Programming Applications and Frameworks
IT3030

ElectroGrid (EG)
Power Management System

Group Number :– 63
Batch :–
Y3.S1.WE.IT.02.02

Group Members :-

Registration Number	**Name**
IT20160098	B.A.D.A.Sathsarani
IT20143190	De Silva P.H.S.Y.
IT20142582	Thilakawardhana B.G.M.S
IT20154394	U.G.H.T.Kumara

Table of Contents

Work Distribution

Member	Web Service	Functions
IT20142582 Thilakawardhana B.G.M.S	Payment Service	• Add Payment • Update Payment Details • Search payment details from list • Remove unnecessary payments • View Payment List
IT20160098 B.A.D.A.Sathsarani	Customer Service	• Login • Customer registration • Update customer profile • Search customer from customer list • Remove inactive customer • View customer list
IT20154394 U.G.H.T.Kumara	Billing Service	• Add bills • Update bill details • Search given bills • View bills • Remove unnecessary bills.
IT20143190 De Silva P.H.S.Y.	Employee Service	• Login • Update employee profile • Search employees from employee list • Remove employees • View employee list

Introduction

ElectroGrid is an online system which allow customers to pay their power bills directly through an online platform. First, the customer should register to the system as a valid user and then they can pay power bills as they wish. Through this system generates the monthly bills and automatically send to the users. The customer was given the opportunity to select the payment method according to their preference.

1) **Payment Service Implementation**

   Payment service is one of the main parts of the system. Payment service is connected with customer service. This is mainly handling by finance manager. After registering to the system customer can add payments to the system directly. Then customer and finance manager can see payment list and payment details. After adding the payment, customer and finance manager can search payment details according to their needs. Customer can update payment details if they want to change anything. As well as Customer can remove payment details . Finance manager can remove unnecessary payment details.

2) **Customer Service Implementation**

There are mainly 3 types of users in the system. Customer, admin, and finance manager. To pay power bill through the system the customer should register to the system as a valid user by providing details. When registering to the system as a customer they should provide valid certification to prove that they are valid customers. The administrator of the system will check the certification and accept their registering request. After that they can login to the system by providing valid credentials and upload their power bills. Moreover, the administrator of the system can search for customers, update details of the customers and remove inactive customers from the system.

3) **Billing Service Implementation**

Billing function is connected with payment function. This is mainly handling by finance manager. After registering to the system customer can add payments to the system directly. Then customer and finance manager can see payment list and payment details. After adding the payment, customer and finance manager can search payment details according to their needs. Customer can update payment details if they want to change anything. As well as Customer can remove payment details . Finance manager can remove unnecessary payment details.

4) **Employee Service Implementation**

There are mainly 2 types of employees in the system. Admin, and finance manager. As same as customer, the employee also has to login to the system to give services to the customers. The finance manager will be added by the administrator of the system. Moreover, the administrator can add, delete, update and remove billing details. Finance manager can add, delete, update and remove payment details from system

GitHub Details

https://github.com/NEXUS-97/ELOCTROGRIDE_PROJECT-63.git

SE Methodologies

- **Description**

Agile methodology is a software development model which is based on iterative development. This is a method to manage project by breaking it into several phases. It involves continuous development in every phase, and the development cycle goes through planning, implementing, executing, and evaluating. Therefore, we can identify the bugs in the software sooner than later. By using this methodology, a working system can be developed frequently.

- **The Usage**

o Requirements are clearly defined.
o Development of the system is conducted using well known tools.

- **Advantages**

o Rapid and continuous development.
o Late changes in requirements are welcomed.
o Early identification of bugs.

o A working software can be delivered frequently.

- **Disadvantages**

o Lack of emphasis on designing.
o Requires considerable skills to complete the implementation successfully.
o Difficult to scale the complexity of the project.

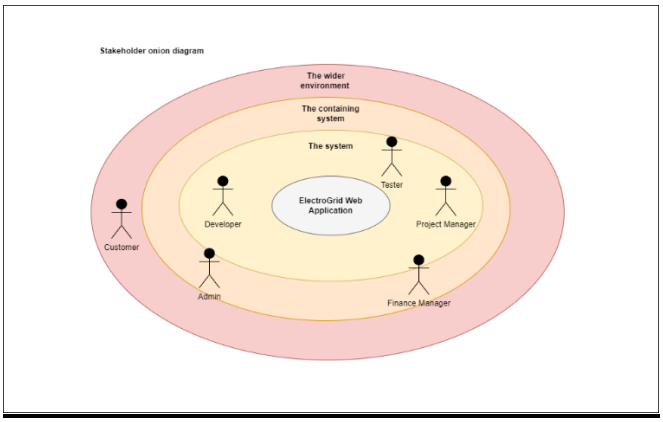
Time schedule (Gantt chart)

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
1)Gathering of information and requirements										
2)Installation of Software										
3)Designing the Database										
4)Implementation										
5)Feedback session										
6)Integration and Testing										
7)Finalizing system and documents										

Requirements Analysis

Requirement analysis includes both user requirements and system requirements. ElectroGrid web system has four main stake holders. Customer, admin, and finance manager. This application is a restful web application and application have 4 services. Customer service, Payment service, Employee service and Billing service.

1) Stakeholder Analysis (onion diagram)



2) Technical Requirements

- Technical requirements are the technical issues that must be considered to make the system successful.
- User, Products, Orders, and Fund's detail can be updated, deleted, and view when needed. And also, sensitive details should be encrypted.

3) **Functional Requirements**

1) **Customer management**
   - Register customers to the system.
   - View customer list
   - Update Remove and Search customers.

2) **Payment management**
   - Add payment to the system.
   - View payment list
   - Update Remove and Search payment.

3) **Employee management**
   - Register employees to the system.
   - View employee list
   - Update Remove and Search employees.

4) **Billing management**
   - Add bills to the system.
   - View bill list
   - Update Remove and Search bills.

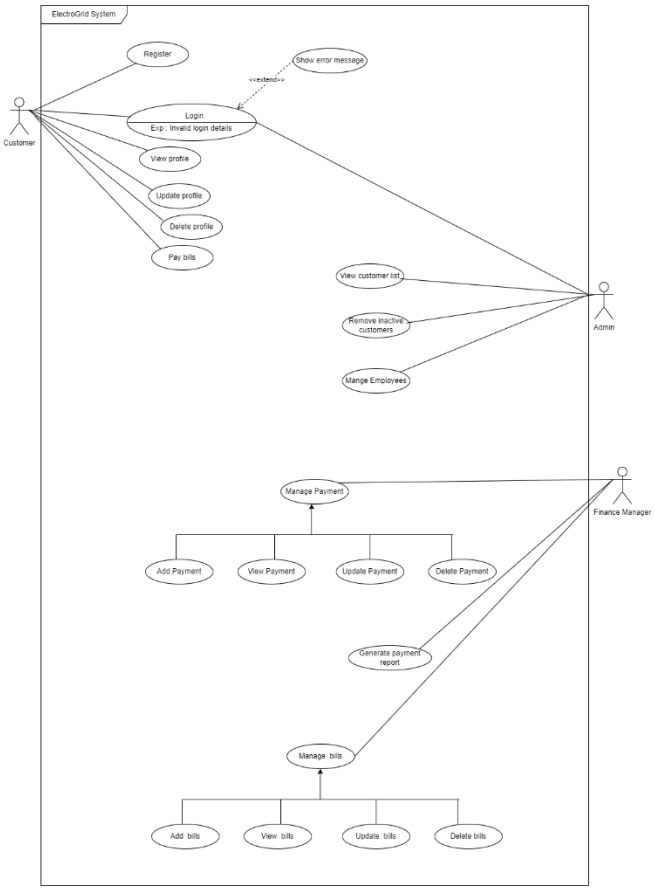
4) **Non – Functional Requirements**

- Performance – Response time, Throughput, Utilization, Static Volumetric, User interface, Conformity
- Security requirements - Security requirements assures that all data inside the system or its part will be protected against malware attacks or unauthorized access. All the users have unique logins. So that, cannot be edited by anyone other than the relevant parties.

