



IT3030

Programming Applications and

Frameworks

3rd Year, 1st Semester

Assignment

Group Project

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology

Group No: Y3S1.WE.IT.02.02_Group211

Table of Contents

1. Cover Page.....	
2.Table of Content.....	
3.Introduction.....	03
4. Members' Details and Work Lord	03
5. Clickable Link	03
6. SE methodologies/Methods	04
7. Time Schedule Gantt Chart.....	05
8. Requirement's analysis (Functional, Non-functional, Technical requirements).....	06
9.Use case Diagram/ Activity Diagram.....	07
10.Overall Architecture.....	07
11.ER Diagram.....	08
12.Individual Section.....	09
13.Appendix.....	

1.Introduction

ElectroGrid (EG) is the company who maintains the power grid of the country. They have a system to monitor the power consumption of the users, generate the

Monthly bills and automatically send to the users and accept the online payments from the users.

Our task was to create the online platform by covering the whole scope of the company. We used java JAX-RS Jersey, tomcat and MySQL as our tools to build the platform.

2.Member's Details

IT Number	Name	Web Service	Description of the Web Service
IT20178154	Dilshan P.A.D.S.D	Customer Management Payment Management	Insert, Update, Delete and View Customer Details Insert, Update, Delete and View Payment Details
IT20140366	Tharushika Devindi M.K.S	Bill Management Feedback Management	Insert, Update, Delete and View Bill Details. Insert, Update, Delete and View Feedback Details.
IT20139544	Kapukotuwa S.A.A.H	Power Consumption Management	Insert, Update, Delete and view Power Consumption Details.
IT20139858	Gunasinghe P.S.L	Electricity Board Management	Insert, Update, Delete and View Electricity Board Details.
IT20235024	Abishaalini.S	Employee Management	Insert, Update, Delete and View Employee Details.

Clickable link

<https://github.com/SaChInD99/PAF-Group-Project-2022.git>

SE Methodology

Agile Methodology is used for the development purposes

The combination of iterative and the incremental models is referred to agile methodology. Due to its adaptiveness the complex projects are very easily to handle. This model will provide chance for the project team to break the task and put into small iteration.

Pros:

- People interaction is frequently happening here, so it very help for people communication.
- Adaption can have any time and any phase

Cons:

- Lack of emphasize designing and documentation.
- Project direction can off track when the discussion gone wrong

We used the unit testing and integrated testing methodologies for testing purposes.

1. Unit testing

- We use this for Unit-testing to test the services like individual component or unit. In every Unit-testing micro service tested as unit.

2. Integrate testing

- After Unit-testing all the services are tested as unit. In integration testing the tested microservice will tested as one service and the service order will decided.

03.Time schedule (Gantt chart)

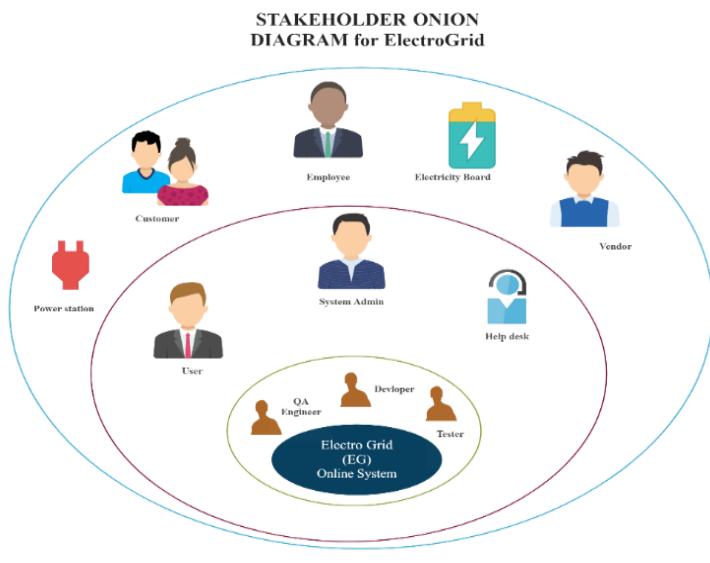


ElectroGrid

GANTT CHART

TASKS	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5
• Planning and Designing overall system designing					
• Requirement analyzing and gathering					
• Design the System DB Design					
• Determine the system development configuration					
• determine about git upload plan and integration plan					
• Develop the web services and testing					
• Testing					
• Documentation					

System's overall design



- Stakeholder analysis (Onion diagram)

Requirement's analysis (Functional, Non-functional, Technical requirements)

Functional Requirements

1. Customer Management: Add, Update, Delete and View Customer Details
2. Payment Management: Add, Update, Delete and View Payment Details
3. Bill Management: Add, Update, Delete and View Details
4. Consumption Management: Add, Update, Delete and View details
5. Employee Management: Add, Update, Delete and View Employee Details
6. Feedback Management: Add, Update, Delete and View Feedback Management Details
7. Electricity Board Management: Add, Update, Delete and View Electricity Management Details

Nonfunctional Requirements

1. Quality, Efficiency, Maintainability, Reliability, Privacy, Accessibility, Compatibility, Extensibility

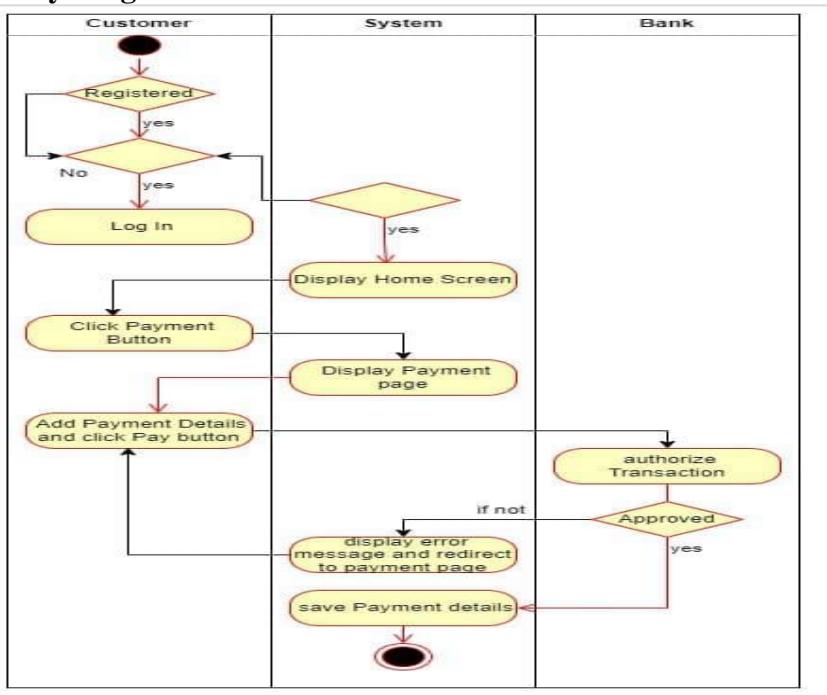
Technical Requirements

The backend is developed using Restful web services **JAX-RS, Jersey, Java on Tomcat** via Eclipse IDE in a windows operating system. My SQL is used in creating database and Postman is an API Client which uses for testing.

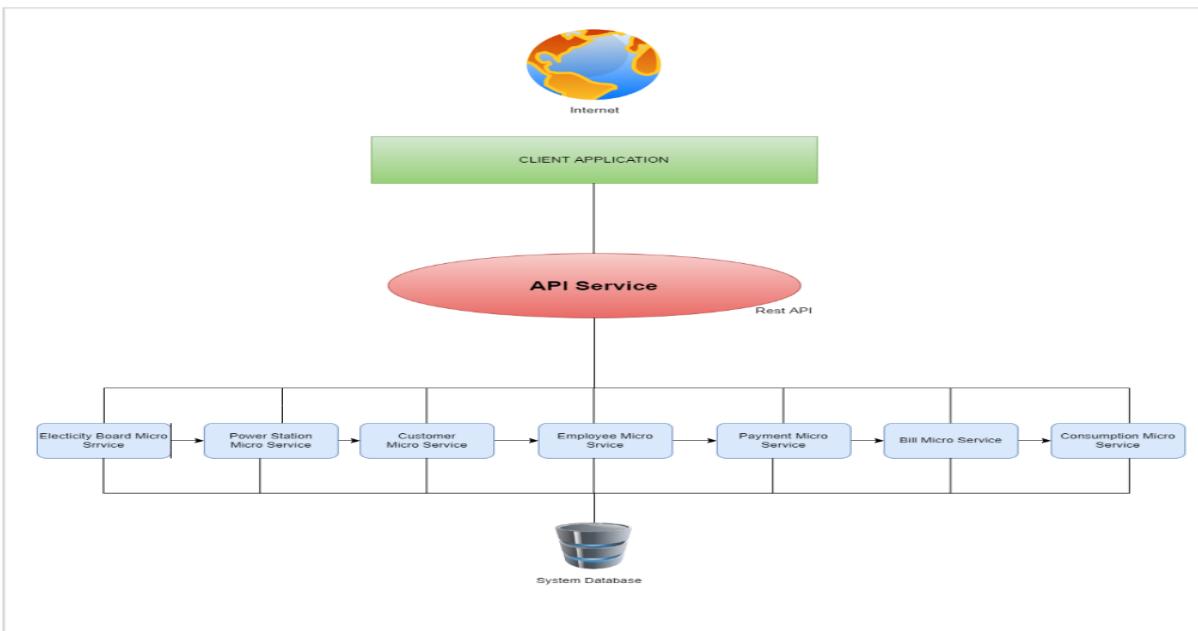
Requirements modelling (Use Case Diagram)



Activity Diagram

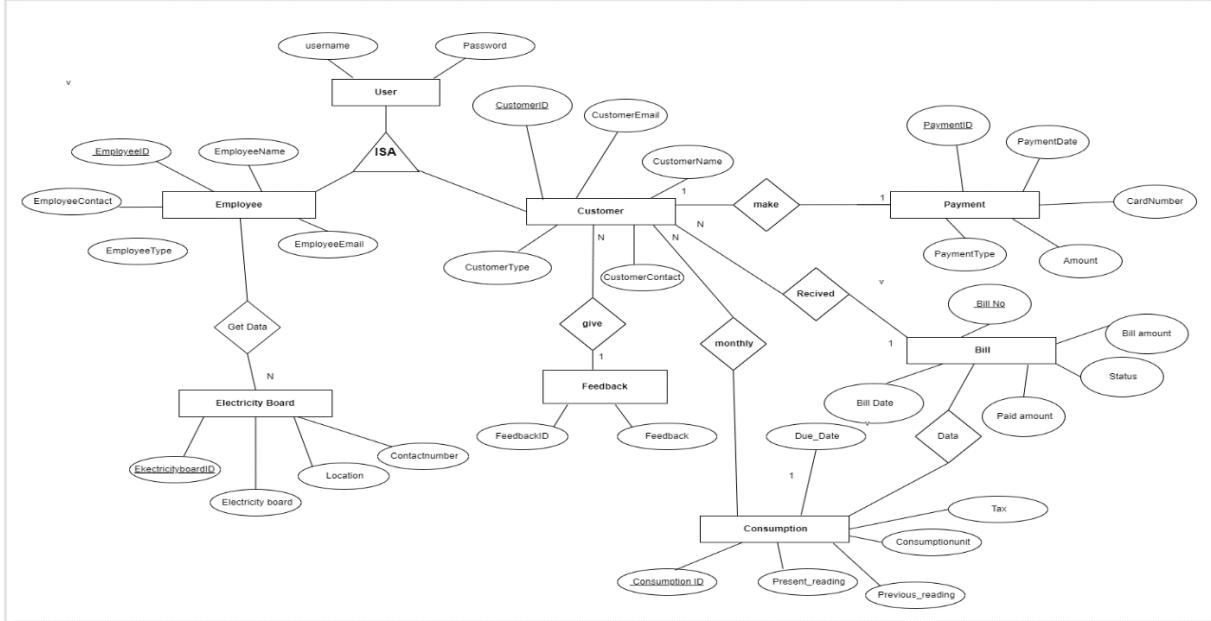


Overall Architecture

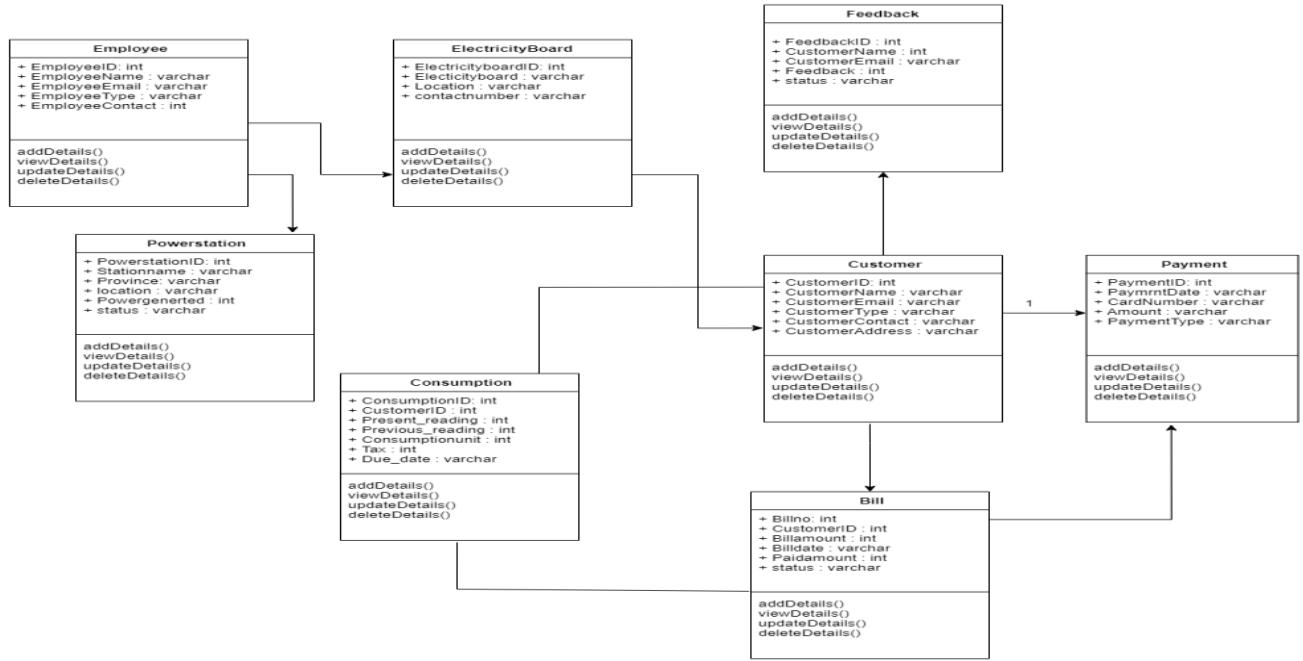


Our task is to create an online platform for a company which maintains the power grid of the country. They have a system to monitor the power consumption of the users, generate the Monthly bills and automatically send to the users and accept the online payments from the users. The number of stakeholders of EG is growing and the current system is not scalable and they asked to built a highly scalable platform

ER Diagram



Class Diagram



IT20178154(Dilshan P.A.D.S.D)

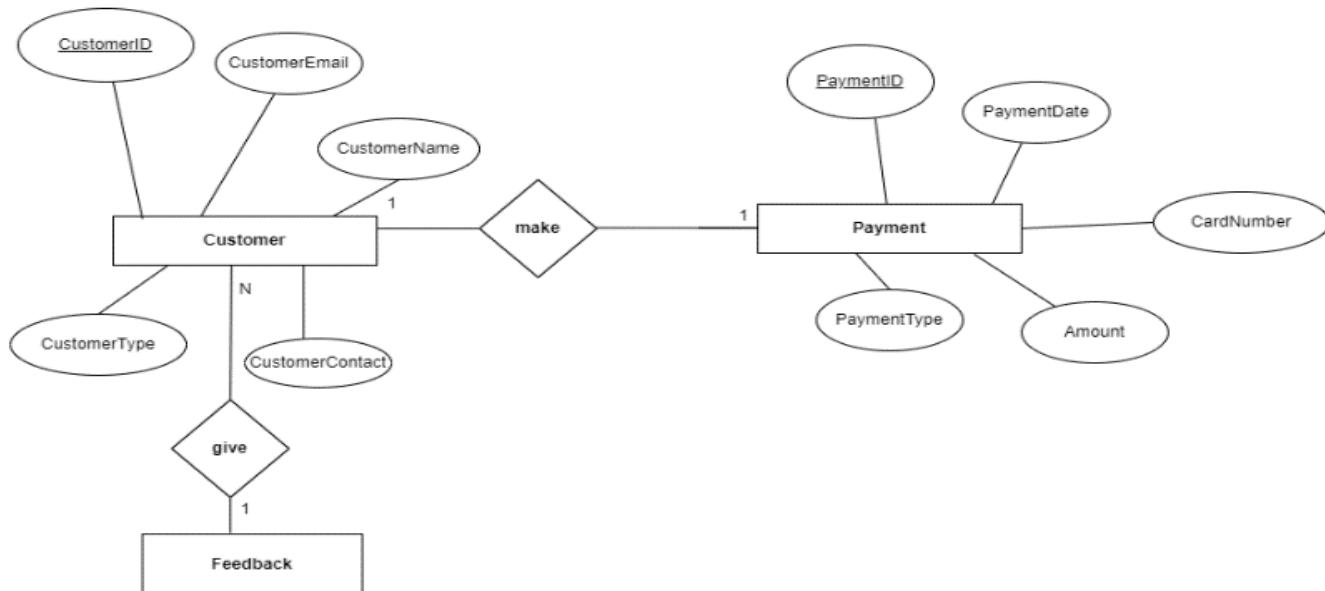
URL <http://localhost:8085/ElectroGrid/CustomerService/Customers>

<http://localhost:8085/ElectroGrid/PaymentService/Payments>

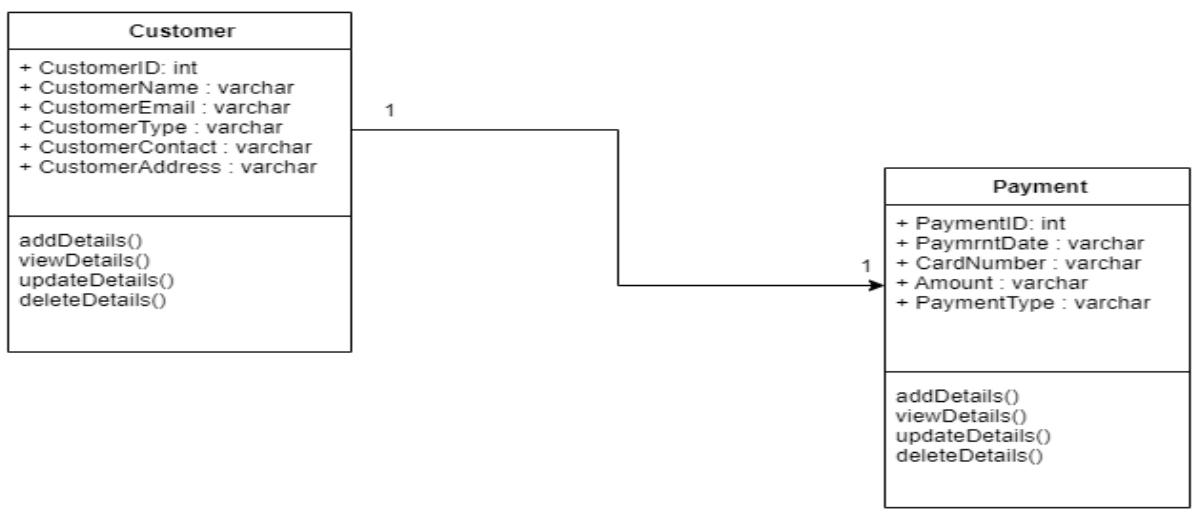
Customer Management and Payment Management

Here is the full scope of the customer and payment management. The requested parties can perform their function successfully through the system. In customer management customer can insert, update delete and view customer details and in payment management the parties can insert, update, delete and view the payment details.

