IT3030
Programming Applications and Frameworks
3rd Year, 1st Semester

Assignment

Group Project

Submitted to

Sri Lanka Institute of Information Technology

Group ID - 18

Batch: Y3.S1.WE.IT.02.02

Table of Contents

1. Introduction

ElectroGrid (EG) is the company who maintains the power grid of the country. This system has the ability 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. In this system there are 5 functions such as Customer Management, Employee Management, Payment Management, Complaint Management and Bill Management. Tools that are used to build this platform were java JAX-RS Jersey, Apache Tomcat 10 and MySQL. Customer can log into the system or register to the system, view and edit their details and they can also delete their profiles. And they can simply add, delete, view and edit their payment details and also, they can make the payments belong to them. And they can also view their bills. Customer should make complaints through employees and employees can create, update, view and delete the complaints. The primary objective of this system is to develop a highly scalable platform for ElectroGrid to cater all the users.

2. Member's Details

IT Number	Name	Web Service	Description of the Web Service
IT20018900	Rathnayaka R.M.A.N.	Bill Management & Complain Management	Insert, Update, Delete and View Bill Details Insert, Update, Delete and View Complain Details
IT20066116	Basnayaka K.V.N.	Employee Management	Insert, Update, Delete and View Employee Details
IT20042660	Salwathura S.R.	Customer Management	Insert, Update, Delete and View Customer Details
IT20026820	Nanayakkara S.	Payment Management	Insert, Update, Delete and View Payment Details

Clickable Link

nimnakaB/Electrogrid-Power-Management-Project (github.com)

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

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

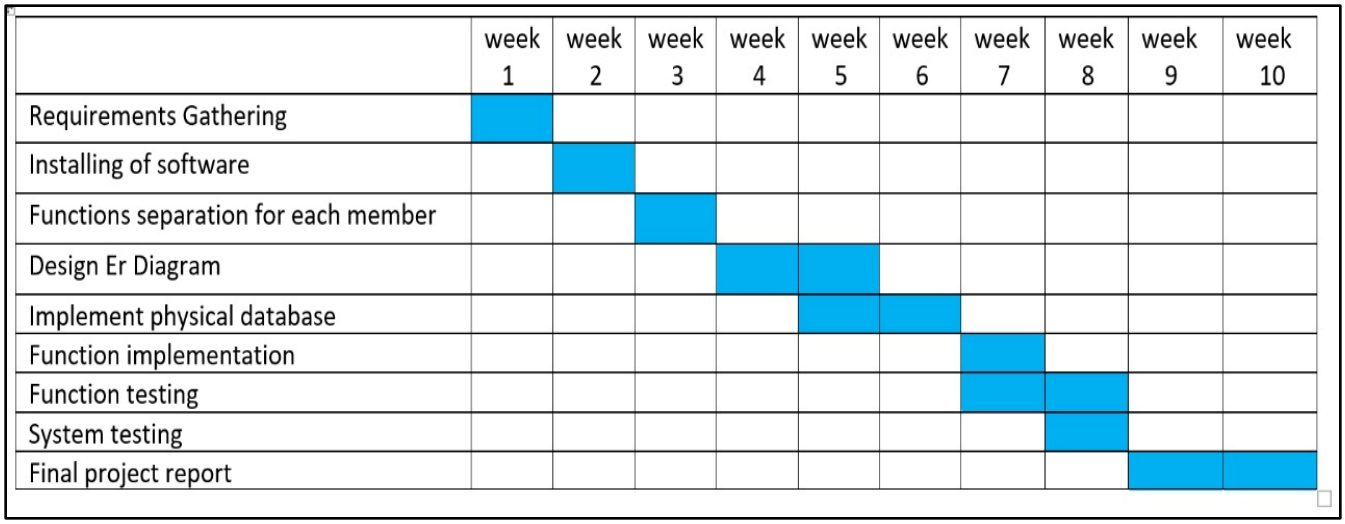
• Advantages

- ❖ Rapid and continuous development.
- ❖ Late changes in requirements are welcomed.
- ❖ Early identification of bugs.
- ❖ A working software can be delivered frequently.

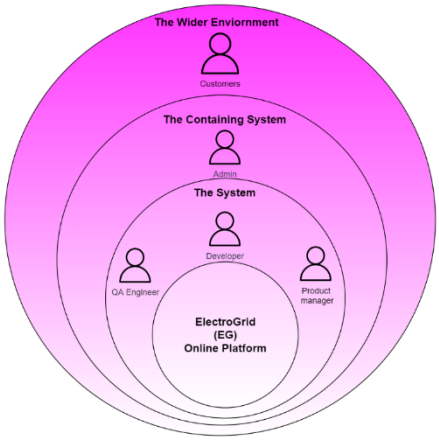
• Disadvantages

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

3. Time schedule (Gantt chart)



System's overall design



Technical Requirements

Technical requirements are the technical issues that must be considered to make the system successful.

Customer, Payments, Employee, Complaints and Bill's detail can be updated, deleted, and view when needed. Also, sensitive details should be encrypted

3)**Functional Requirements**

- Customer Management: Add, Update, Delete and View Customer Details
- Payment Management: Add, Update, Delete and View Payment Details
- Employee Management: Add, Update, Delete and View Employee Details
- Complaint Management: Add, Update, Delete and View Complaint details
- Bill Management: Add, Update, Delete and View Bill Details

4)**Non-Functional Requirements**

• Performance – Response time, User interface, Static Volumetric, Utilization, Conformity

• Security requirements – All data inside the system or its part will be protected against malware attacks or unauthorized access.