(User authentication – Username & password, Insertion, Modification and Administrators' rights)

- Software Quality Attributes
  - Availability
  - Maintainability
  - Usability
  - Accuracy
  - Stability
  - Correctness
  - Accessibility
- Reliability

5) Requirements Modelling (Use case Diagram)

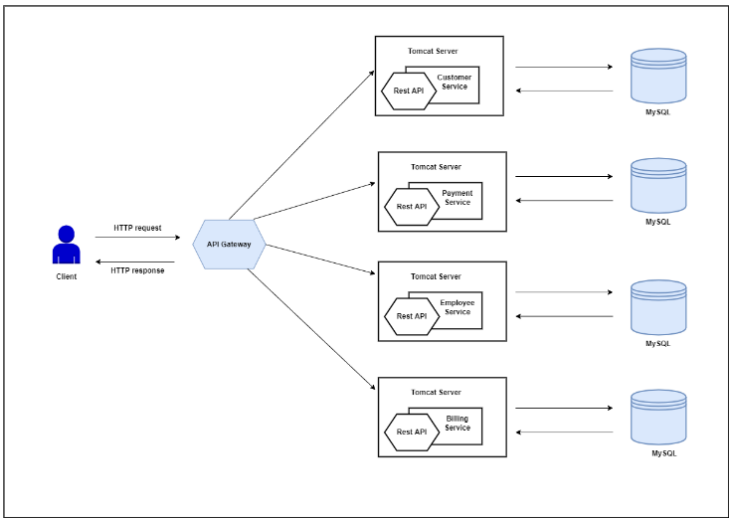


System's overall design

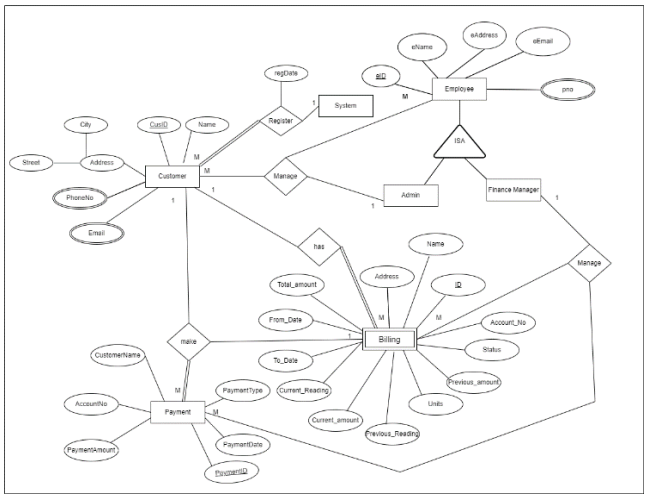
1) Overall Architecture

ElectroGrid is a power management system where the registered customer can pay their power bill payments to the system. Admin can add , delete, view and update customer details as well as billing details. Finance manager can add , delete, view and update payment details.

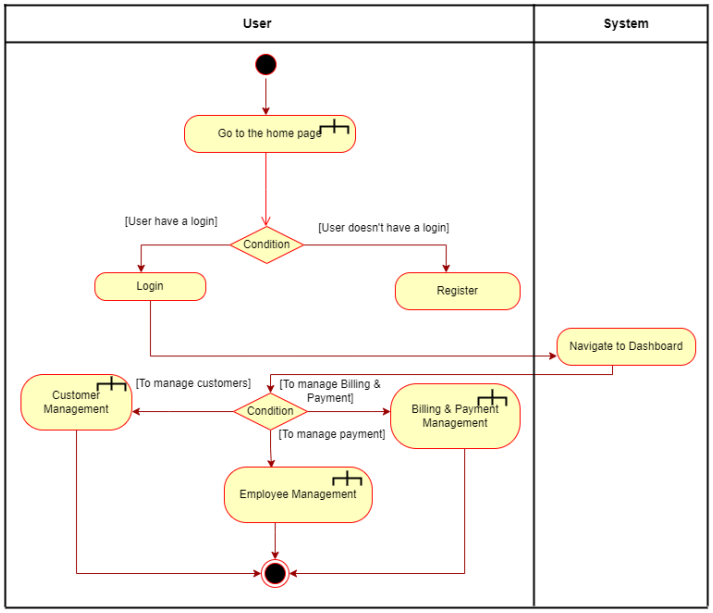
The System can be used by entering the respective username and password. This management system consists of four services as Customer Service, Payment Service, Employee Service, Billing Service. Separated databases are used for each service. Also, Postman was used to take the inputs and test the output results. Here the data is passed when a client make a request it filters through a gateway to the relevant service and the respond is sent back in the same way through the gateway to the client. Application Programming Interfaces (APIs) allow these operations like improve existing services, that can work isolate and more efficiently.



2) Overall DB Design (ER)

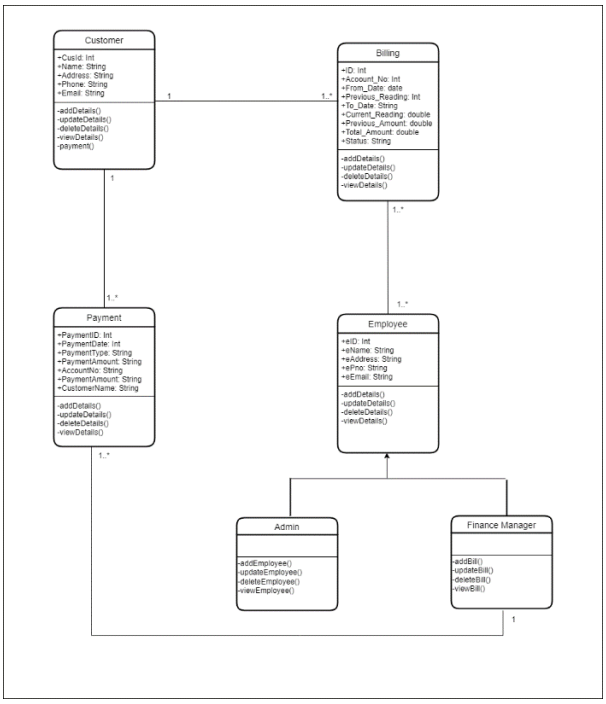


3) Overall DB Design (ER)



❖ **The above-mentioned action calls are depicted in the individual sections**

4) Any other relevant design diagrams for the overall system



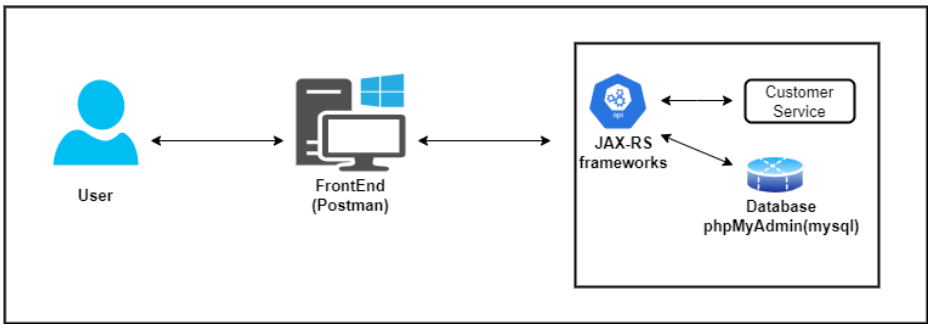
Individual Sections

Customer Service

1) Service design

Customers can register to the system using registration form by providing their details. After that customers can login to the system as a valid user by providing valid credentials. The access to the system varies according to the user type.

- **API of the service**



1. **Read Customer(GET)**

   Resource: - Customer
   Request: - GET CustomerRegistration/RegisterService/Customer
   Media: - Form Data
   Response: - HTML table with all attributes in customer table
   URL: - http://localhost:8010/CustomerRegistration/CustomerService/Customer

2. **Create Customer(POST)**

   Resource: - Customer
   Request: - POST CustomerRegistration/RegisterService/Customer
   Media: - URL encoded form
   Data: - cName: Sasini
   cAddress: Galle
   cEmail:sasini@gmail.com
   cDate: 2022-04-24 ,
   pno: 0755748964
   Response: - String message displayed as "Inserted Successfully"
   URL: - http://localhost:8010/CustomerRegistration/CustomerService/Customer

3. **Update Customer (PUT)**

   Resource: - Customer
   Request: - PUT CustomerRegistration/RegisterService/Customer
   Media: - Application JSON
   Data: - {
   "cID": "6",
   "cName": "Sasini" ,
   "cAddress": "Mathara",
   "cEmail":"sasini@hotmail.com",
   "cDate": "2022-04-21",
   "pno": "0788748569"
   }
   Response: - String message displayed as "Updated Successfully"
   URL: http://localhost:8010/CustomerRegistration/CustomerService/Customer

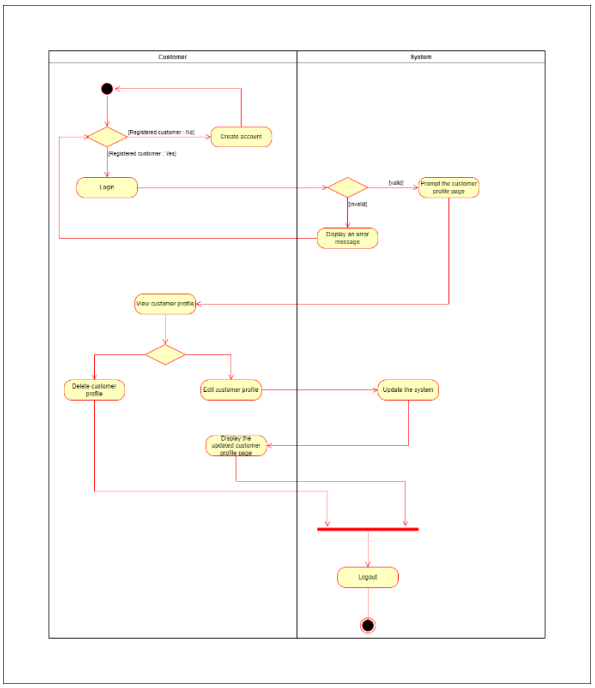
4. **Delete Customer (DELETE)**
   Resource: - Customer
   Request: - DELETE CustomerRegistration/RegisterService/Customer
   Media: - Application XML

   Data: -
   <customerData>
   <cID>3</cID >
   </ customerData >
   Response: - String message displayed as "Deleted Successfully"
   URL: : http://localhost:8010/CustomerRegistration/CustomerService/Customer

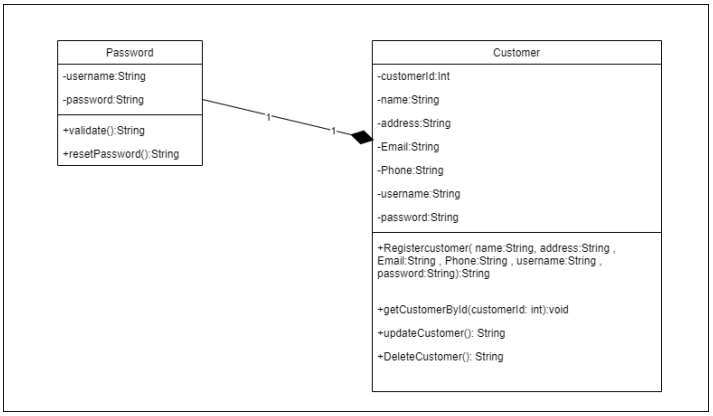
b) Internal logic (Class Diagram / Activity Diagram/Use case Diagram/ER Diagram /Flow Chart)

The responsibility of the customer service is to manage the details of the users who register to the system. Mainly there are 3 types of users are customer, admin and finance manager. When registering to the system as a customer they should provide valid certification to prove that they are valid customers. The administrator of the system will check the certification and accept their registering request. Then they were given the access to pay power bills.

