# CHAPTER THREE

**PROPOSED SYSTEM**

**3.1. Overview**

The proposed system aims to solve the situation that is currently intact, which is the system in which we use to read meters in the city and around the country. It aims to improve and make the existing system easier.

**3.2. Functional Requirement**

* **Reader will interpret the data**

The metering reader will read and interpret the data on the meter to a comprehensible figure.

* **Enable Two way communication**

The main objective of AMI is to enable two way communications between smart energy meter and the electric utility providers.

* **Handle power outage and maintenance**

If any inconvenience occurs like power outage energy surge, it will notify the service providers for maintenance.

* **Customers can look at their energy consumption**

The customers will be able to look at the amount of electricity they used or consumed.

* **Customers can pay their bill**

The customers will be able to pay for the amount of energy they used monthly.

**3.3. Non-Functional Requirement**

They define *how* the system should perform and specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

**3.3.1. Usability**

The system should be user friendly and how effectively they can learn and use a system.

**3.3.2. Reliability**

Reliability is the probability and percentage of the software performing without failure for a specific number of uses or amount of time.

* **Percentage of the probability of failure:** You can check the percentage of the probability of failure, or failure rate, to determine the reliability of a system. If the percentage is higher, the system is likely to function normally after substantial use.
* **Number of critical failures:** Consider recording the amount of critical failures a system experiences during testing to check its reliability. If the number of failures is low, it means that the system operates properly.
* **Time between critical failures:** Tracking the time between critical failures can help you understand the reliability of a system. When critical failures occur rarely, it means that a system functions normally most of the time.

**3.3.3. Maintainability**

This indicates the average time and ease and rapidity with which a system can be restored after a failure.

The ease with which our software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.

**3.3.3 Performance**

System performance is the most important quality in non-functional requirements and affects almost all the other preceding ones.

Furthermore, reliability, availability, and maintainability features fall exclusively under these requirements.

System performance defines how fast a system can respond to a particular user’s action under a certain workload.

**3.3.4. Security**

Security measures ensure your software’s safety against espionage or sabotage.

These features are necessary even for stand-alone systems; you don’t want anyone to have access to your sensitive data.

**3.3.5. Error Handling** **and Extreme Conditions**

Error handling refers to the routines in a program that respond to abnormal input or conditions. The quality of such routines is based on the clarity of the error messages and the options given to users for resolving the problem**.**

# 3.4. System Model

**3.3.1. Use Case Scenarios**

**Scenario 1**

**Name of the scenario:** Register to the system

**Participating instance actor:** Customers

**Entry condition:**

* User must have valid email address and phone number
* User should be above 18 years of age

**Flow of events:**

* User browse to web app or open mobile app
* User goes to register page
* Fill all the required form
* Read and accept terms and conditions of the system
* Submit the input

**Alternative condition:** If the input is wrong the system will generate error message and let the user to refill forms

**Exit condition:** The system saves the entered data into database

**Scenario 2**

**Name of the scenario:** Login to the system

**Participating instance actor:** Every user

**Entry condition:**

* The user should have an account
* The system should be installed

**Flow of events:**

* User opens the web page
* User inserts ID and password
* User submits input

**Alternative Condition:**

* If the input is wrong the system will generate error message and let the user refill the form.

**Exit condition:**

* The user will be logged in to the system

**Scenario 3**

**Name of the scenario:** Resetting password

**Participating instance actor:** Every user

**Entry condition:**

* User should have account that is previously logged into the system
* User forgot or lost his/her password

**Flow of events:**

* User should go to the password recovery form
* User should insert their ID to recover password
* System sends recovery link
* User should use the link given to reset the password

**Alternative Condition:**

* If user enters invalid recovery email system will generate error or show message that states invalid ID.

**Exit condition:**

* User uses new password to login the system.

**Scenario 4**

**Name of the scenario:** Update profile

**Participating instance actor:** Users

**Entry condition:**

* Open the web page

**Flow of events:**

* Login to the system
* Go to profile section
* Display full profile information
* Selected edit profile
* Edit required form and update information
* Click ‘update profile to update
* Show updated profile

**Alternative Condition:**

* If the user fills the form incorrectly then the system will generate error message.

**Exit condition:**

* Save updated information to database

**Scenario 5**

**Name of the scenario:** Create technicians account

**Participating instance actor:** Admin

**Entry condition:**

* Admin must browse the web page

**Flow of events:**

* Admin login into their account
* Admin clicks ‘manage accounts’ link and browse manage accounts page
* Admin selects ‘create new technician’ from the options in that page
* Admin fills the form in create new technician page
* Admin submits form

**Alternative Condition:**

* If the input is wrong the system will generate error message and let the admin refill the form

**Exit condition:** New technician is saved to the database

**Scenario 6**

**Name of the scenario:** Delete Technician account

**Participating instance actor:** Admin

**Entry condition:**

* Admin must navigate the web page

**Flow of events:**

* Admin login into his/her account
* Admin clicks ‘manage accounts’ link and browse manage accounts page
* Admin selects ‘Delete Technician’ link
* Admin choose the Technician from dropdown list
* Admin confirms the pop-up to delete Technician

**Exit condition:** Technician deleted from database

**Scenario 7**

**Name of scenario:** Suspend Users

**Participating instance actor:** Technician

**Entry condition:**

* Has valid technician username and password
* Navigate through our web page

**Flow of events:**

* Login to the system
* The system displays main page
* Technician clicks “User management” button
* Technician searches a user from users list
* Select a specific user
* Technician click “Suspend” button

**Exit condition:** Update user’s database table.

**Scenario 8**

**Name of scenario:** MakePayment

**Participating instance actor:** User

**Entry condition:** User must have the system and web page

**Flow of events:**

* Login into the system
* The system displays the main page
* User enters serial number
* User clicks ‘Make Payment’
* Confirms Payment by displaying a ‘Payment Complete’ pop-up

**Alternative condition:** If user does not pay the required among the system will generate and display ‘Payment NOT Completed’

**Exit condition:** Payment will be done

**Scenario 9**

**Name of scenario:** Enter Serial Number

**Participating instance actor:** User

**Entry condition:**

* User must have the system and web page

**Flow of events:**

* Login to the system
* The system displays main page
* Click on’ Enter Serial Number’ button
* Enters the serial number given