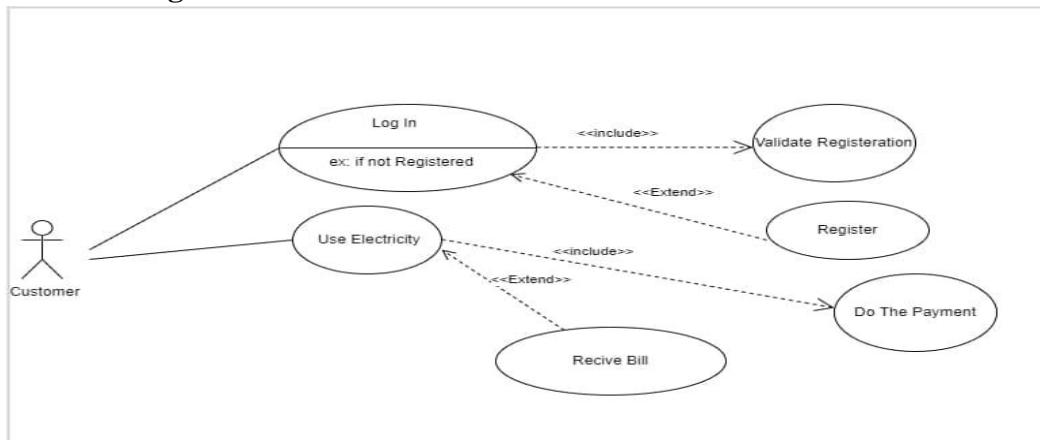
- Er Diagram



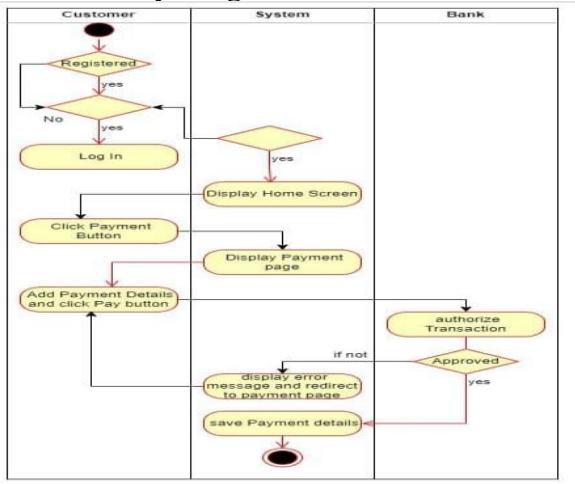
- Class Diagram



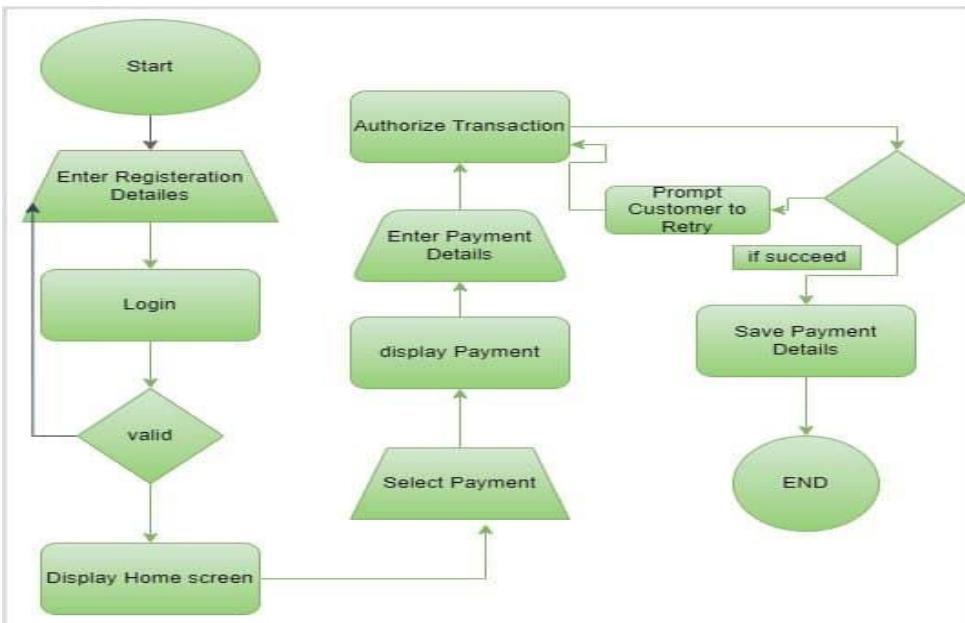
- **Usecase Diagram**



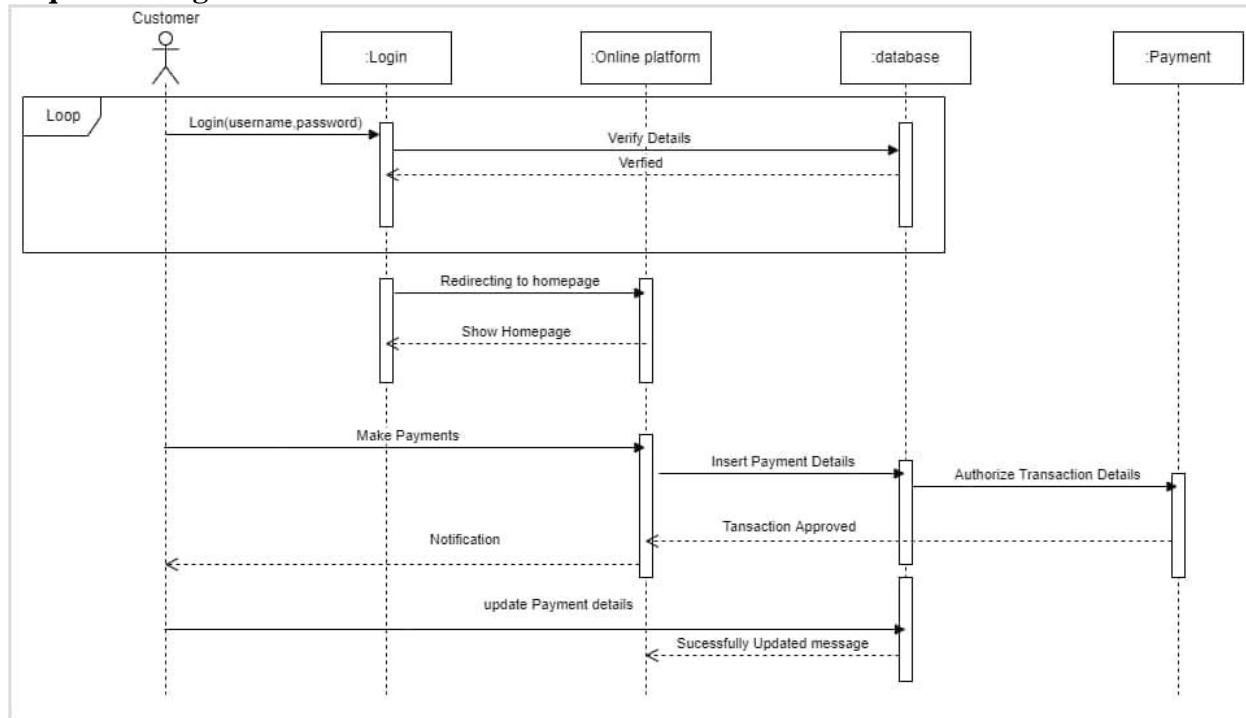
Activity Diagram



- **Flow Chart**

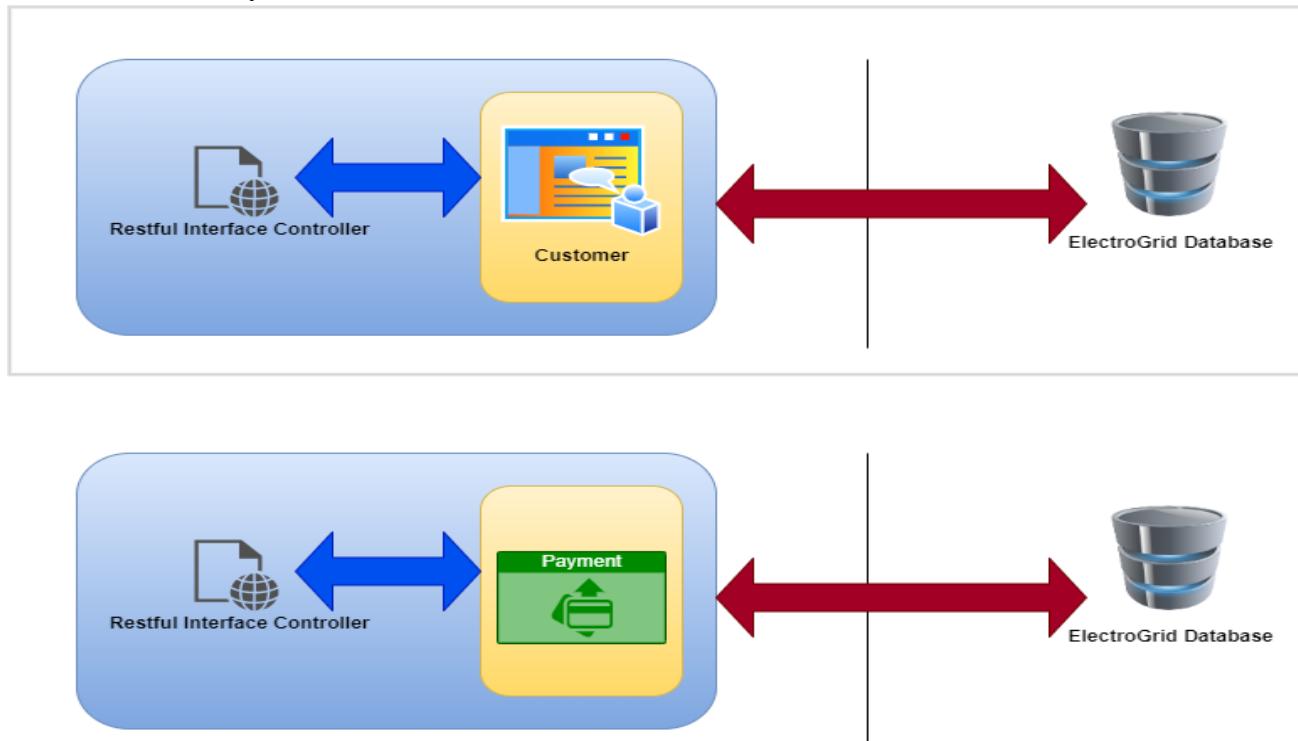


Sequence Diagram



Internal workflow

Customer and Payment



API Design Relational

Customer management and payment management is main part of this project. Customer can easily get the bills from the electrogrid system and they can easily get bills and pay their bills from the system. The system is user friendly and low complicated.

- **Customer can insert, delete, update, view the customer details.**
- **Payment details can be inserted, viewed, deleted, and updated by the respected party.**

Service Development

- Technologies used == Java-JAX-RS(Jersey) on Tomcat
- IDE == Eclipse
- Database == MySQL
- Testing == POSTMAN

API for get all the Customer Details in the database (GET Request)

URL:- <http://localhost:8085/ElectroGrid/CustomerService/Customers>

Request : - { }

Response : - { CustomerID : "Auto generated integer value" }
{CustomerName =:"sachin"}
{CustomerEmail :sachin@gmail.com }
{CustomerType : Old customer"}
{CustomerContact : "071513456"}
{CustomerAddress:"No 40, Malabe"}

API for insert a new Customers to the database (POST Request)

URL:- <http://localhost:8085/ElectroGrid/CustomerService/Customers>

Request : - { CustomerID : "Auto generated integer value" }
{CustomerName =:"sachin d"}
{CustomerEmail : Sachind@gmail.com }
{CustomerType : old customer"}
{CustomerContact : "075123456"}
{CustomerAddress:"No 40, Malabe"}

Response:- { Result = "Inserted successfully" }
{CustomerID = "Auto generated integer value"
{Error = "Error while inserting"}

API for update a customer which is existing in database (PUT Request)

URL:- <http://localhost:8085/ElectroGrid/CustomerService/Customers>

Request : - { CustomerID : "Auto generated integer value" }
{CustomerName =:"sachin d"}
{CustomerEmail : Sachind@gmail.com }
{CustomerType : old customer"}
{CustomerContact : "075123456"}
{CustomerAddress:"No 40, Malabe"}

Response:- { Result = "Updated successfully" }
{CustomerID = "Auto generated integer value"
{Error= "Error while Updating"}

API for delete a Customers which is existing in database (DELETE Request)

URL:- <http://localhost:8085/ElectroGrid/CustomerService/Customers>

Request :- { CustomerID = " Selected Customer ID from the service " }

Response :- { Result = "Deleted successfully" }
{ Error = "Error while deleting the hospital" }

API for get all the Payment details in the database (GET Request)

URL: - <http://localhost:8085/ElectroGrid/PaymentService/Payments>

Request: - { }

Response: - { PaymentID = "Auto generated integer value" }
{ PaymentDate" = "12.09.2017" }
{ CardNumber = "12334335454664" }
{ Amount = "130000.00 " }
{ PaymentType = "Cash" }

API for insert a new Payment to the database (POST Request)

URL: - <http://localhost:8085/ElectroGrid/PaymentService/Payments>

Request: - { PaymentID: "Auto generated integer value" }
{ PaymentDate" = "13.11.2021" }
{ CardNumber = "1213123434" }
{ Amount = "1234.00 " }
{ PaymentType = "Credit" }

Response: - { Result = "Inserted successfully" }
{ PaymentID = "Auto generated integer value" }
{ Error= "Error while inserting" }

API for update a Payment which is existing in database (PUT Request)

URL: - <http://localhost:8085/ElectroGrid/PaymentService/Payments>

Request : - { PaymentID: "Auto generated integer value" }
 { PaymentDate" = "12.03.2022" }
 { CardNumber = "1121233123" }
 { Amount = "123450.00 " }
 { PaymentType = "Debit Card" }

Response: - { Result = "Updated successfully" }
{ PaymentID = "Auto generated integer value" }
{ Error= "Error while Updating " }

API for delete a Payment which is existing in database (DELETE Request)

URL: - <http://localhost:8085/ElectroGrid/PaymentService/Payments>

Request :- { PaymentID = " Selected PaymentID from the service "}

Response :- {Result = "Deleted successfully" }

{Error = "Error while deleting the Payment Details" }

Test Cases TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Customer Details	{CustomerName =:"sachin d"} {CustomerEmail : Sachind@gmail.com } {CustomerType : old customer"} {CustomerContact : "075123456"} {CustomerAddress:"No 40, Malabe"}	New Customer details are added to the database. Show message as "Inserted successfully".	New Customer details are added to the database. Show message as "Inserted successfully".	PASS
2	Update Customer Details	{CustomerName =:"sachin d"} {CustomerEmail : Sachind@gmail.com } {CustomerType : old customer"} {CustomerContact : "075123456"} {CustomerAddress:"No 40, Malabe"}	Customer details are updated according to relevant input Customer details. Show message as "Updated successfully".	Customer details are updated according to relevant input Customer details. Show message as "Updated successfully".	PASS
3	Delete Customer Details	<customerData> <CustomerID>1</CustomerID> </customerData>	Show message as "Deleted Successfully"	Customer Details are deleted successfully. Show message as "Deleted Successfully"	PASS

TestID	Test Description/Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Payment Details	{ PaymentDate = "13.11.2021"} {CardNumber = "1213123434"} {Amount = "1234.00"} {PaymentType = "Credit"}	New Payment details are added to the database. Show message as "Inserted successfully".	New Payment details are added to the database. Show message as "Inserted successfully".	PASS
2	Update Payment Details	{ PaymentDate = "12.03.2022"} {CardNumber = "1121233123"} {Amount = "123450.00"} {PaymentType = "Debit Card"}	Payment details are updated according to relevant input Payment details. Show message as "Updated successfully".	Customer details are updated according to relevant input Payment details. Show message as "Updated successfully".	PASS
3	Delete Payment Details	<paymentData> <PaymentID>1</PaymentID> </paymentData>	Show message as "Deleted Successfully"	Payments Details are deleted successfully. Show message as "Deleted Successfully"	PASS

Testing

Customer Management

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with collections like 'check', 'Paf', 'BILL', 'ELECTRICITY', and 'CUSTOMER'. Under 'CUSTOMER', there are several API endpoints listed. The main workspace shows a 'GET' request to 'http://localhost:8085/ElectroGrid/CustomerService/Customers'. The 'Params' tab shows a single parameter 'Key' with value 'Value'. The 'Body' tab is empty. The 'Tests' tab contains a script: `if(pm.response.code == 200){pm.response.text = pm.response.text.replace(/\n/g, "
");}`. The 'Settings' tab is collapsed. Below the request, the 'Body' tab is selected, showing the response body as a table:

Customer Name	Customer Email	Customer Type	Customer Contact	Customer Address
Dil	dil@gmail.com	Old Customer	076305153	No 90
sachin Dileepa	sachinD@gmail.com	New Customer	076305154	no 90
anjalee	anjalee@gmail.com	Old	087382345	No 40, Malabe
SHASHINTHA	anjalee@gmail.com	Old	087382345	No 40, Malabe

The status bar at the bottom indicates 'Status: 200 OK Time: 2.15 s Size: 808 B Save Response'.