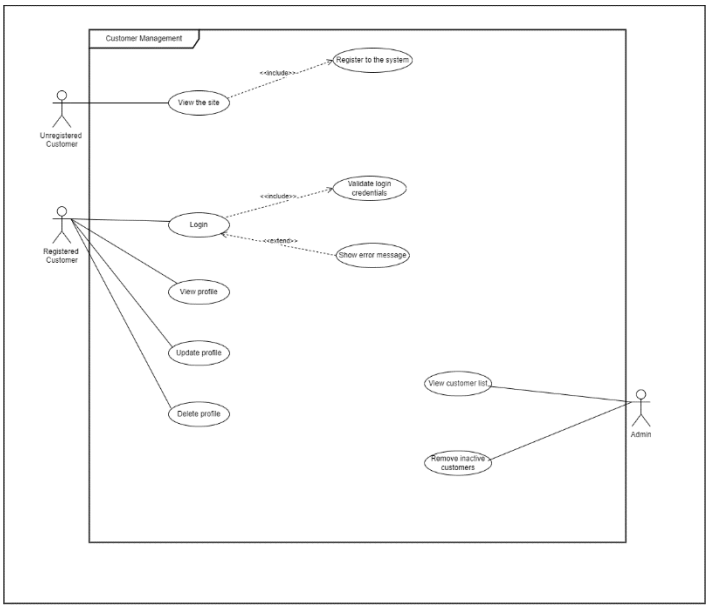
- **Activity Diagram**



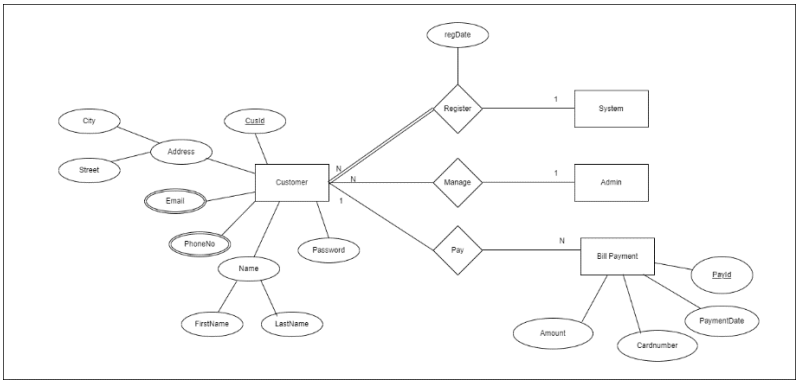
- **Class Diagram**



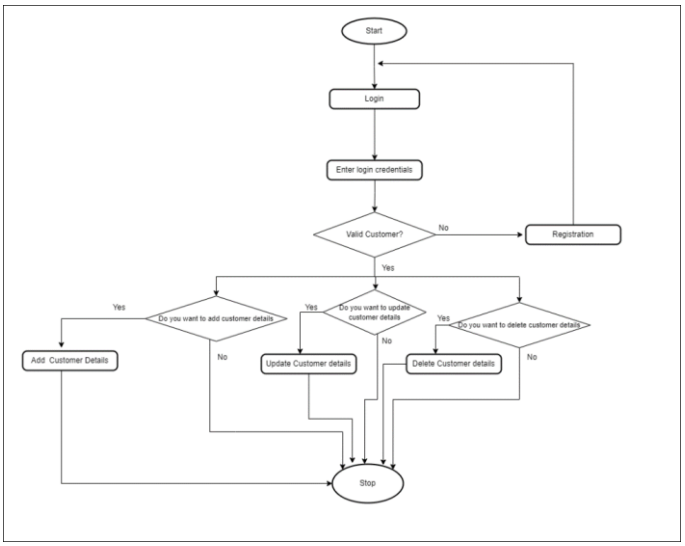
- **Use case Diagram**



- **Database for the service (ER)**



- **Flow Chart**



2). **Service development and testing**

- **Tools used**
  - Dependency Management Tool : Maven
  - Testing Tool : Postman
  - Verion Control System : Git
  - IDE : eclipse
  - Programming Language : Jersey framework (JAX-RS)
  - Programming Language : Java
  - Database : phpMyAdmin(Mysql)
  - Server : Apache Tomcat Server

- **Testing methodology and results**

Test ID	Description	Input	Expected Output	Actual Output	Result
1	Create Customer	cName: Sasini cAddress: Galle cEmail:sasini@gmail.com cDate: 2022-04-24 , pno: 0755748964	Inserted Successfully	Inserted Successfully	Pass
2	View Customer		Display a HTML table with all the attributes in customer table	Displayed HTML Table with all the attributes in customer table	Pass
3	Update Customer	cID: 6 cName: Sasini cAddress: Mathara cEmail:sasini@hotmail.com cDate: 2022-04-21, pno: 0788748569	Updated Successfully	Updated Successfully	Pass
4	Delete Customer	cID – "3"	Deleted Successfully	Deleted Successfully	Pass

2) Assumptions

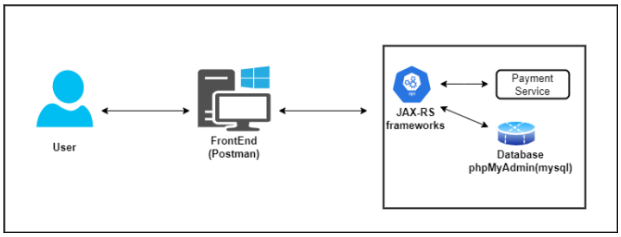
- There are mainly three types of users in the system customer, admin and financial manager
- Admin has the responsibility to manage all the users of the system
- Customers only can send registration requests, After the approval of the administrator they will be considered as a valid user

Payment Service

- **Service design**

Payment management part mainly handle by finance manager. Customers can add payment details by using this system. After that customer can pay power bills through the system.

a) API of the service



I. Add Payment(POST)

Resource: payment
Request:POST Payments/PaymentService/Payments
Media:Form data - URL encoded
Data: - PaymentDate:12.02.2022,
CustomerName: Nuwan Prasad,
AccountNo: 800085231,
PaymentType: Check,
Amount : 12000.00
Response: Inserted successfully
URL : **http://localhost:8080/Payments/PaymentService/Payments**

II. View Payment (GET)

Resource : payment
Request: GET Payments/PaymentService/Payments
Media:Form Data
Response:HTML table with all attributes in the payment table
URL : **http://localhost:8080/Payments/PaymentService/Payments**

III. Update Payment (PUT)

Resource: payment

Request:PUT Payments/PaymentService/Payments
Media:Form data – Application JSON
Data: { PaymentID: "1" ,
PaymentDate: "12.02.2022" ,
CustomerName: "Nuwan Prasad" ,
AccountNo: "8000852314" ,
PaymentType: "Check" ,
Amount : "15000.00"}
Response: Updated successfully
URL : **http://localhost:8080/Payments/PaymentService/Payments**

IV. Delete Payment (DELETE)

Resource: - Payment
Request: - DELETE
Media: - Application XML
Data: -
<paymentData>
<paymentID>3</ paymentID >
</ paymentData >