**Alternative condition:** If user does not provide the correct serial number among the system will generate and display

**Exit condition:** Correct serial number entered; It continues to the payment page.

**Scenario 10**

**Name of scenario:** Logout of the system

**Participating instance actor:** User and Technician

**Entry condition:**

* User must have an account

**Flow of events:**

* User clicks on ‘Log Out’ button
* Confirms the action

**Exit condition:** Logout of the account

**Scenario 11**

**Name of scenario:** Update monthly bills

**Participating instance actor:** Technician

**Entry condition:**

* Has valid technician username and password
* Navigate through our web page
* Proper authorization

**Flow of events:**

* Login into admin page
* View energy consumption amount
* Click on ‘Payment amount’ button
* Enter the amount of bill that needs to be paid
* Confirm that the payment has been completed

**Alternative condition:** If payment has not been done send a notice that the payment has not been completed

**Exit condition:** Monthly payment update done.

**Scenario 12**

**Name of scenario:**  Maintenance issue

**Participating instance actor:** Users

**Entry condition:**

* A customer with an account
* User with the web page

**Flow of events:**

* Login into their account
* Click on ‘State The Issue ’
* Confirm with ‘Message Sent For Maintenance’

**Alternative condition:** If feedback has not been sent an error message will be generated.

**Exit condition:** Issue concerning maintenance will be sent to the technician.

**Scenario 13**

**Name of scenario:**  Provide feedback for maintenance issues

**Participating instance actor:** Technician

**Entry condition:**

* A Technician with an account
* Must have the web page

**Flow of events:**

* Login into their account
* Click on ‘Provide Feedback’
* Confirm with ‘Feedback Sent’

**Alternative condition:** If feedback has not been sent an error message will be generated.

**Exit condition:** Issues will be fixed immediately.

**3.5. Use case Model**

**3.5.1 Actor Identification**

**Name:** Customer

**Description:** The person who uses the system for electricity

**Role:** pay bills

**Name:** Technician

**Description:** The person who operate the system

**Role:** make feedback and control the users

**Name:** Admin

**Description:** The person who is behind the system

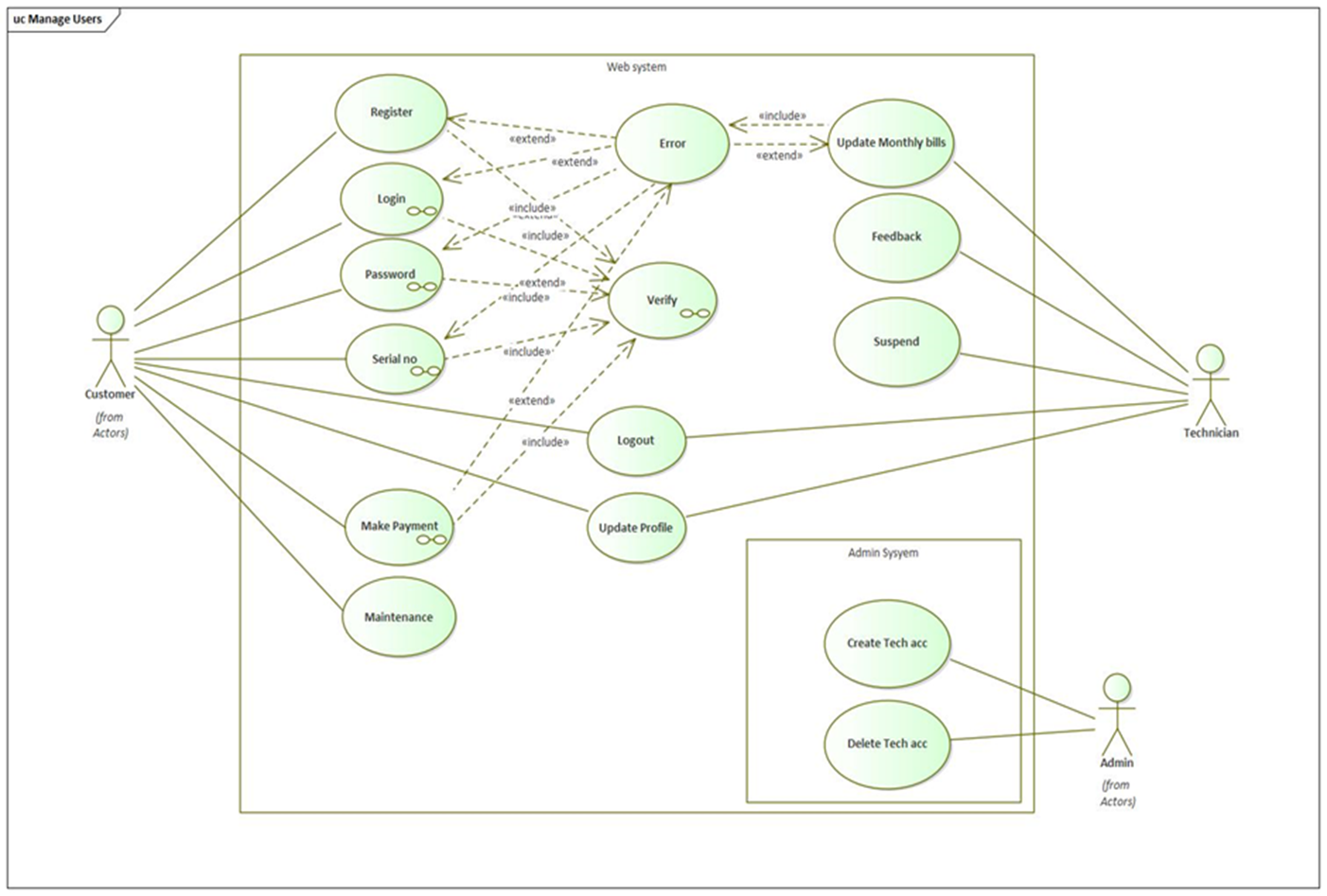
**Role:** controls and keeps track of the system.