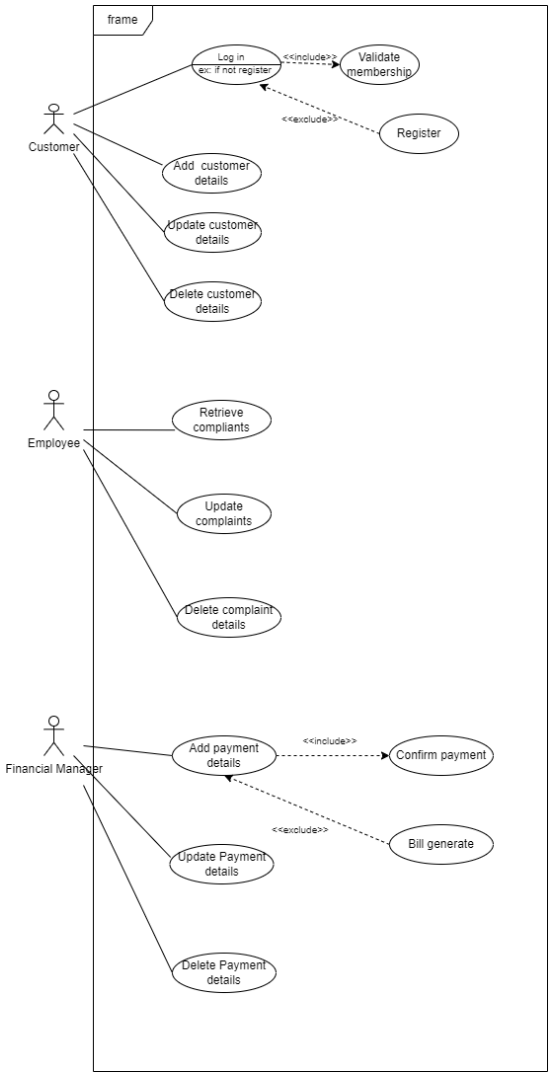
• Software Quality Attributes

  ▪ Availability
  ▪ Maintainability
  ▪ Usability
  ▪ Accuracy
  ▪ Stability
  ▪ Correctness
  ▪ Accessibility