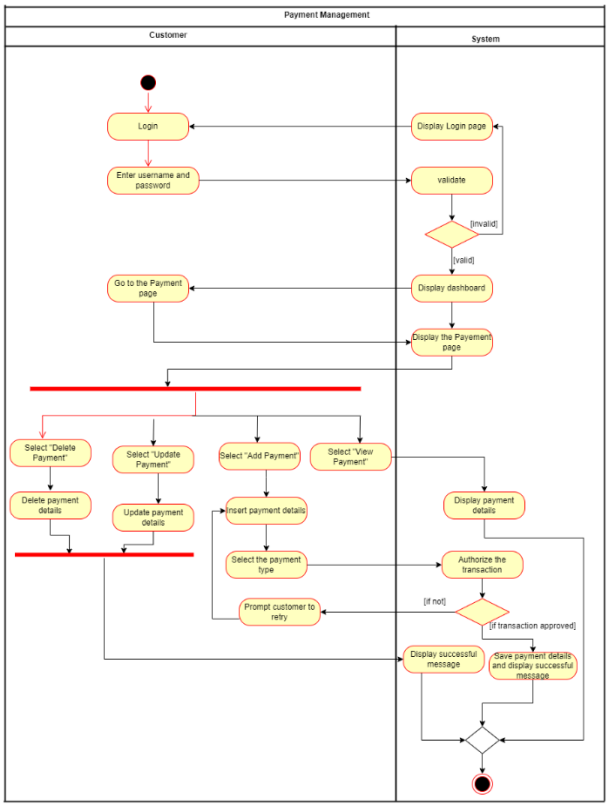
Response: - Deleted successfully

URL: - **http://localhost:8080/Payments/PaymentService/Payments**

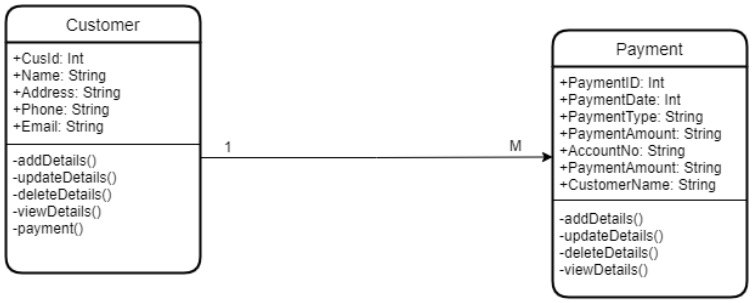
c) Internal logic (Class Diagram / Activity Diagram/Use case Diagram/ER Diagram /Flow Chart)

The main responsibility of payment service is payment management. Both customer and finance manager are connected with this service. Before doing the payments, user can edit details or remove. Then the user can select their payment type. After that users can enter their payment details. There are 2 types of payments. Credit and Debit card payments and Cash payments. After that user can check the orders in the order- list.

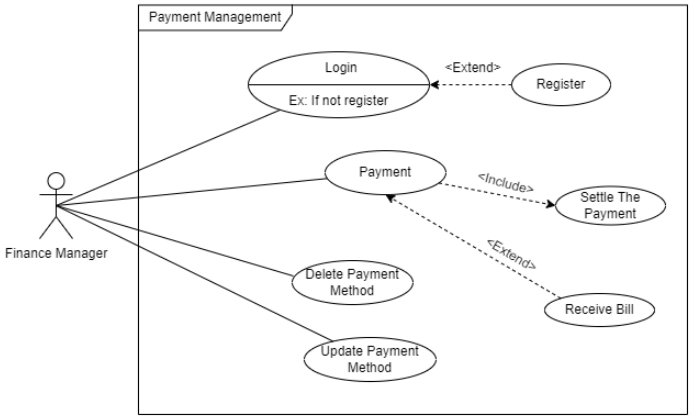
- **Activity Diagram**



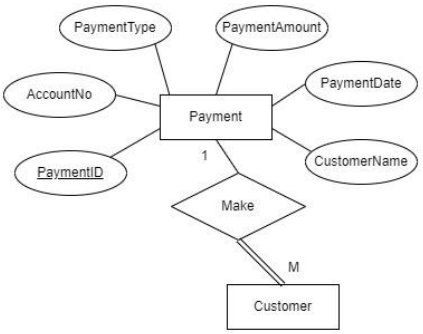
- **Class Diagram**



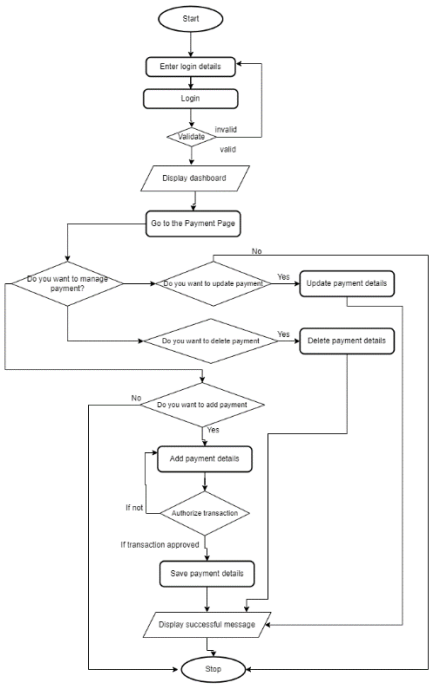
- **Use Case Diagram**



- **Database for the service**



- **Flow Chart**



a) Tools used

- Dependency Management Tool : Maven
- Testing Tool : Postman
- Verion Control System : Git
- IDE : eclipse
- Programming Language : Jersey framework (JAX-RS)
- Programming Language : Java
- Database : phpMyAdmin(Mysql)
- Server : Apache Tomcat Server

b) Testing methodology and results

Test ID	Description	Input	Expected Output	Actual Output	Result
1	Add Payment	PaymentDate:12.02.2022 CustomerName: Nuwan Prasad AccountNo: 800085231 PaymentType: Check Amount : 12000.00	Inserted successfully	Inserted successfully	Pass
2	View Payment		Display a HTML table with all the attributes in payment table	Displayed HTML Table with all the attributes in payment table	Pass
3	Update Payment	PaymentID: 1 PaymentDate:12.02.2022 CustomerName: Nuwan Prasad AccountNo: 8000852314 PaymentType: Check Amount : 15000.00	Updated successfully	Updated successfully	Pass
4	Delete Payment	PaymentID: 3	Deleted successfully	Deleted successfully	Pass

5) Assumptions

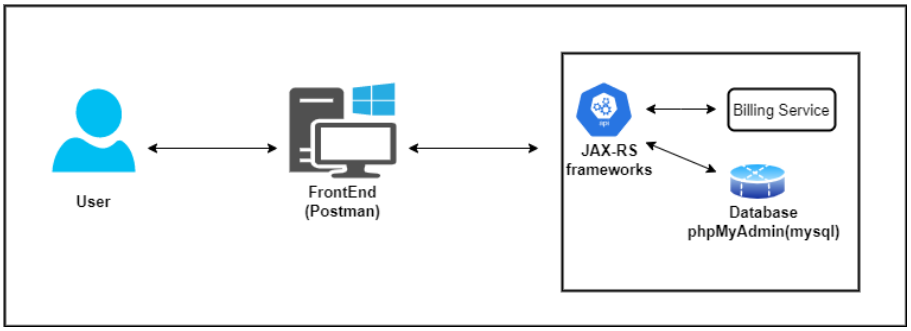
- Payment can be added by customers and they can update and delete payment details according to their preferences.
- Finance manager can see payment details.

Billing Service

1) Service design

The billing management part is handled by the administrator. By using this system, the customer can add bills and pay. And also administrator can view billing details.

a) API of the service



1. **Read Bills (GET)**

   Resource: - Bill
   Request: - GET CustomerRegistration/BillingService/Billing/read
   Media: - Form Data
   Response: - HTML table with all attributes in bill table
   URL: - http://localhost:8010/CustomerRegistration/BillingService/Billing/read

2. **Create Bills (POST)**
   Resource: - Bill
   Request: - POST CustomerRegistration/BillingService/Billing/insert
   Media: - URL encoded form
   Data: - Account_No : 2547
   To_Date : 2022- 04 - 25
   From_Date : 2022-04-01
   Current_Reading: 80
   Status : Done
   Response: - String message displayed as “Billing data added successfully”
   URL: - http://localhost:8010/CustomerRegistration/BillingService/Billing/insert

3. **Update Bills (PUT)**
   Resource: - Bill
   Request: - PUT CustomerRegistration/BillingService/Billing/updatebill
   Media: - Application JSON
   Data: - {
   “ID”: “ 2” ,
   “Account_No” : “12344321”,
   “To_Date” : “2022- 03 – 24”,
   “From_Date” : “2022-02-25”,
   “Current_Reading”: “159”,
   “Status” : “Done “
   }
   Response: - String message displayed as “Updated Successfully”
   URL: - http://localhost:8010/CustomerRegistration/BillingService/Billing/updatebil

4. **Delete Bills (DELETE)**
   Resource: - Bill

Request: - DELETE CustomerRegistration/BillingService/Billing/delete
Media: - Application XML

Data: -

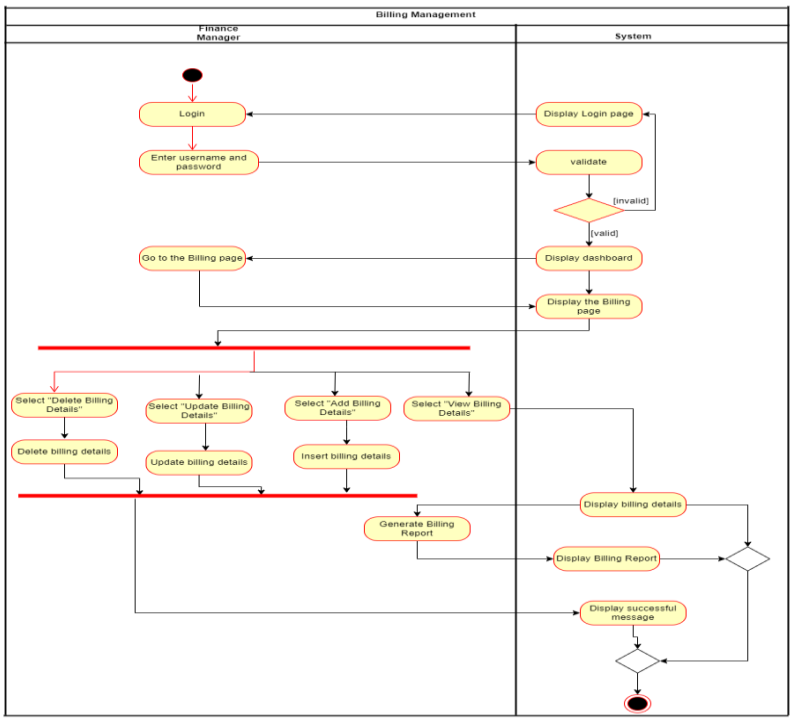
```
<billData>
    <Account_No>12344321</ Account_No >
</ billData >
```

