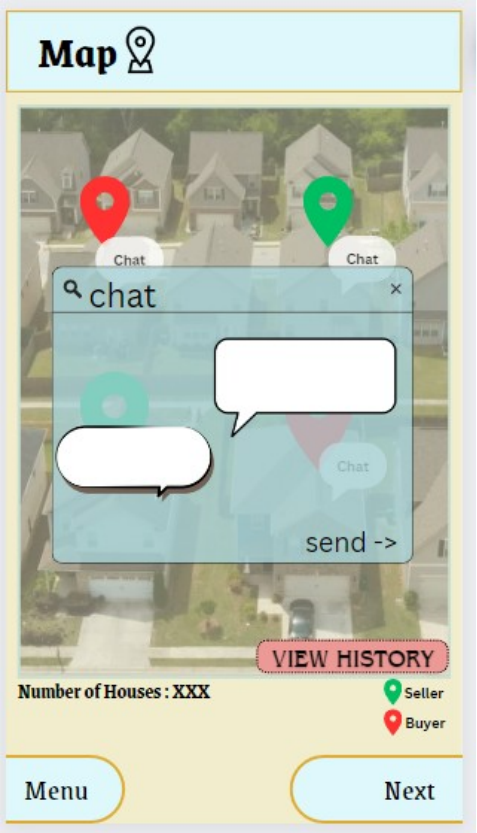


Check balance function

This is the neighborhood map where residents can access and engage in buying and selling activities.

View map

By clicking on the chat icon next to any house on the map, a chat box will pop up, allowing you to communicate with the residents. This feature enables convenient discussions for buying or selling purposes.



Chat box

4.1.2 Objects and actions

The above shown GUI consist of several components.

- A frame That consists of label for displaying text.
- Drop down list in sign up page.
- Pop ups for messages .
- Menu that shows options.
- Command buttons are used for execution of procedures or opening new forms.
- Scroll Bar to scroll through the page when it does not fit into the visible area of a page .
- Chat box . For all type of users .
- A tool bar or key buttons can be used to do several functions.
- Text fields for entering or editing text.

4.2 Interface Design Rules

To improve the usability of an application it is important to have a well designed interface that can be used easily without facing challenges . These “ Nine important Rules of Interface Design" are a guide to create good interaction design.

1) **Continuity:**

Make sure that every interface element has the same design. This covers the general layout, button styles, fonts, and colors. Users can navigate and comprehend the application more easily when there is consistency in their mental image of it.

2) **Streamline**:

Avoid needless complexity and maintain a basic interface. Set important features and information first to avoid overloading consumers. In general, simpler interfaces are easier to use and more intuitive.

3) Comments:

Give users prompt, understandable feedback for their actions. This contains visual cues to verify that an action has been identified, including button push animations or shifting cursor states. Users can better understand how the system reacts to their inputs by using feedback.

4) Error Control and Prevention:

Design with the intention of avoiding mistakes, or at the very least offering precise instructions on how to correct them. Provide detailed error messages together with recommendations for fixing the issue. Reduce the likelihood of user annoyance by helping them avoid possible hazards.

5) Both Visual and Aesthetic Appeal:

A visually pleasing design can improve the entire user experience, even though functionality is still quite important. Employ a unified color scheme, brand-consistent imagery, and design components that elicit a favorable feeling in the viewer.

6) Responsiveness on Mobile:

Make sure the program is responsive to various screen sizes if it is meant to be used on many devices. Provide a layout that is adaptable to different screen sizes so that consumers may enjoy a consistent experience on desktop, tablet, and mobile devices.

7) Clarity of Navigation:

Create a structure for easy navigating. Make sure that the application's various sections are simple for users to go through. To make it easier for people to locate what they need, organize items logically and with clear labels.

8) **Readability and Hierarchy:**

Create an informational structure that is obvious. Employ visual signals to establish hierarchy and draw attention to key components, such as font size, color, and spacing. By selecting suitable fonts and contrast, you can make sure that the text is easy to read.

9) **User Command:**

Give users authority over how they interact. Give people the ability to customize and adjust the UI to suit their tastes. Give consumers explicit options for reversing or redoing actions rather than pressuring them into taking them.

4.3 Components Available

For the GUI in this phase, we used CANVA which is an online graphic design platform. However, in phase 3 we will be using FIGMA to implement our application.

4.4 UIDS description

No user interface development system has been used in the development of the Soloro application.