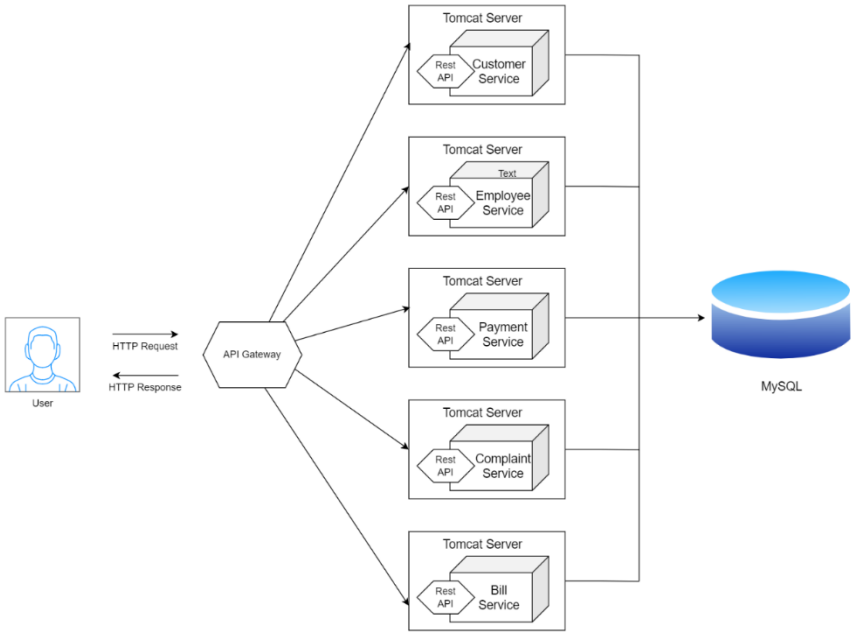
• Reliability

Systems Overall Design

Use case Diagram



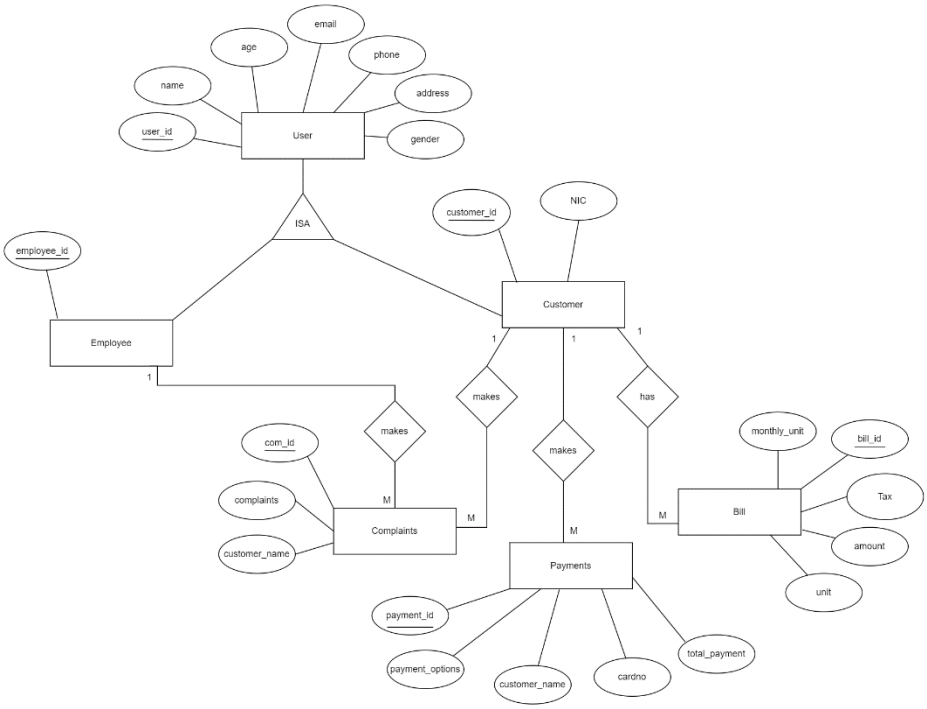
Overall System Architecture



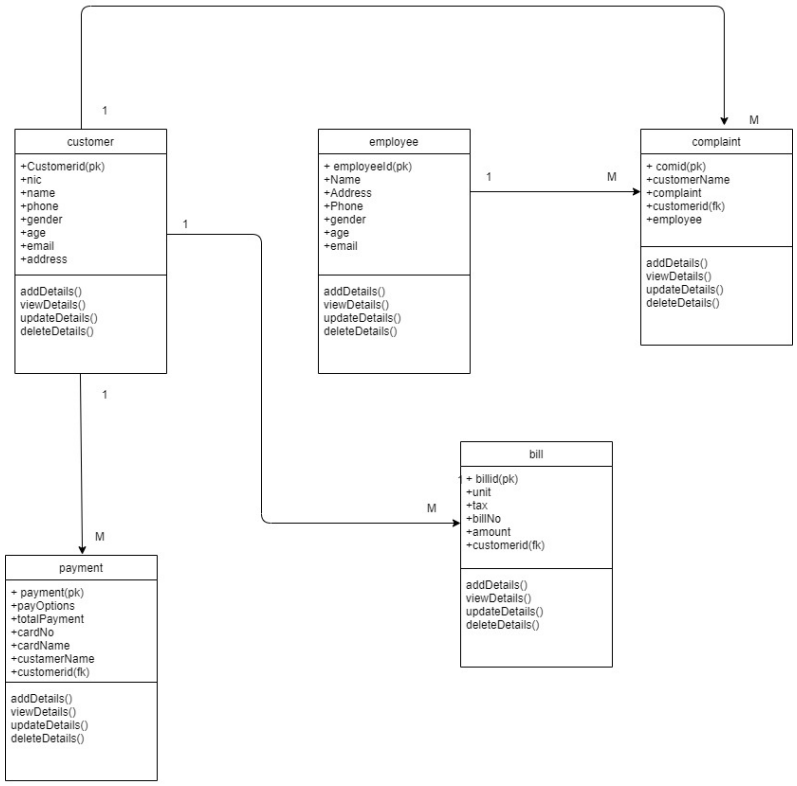
ElectroGrid (EG) is the company who maintains the power grid of the country. This system has the ability to monitor the power consumption of the users, generate the monthly bills and automatically send to the users, and accept the online payments from the users.

This management system consists of four services as Customer Service, Employee Service, Payment Service, Complaint Service. One database was used for every 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 response 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 more efficiently.

ER Diagram



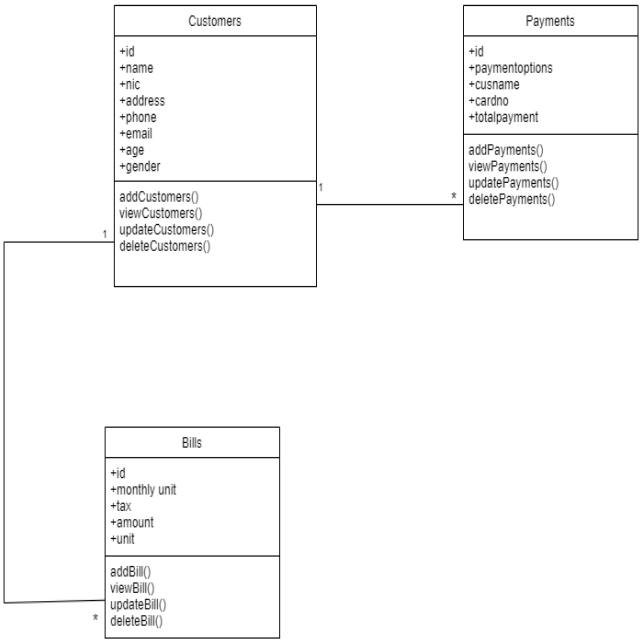
Class Diagram



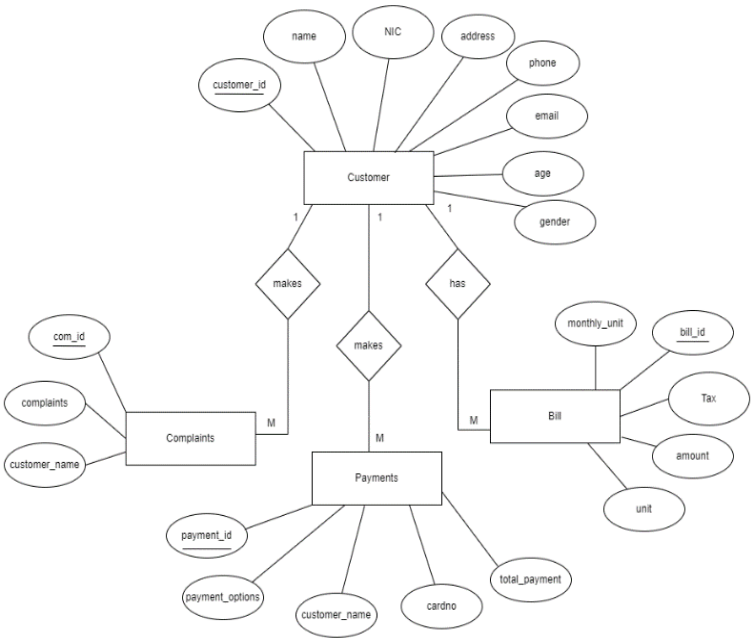
IT20042660 (Salwathura S.R.) – URL: - http://localhost:8080/ElectroGrid/rest/customers
:

Customer Management System

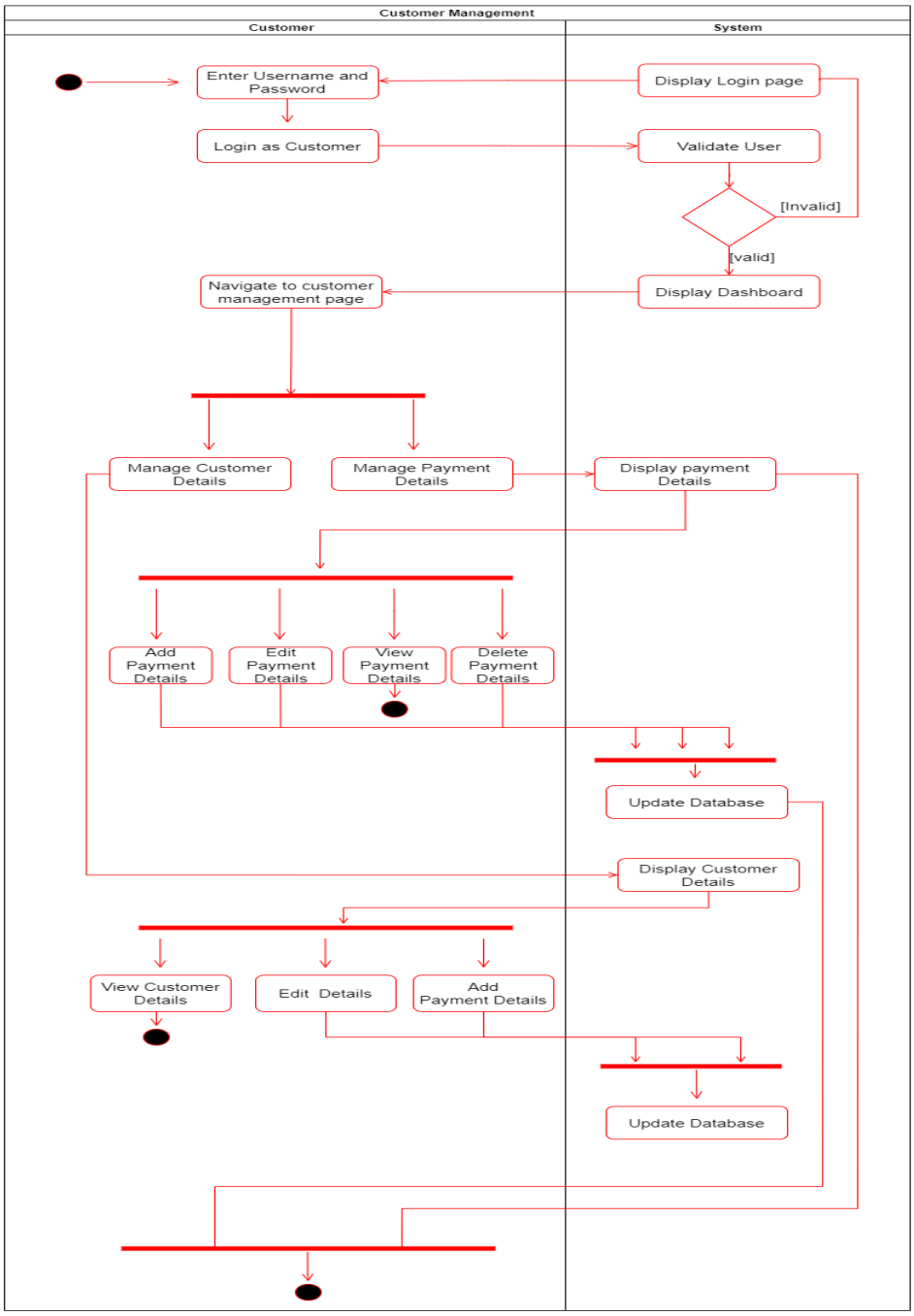
Here the full scope of the Customer management is completed. The respected parties can perform their function successfully through this platform. In Customer management the customers can add, update, delete and view customer details.



Class Diagram



ER Diagram



Activity Diagram

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

URL: - http://localhost:8080/ElectroGrid/rest/customers/2

API for get all the Customer details in the database (GET Request)
URL: - http://localhost:8080/ElectroGrid/rest/customers
Response

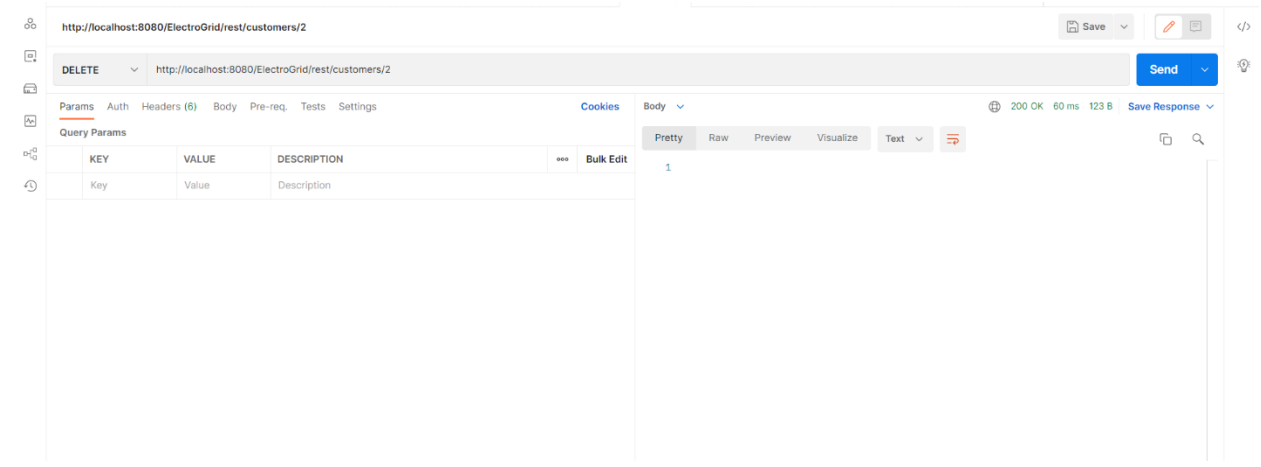

```
[
    {
        "customerId": 1,
        "nic": "992133786V",
        "name": "Kavindu",
        "address": "Colombo",
        "phone": "0711672290",
        "gender": "male",
        "age": 21,
        "email": "kavindu@gmail.com"
    }
]
```