Response: - String message displayed as “Deleted Successfully”
URL: - http://localhost:8010/CustomerRegistration/BillingService/Billing/delete

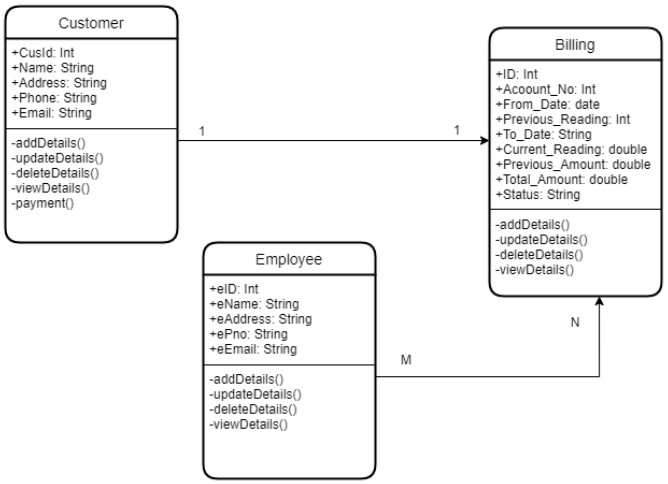
b) Internal logic (Class Diagram / Activity Diagram/Use case Diagram/ER Diagram /Flow Chart)

The Funds web service provides young researchers to get funds through the GadgetBadget company. All the necessary details to issue funds will be captured through a form, which allows the researchers to apply for funds by inserting their details. If the applicant is qualified for the funds, the company will approve the request, update the request status as “Approved” and allocate funds and inform the researcher. Otherwise, if the request does not meet the requirements to issue funds, the request will be rejected. Moreover, the finance manager can update fund details, and delete unnecessary funds from the system.

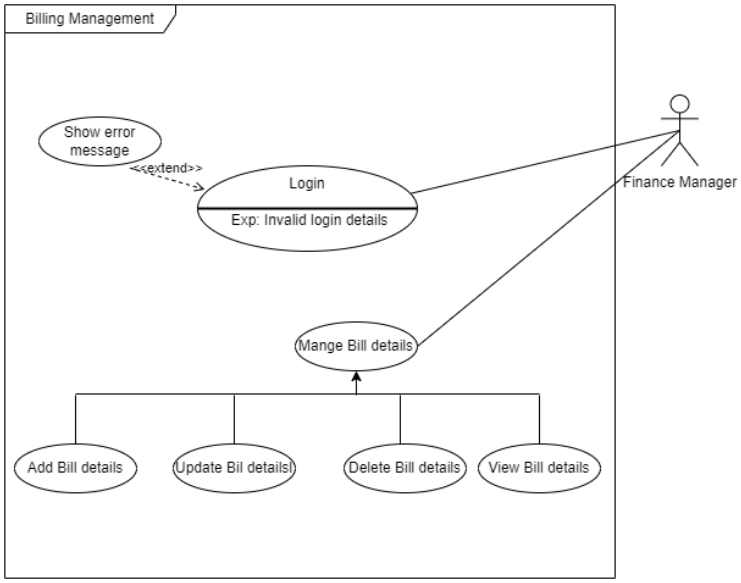
- **Activity Diagram**



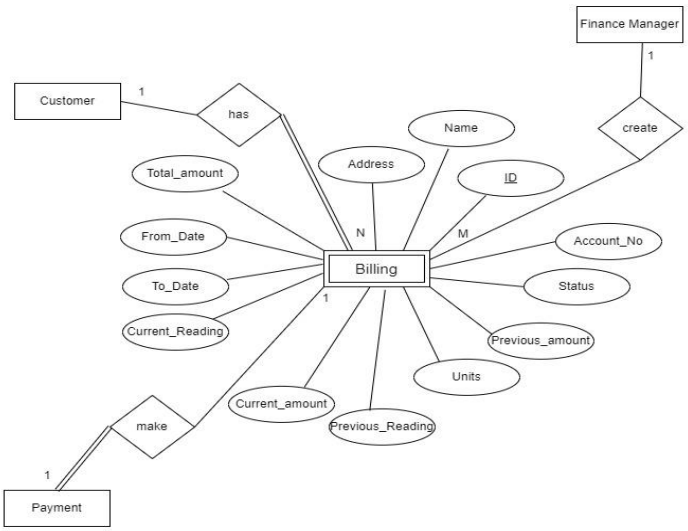
- **Class Diagram**



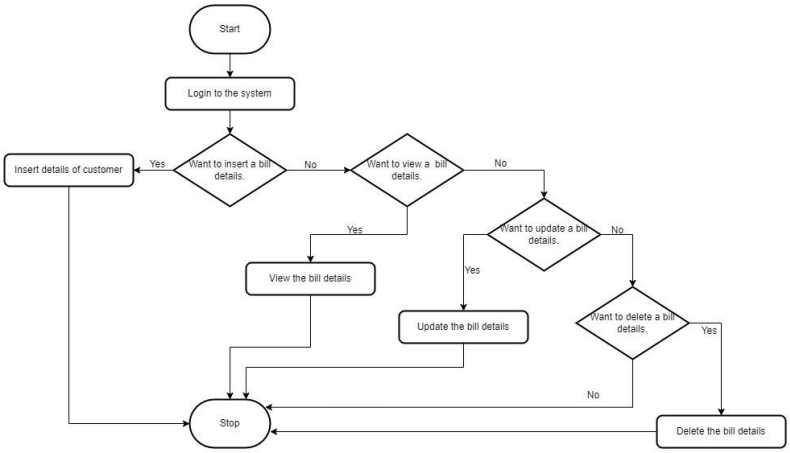
- **Use case Diagram**



- **Database for the service (ER)**



- **Flow Chart**



2) Service development and testing

a) Tools used

- Dependency Management Tool: - Maven
- Testing Tool: - Postman
- IDE: - Eclipse
- Programming Language: Java
- Framework: - JAX - RS
- Database: - phpMyAdmin (MySQL)
- Server: - Apache Tomcat
- Version Control System: - Git

b) Testing methodology and results

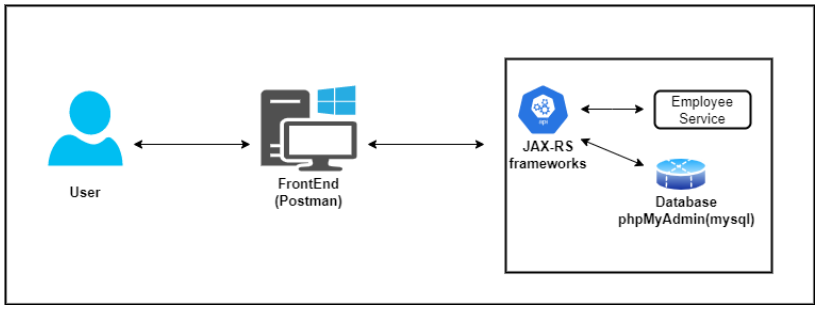
Test ID	Description	Input	Expected Output	Actual Output	Result
1	Create Bill	Account_No : 2547 To_Date : 2022- 04 - 25 From_Date : 2022-04-01 Current_Reading: 80 Status : Done	Billing data added successfully	Billing data added successfully	Pass
2	View Bills		Display a HTML table with all the attributes in billing table	HTML Table displayed with all the attributed in billing table	Pass
3	Update Bill	ID : 2 Account_No : 12344321 To_Date : 2022- 03 - 24 From_Date : 2022-02-25 Current_Reading: 159 Status : Done	Updated Successfully	Updated Successfully	Pass
4	Delete Bill	Account_No: 12344321	Deleted Successfully	Deleted Successfully	Pass

3) Assumptions

- Researcher can request for funds, update fund details, and delete their requests.
- After updating the fund request status into “Approved” funds will be allocated.
- Finance Manger has the responsibility of managing funds.
-

Employee Service

1) Service design



1. **Read Employee(GET)**

   Resource: - Employee
   Request: - GET EmployeeManagement/EmployeeService/Employee
   Media: - Form Data
   Response: - HTML table with all attributes in employee table
   URL: - http://localhost:8082/EmployeeManagement/EmployeeService/Employee