**3.5.2. Use Case Identification**

Our system includes the following use cases:

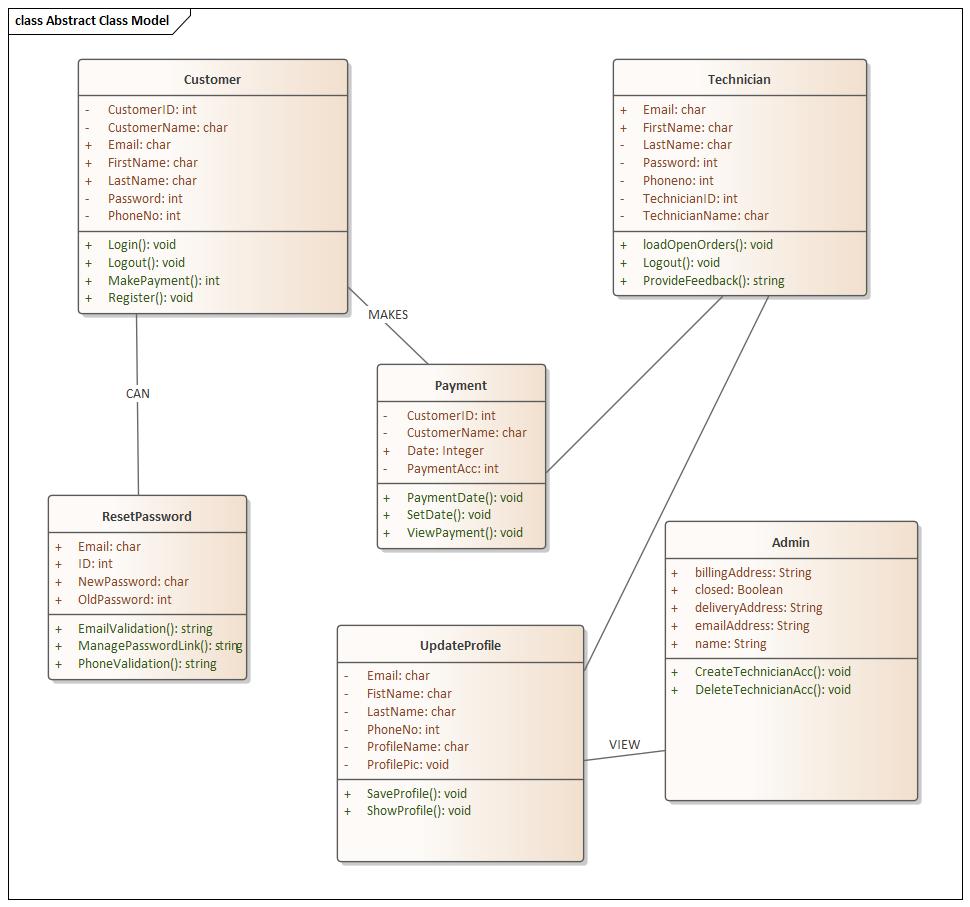
* Register to the system
* Login to the system
* Resetting password
* Update profile
* Create technicians account
* Delete Technician account
* Suspend Users
* MakePayment
* Enter Serial Number
* Logout of the system
* Update monthly bills
* Maintenance issue
* Provide feedback for maintenance issues

**3.5.3. Use Case Diagram**

****

**3.5.4. Class Diagram**

The Class diagram captures the logical structure of the system; the classes and things that make  
up the model. It is a static model, describing what exists and what attributes and behavior it has,  
rather than how something is done. Class diagrams are most useful to illustrate relationships  
between classes and interfaces.

****

**3.6. Dynamic model  
 3.6.1.Sequence Diagram**A sequence diagram in a UML is a kind of interaction diagram that shows how processes operate  
with one another and in what order. A sequence diagram shows object interactions arranged in  
time sequence. It depicts the objects and classes involved in the scenario and the sequence of  
messages exchanged between the objects needed to carry out the functionality of the scenario. It  
shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously,  
and, as horizontal arrows, the messages exchanged between them, in the order in which they  
occur. This allows the specification of simple runtime scenarios in a graphical manner.