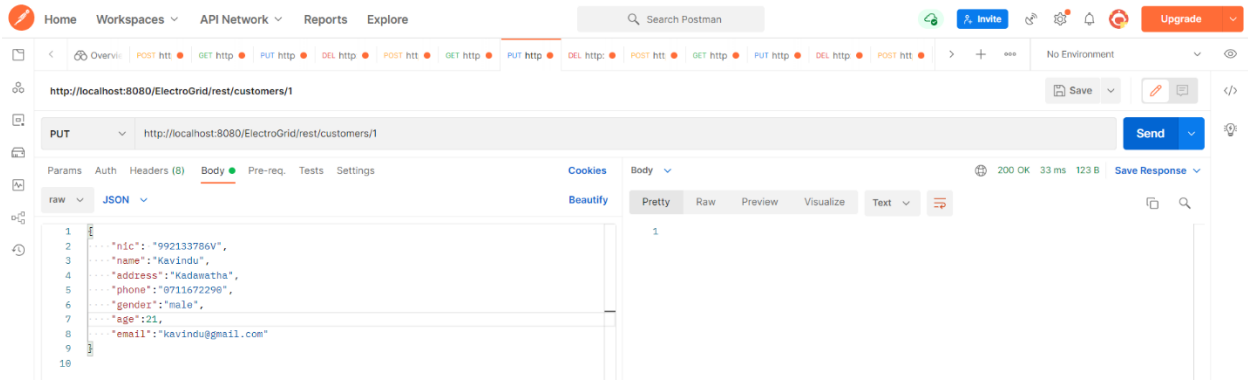

API for insert a new Customer to the database (POST Request)
URL: - http://localhost:8080/ElectroGrid/rest/customer
Media raw JSON

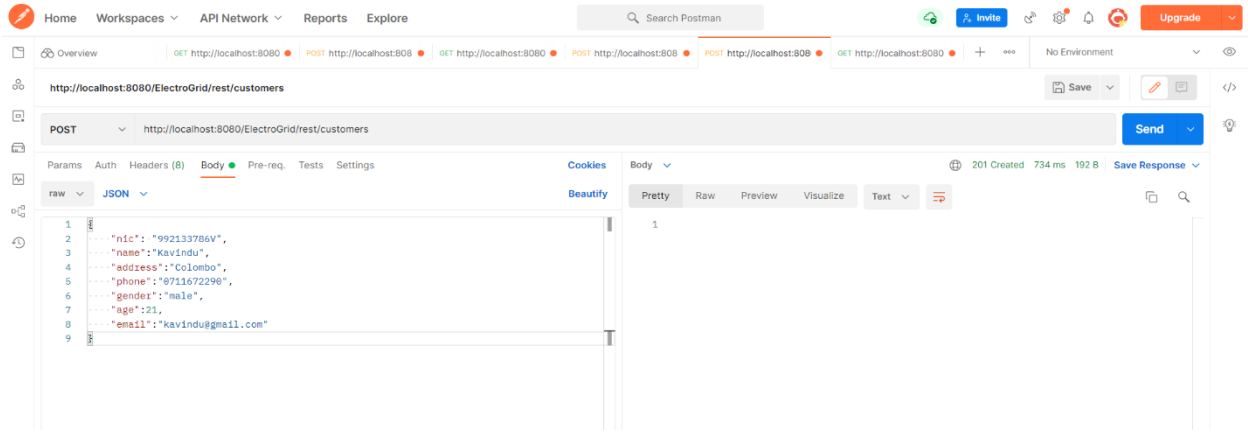

```
{
    "nic": "992133786V",
    "name":"Kavindu",
    "address":"Colombo",
    "phone":"0711672290",
    "gender":"male",
    "age":21,
    "email":"kavindu@gmail.com"
}
```

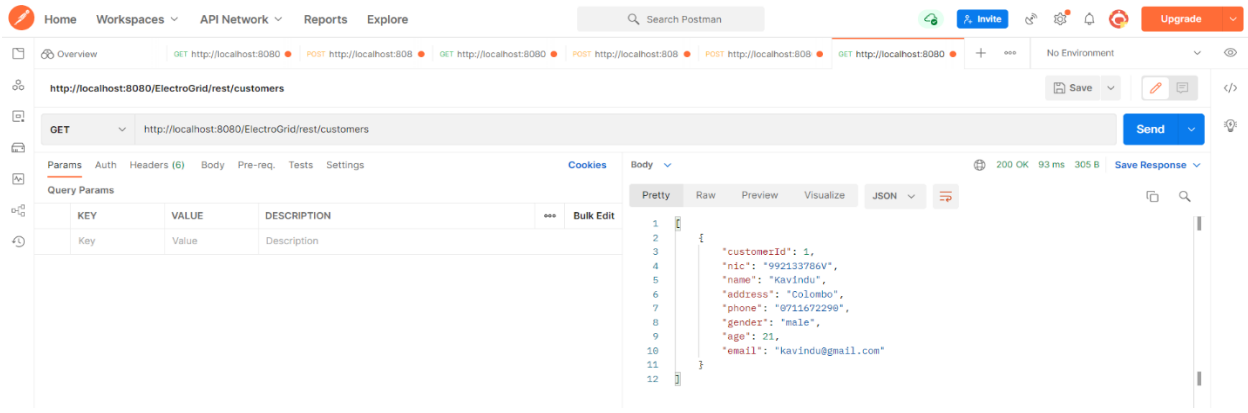

API for update a Customer which is existing in database (PUT Request)
URL: - http://localhost:8080/ElectroGrid/rest/customers/1
Media raw JSON


```
{
    "nic": "992133786V",
    "name":"Kavindu",
    "address":"Kadawatha",
    "phone":"0711672290",
    "gender":"male",
    "age":21,
    "email":"kavindu@gmail.com"
}
```


http://localhost:8080/ElectroGrid/rest/customers/2
Save
DELETE
http://localhost:8080/ElectroGrid/rest/customers/2
Send
Params Auth Headers (6) Body Pre-req. Tests Settings
Cookies
Body
200 OK 60 ms 123 B
Save Response
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Pretty Raw Preview Visualize Text
1