Three screenshots of the Postman API testing tool interface, showing the execution of POST, PUT, and DELETE requests to the 'Customer' endpoint of the 'ElectroGrid' service.

POST Request (Screenshot 1):

URL: `http://localhost:8085/ElectroGrid/CustomerService/Customers`

Body (x-www-form-urlencoded):

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> CustomerID	5	
<input checked="" type="checkbox"/> CustomerName	sachin d	
<input checked="" type="checkbox"/> CustomerEmail	sachind@gmail.com	
<input checked="" type="checkbox"/> CustomerType	Old customer	
<input checked="" type="checkbox"/> CustomerContact	075123456	
<input checked="" type="checkbox"/> CustomerAddress	No 40, Malabe	

Response:

```
Inserted successfully
```

PUT Request (Screenshot 2):

URL: `http://localhost:8085/ElectroGrid/CustomerService/Customers`

Body (JSON):

```
{
  "CustomerID": "2",
  "CustomerName": "sachind",
  "CustomerEmail": "sachind@gmail.com",
  "CustomerType": "Old customer",
  "CustomerContact": "075123456",
  "CustomerAddress": "No 40, Malabe"
}
```

Response:

```
Updated successfully
```

DELETE Request (Screenshot 3):

URL: `http://localhost:8085/ElectroGrid/CustomerService/Customers`

Body (XML):

```
<CustomerData>
  <CustomerID>1</CustomerID>
</CustomerData>
```

Response:

```
Deleted successfully
```

Payment Management

The screenshot shows the Postman interface with a successful GET request to `http://localhost:8085/ElectroGrid/PaymentService/Payments`. The response body displays a table of payment records:

Payment Date	Card Number	Amount	Payment Type
12.03.2022	1121233123	123450.00	Debit Card
13.11.2021	1213123434	1234.00	Credit

The screenshot shows the Postman interface with a successful POST request to `http://localhost:8085/ElectroGrid/PaymentService/Payments`. The request body contains the following data:

KEY	VALUE	DESCRIPTION
PaymentID	3	
PaymentDate	13.11.2021	
CardNumber	1213123434	
Amount	1234.00	
PaymentType	Credit	

The response body indicates: "Inserted successfully".

The screenshot shows the Postman application interface. A collection named 'check' is selected on the left. In the main workspace, a PUT request is made to `http://localhost:8085/ElectroGrid/PaymentService/Payments`. The request body is a JSON object:

```
1 {  
2   "PaymentID": "2",  
3   "PaymentDate": "12.03.2022",  
4   "CardNumber": "1121233123",  
5   "Amount": "123450.00",  
6   "PaymentType": "Debit Card"  
7 }
```

The response status is 200 OK, and the message is "Updated successfully".

The screenshot shows the Postman application interface. A collection named 'check' is selected on the left. In the main workspace, a DELETE request is made to `http://localhost:8085/ElectroGrid/PaymentService/Payments`. The request body is a XML object:

```
1 <paymentData>  
2   <PaymentID>3</PaymentID>  
3 </paymentData>
```

The response status is 200 OK, and the message is "Deleted successfully".

IT20140366 (Tharushika Devindi M.K.S.)

<http://localhost:8085/ElectroGrid/BillService/Bills>

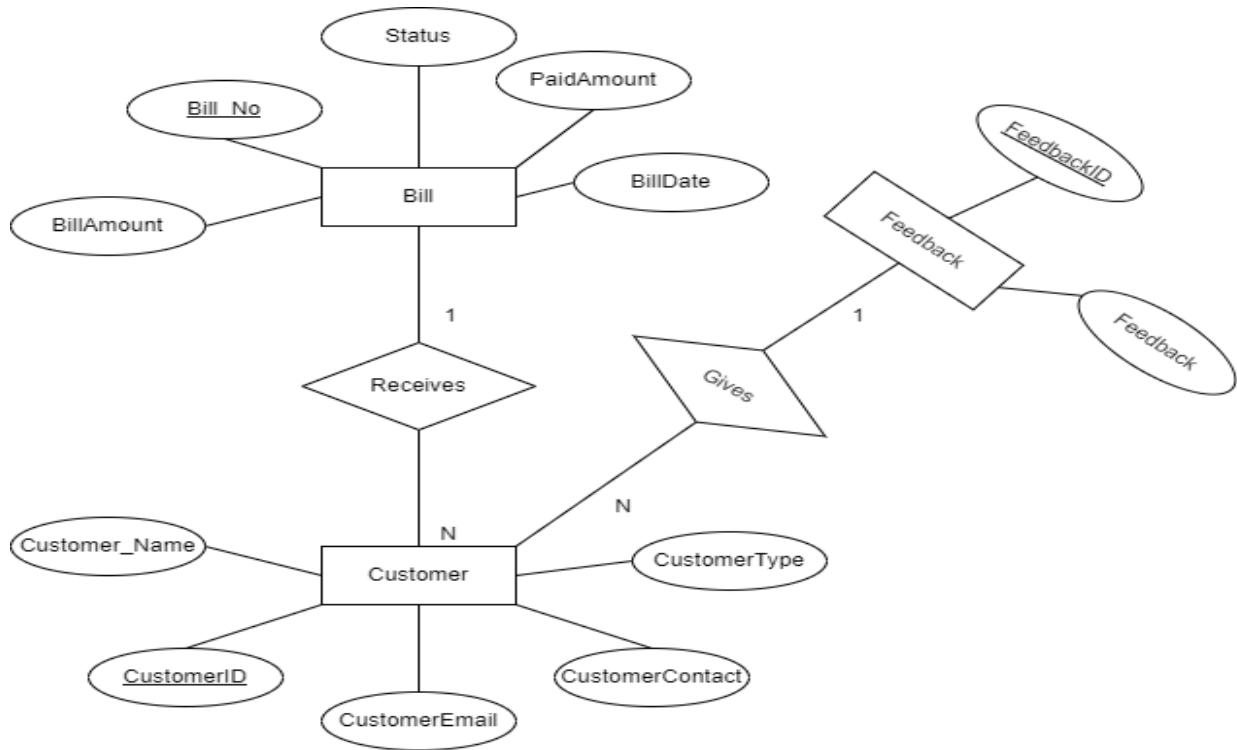
<http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks>

Bill Management and Feedback Management

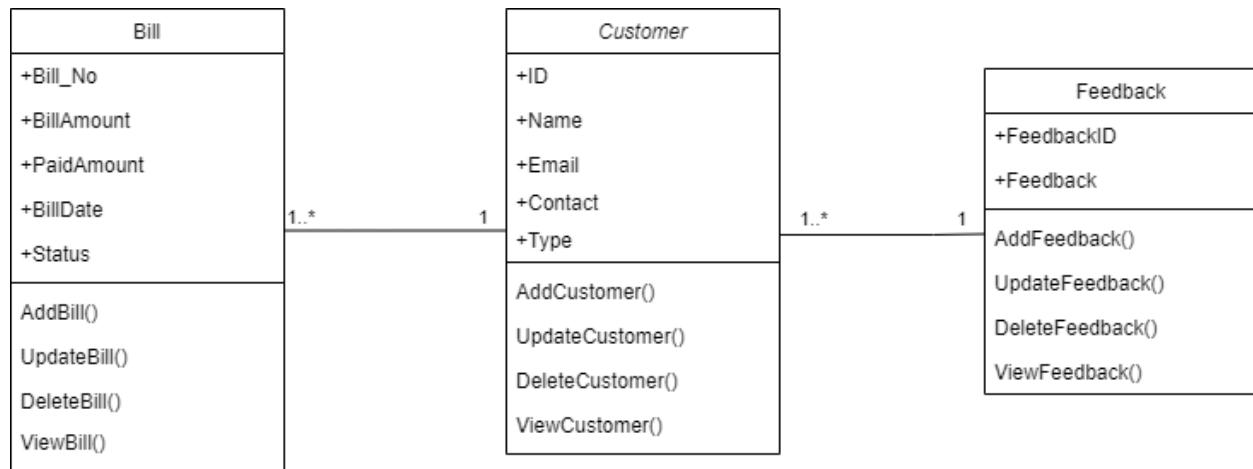
Energy consumption is the amount of energy used per unit time. ElectroGrid's system calculates how much electricity consumers electricity consume per month and issues bills for it. In addition, those bills are automatically sent to users. Through the system can enter, update, and delete bill details to those bills as well as view those bills.

Customers can provide feedback about the service provided by ElectroGrid, where they can add, update, delete and review feedback.