-

5.0 Restrictions, limitations, and constraints

Special design issues which impact the design or implementation of the software:

1. Ensuring application is responsive and efficient.
2. Having a suitable version for each different mobile platform (iOS and android)
3. Changes in other services that our app relies on will affect the software.
4. Limited funding and resources.
5. Security in processes related to payments.
6. Limited expertise regarding solar panels and renewable energy.
7. The availability of the needed technology and hardware to create our solar panel application and the technology being compatible with our application.
8. Integration with electricity companies.
9. Integration with a solar panel supplier.
10. Maintenance and inspection might be costly.
11. Requesting changing location might be costly.
12. Limited user engagement.
13. Accessing different area maps.
14. Finding employees with expertise in solar panels and their inspection.

6.0 Testing Issues:

6.1 Classes of tests

Test cases are made based on black box testing method, including all its techniques (boundary value analysis, Equivalence partitioning and decision table testing). Noting that white box testing wasn't used, since it requires a full code, which is not applicable yet.

6.2 Expected software response

Name of Technique for BB testing	Restrictions	user	Result	
			Valid cases	Invalid cases
Boundary Value Analysis	Password should be between 8 to 12 characters	Admin/ resident/ handyworker	8: "Admin231" 9: "cop28yay!" 11:"Manegedmin2" 12:"123456789098"	7: "ceo8491" 13:"Ghr65admin324"
Equivalence partitioning	ID should be between 6 to 9 characters	Resident	7: "res4289" 9: "villares2"	4: "kqt2" 1: "a"
Boundary Value Analysis	Resident can't access admin domain	Resident	Resident can view admins	Resident try to change admin details
Boundary Value Analysis	admin can't access resident domain	admin	Admin can deactivate resident account only	Admin try to delete resident account
Equivalence partitioning	Handy worker can only perform one task at a time.	Handy worker	One resident wants to get solar panel for his neighbor	More that one residents call the same handy worker for shifting a solar panel

Decision Table Testing:

condition	Rule 1	Rule 2	Rule 3
Admin	1	0	0
Handy worker	0	1	0
Resident	0	0	1
Action	Admin	Handyworker	Resident
Admin account	Valid access	No access	No access
Resident account	Valid access	No access	Valid access
Handy worker task	No access	Valid access	No access
Solar panel modification	Yes	Yes	Yes
Doing critical changes in system	Yes	No	No
Entering data into system	Yes	Yes	Yes

6.3 Performance bounds

1. Temperature tolerance bound: As solar panel need a specific temperature to be preforming with its full power, this temperature should be too low or too high as very high temperature may cause damage to our solar panel system, also very low temperature will not be enough to preform and work.
2. Maintenance bound solar panel require cconsecutive check-up in order to maintain an optimal performance also solar panel will need cleaning services.
3. Safety shutdown: performance bound may require the solar panel to have a safety action during the emergencies situations. these bound could define the maximum time it takes for the solar panel system to shutdown and stop working completely.

6.4 identification of critical components

Critical components could include payments and transactions that happen in residents account, as it depend on many things like type of house, number of solar panels and amount of energy needed.

Table of members contribution:

Member name	Contributions
Asil	1. Full creation of sequence diagram and drawing on draw io, with the needed description in report: - Modify my panel 2. Full creation of statechart diagram and drawing on draw io, with the needed description in report: - Supplier - Admin 3. Designing the GUI with the needed description in report (section 4)
Mustafa	1. Full creation of sequence diagram and drawing on draw io, with the needed description in report: - Check energy level 2. Full creation of statechart diagrams and drawing on draw io, with the needed description in report: - Resident 3. Section 2 in report 4. Creating ppt for presentation
Rena	1. Full creation of sequence diagram and drawing on smart draw, with the needed description in report: - View points 2. Full creation of statechart diagrams and drawing on draw io, with the needed description in report: - Reward - Awareness 3. Section 2 in report 4. Creating ppt for presentation
Renad	1. Full creation of sequence diagrams and drawing on draw io, with the needed description in report: - Start payment - Generate suitable panel - Update balance 2. Updating the class diagram 3. Full creation of statechart diagrams and drawing on draw io, with the needed description in report: - Payment - HandyWorker 4. Software Architecture style (section 3), section 5
Salam	1. Full creation of sequence diagrams and drawing on draw io, with the needed description in report: - Inspect solar panel 2. Full creation of statechart diagrams and drawing on io, with the needed description in report: - Account - System Management

	3. Designing the GUI with the needed description in report (section 4)
Sulaf	1. Full creation of sequence diagram and drawing on draw io, with the needed description in report: - View map 2. Updating the class diagram 3. Full creation of statechart diagram and drawing on draw io, with the needed description in report: - Map
Yasmin	1. Full creation of sequence diagram and drawing on draw io, with the needed description in report: - Raise awareness 2. Full creation of statechart diagram and drawing on draw io, with the needed description in report: - House - SolarPanel 3. Section 6 in report