Home Workspaces API Network Reports Explore
Search Postman
Invite
Upgrade
No Environment
http://localhost:8080/ElectroGrid/rest/customers/1
Save
PUT
http://localhost:8080/ElectroGrid/rest/customers/1
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
200 OK 33 ms 123 B
Save Response
raw JSON
Beautify
Pretty Raw Preview Visualize Text
{
"nic": "992133786V",
"name":"Kavindu",
"address":"Kadawatha",
"phone":"0711672290",
"gender":"male",
"age":21,
"email":"kavindu@gmail.com"
}


Home Workspaces API Network Reports Explore
Search Postman
Invite
Upgrade
Overview
No Environment
http://localhost:8080/ElectroGrid/rest/customers
Save
POST
http://localhost:8080/ElectroGrid/rest/customers
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body
201 Created 734 ms 192 B
Save Response
raw JSON
Beautify
Pretty Raw Preview Visualize Text
{
"nic": "992133786V",
"name":"Kavindu",
"address":"Colombo",
"phone":"0711672290",
"gender":"male",
"age":21,
"email":"kavindu@gmail.com"
}


Home Workspaces API Network Reports Explore
Search Postman
Invite
Upgrade
Overview
No Environment
http://localhost:8080/ElectroGrid/rest/customers
Save
GET
http://localhost:8080/ElectroGrid/rest/customers
Send
Params Auth Headers (6) Body Pre-req. Tests Settings
Cookies
Body
200 OK 93 ms 305 B
Save Response
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Pretty Raw Preview Visualize JSON
[
{
"customerId": 1,
"nic": "992133786V",
"name": "Kavindu",
"address": "Colombo",
"phone": "0711672290",
"gender": "male",
"age": 21,
"email": "kavindu@gmail.com"
}
]

Test Cases

TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/ Fail)
1	Add Customer Details	{ "nic": "992133786V", "name":"Kavindu", "address":"Colombo", "phone":"0711672290", "gender":"male", "age":21, "email":"kavindu@gmail.com" }	New Customer details are added to the database.	New Customer details are added to the database.	**PASS**
2	Update Customer Details	{ "nic": "992133786V", "name":"Kavindu", "address":"Kadawatha", "phone":"0711672290", "gender":"male", "age":21, "email":"kavindu@gmail.com" }	Customer details are updated according to relevant input Customer details.	Customer details are updated according to relevant input Customer details.	**PASS**
3	Delete Customer Details		Delete customer details from the Database.	Customer Details are deleted successfully.	**PASS**

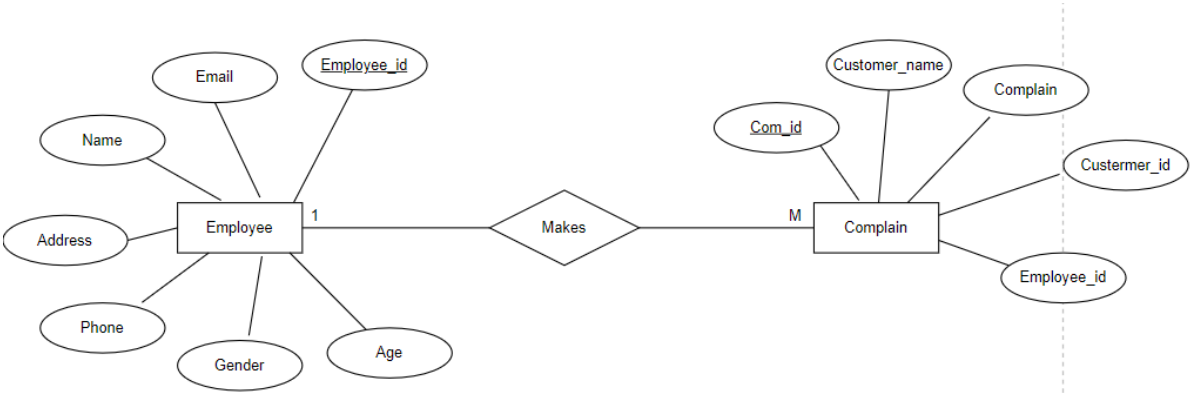
IT20066116 (Basnayaka K.V.N)

URL - http://localhost:8080/ElectroGrid/rest/employees-

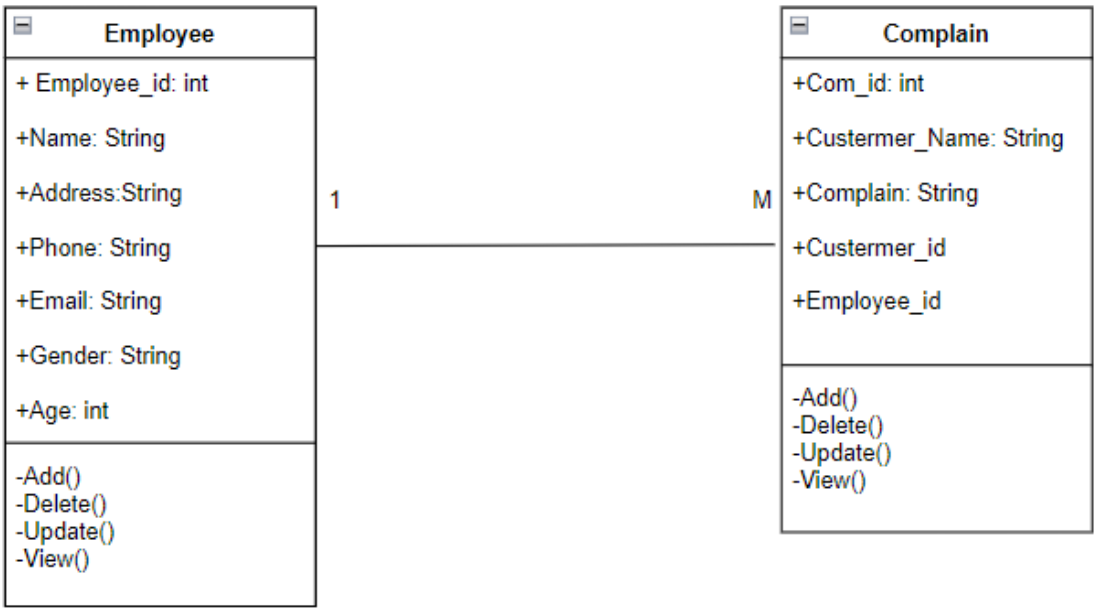
Employee Management

In here Employee management an employee can create their profile providing Required details, and then they can simply Manage the complains that are created by the customers. the employee can Add new employee to the system View their details, update their details, and also can delete any employee details