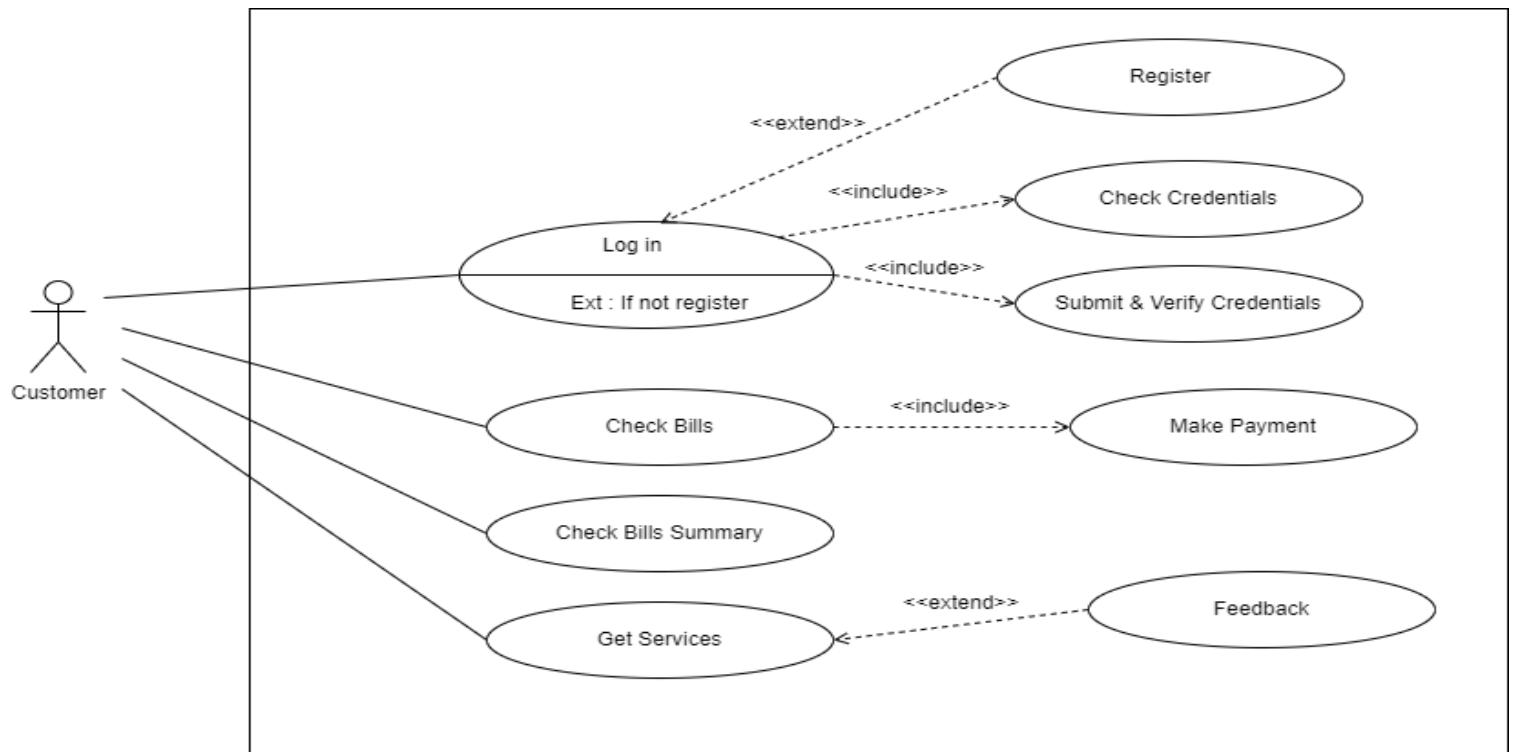
- **ER Diagram**



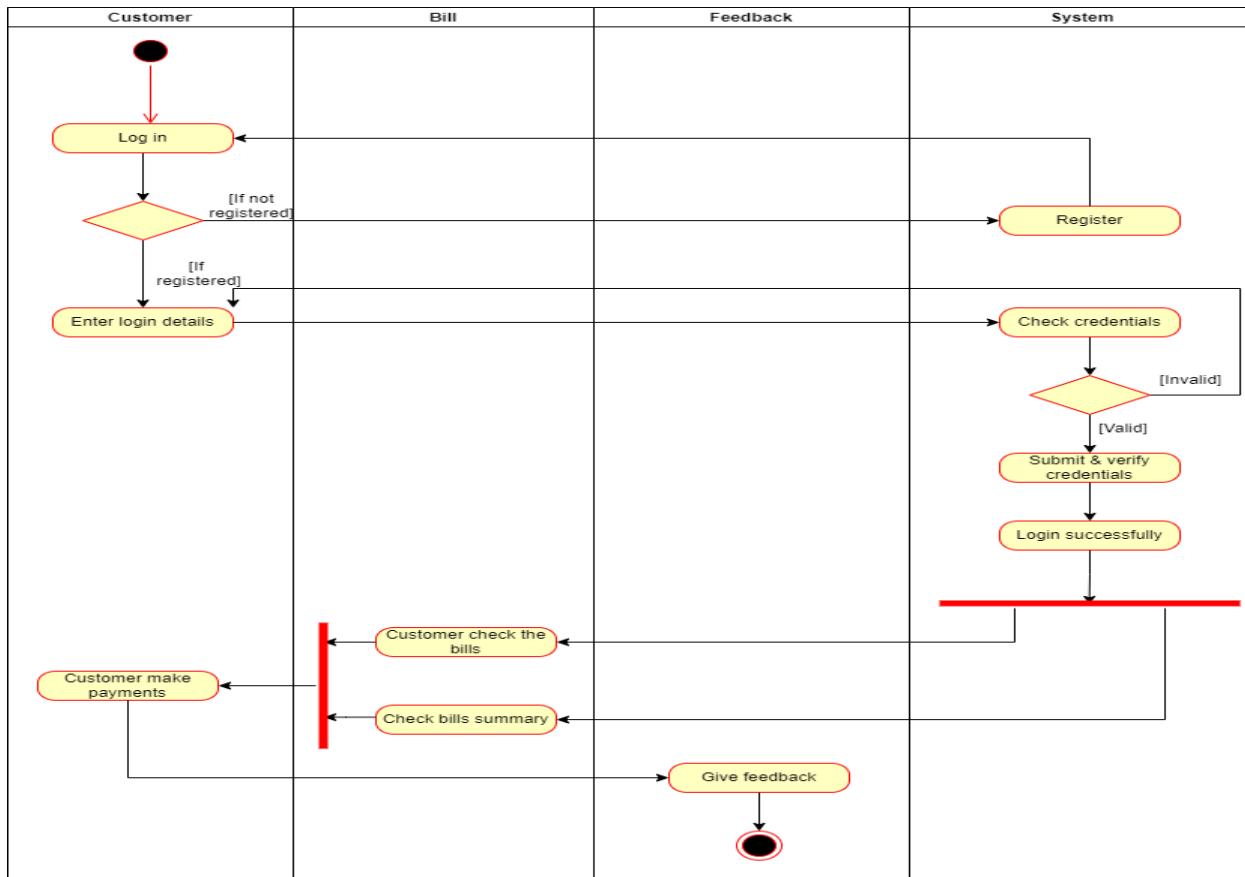
- **Class Diagram**



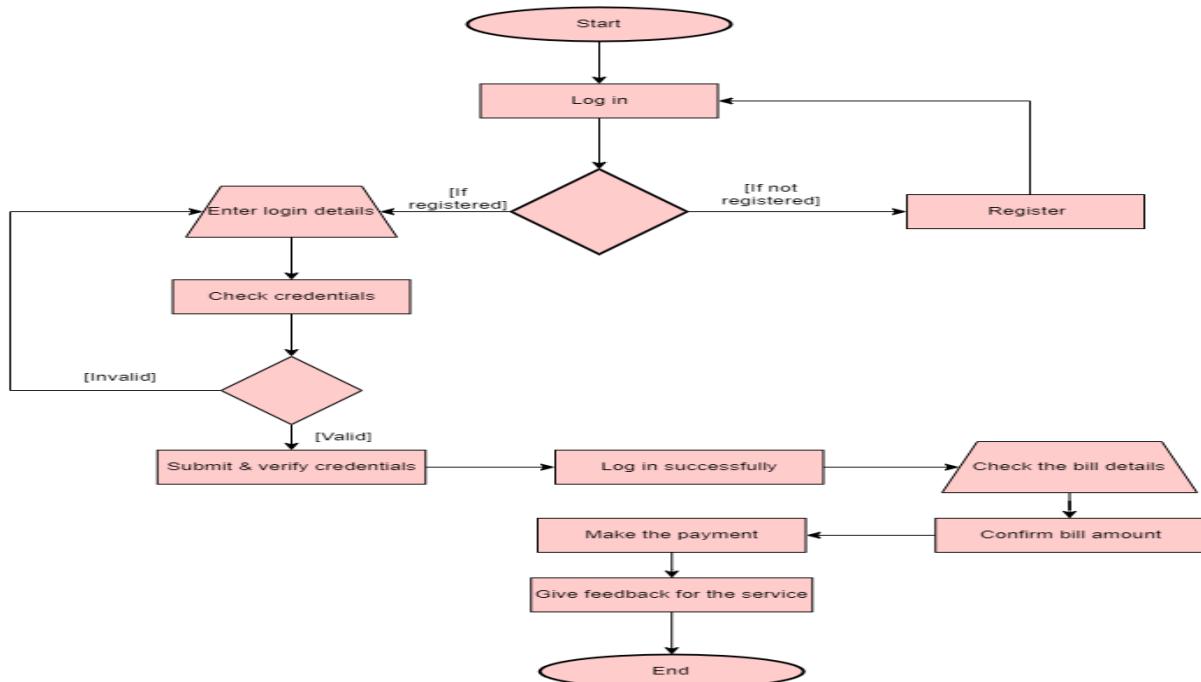
- **Usecase Diagram**



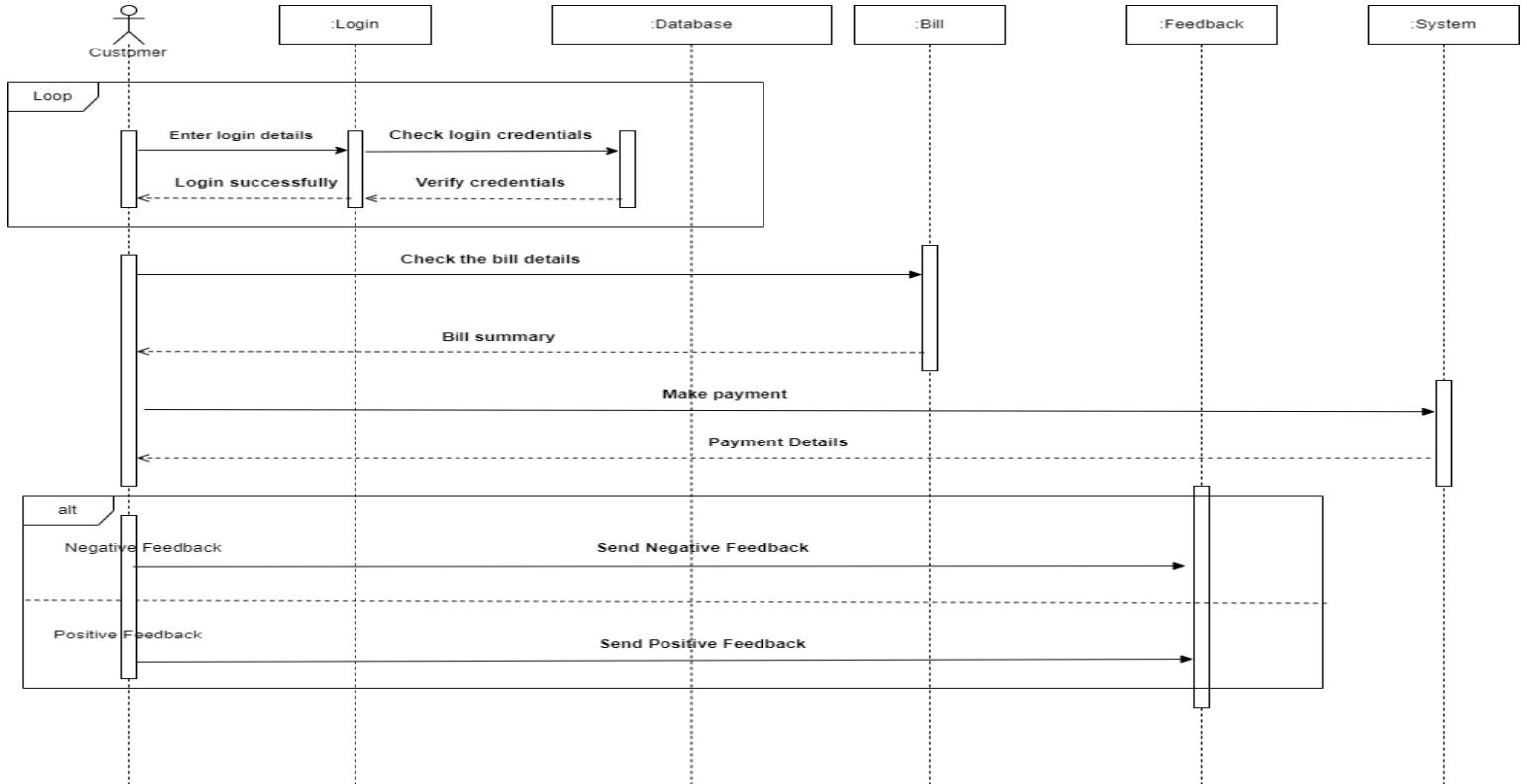
- **Activity Diagram**



- **Flow Chart**

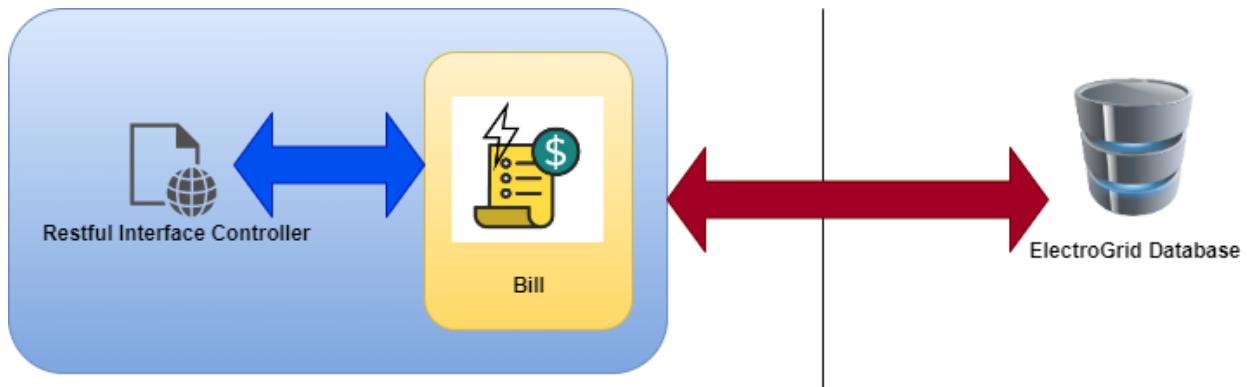


- Sequence Diagram

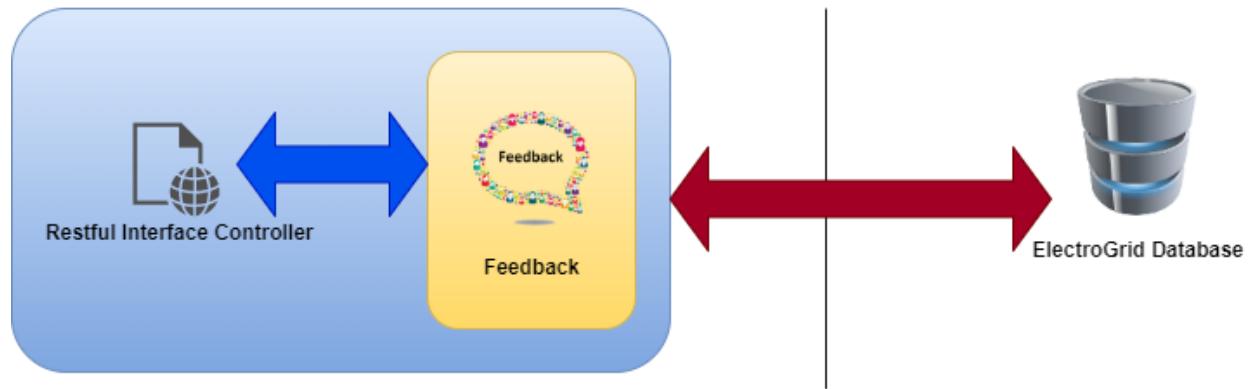


Internal Workflow

Bill:



Feedback:



API Design Relational

ElectroGrid's system calculates how much electricity consumers electricity consume per month and issues bills for it. Customers can provide feedback about the service provided by ElectroGrid. The bill management and feedback management is user friendly and easy to understand with less complication.

- System admin can enter, update, and delete bill details to those bills as well as view those bills.
- Customer can add, update, delete and review feedback.

Service Development

- Technologies used : Java-JAX-RS(Jersey) on Tomcat
- IDE : Eclipse
- Database : MySQL
- Testing : POSTMAN

- **Bill Management**

API for get all the Bill Details in the database (GET Request)

URL:- <http://localhost:8085/ElectroGrid/BillService/Bills>

Request :- { }

Response :- {Billno : "Auto generated integer value"
 {Billamount : 2500.00 }
 {Billdate : "12/11/2021"}
 {Paidamount : 1500.00 }
 {Status : "paid"}}

API for insert a new Bill Details to the database (POST Request)

URL:- <http://localhost:8085/ElectroGrid/BillService/Bills>

Request :-

```
{Billno      : "Auto generated integer value"
{Billamount : 2500.00 }
{Billdate    : "12/11/2021"}
{Paidamount  : 1500.00 }
{Status      : " paid "}
```

Response:-

```
{ Result   = "Inserted successfully" }
{Billno    = "Auto generated integer value" }
{Error     = "Error while inserting"}
```

API for update a Bill Details which is existing in database (PUT Request)

URL:- <http://localhost:8085/ElectroGrid/BillService/Bills>

Request :-

```
{Billno      : "Auto generated integer value"
{Billamount : 3000.00 }
{Billdate    : "10/11/2021"}
{Paidamount  : 1500.00 }
{Status      : " not paid "}
```

Response:-

```
{ Result   = "Updated successfully" }
{Billno    = "Auto generated integer value" }
{Error     = "Error while Updating"}
```

API for delete Bill Details which is existing in database (DELETE Request)

URL:- <http://localhost:8085/ElectroGrid/BillService/Bills>

Request :-

```
{ Billno = " Selected Billno from the service " }
```

Response :-

```
{ Result   = "Deleted successfully" }
{Error     = "Error while deleting the bill details"}
```

• Feedback Management

API for get all the Feedback Details in the database (GET Request)

URL:- <http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks>

Request :- { }

Response :- { FeedbackID : "Auto generated integer value" }
 { Feedback : " Good " }

API for insert a new Feedback Details to the database (POST Request)

URL:- <http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks>

Request :- { FeedbackID : "Auto generated integer value" }
 { Feedback : " Good " }

Response:- { Result = "Inserted successfully" }
 { FeedbackID = "Auto generated integer value" }
 { Error = "Error while inserting" }

API for update a Feedback Details which is existing in database (PUT Request)

URL:- <http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks>

Request :- { FeedbackID : "Auto generated integer value" }
 { Feedback : " bad " }

Response:- { Result = "Updated successfully" }
 { FeedbackID = "Auto generated integer value" }
 { Error = "Error while Updating" }

API for delete Feedback Details which is existing in database (DELETE Request)

URL:- <http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks>

Request :- { FeedbackID = " Selected FeedbackID from the service " }

Response :- { Result = "Deleted successfully" }
 { Error = "Error while deleting the feedback details" }

Test Cases TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Bill Details	{BillNo : "Auto generated integer value"} {BillAmount : 960.75 } {PaidAmount : 1000.00 } {BillStatus : "Electricity Bill"} {BillDate : "15/04/2022"}	New Bill details are added to the database. Show message as "Inserted successfully".	New Bill details are added to the database. Show message as "Inserted successfully".	PASS
2	Update Bill Details	{BillNo : "Auto generated integer value"} {BillAmount : 960.75 } {PaidAmount : 1000.00 } {BillStatus : "Internet Bill"} {BillDate : "15/04/2022"}	Bill details are updated according to relevant input Bill details. Show message as "Updated successfully".	Bill details are updated according to relevant input Bill details. Show message as "Updated successfully".	PASS
3	Delete Bill Details	<billData> <BillNo >1</ BillNo > </billData>	Show message as "Deleted Successfully"	Bill Details are deleted successfully. Show message as "Deleted Successfully"	PASS

Test Cases TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)

1	Add Feedback Details	<pre>{FeedbackID : "Auto generated integer value"} {Feedback : Excellent customer service. Thank you for the amazing support!}</pre>	New feedback details are added to the database. Show message as "Inserted Successfully".	New feedback details are added to the database. Show message as "Inserted Successfully".	PASS
2	Update Feedback Details	<pre>{FeedbackID : "Auto generated integer value"} {Feedback : Excellent customer service at ElectroGrid! Thank you for the amazing support!}</pre>	Feedback details are updated according to relevant input Bill details. Show message as "Updated successfully".	Feedback details are updated according to relevant input Bill details. Show message as "Updated successfully".	PASS
3	Delete Bill Details	<pre>< feedbackData> < FeedbackID >1</ FeedbackID > </ feedbackData></pre>	Show message as "Deleted Successfully"	Feedback Details are deleted successfully. Show message as "Deleted Successfully"	PASS

Testing

Bill Management

Contributors to SachinD99/RAF X http://localhost:8085/ElectroGrid X

GET http://localhost:8085/Bills POST http://localhost:8085/Bills + ***

GET /Paf / BILL / http://localhost:8085/ElectroGrid/BillService/Bills

Headers (7)

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

Customer ID	Bill Amount	Bill Date	Paid Amount	Status
3	3000	10.11.2021	1500	not paid
4	2500	12.11.2021	1500	paid

Console

Cookies Bootcamp Desktop Agent Runner Trash 10:00 AM 4/24/2022

Contributors to SachinD99/RAF X http://localhost:8085/ElectroGrid X

POST http://localhost:8085/Bills POST http://localhost:8085/Bills + ***

POST /Paf / BILL / http://localhost:8085/ElectroGrid/BillService/Bills

Headers (10)

Key	Value
Billno	2
CustomerID	4
Billamount	2500.00
Billdate	12.11.2021
Paidamount	1500.00
Status	paid

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

Inserted successfully.

Console

Cookies Bootcamp Desktop Agent Runner Trash 10:02 AM 4/24/2022

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar is visible with a 'Collections' section containing a 'check' folder which has a 'Paf' folder and a 'BILL' folder. The 'BILL' folder contains several requests: GET http://localhost:8085/ElectroGrid/BillService/Bills, POST http://localhost:8085/ElectroGrid/BillService/Bills, PUT http://localhost:8085/ElectroGrid/BillService/Bills, and others. The main workspace shows a 'PUT' request to 'http://localhost:8085/ElectroGrid/BillService/Bills'. The 'Body' tab is selected, showing a JSON payload:

```
1   {
2     "Billno": "1",
3     "CustomerID": "3",
4     "Billamount": "3000.00",
5     "Billdate": "10.11.2021",
6     "Paidamount": "1500.00",
7     "Status": "not paid"
8 }
```

The status bar at the bottom indicates 'Status: 200 OK'.

This screenshot shows the same Postman interface as the previous one. The 'My Workspace' sidebar is identical. In the main workspace, a 'DELETE' request is shown to 'http://localhost:8085/ElectroGrid/BillService/Bills'. The 'Body' tab is selected, showing XML payload:

```
1 <BillData>
2   <Billno>2</Billno>
3 </BillData>
```

The status bar at the bottom indicates 'Status: 200 OK'.