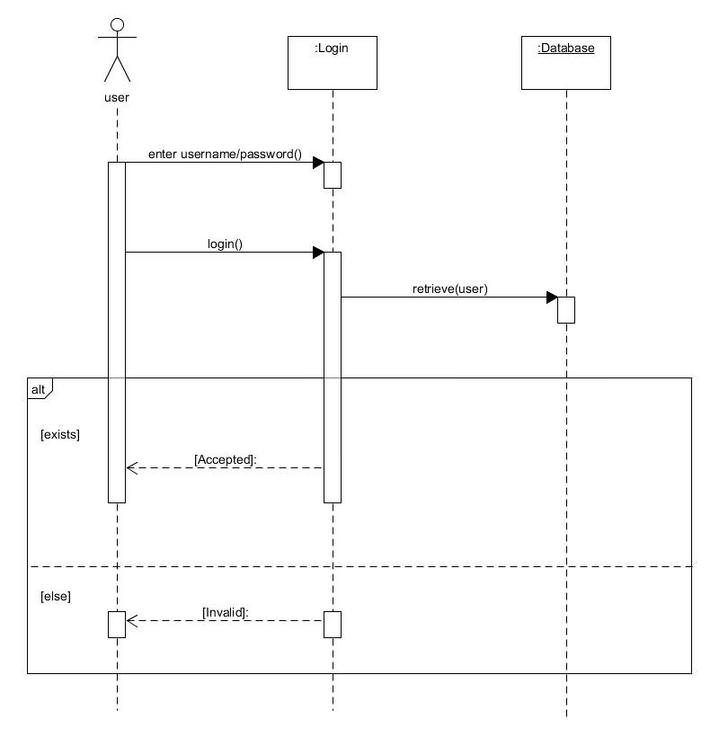
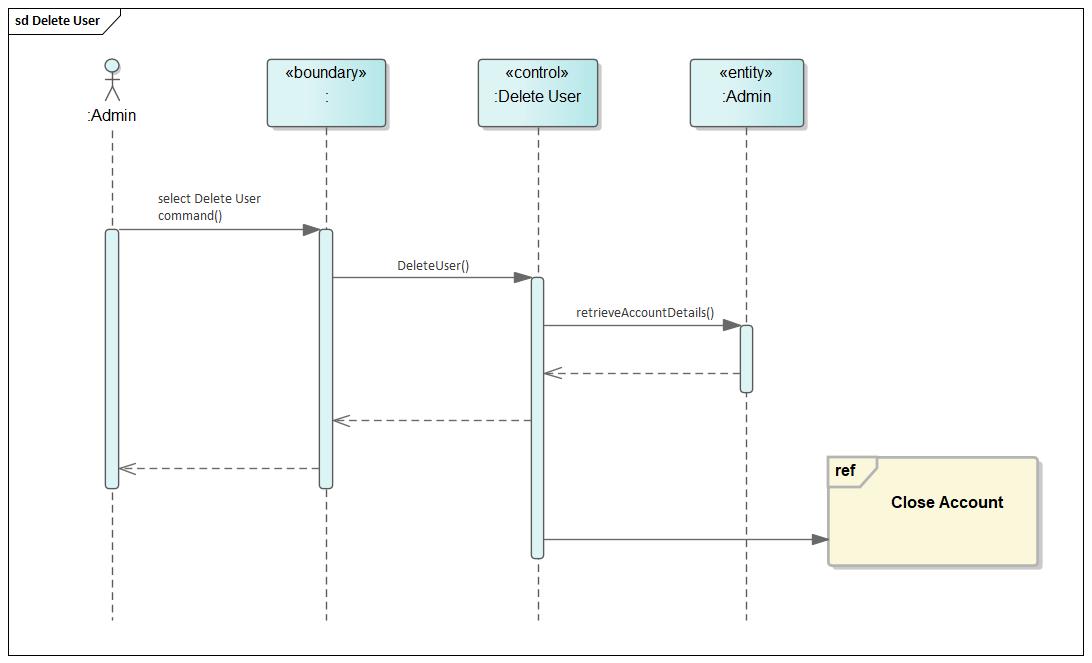
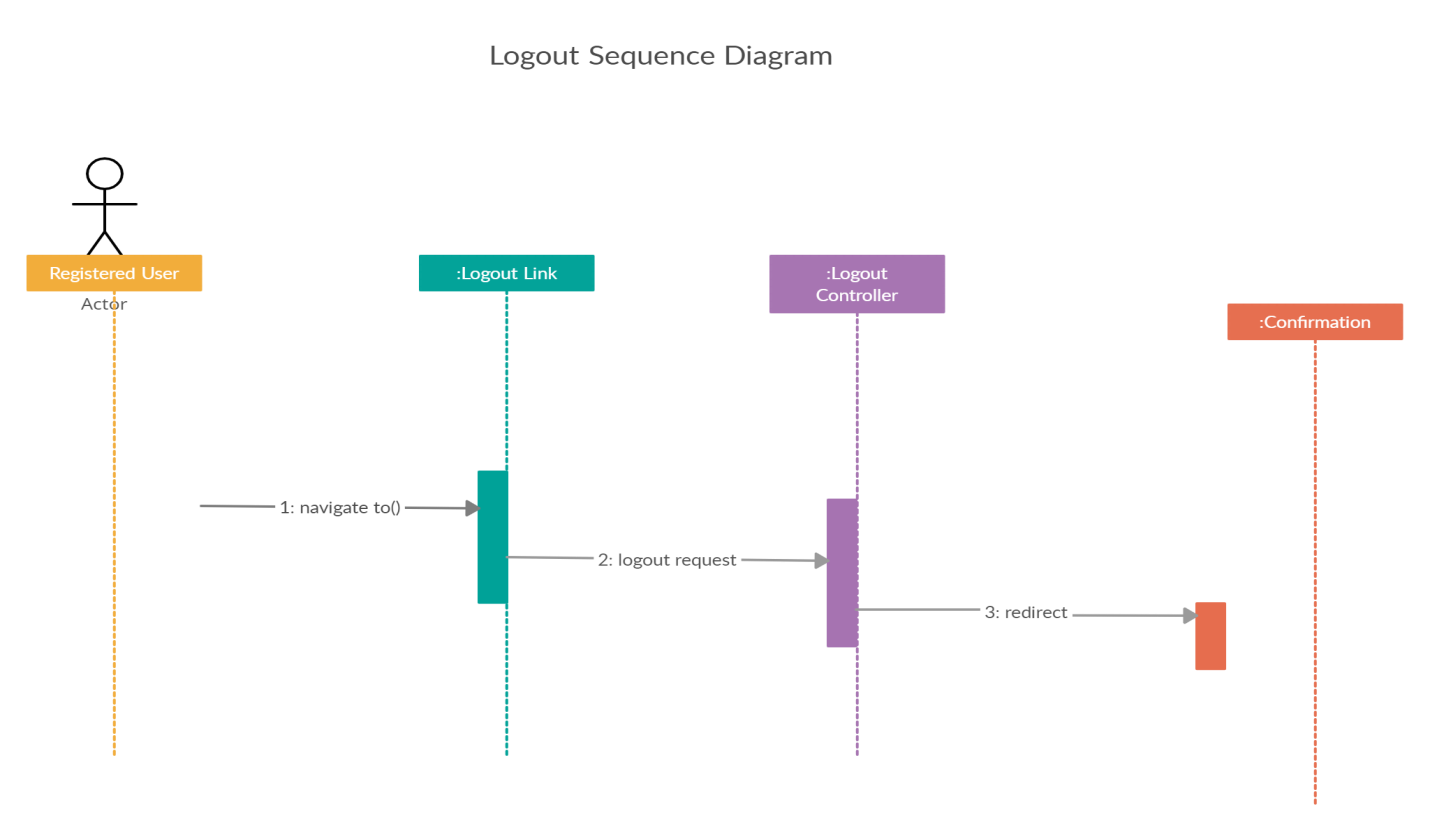