2. **Create Employee (POST)**

   Resource: -
   Employee
   Request: - POST EmployeeManagement/EmployeeService/Employee
   Media: - URL encoded form
   Data: - eName: Thisari Navoda
   eAddress: Galle
   eEmail:thisari2@gmail.com
   eDate: 18.04.2022,
   pno: 0715789654
   Response: - String message displayed as “Inserted Successfully”
   URL: - http://localhost:8082/EmployeeManagement/EmployeeService/Employee

3. **Update Employee (PUT)**

   Resource: - Employee
   Request: - PUT EmployeeManagement/EmployeeService/Employee
   Media: - Application JSON
   Data: - {
   “eID”: “6” ,
   “eName”: “Dinithi Chandula”,
   “eAddress”: “Kandy”
   “eEmail”:”dinithi12@gmail.com”
   “eDate”: “19.04.2022”,
   “pno”: “0715678956”
   }
   Response: - String message displayed as “Updated Successfully”

URL: - http://localhost:8082/EmployeeManagement/EmployeeService/Employee

4. **Delete Employee (DELETE)**
   Resource: - Employee
   Request: - DELETE EmployeeManagement/EmployeeService/Employee
   Media: - Application XML

   Data: -

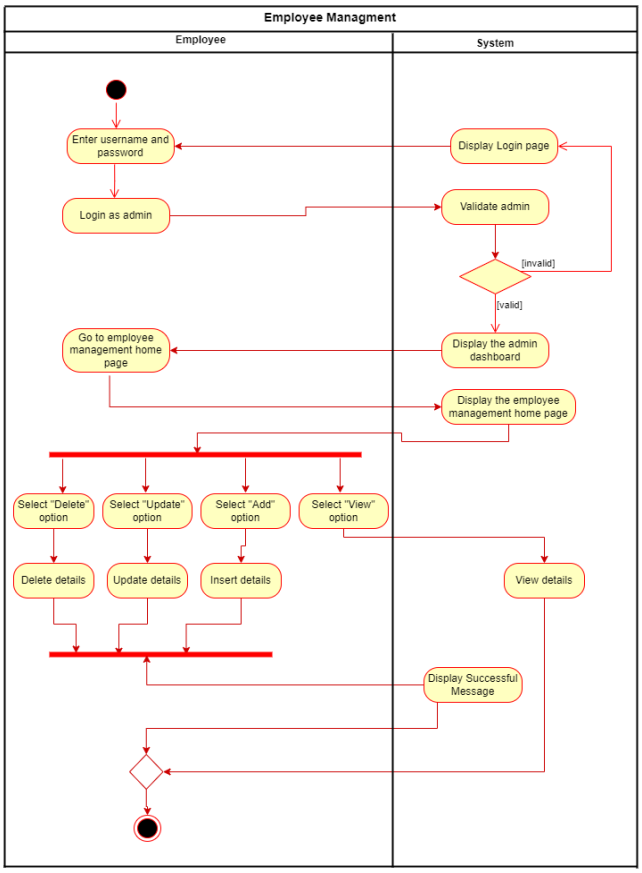
```
<employeeData>
    <eID>6</eID >
</ employeeData >
```

   Response: - String message displayed as "Deleted Successfully"
   URL: http://localhost:8082/EmployeeManagement/EmployeeService/Employee

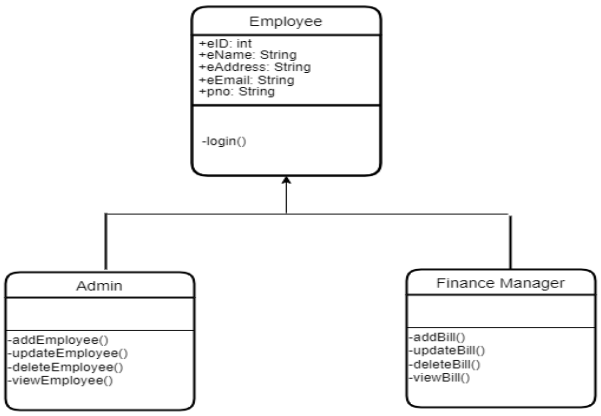
b) Internal logic (Class Diagram / Activity Diagram/Use case Diagram/ER Diagram /Flow Chart)

Mainly there are 2 types of employees such as admin and finance manager . The administrator of the system will check the certification and accept their registering request. An also admin can add. Delete , view and update billing details as well as customer details. Finance manager is added by the administrator by providing credentials for login. He can view payment details.

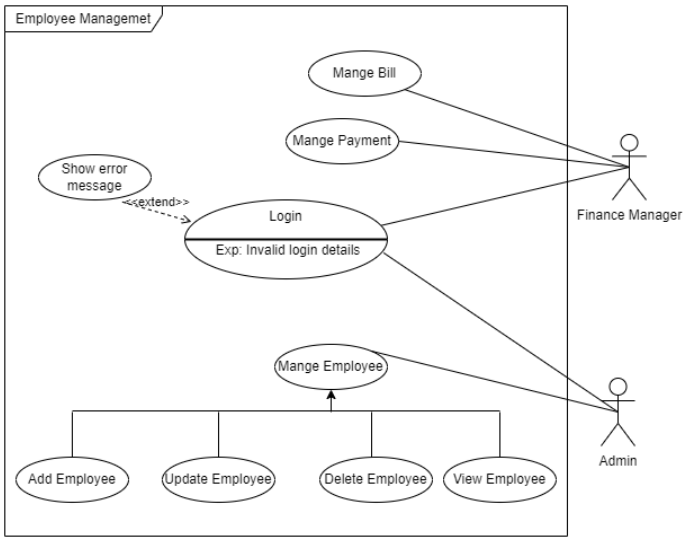
- **Activity Diagram**



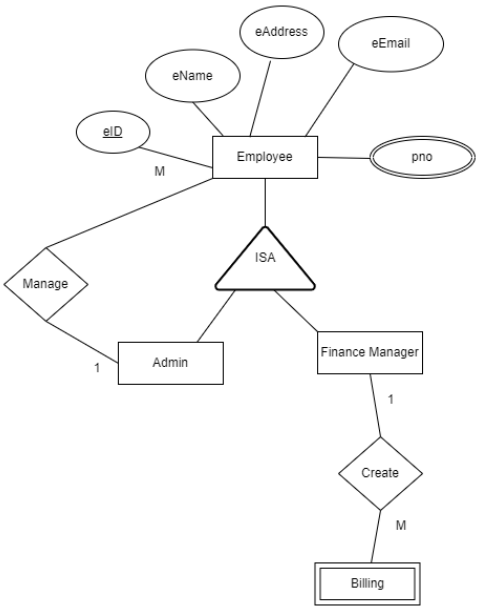
- **Class Diagram**



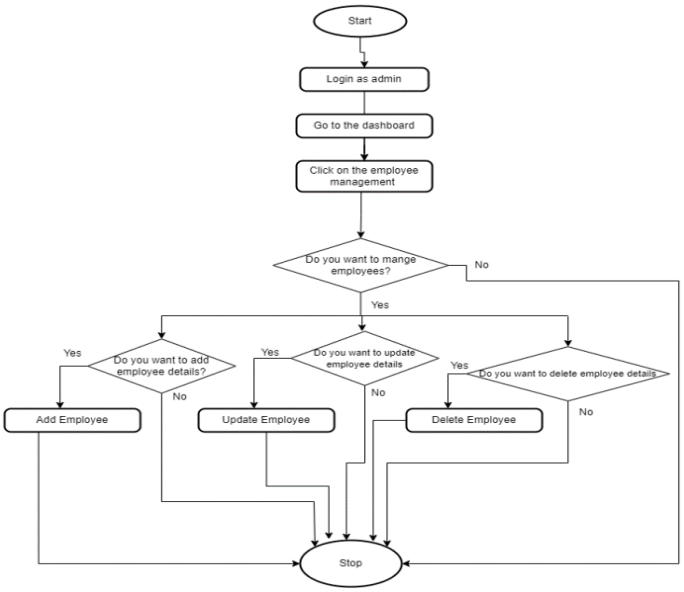
- **Use case Diagram**



- **Use case Diagram**



- **Flow Chart**



2) Service development and testing

- **Tools used.**
  - Dependency Management Tool : Maven
  - Testing Tool : Postman (because user interfaces are not used)
  - Verion Control System : Git (This was used to share the code between the team members)
  - IDE : eclipse
  - Programming Language : Jersey framework (JAX-RS)
  - Programming Language : Java
  - Database: phpMyAdmin (Mysql)
  - Server: Apache Tomcat Server
- **Testing methodology and results**

Test ID	Description	Input	Expected Output	Actual Output	Result
1	Create Employee	eName: Thisari Navoda eAddress: Galle eEmail:thisari2@gmail.com eDate: 18.04.2022, pno: 0715789654	Inserted Successfully	Inserted Successfully	Pass
2	View Employee		Display a HTML table with all the attributes in employee table	Displayed HTML Table with all the attributes in employee table	Pass
3	Update Employee	eID: 6 eName: Dinithi Chandula eAddress: Kandy eEmail:dinithi12@gmail.com eDate: 19.04.2022, pno: 0715678956	Updated Successfully	Updated Successfully	Pass
4	Delete Employee	eID – “6”	Deleted Successfully	Deleted Successfully	Pass

3) Assumptions

- Payment of the order is included in the order service.
- Even though there are multiple payment options, only two categories; card payment and cash payment are used.
    - Financial manager will be added by the administrator to maintain the transactions of the system.