Feedback Management

http://localhost:8085/ElectroGrid

GET / Paf / FEEDBACK / http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks

Customer Name | customer Email | Customer Feedback

Sachin	sachind@gmail.com	Good
Sachin	sachind@gmail.com	Good
Sachin	sachind@gmail.com	Good

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 10 ms Size: 432 B Save Response

http://localhost:8085/ElectroGrid

POST / Paf / FEEDBACK / http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks

FeedbackID | CustomerName | CustomerEmail | Feedback

1	Sachin	sachind@gmail.com	Good
---	--------	-------------------	------

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 90 ms Size: 179 B Save Response

Inserted successfully

The screenshot shows the Postman application interface. On the left, the 'My Workspaces' sidebar is visible, containing collections like 'check', 'Paf', 'BILL', 'ELECTRICITY', 'CUSTOMER', 'FEEDBACK', 'PAYMENT', 'POWERSTATION', 'EMPLOYEE', 'CONSUMPTION', and several 'Electro...' items. The main workspace is titled 'check' and contains a 'Paf' collection. Under 'Paf', there is a 'FEEDBACK' folder which includes a 'PUT' request for 'http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "FeedbackID": "8",
3   "CustomerName": "Dil",
4   "CustomerEmail": "dil@gmail.com",
5   "Feedback": "bad"
6 }
```

Below the body, the response status is shown as 'Status: 200 OK Time: 968 ms Size: 178 B'. The response body contains the message 'Updated successfully'.

This screenshot shows the same Postman interface as the first one, but with a different request. The 'Body' tab is now selected for a 'DELETE' request to 'http://localhost:8085/ElectroGrid/FeedbackService/Feedbacks'. The body contains XML data:

```
1 <FeedbackData>
2   <FeedbackID>8</FeedbackID>
3 </FeedbackData>
```

The response status is 'Status: 200 OK Time: 38 ms Size: 178 B'. The response body contains the message 'Deleted successfully'.

IT20139544 (Kapukotuwa S.A.A.H)

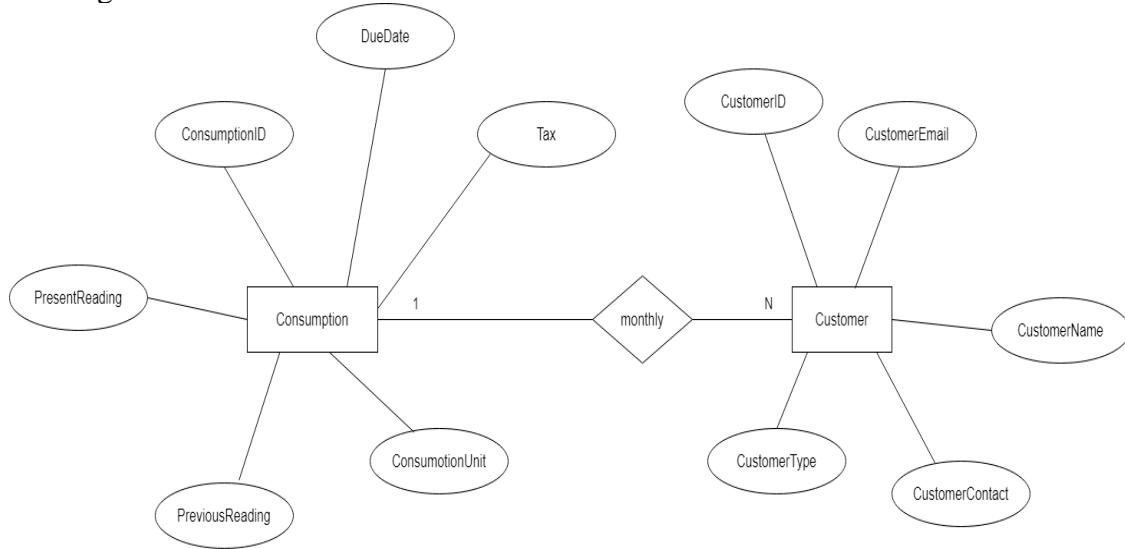
URL:

<http://localhost:8085/ElectroGrid/PowerConsumptionService/Consumptions>

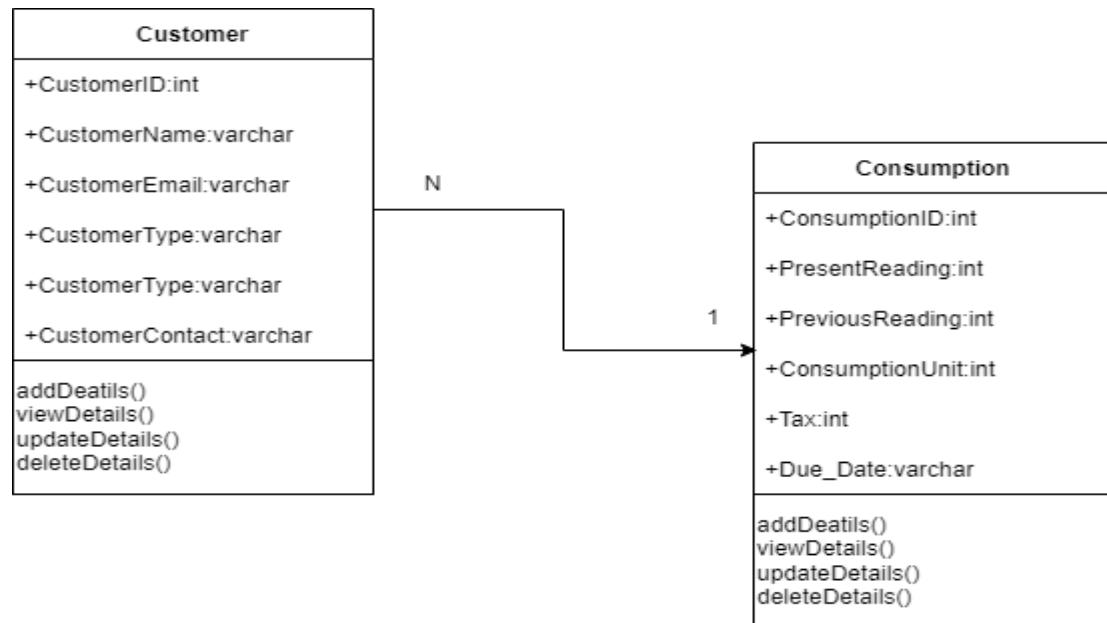
Consumption Management

Energy consumption is the amount of energy used per unit time. So this duty is assigned to consumption management. Here is the full scope of consumption management. Through this system user can enter, update and delete the details.

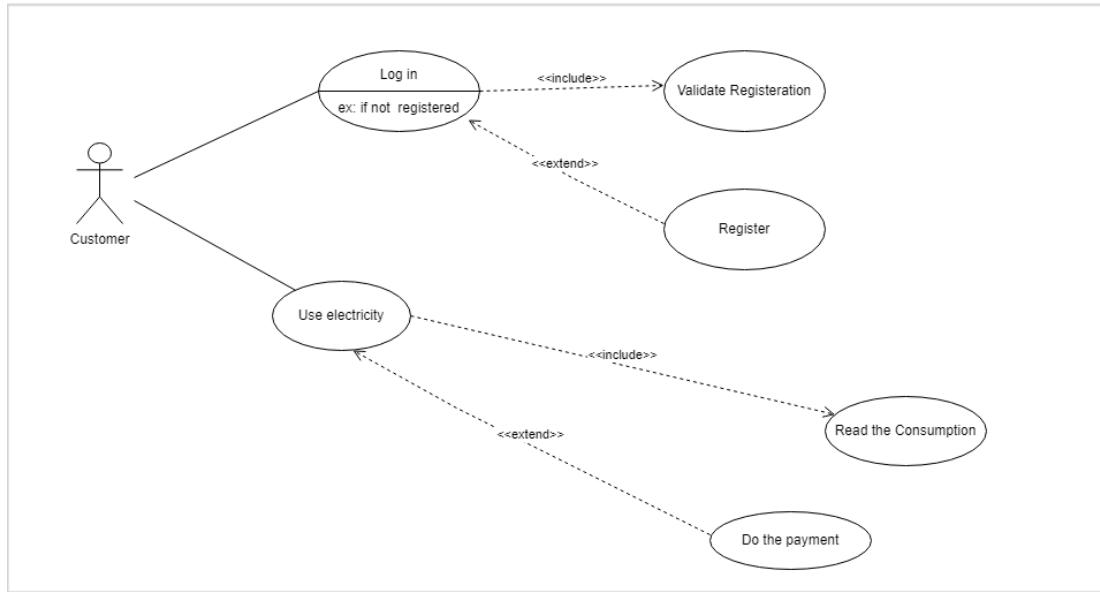
- ER diagram



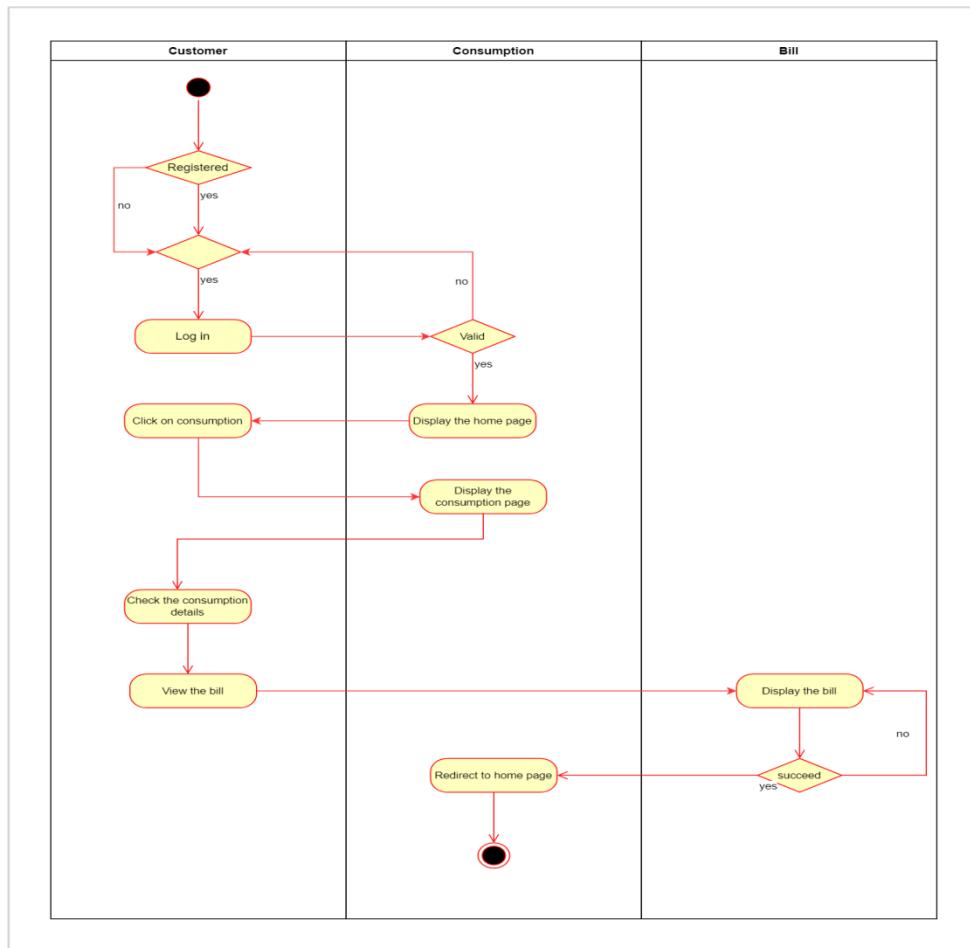
- Class Diagram



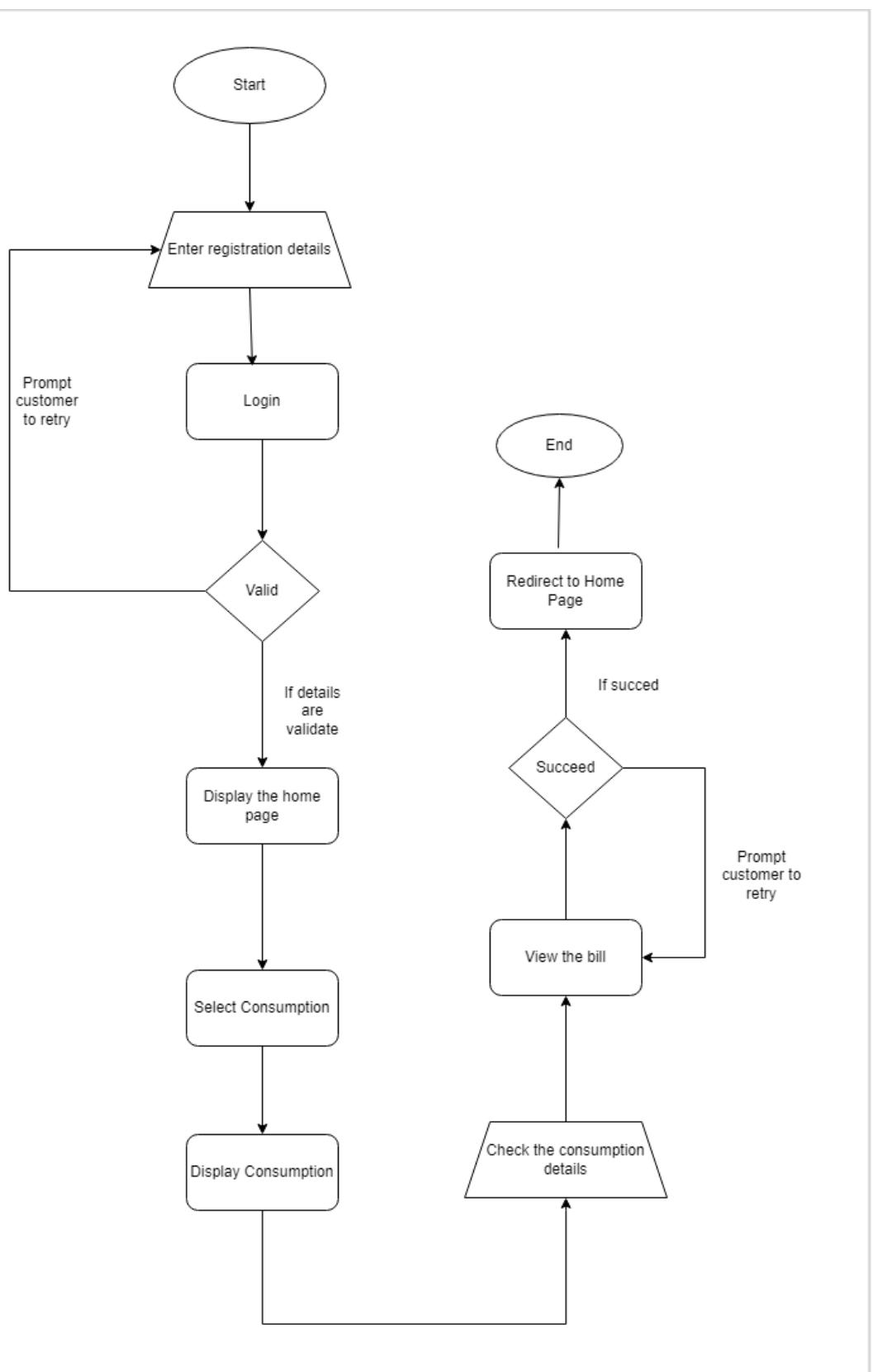
Usecase Diagram

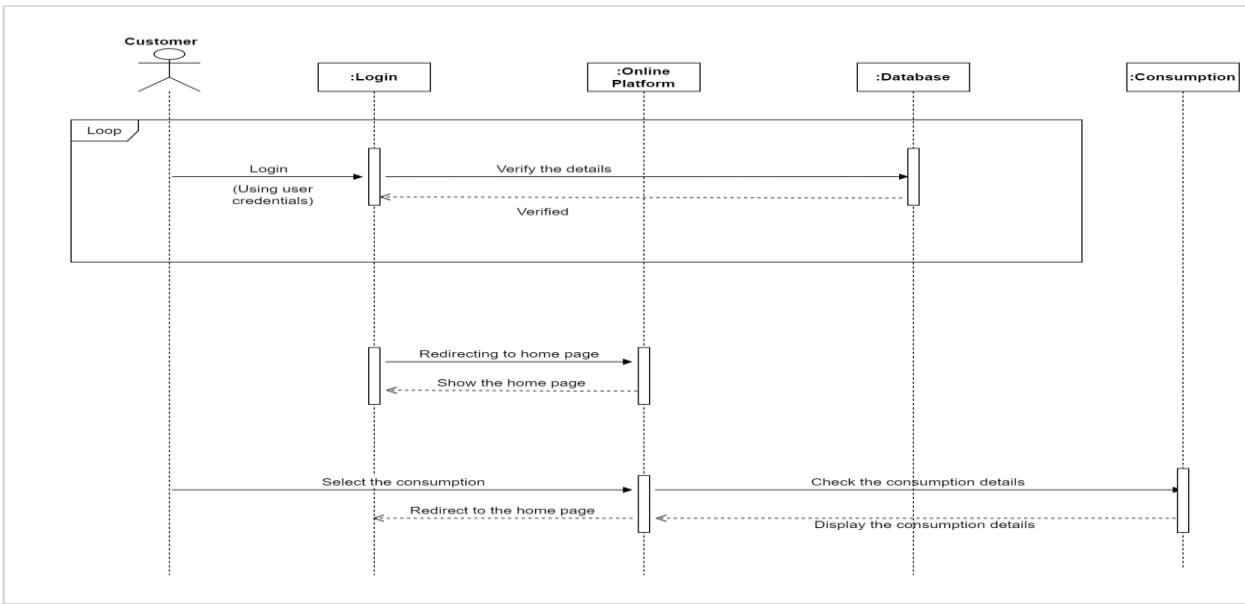


• Activity Diagram



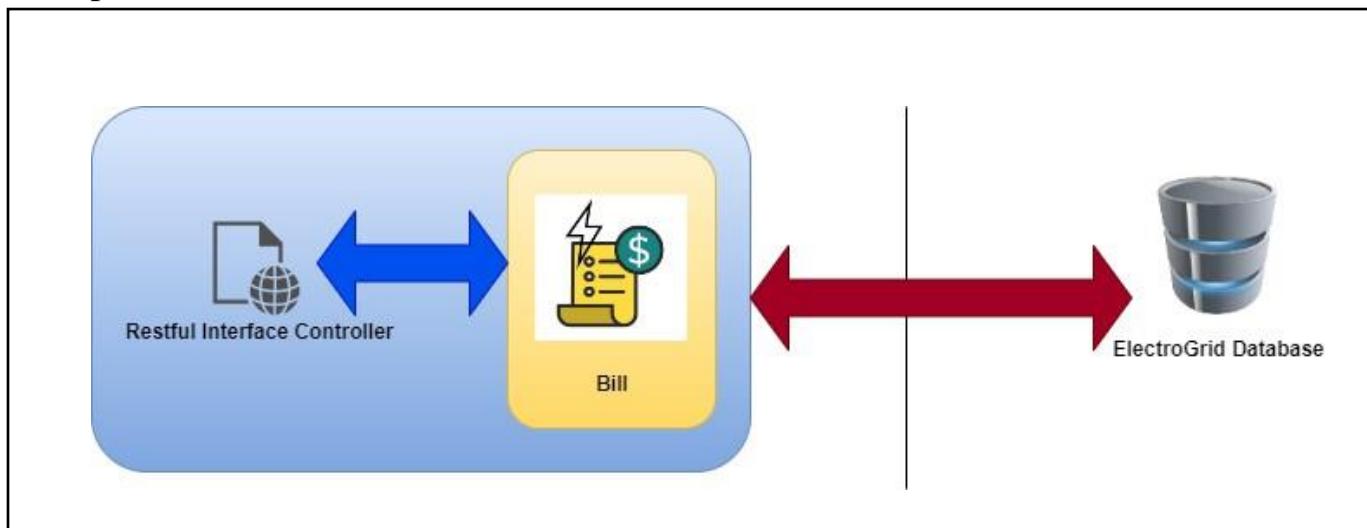
- Flow Chart





Internal workflow

Consumption:



Design Relational

Consumption management system is user friendly and low complicated. Present and previous unit readings, due dates and tax which charged for unit is managed properly by this system.

- Consumption details can be inserted, viewed, updated and deleted by user.

- Technologies used == Java-JAX-RS(Jersey) on Tomcat
- IDE == Eclipse
- Database == MySQL
- Testing == POSTMAN

API for get all the Consumption Details in the database (GET Request)

URL:-

<http://localhost:8085/ElectroGrid/PowerConsumptionService/Consumptions>

Request : - { }

Response : - {ConsumptionID : "Auto generated integer value"}
{CustomerID : 1}
 {PresentReading :1000}
 {PreviousReading:855}
 {ConsumptionUnit: 235}
 {Tax : 2400.00}
 {DueDate:"18.02.2021"}

API for insert a new Consumption Details to the database (POST Request)

URL:- <http://localhost:8085/ElectroGrid/PowerConsumptionService/Consumptions>

Request : - {ConsumptionID : "Auto generated integer value"}
{CustomerID : 2}
 {PresentReading :1343}
 {PreviousReading:1000}
 {ConsumptionUnit: 235}
 {Tax : 2600.00}
 {DueDate:"20.03.2021"}

Response:- { Result = "Inserted successfully" }
{ConsumptionID = "Auto generated integer value"}
{Error = "Error while insertin

API for update a consumption detail which is existing in database (PUT Request)

URL:- <http://localhost:8085/ElectroGrid/PowerConsumptionService/Consumptions>

Request :- {ConsumptionID : "Auto generated integer value"}
{CustomerID :4}
{PresentReading :1200}
{PreviousReading:1000}
{ConsumptionUnit: 200}
{Tax : 3000.00}
{DueDate:"10.04.2025"}

Response:- { Result = "Updated successfully" }
{ConsumptionID = "Auto generated integer value"}
{Error= "Error while Updating"}

API for delete a consumption detail which is existing in database (DELETE Request)

URL:- <http://localhost:8085/ElectroGrid/PowerConsumptionService/Consumptions>

Request :- { ConsumptionID = " Selected Consumption ID from the service "}

Response :- { Result = "Deleted successfully" }
{Error = "Error while deleting the Consumption details"}

Test Cases TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s))	Result (Pass/Fail)
1	Add Consumption Details	{CustomerID :2} {PresentReading :1343} {PreviousReading : 1000} {ConsumptionUnit : 235} {Tax:2600.00} {DueDate:"20.03.2021"}	New Consumption details are added to the database. Show message as "Inserted successfully".	New Consumption details are added to the database. Show message as "Inserted successfully".	PASS
2	Update Customer Details	{CustomerID :4} {PresentReading :1200} {PreviousReading : 1000} {ConsumptionUnit : 200} {Tax:3000.00} {DueDate:"10.04.2025"}	Consumption details are updated according to relevant input Consumption details. Show message as "Updated successfully".	Consumption details are updated according to relevant input Consumption details. Show message as "Updated successfully".	PASS

			successfully".		
3	Delete Customer Details	<consumptionData> <ConsumptionID>1</ConsumptionID> </consumptionData>	Show message as "Deleted Successfully"	Consumption details are deleted successfully. Show message as "Deleted Successfully"	PASS

Testig

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'check', 'Paf', 'BILL', 'ELECTRICITY', 'CUSTOMER', 'FEEDBACK', 'PAYMENT', 'POWERSTATION', 'EMPLOYEE', and 'CONSUMPTION'. Under 'CONSUMPTION', there are four items: a green 'GET' button for 'http://localhost:8085/Electro...', a blue 'POST' button for 'http://localhost:8085/Electro...', a red 'PUT' button for 'http://localhost:8085/Electro...', and a blue 'DELETE' button for 'http://localhost:8085/Electro...'. The main area shows a 'GET http://localhost:8085/ElectroGrid/PowerconsumptionService/Consumptions' request. The 'Params' tab is selected, showing a single parameter 'Key' with value 'Value'. Below it, the 'Body' tab shows a JSON response:

```

{
    "Consumption ID": 2,
    "Customer ID": 2,
    "Present Reading": 1235,
    "Previous reading": 1000,
    "Consumption unit": 235,
    "Tax": 2600,
    "Due Date": "20.03.2022"
}

```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'check', 'Paf', 'BILL', 'ELECRICITY', 'CUSTOMER', 'FEEDBACK', 'PAYMENT', 'POWERSTATION', 'EMPLOYEE', and 'CONSUMPTION'. The 'CONSUMPTION' collection is expanded, showing several API endpoints. One endpoint, 'POST http://localhost:8085/ElectroGrid/PowerconsumptionService/Consumptions', is selected. The 'Body' tab is active, showing a JSON payload with fields: CustomerID (2), Present_reading (1343), Previous_reading (1000), Consumptionunit (235), Tax (2600.00), and Due_date (20.03.2021). The 'Preview' tab shows the response: 'Inserted successfully'. The status bar at the bottom indicates 'Status: 200 OK Time: 831 ms Size: 179 B Save Response'.

This screenshot shows another instance of the Postman application. The left sidebar has the same structure as the first one. The 'EMPLOYEE' collection is expanded, showing an endpoint 'PUT http://localhost:8085/ElectroGrid/EmployeeService/Employee'. The 'Body' tab is active, displaying a JSON payload with fields: ConsumptionID (1), CustomerID (4), Present_reading (1280), Previous_reading (1000), Consumptionunit (200), Tax (3000), and Due_date (18-04-25). The 'Preview' tab shows the response: 'Updated successfully'. The status bar at the bottom indicates 'Status: 200 OK Time: 66 ms Size: 178 B Save Response'.

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with a collection named 'check' containing several API endpoints under categories like BILL, ELECTRICITY, CUSTOMER, FEEDBACK, PAYMENT, POWERSTATION, EMPLOYEE, and CONSUMPTION. The main workspace shows a DELETE request to `http://localhost:8085/ElectroGrid/PowerconsumptionService/Consumptions`. The 'Body' tab is selected, showing the following XML payload:

```
1 <consumptionData>
2   <ConsumptionID>1</ConsumptionID>
3 </consumptionData>
```

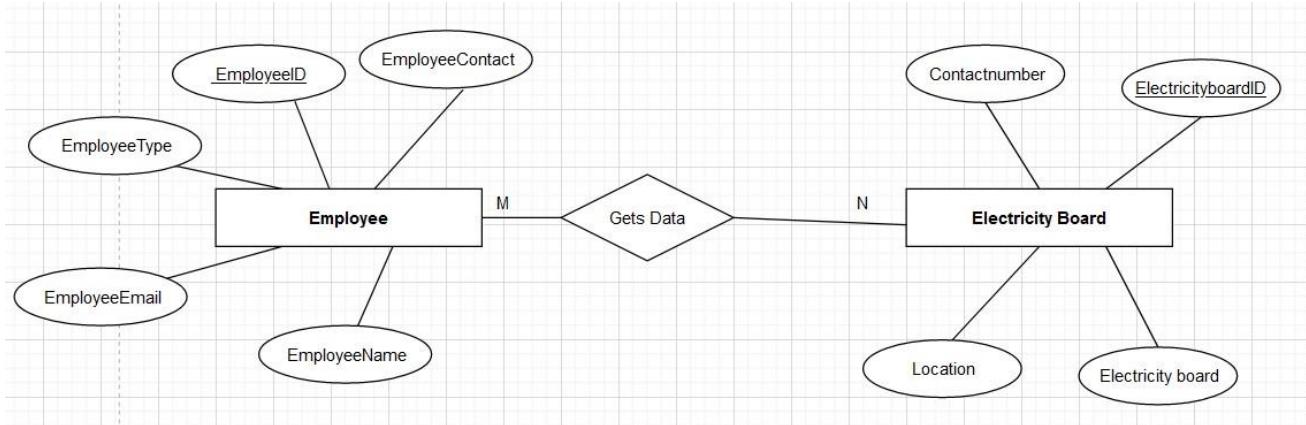
The response section indicates a successful deletion with the message "Deleted successfully".

<http://localhost:8085/ElectroGrid/EmployeeService/Employees>

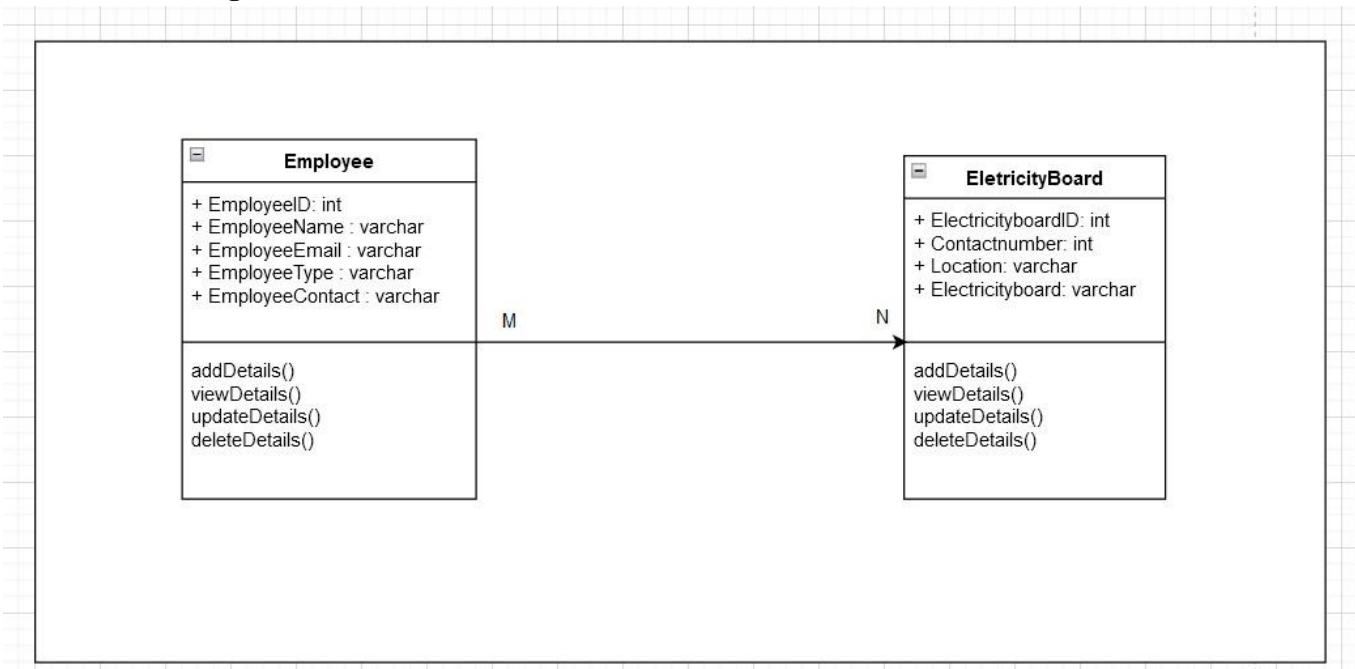
Employee Management

Here is the full scope of the employee management. The requested parties can perform their function successfully through the system. In employee management user can insert, update delete and view employee details.

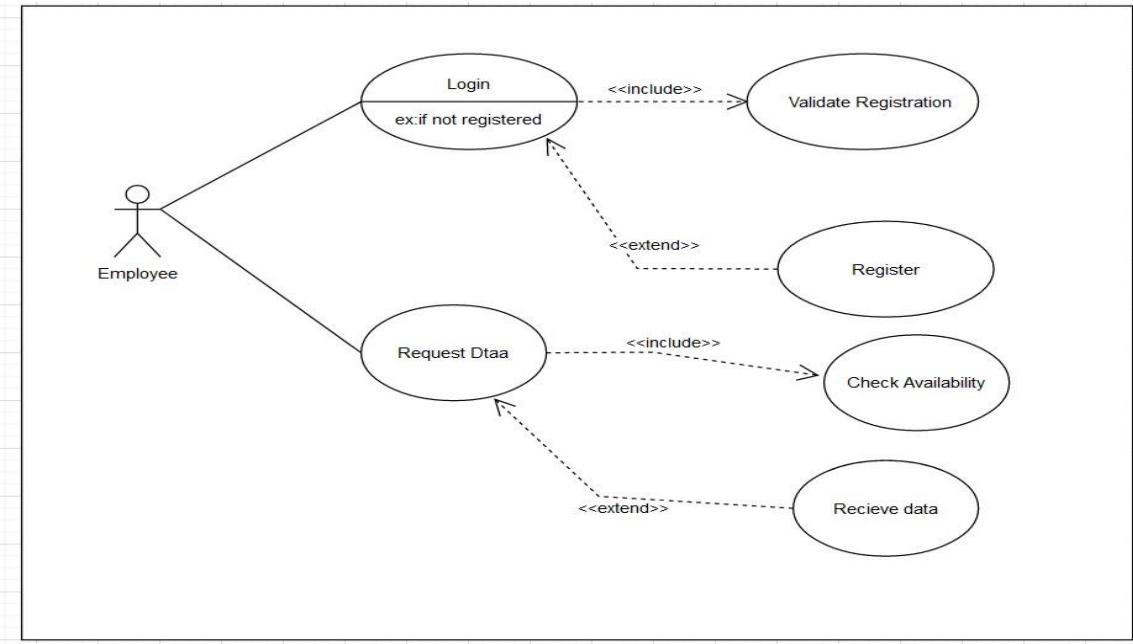
- **Er Diagram**



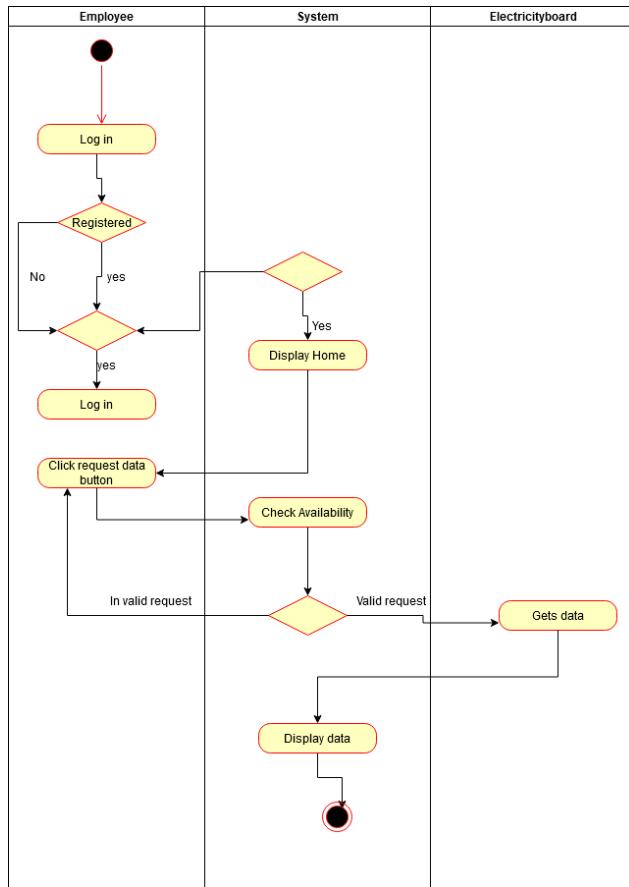
- **Class Diagram**



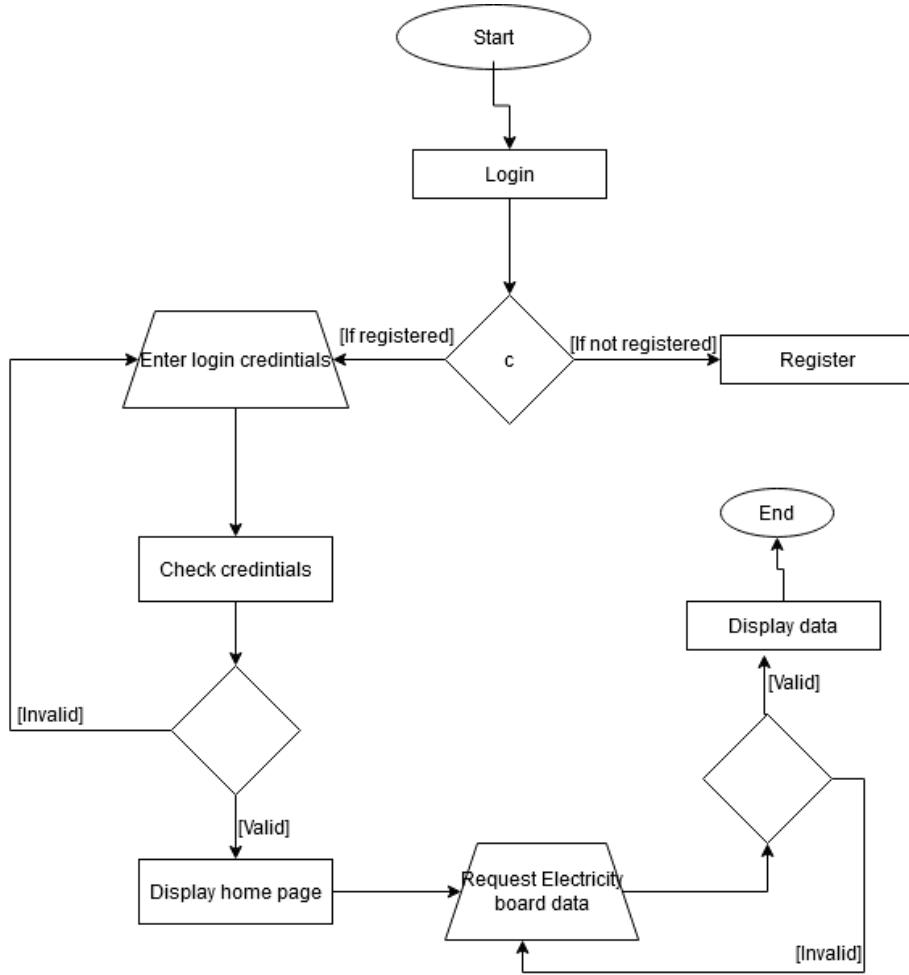
- Use case Diagram



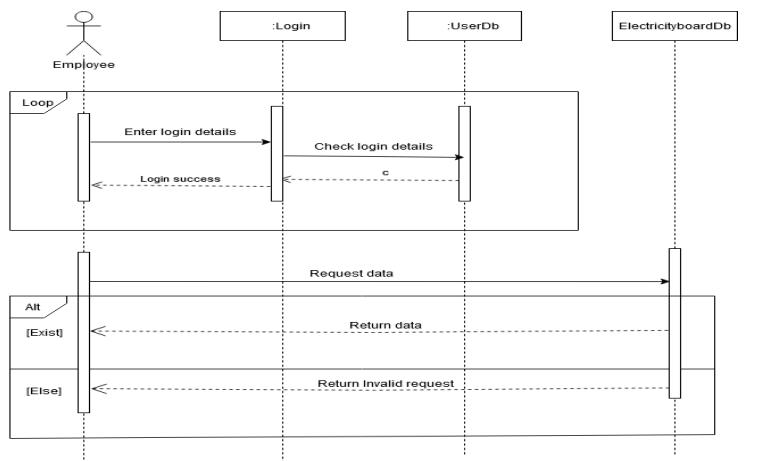
- Activity Diagram



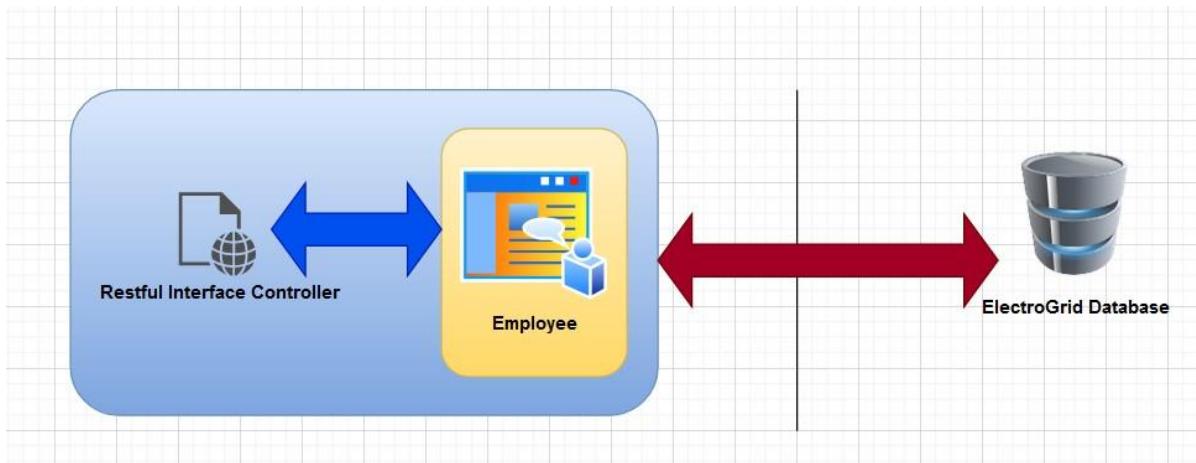
- **Flow Chart**



- **Sequence Diagram**



- Internal workflow



API Design Relational

Employee management handles all the information related to employees through insert, delete update, read methods.

- Employee can insert, delete, update, view the employee details.

Service Development

- Technologies used == Java-JAX-RS(Jersey) on Tomcat
- IDE == Eclipse
- Database == MySQL
- Testing == POSTMAN

API for get all the Employee Details in the database (GET Request)

URL:- <http://localhost:8085/ElectroGrid/EmployeeService/Employees>

Request :- { }

Response :- {EmployeeID : "Auto generated integer value"}
 {EmployeeName =:"Kamal"}
 {EmployeeEmail :kamal@gmail.com }
 {EmployeeType : Old Employee"}