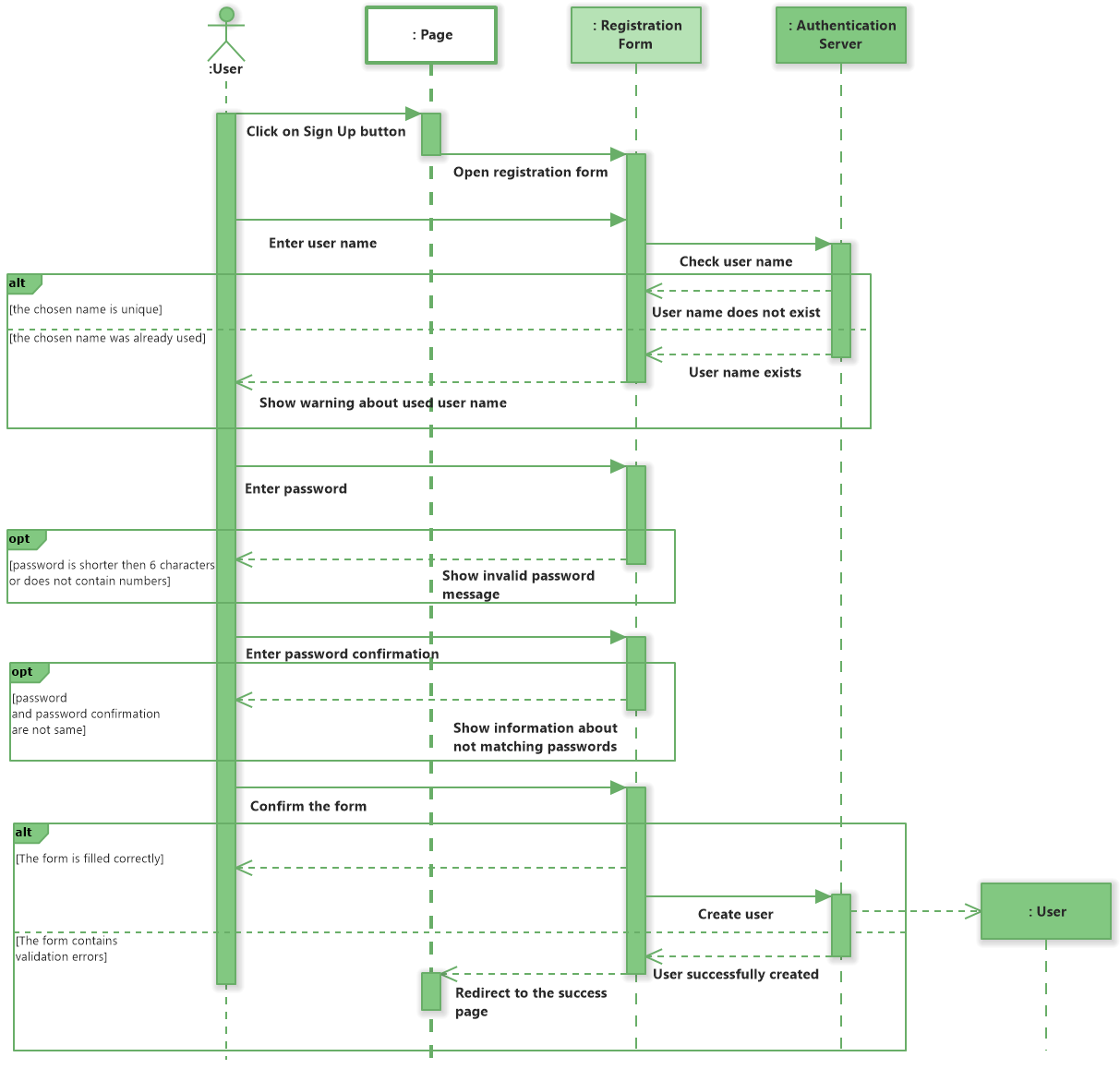
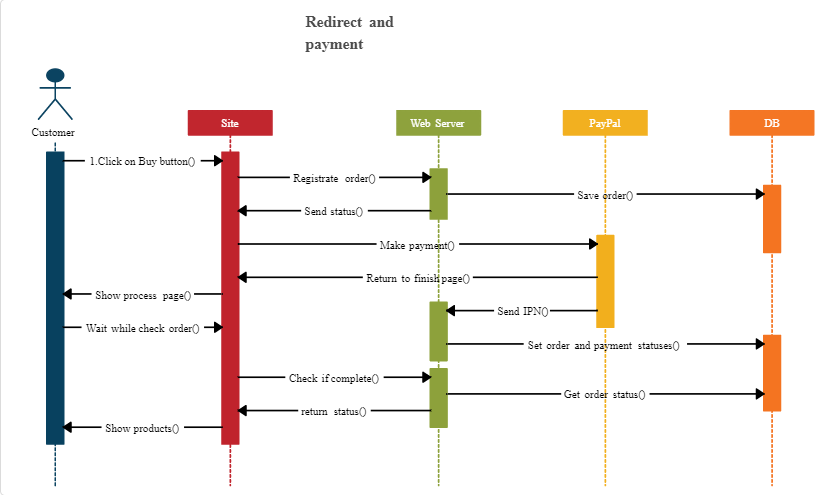
****