System's Integration Details

1) Tools Used, Testing Methodology and Results & API Documentation

- The following tools were used to develop the project.
    - Dependency Management Tool : Maven
    - Testing Tool : Postman
    - Verion Control System : Git
    - IDE : eclipse
    - Programming Language : Jersey framework (JAX-RS)
    - Programming Language : Java
    - Database : phpMyAdmin(Mysql)
    - Server : Apache Tomcat Server
- For testing purpose postman was used.
- For integration GitHub was used.

2) The Architecture used to Design the System

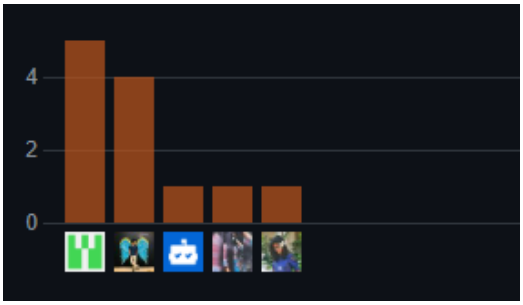
- The high-level architecture diagram was used to design the overall architecture of the system.
- Use case diagram was used to identify the use cases.
- ER diagram to identify the tables of the database.
- Activity diagram and flow charts to identify the flow of the system.
- Class diagram to identify the classes for the implementation.

References

- JAX-RS Documentation https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html
- SE Methodologies https://acodez.in/12-best-software-development-methodologies-pros-cons/
- Maven Documentation https://maven.apache.org/guides/