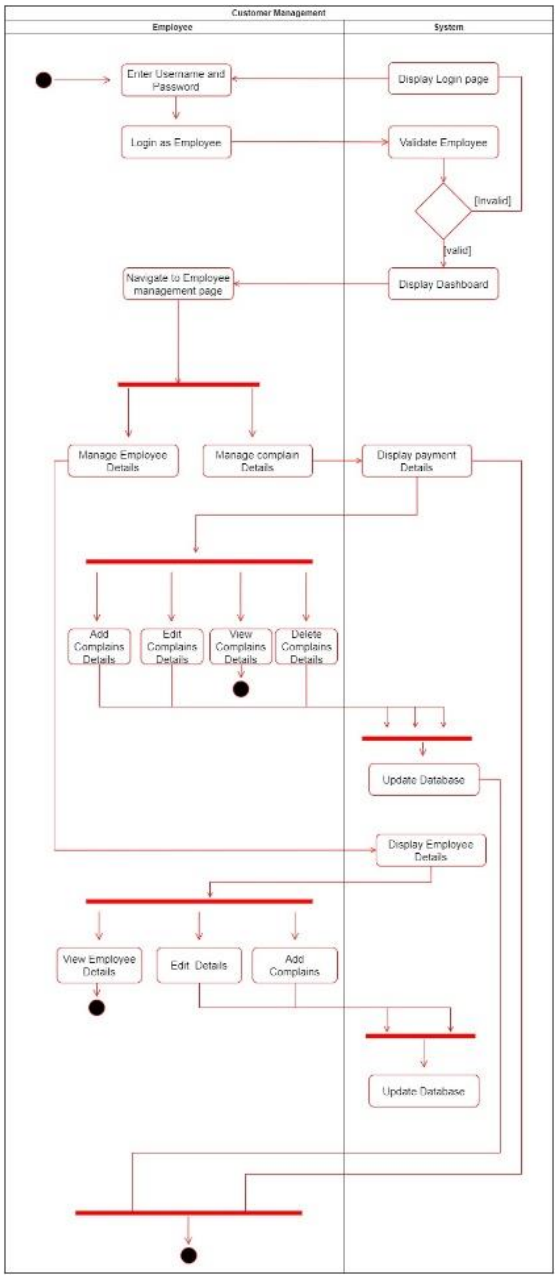
ER diagram



Class diagram



Activity Diagram



Delete – DELETE

URL - http://localhost:8080/ElectroGrid/rest/employees/2

Create – POST

URL - http://localhost:8080/ElectroGrid/rest/employees

Media raw JSON

```
{
    "nic": "992133786V",
    "name":"Nimnaka",
    "address":"Kandy",
    "phone":"0723481901",
    "gender":"male",
    "age":21,
    "email":"kavindu@gmail.com"
}
```

Read – GET

URL - http://localhost:8080/ElectroGrid/rest/employees

Response

```
[
    {
        "employeeId": 1,
        "name": "Nimnaka",
        "address": "Kandy",
        "phone": "0723481901",
        "gender": "male",
        "age": 21,
        "email": "nimnaka@gmail.com"
    }
]
```

Update – PUT

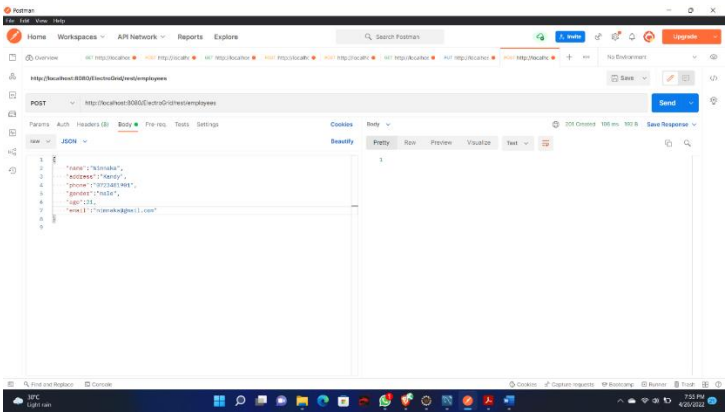
URL - http://localhost:8080/ElectroGrid/rest/employees/1