Fig: Logging in sequence

 Fig: Deleting account sequence

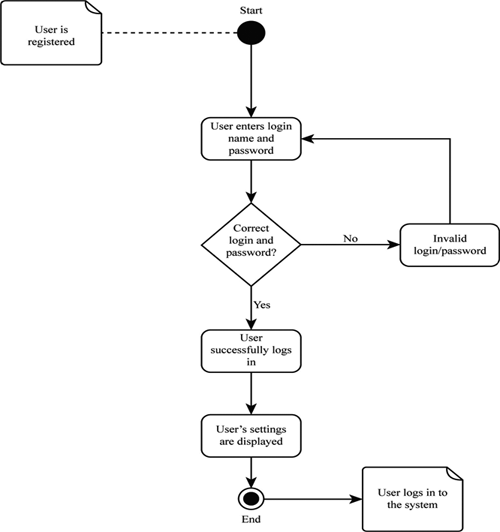
 Fig: Registration sequence

h

**Money Banking**

Fig above: payment sequence

**3.6.2. Activity Diagram**

 Login Activity Diagram

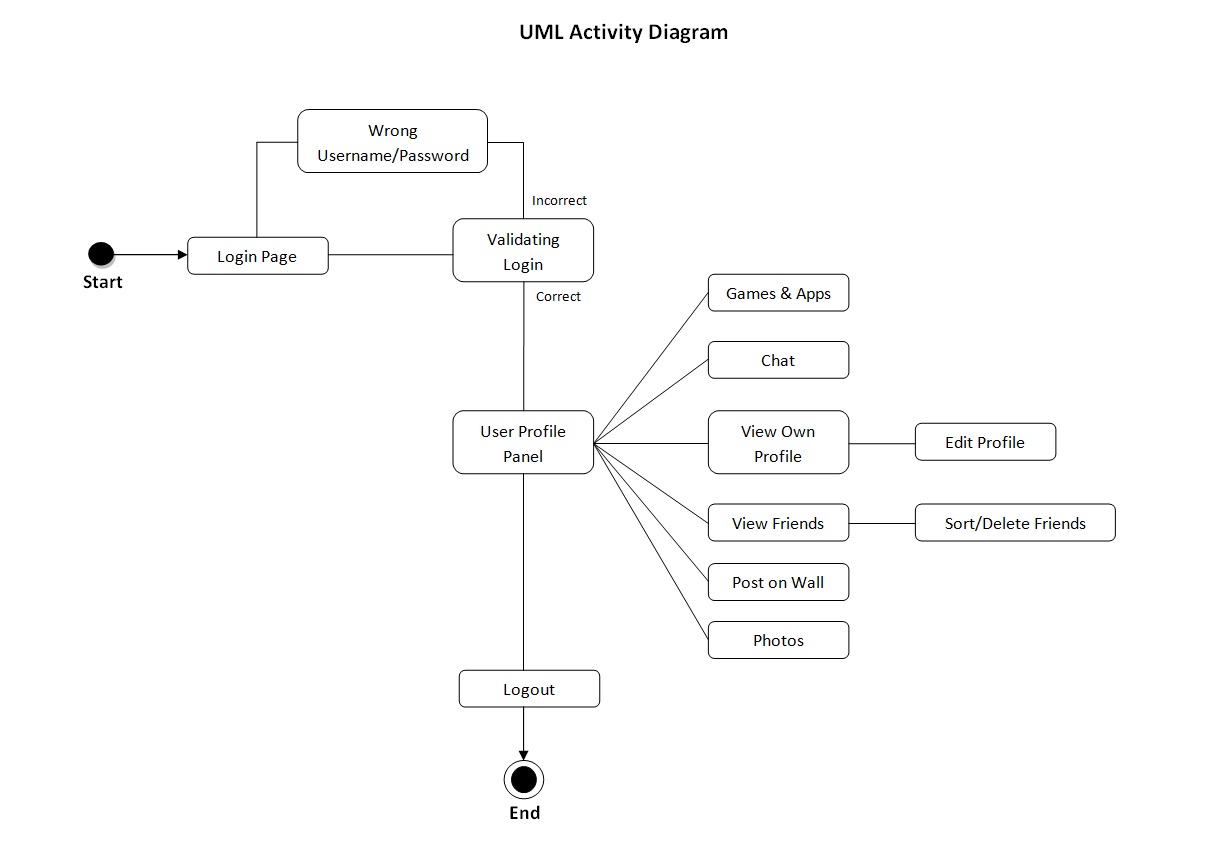
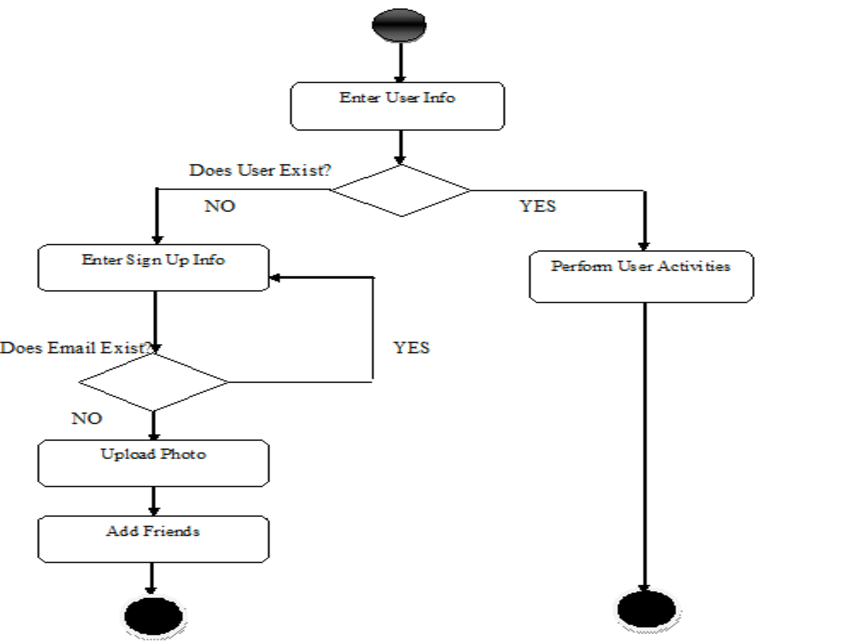
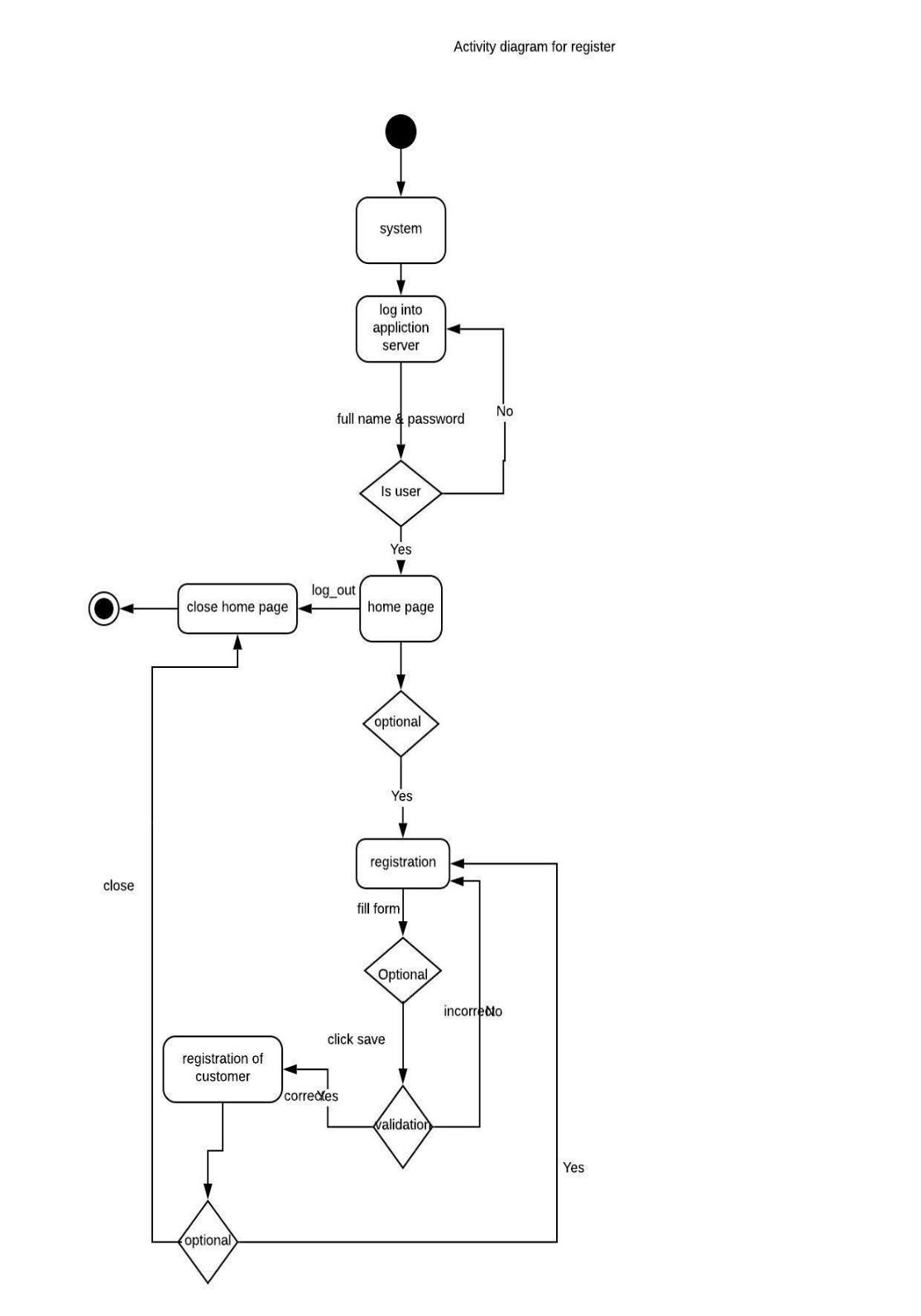