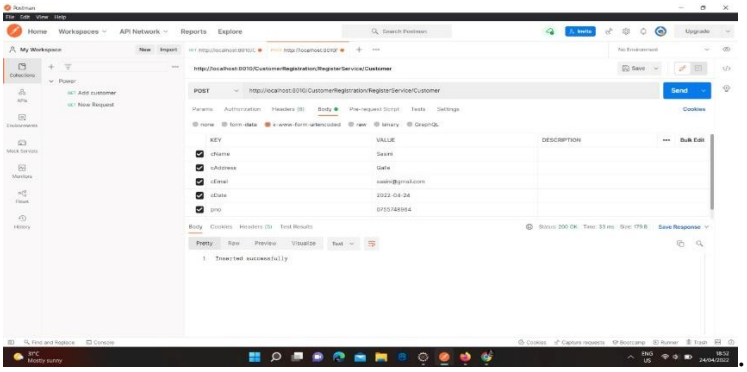
Appendices

- **Commit Log**

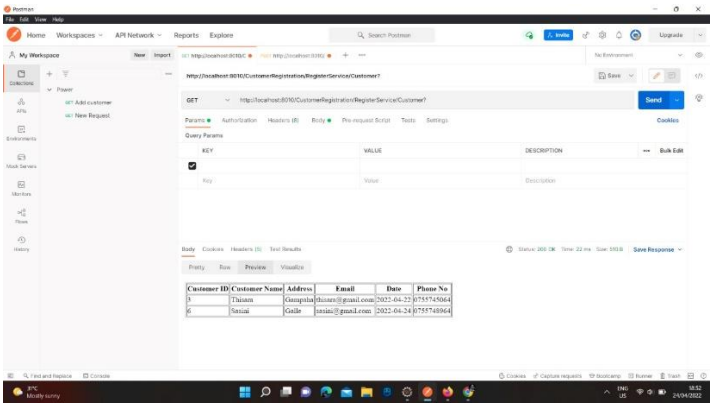


➢ Customer Service postman screenshots

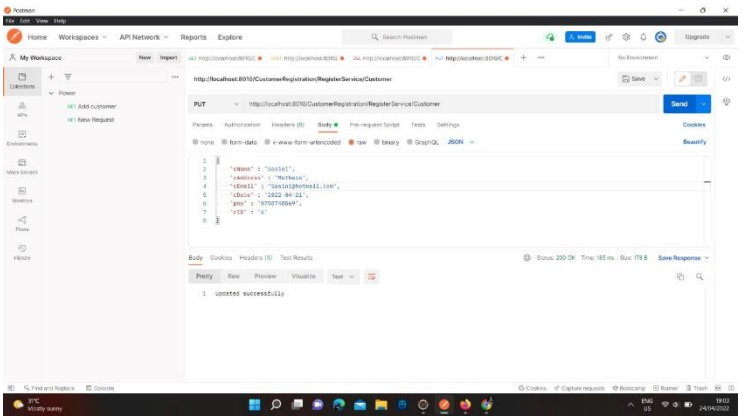
1. Insert Customer



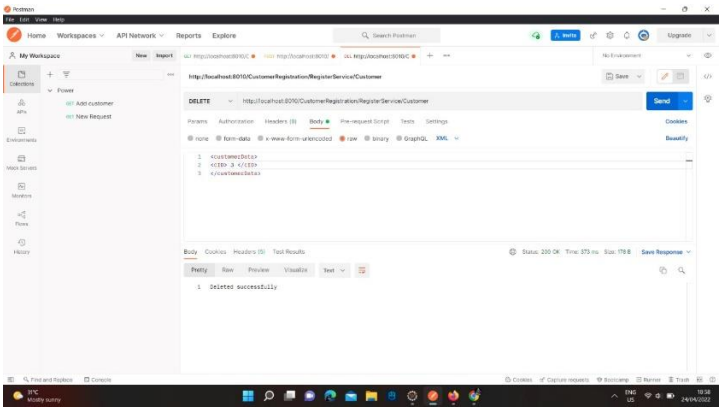
2. Read Customers



3. Update Customer

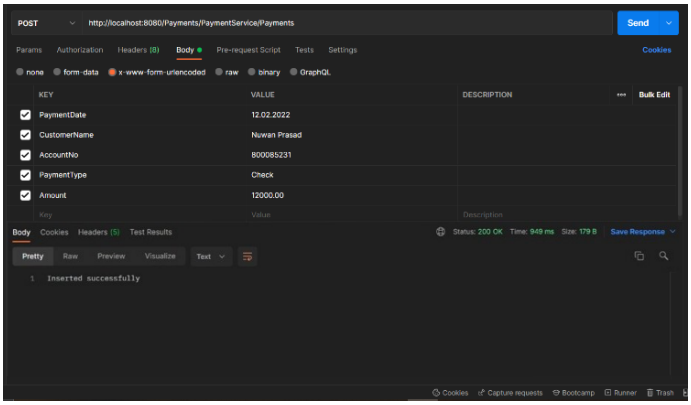


4. Delete Customer

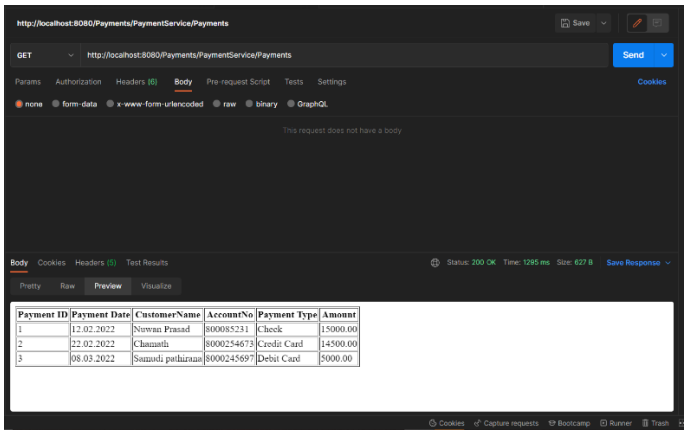


- **Payment Service postman screenshots**

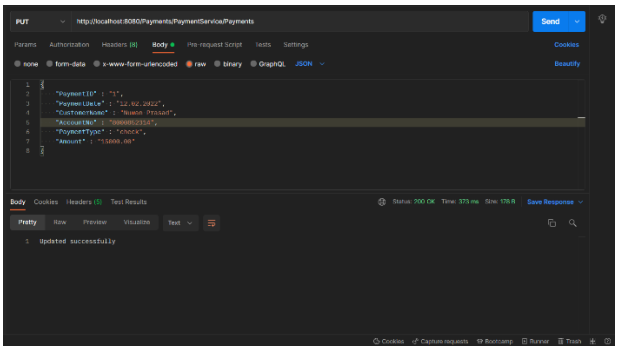
1. Insert Payment



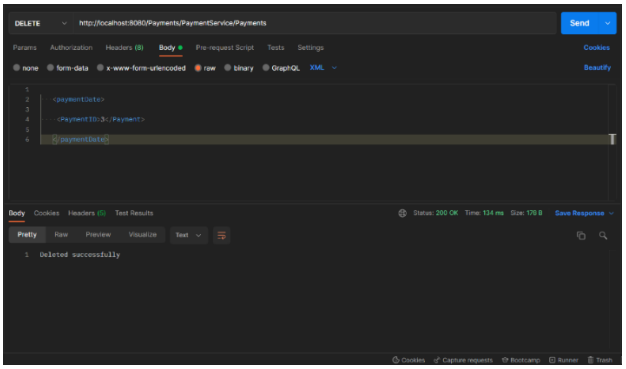
2. Read Payments



3. Update Payment

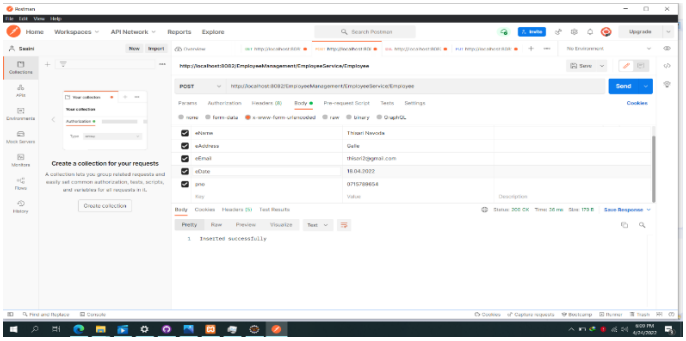


4. **Delete Payment**

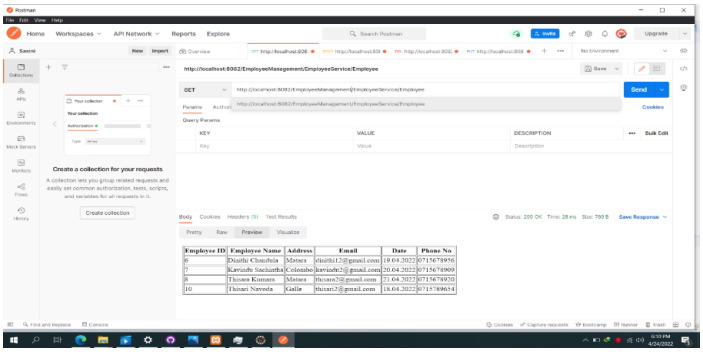


- **Employee Service postman screenshots**

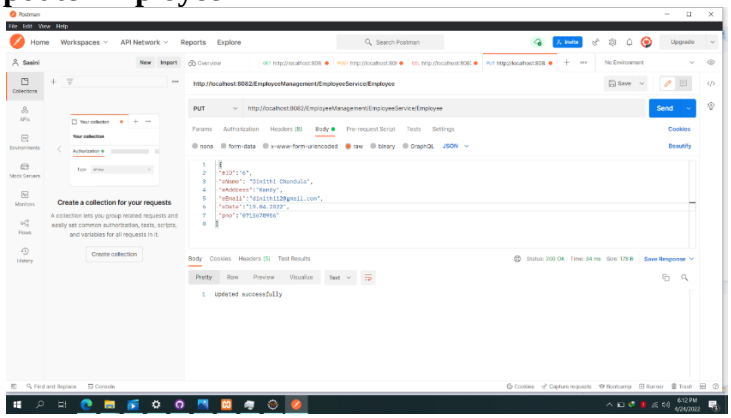
1. **Insert Employee**



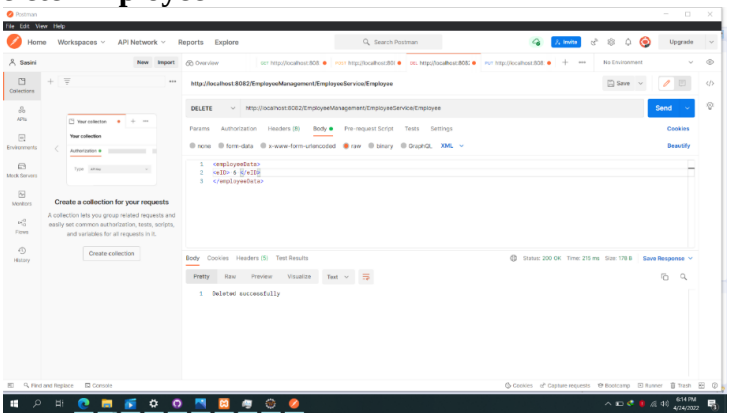
2. **Read Employee**



3. Update Employee

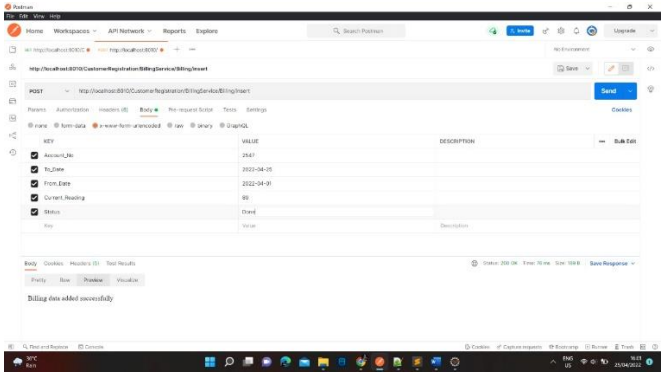


4. Delete Employee

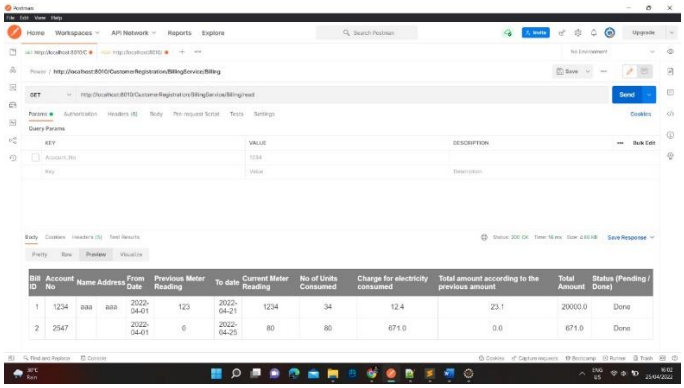


➢ Billing Service postman screenshots

1. Insert Bill

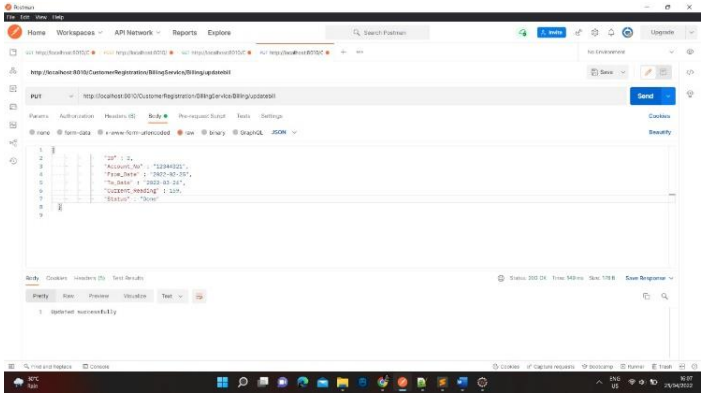


2. Read Bills

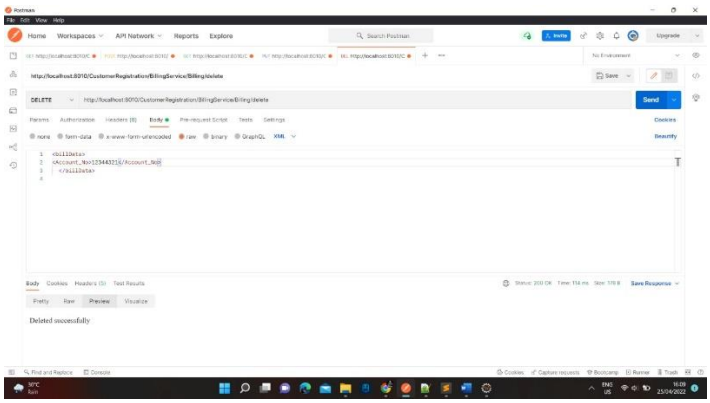


Bill ID	Account No	Name	Address	From Date	Previous Meter Reading	To date	Current Meter Reading	No of Units Consumed	Charge for electricity consumed	Total amount according to the previous amount	Total Amount	Status (Pending / Done)
1	1234	aaa	aaa	2022-04-01	123	2022-04-21	1234	34	12.4	23.1	20000.0	Done
2	2547			2022-04-01	0	2022-04-25	80	80	671.0	0.0	671.0	Done

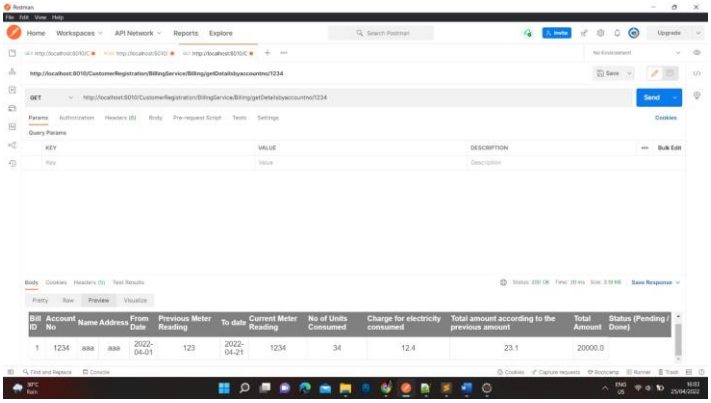
3. Update Bills



4. Delete Bills



5. Search Bills



Bill ID	Account No	Name	Address	From Date	Previous Meter Reading	To date	Current Meter Reading	No of Units Consumed	Charge for electricity consumed	Total amount according to the previous amount	Total Amount	Status (Pending / Done)
1	1234	aaa	aaa	2022-04-01	123	2022-04-21	1234	34	12.4	23.1	20000.0	