```
{EmployeeContact : "071222256"}
```

API for insert a new Employee to the database (POST Request)

URL:- <http://localhost:8085/ElectroGrid/EmployeeService/Employees>

Request :- {EmployeeID : "Auto generated integer value"}
{EmployeeName =:"Nimal"}
{EmployeeEmail :nimal@gmail.com }
{EmployeeType : New Employee"}
{EmployeeContact : "071222256"}

Response:- { Result = "Inserted successfully" }
{EmployeeID = "Auto generated integer value"
{Error = "Error while inserting"}

API for update an Employee which is existing in database (PUT Request)

URL:- <http://localhost:8085/ElectroGrid/EmployeeService/Employees>

Request :- {EmployeeID : "Auto generated integer value"}
{EmployeeName =:"Nimal"}
{EmployeeEmail :nimal@gmail.com }
{EmployeeType : New Employee"}
{EmployeeContact : "071222256"}

Response:- { Result = "Updated successfully" }
{ EmployeeID = "Auto generated integer value"
{Error= "Error while Updating"}

API for delete an Employee which is existing in database (DELETE Request)

URL:- <http://localhost:8085/ElectroGrid/EmployeeService/Employees>

Request :- { EmployeeID = " Selected Employee ID from the service "}

Response :- { Result = "Deleted successfully" }
{Error = "Error while deleting the Employee" }

Test Cases TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
----------------------	---------------------------------	---------------	--------------------	------------------	-----------------------

1	Add Employee Details	{EmployeeName :"Abishaalini"} {EmployeeEmail : abi@gmail.com } {EmployeeType : old employee"} {EmployeeContact : "075123456"}	New Employee details are added to the database. Show message as "Inserted successfully".	New Employee details are added to the database. Show message as "Inserted successfully".	PASS
2	Update Employee Details	{EmployeeName :"Abishaalini"} {EmployeeEmail : abi@gmail.com } {EmployeeType : old employee"} {EmployeeContact : "075123456"}	Employee details are updated according to relevant input Employee details. Show message as "Updated successfully".	Employee details are updated according to relevant input Employee details. Show message as "Updated successfully".	PASS
3	Delete Employee Details	<EmployeeData> <EmployeeID>1</EmployeeID> </EmployeeData>	Show message as "Deleted Successfully"	Employee Details are deleted successfully. Show message as "Deleted Successfully"	PASS

Testing (Employee)

The screenshot shows the Postman application interface. On the left, the sidebar lists collections like 'check', 'EMPLOYEE' (which contains 'BILL', 'ELECTRICITY', 'CUSTOMER', 'FEEDBACK', 'PAYMENT', 'POWERSTATION'), and 'CONSUMPTION'. The main workspace shows a 'GET' request to 'http://localhost:8085/ElectroGrid/EmployeeService/Employee'. The 'Preview' tab of the response section displays a table with the following data:

Employee Name	Employee Email	Employee Type	Employee Contact
shashintha	shashintha@gmail.com	manager	78654321
abishalini	abishalini@gmail.com	manager	786542121

POST http://localhost:8085/ElectroGrid/EmployeeService/Employee

KEY	VALUE	DESCRIPTION
EmployeeID	1	
EmployeeName	abishalini	
EmployeeEmail	abishalini@gmail.com	
EmployeeType	manager	
EmployeeContact	0786542121	

Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize

Status: 200 OK Time: 78 ms Size: 179 B Save Response

Inserted successfully

PUT http://localhost:8085/ElectroGrid/EmployeeService/Employee

Body	Cookies	Headers (5)	Test Results
<pre>1 2 "EmployeeID": "1", 3 "EmployeeName": "shashintha", 4 "EmployeeEmail": "shashintha@gmail.com", 5 "EmployeeType": "Manager", 6 "EmployeeContact": "087234567"</pre>			

Pretty Raw Preview Visualize

Status: 200 OK Time: 29 ms Size: 178 B Save Response

Updated successfully

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with a collection named 'check'. This collection contains several sub-collections: 'Paf', 'BILL', 'ELECTRICITY', 'CUSTOMER', 'FEEDBACK', 'PAYMENT', 'POWERSTATION', 'EMPLOYEE', 'CONSUMPTION', and 'CONSULTATION'. The 'EMPLOYEE' collection is currently selected. In the main workspace, a DELETE request is being made to the URL `http://localhost:8085/ElectroGrid/EmployeeService/Employee?content`. The 'Body' tab is active, showing the following XML content:

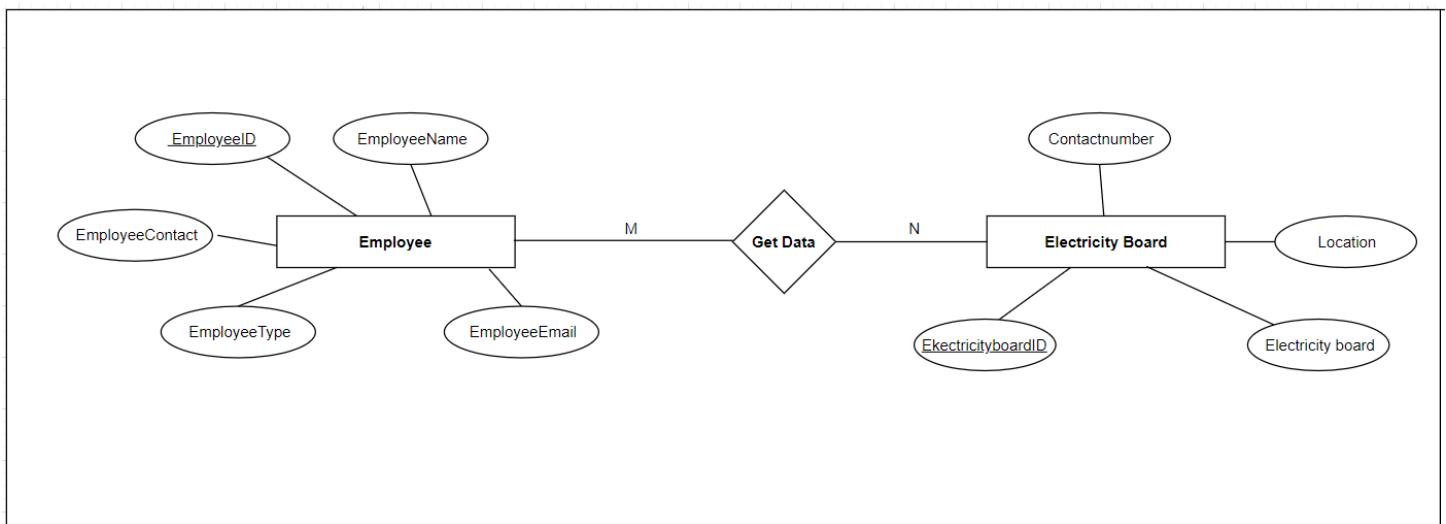
```
1 <EmployeeData>
2 <EmployeeID>1</EmployeeID>
3 </EmployeeData>
```

Below the request, the response status is shown as `Status: 200 OK`, `Time: 30 ms`, and `Size: 178 B`. The response body contains the message `Deleted successfully`.

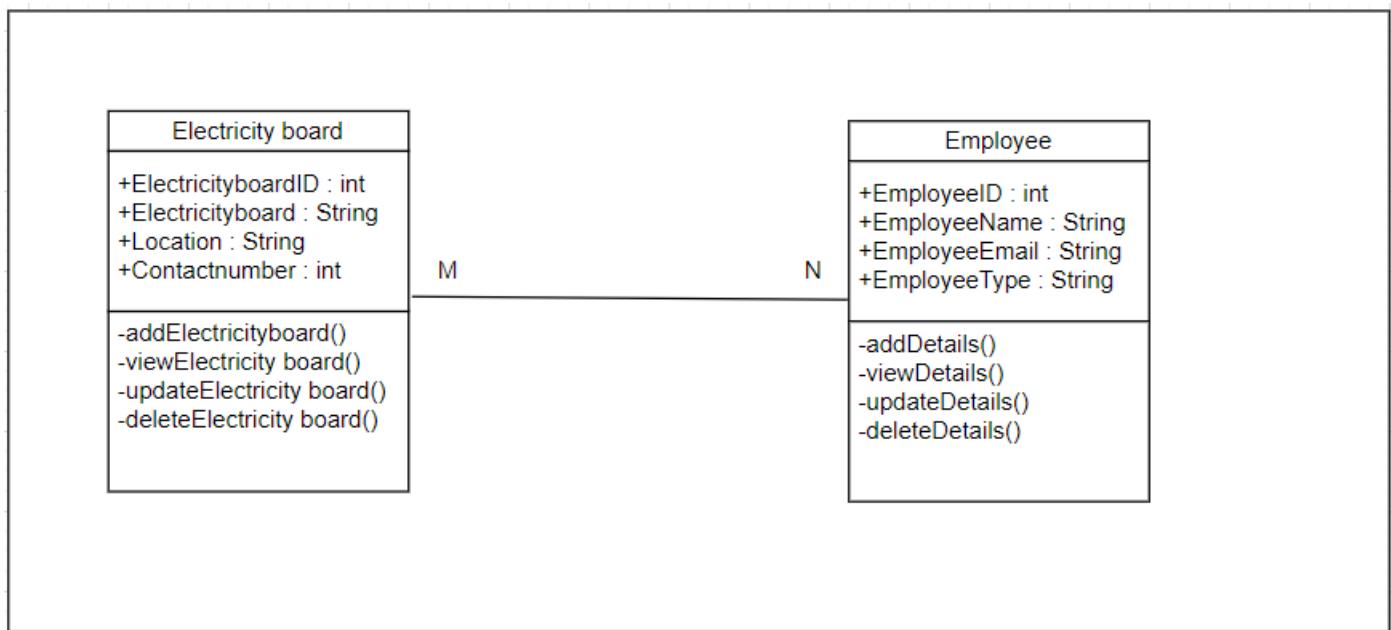
Electricity board Management

Here is the full scope of the electricity board management. The requested parties can perform their function successfully through the system. In electricity board management, user can insert, update delete and view Electricity board branch details.

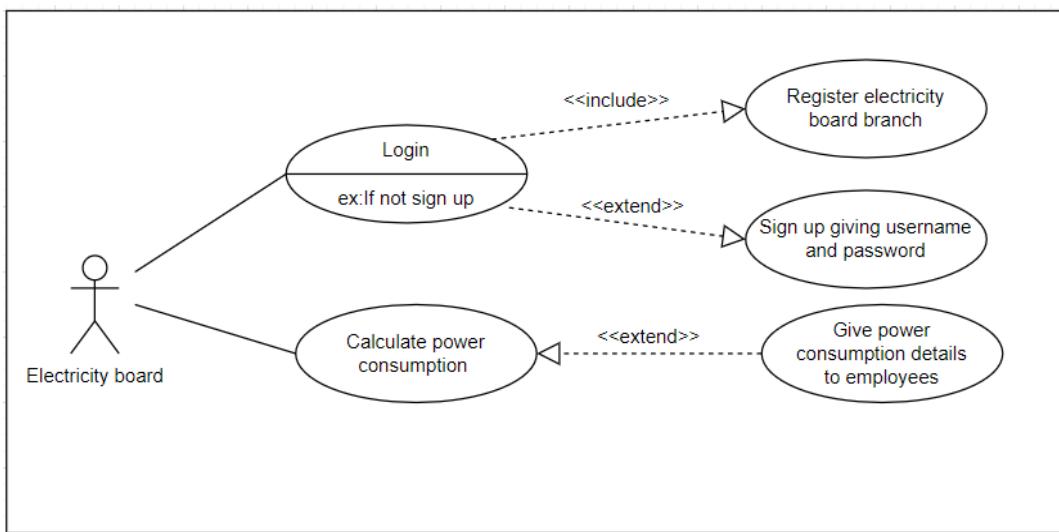
- **Er Diagram**



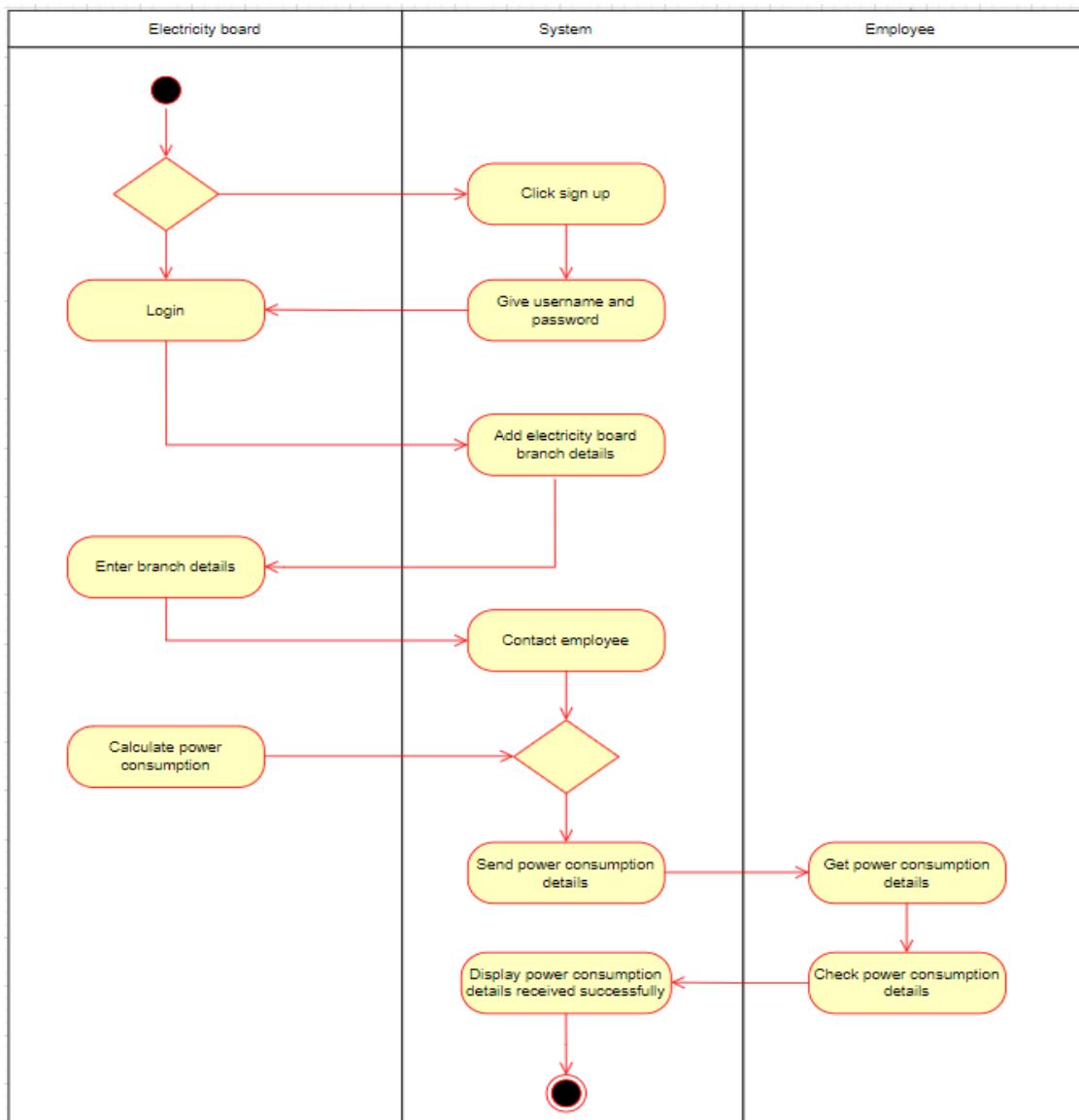
- **Class Diagram**



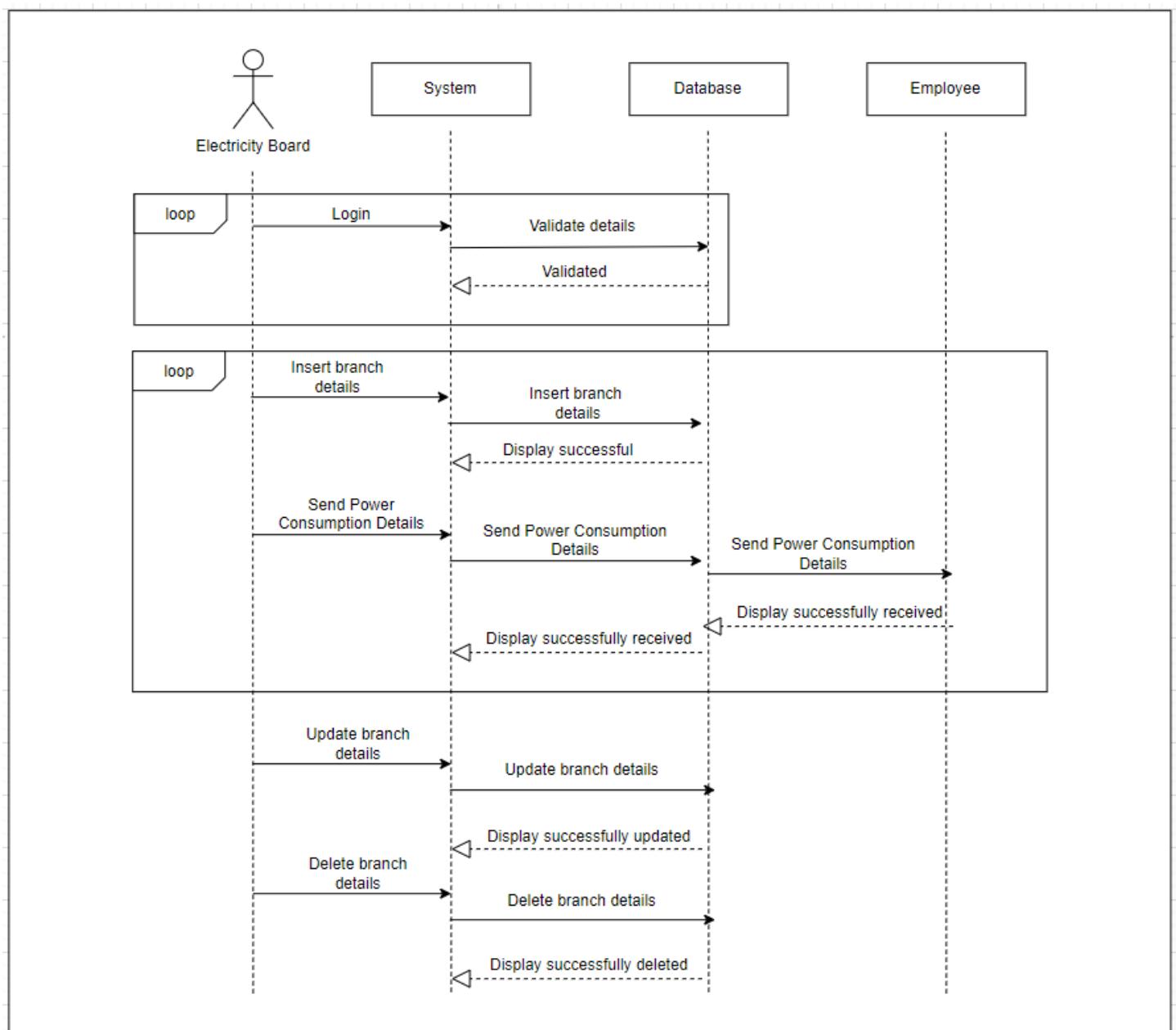
- **Use case Diagram**



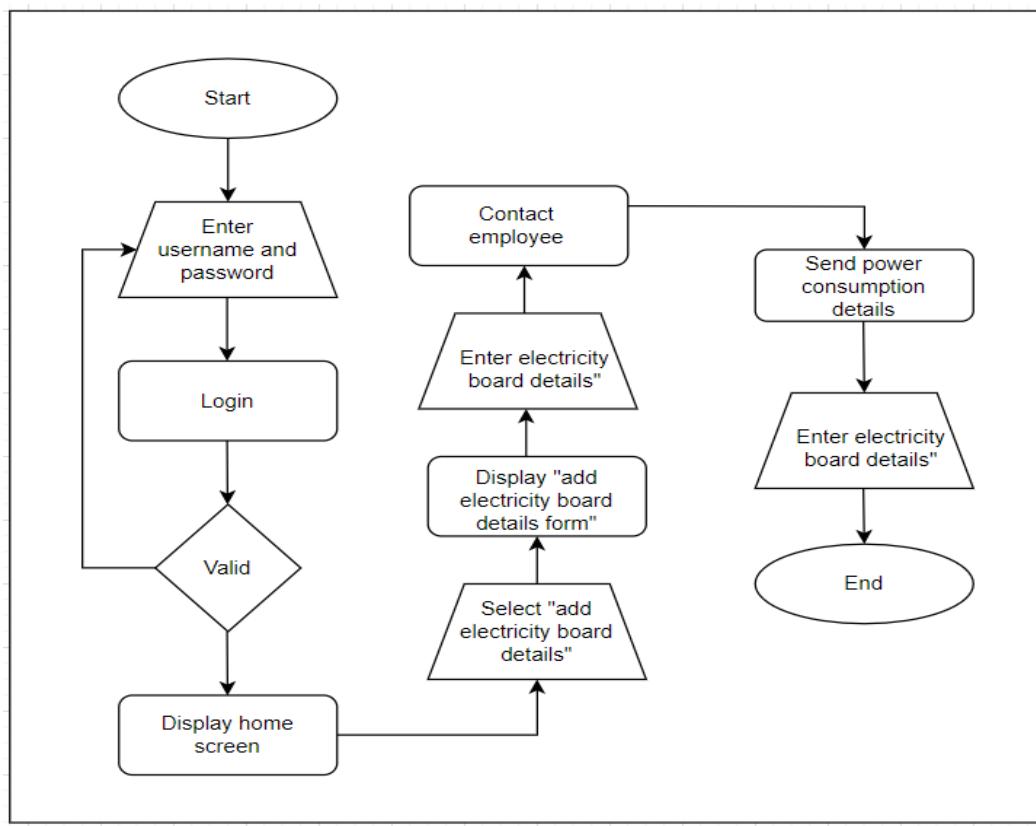
- **Activity Diagram**



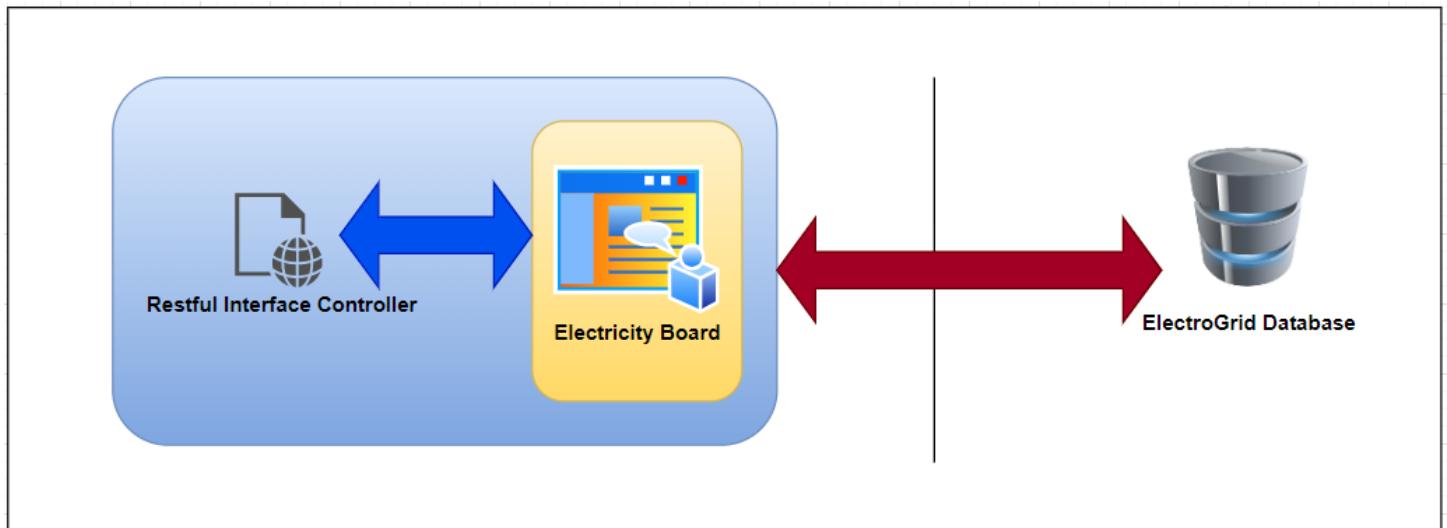
- Sequence Diagram



- **Flow chart**



- **Internal workflow**



API Design Relational

Electricity board management handles all the information related to electricity boards

through insert, delete, update, read methods.

- **Electricity board can insert, delete, update, view the electricity board branch details.**

Service Development

- Technologies used : Java-JAX-RS(Jersey) on Tomcat
- IDE : Eclipse
- Database : MySQL
- Testing : POSTMAN

1) API for get all the Electricity board branch details in the database (GET Request)

URL :- <http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards>

Request :- {}

Response :- {ElectricityboardID : "Auto generated integer value"}

- {Electricityboard =:"LECO"}