Fig: Logout Activity Diagram

 Fig: Update Profile

Activity Diagram for Registration

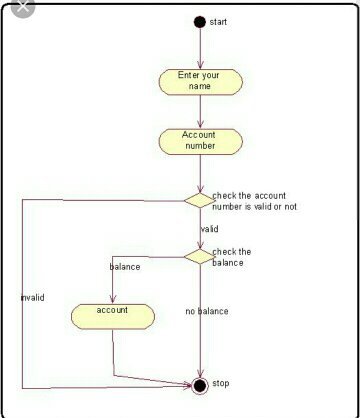
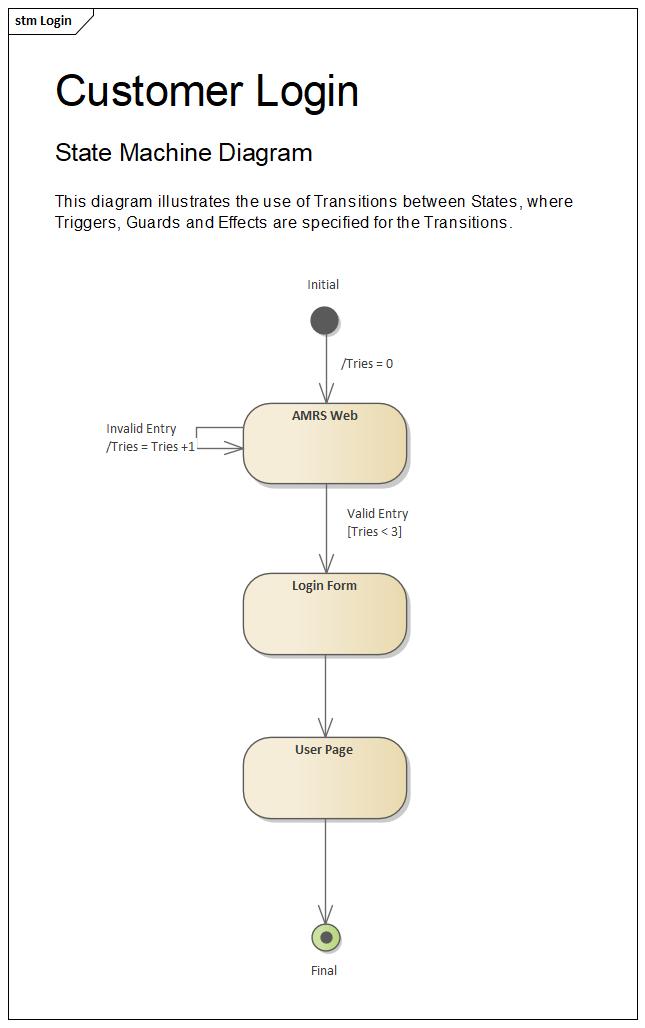


Fig: Payment Activity Diagram

**3.6.3. State Diagram**State diagrams are used to give an abstract description of the behavior of a system. This behavior  
is analyzed and represented in series of events that could occur in one or more possible states.

**Fig: LOGIN**

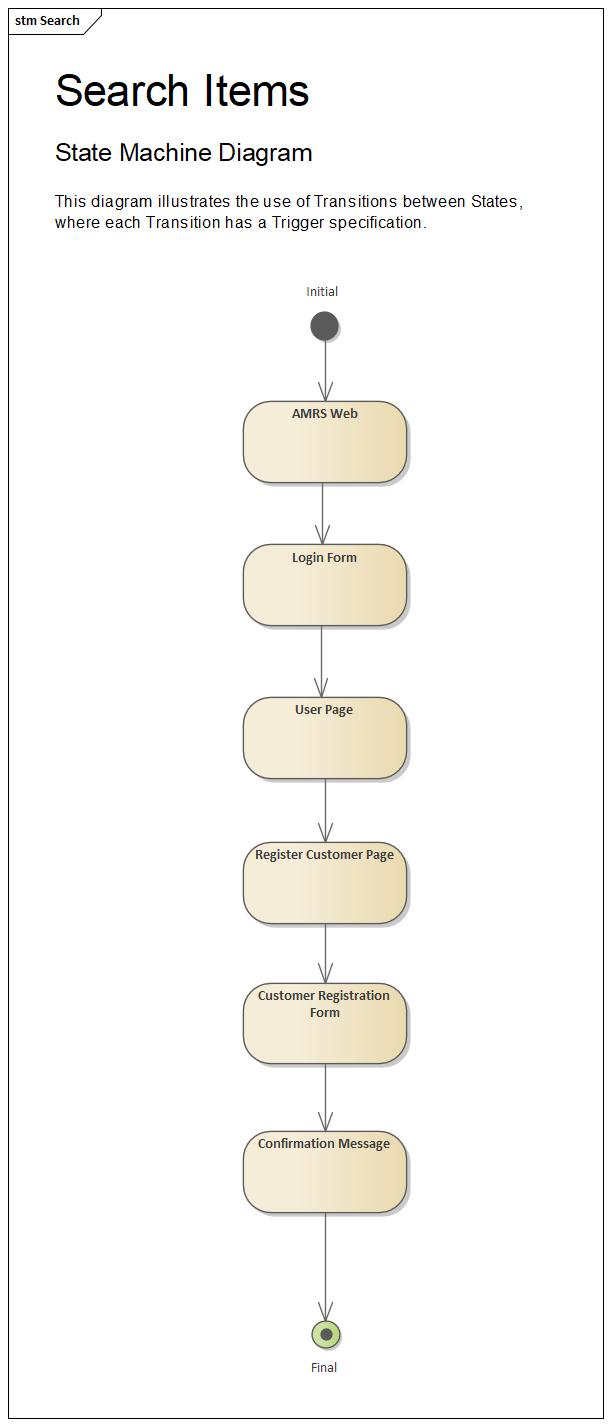
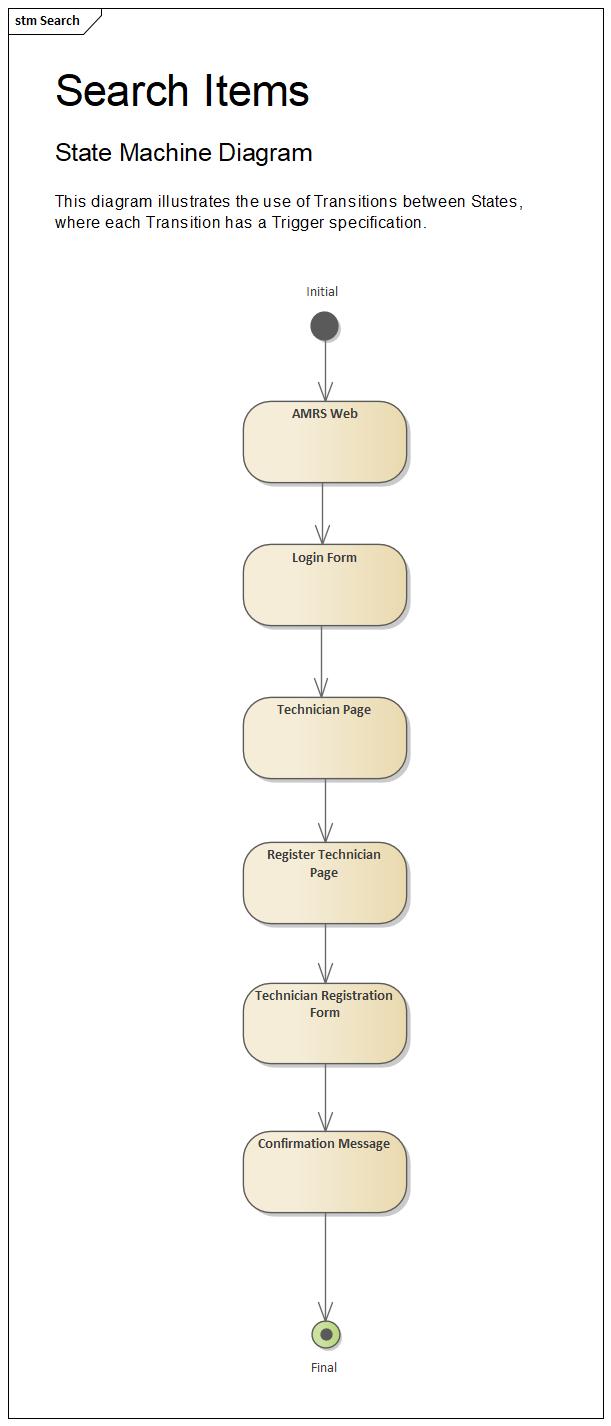
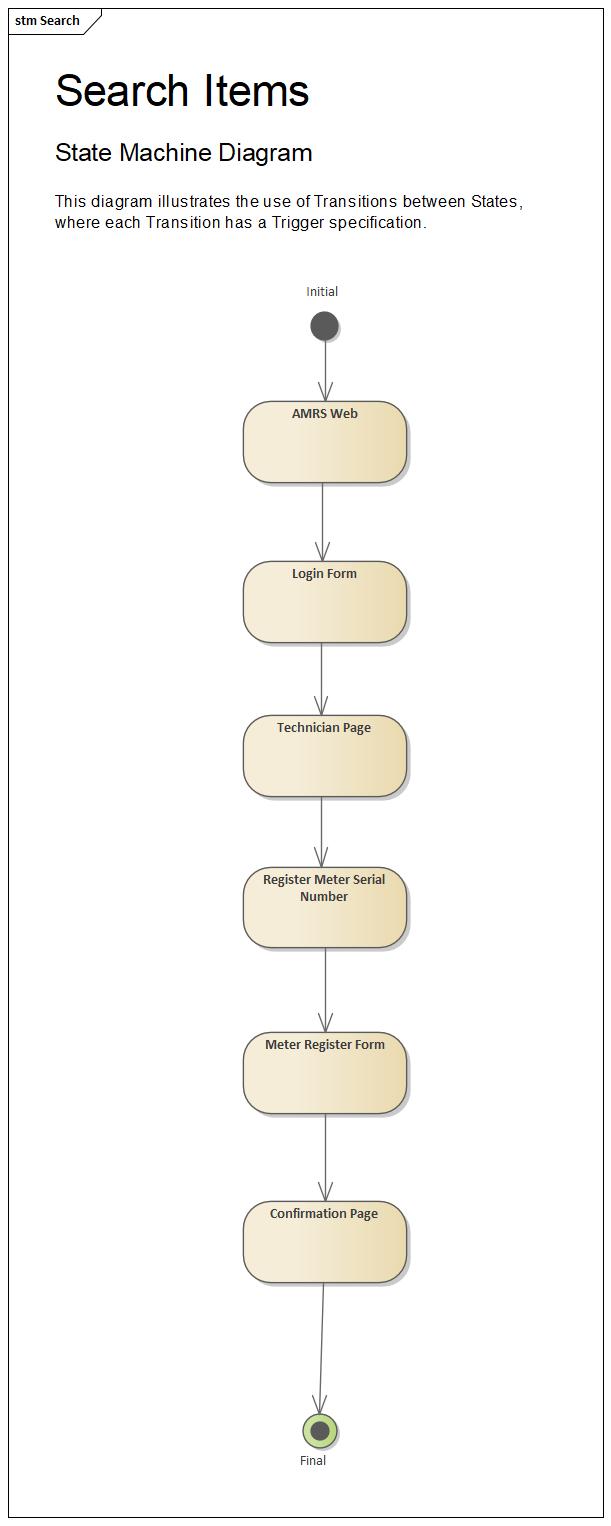


Fig: **CUSTOMER REGISTRATION**

 **Fig: TECHNICIAN REGISTRATION**

 **Fig: METER REGISTRATION**