- {Location : "Colombo" }
- {Contactnumber : "0814563297"}

2) API for insert a new Electricity board branch details in the database (POST Request)

URL :- <http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards>

Request :- {ElectricityboardID : "Auto generated integer value"}

- {Electricityboard :"CEB"}
- {Location : "Kandy" }
- {Contactnumber : "0814563297"}

Response :- { "Inserted successfully"}

3) API for update existing Electricity board branch details in the database (PUT Request)

URL :- <http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards>

Request :- {ElectricityboardID : "Auto generated integer value"}

- {Electricityboard :"CEB"}
- {Location : "Kurunegala" }
- {Contactnumber : "0379629186"}

Response :- { Result = "Updated successfully" }

4) API for delete existing Electricity board branch details in the database (DELETE Request)

URL :- <http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards>

Request :- {ElectricityboardID : "Selected ElectricityboardID from the service"}

Response :- { Result = "Deleted successfully" }

Test Cases

Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Electricity board branch details	{ElectricityboardID : ""} {Electricityboard :"CEB"} {Location : "Kandy" } {Contactnumber : "0814563297"}	New Electricity board branch details are added to the database. Show message as "Inserted successfully".	New Electricity board branch details are added to the database. Show message as "Inserted successfully".	PASS
2	Update Electricity board branch details	{ElectricityboardID : ""} {Electricityboard :"CEB"} {Location : "Kurunegala" } {Contactnumber : "0379629186"}	Electricity board branch details are updated according to relevant input details. Show message as "Updated successfully".	Electricity board branch details are updated according to relevant input details. Show message as "Updated successfully".	PASS
3	Delete Electricity board branch details	<elecData> <ElectricityboardID>12</ElectricityboardID> </elecData>	Show message as "Deleted Successfully"	Electricity board branch details are deleted successfully. Show message as "Deleted Successfully"	PASS

Testing

The screenshot shows the Postman application interface. At the top, there is a search bar labeled "Search Postman" and several status indicators for other requests. Below the header, the URL "http://localhost:8090/ElectricGrid/ElectricityboardService/Electricityboards" is entered into the main request field. The method dropdown shows "GET". To the right of the URL, there are "Save", "Edit", and "Copy" buttons, along with a "Send" button. Below the URL, tabs for "Params", "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", and "Settings" are visible. The "Headers" tab is selected. Under "Query Params", there is a table with one row: "Key" (Value) and "Value" (Description). In the bottom section, tabs for "Body", "Cookies", "Headers (5)", and "Test Results" are shown. The "Test Results" tab is selected, displaying a status of "200 OK" and a response body table:

Electricity Board ID	Electricity Board	Location	Contact Number
11	LECO	Colombo	114598745

The screenshot shows the Postman application interface. At the top, there is a search bar labeled "Search Postman" and several status indicators for other requests. Below the header, the URL "http://localhost:8090/ElectricGrid/ElectricityboardService/Electricityboards" is entered into the main request field. The method dropdown shows "POST". To the right of the URL, there are "Save", "Edit", and "Copy" buttons, along with a "Send" button. Below the URL, tabs for "Params", "Authorization", "Headers (8)", "Body", "Pre-request Script", "Tests", and "Settings" are visible. The "Body" tab is selected. Under "Body", there are radio buttons for "none", "form-data", "x-www-form-urlencoded" (which is selected), "raw", "binary", and "GraphQL". A table below lists form fields: "ElectricityboardID" (Value: 12), "Electricityboard" (Value: CEB), "Location" (Value: Kandy), and "Contactnumber" (Value: 0814563297). In the bottom section, tabs for "Body", "Cookies", "Headers (5)", and "Test Results" are shown. The "Test Results" tab is selected, displaying a status of "200 OK" and the message "Inserted successfully".

Search Postman

Sign In Create Account

Overview POST http://loc... ● POST http://loc... ● PUT http://local... ● DEL http://local... ● GET http://local... ● + ⚙ No Environment

<http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards>

PUT http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards

Params Authorization Headers (9) Body **JSON** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Beautify

```

1 {
2   "ElectricityboardID": "12",
3   "Electricityboard": "CEB",
4   "Location": "Kurunegala",
5   "Contactnumber": "0379629186"
6 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 129 ms Size: 178 B Save Response

Pretty Raw Preview Visualize

Updated successfully

Search Postman

Sign In Create Account

Overview POST http://loc... ● POST http://loc... ● PUT http://local... ● DEL http://local... ● GET http://local... ● + ⚙ No Environment

<http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards>

DELETE http://localhost:8090/ElectroGrid/ElectricityboardService/Electricityboards

Params Authorization Headers (9) Body **XML** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL XML Beautify

```

1 <elecData>
2 | <ElectricityboardID>12</ElectricityboardID>
3 </elecData>
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 106 ms Size: 178 B Save Response

Pretty Raw Preview Visualize

Deleted successfully

Tool Used

- Java-JAX-RS (Jersey): Used for backend development
- Eclipse: Use as IDE for developing this project
- MVC Architecture: used to easily code our project
- Apache Tomcat V.09: Used as the server
- My SQL Used as our database