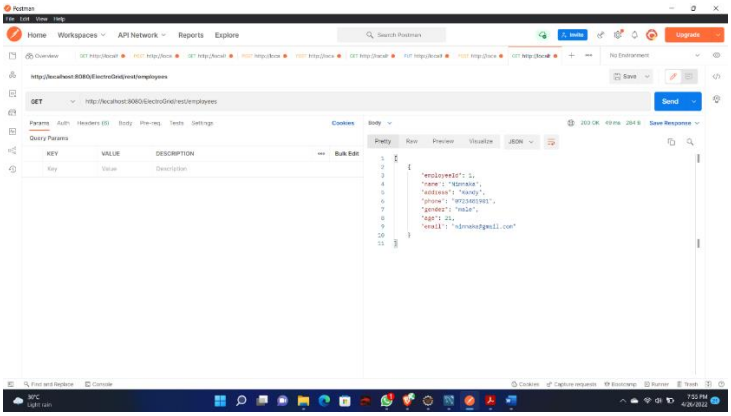
Media raw JSON

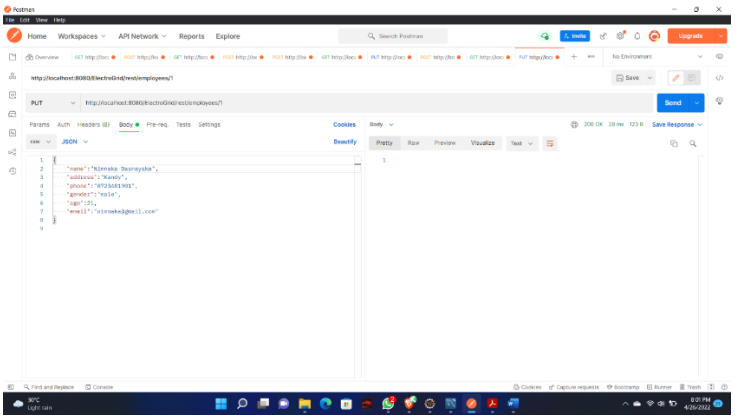
```
{
    "name":"Nimnaka Basnayaka",
    "address":"Kandy",
    "phone":"0723481901",
    "gender":"male",
    "age":21,
    "email":"nimnaka@gmail.com"
}
```

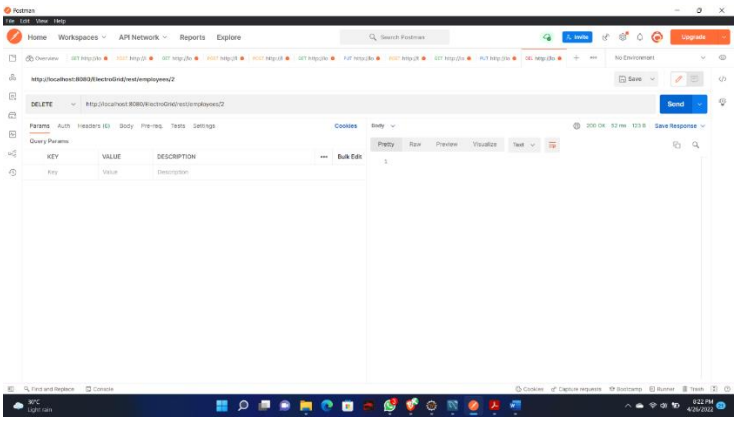
Test Cases

TestID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)
1	Add Employee Details	{Name:Nimnaka} {Address : Colombo"} {Phone:" 0723481901"} {Email:"nimnaka@gmail.com"} {Gender:"male"} {Age:" 21"}	New Employee details are added to the database.	New Customer details are added to the database.	**PASS**
2	Update Employee Details	{Name: Nimnaka Basnayaka} {Address : Kandy"} {Phone:"0715534356"} {Email:" nimnaka@gmail.com"} {Gender:"male"} {Age:"21"}	Employee details are updated according to relevant input Employee details.	Employee details are updated according to relevant input Employee details. Show	**PASS**
3	Delete Employee Details		Delete Employee details from the DB"	Delete Employee details from the DB"	**PASS**







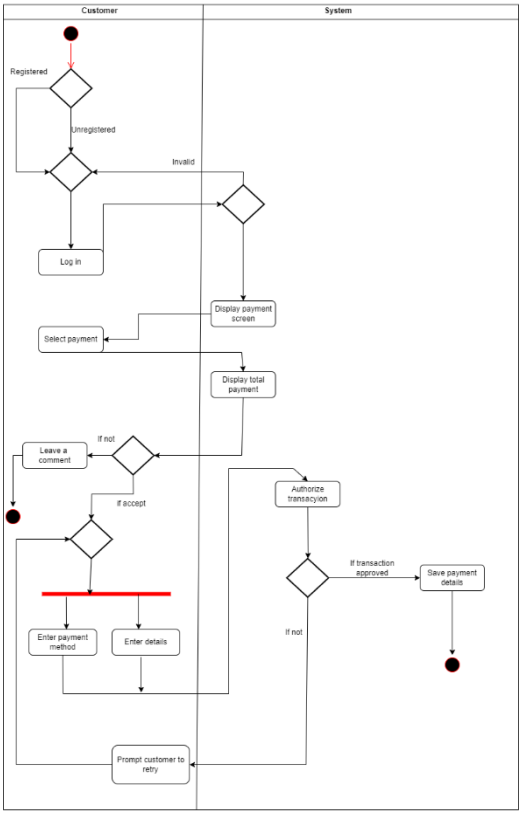


IT20026820(Nanayakkara.S) – URL – http://localhost:8080/ElectroGrid/rest/payments

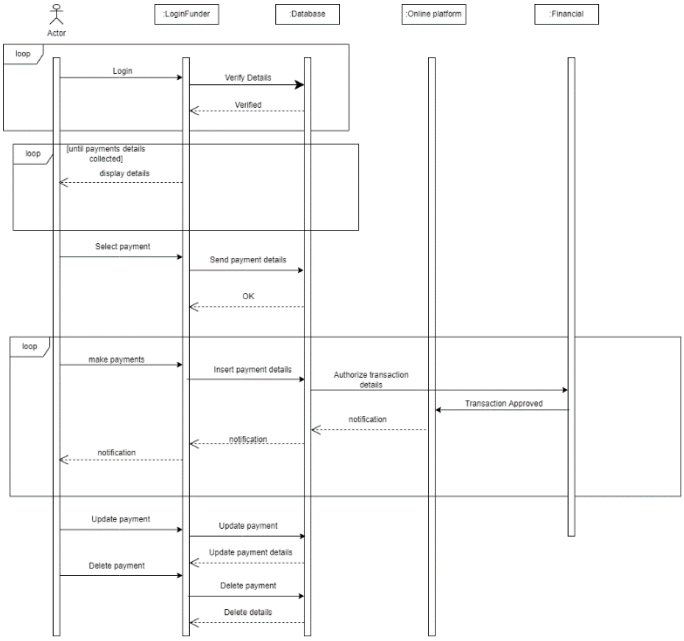
Payment Management

In payment management system customers can make their payments online. once the payment is made a bill will be auto generated. Customers can simply add and delete their details and also, they can view and edit their payment details. Customers can make the payments belongs to them and they can also view their bills. This is the whole process of payment management system.

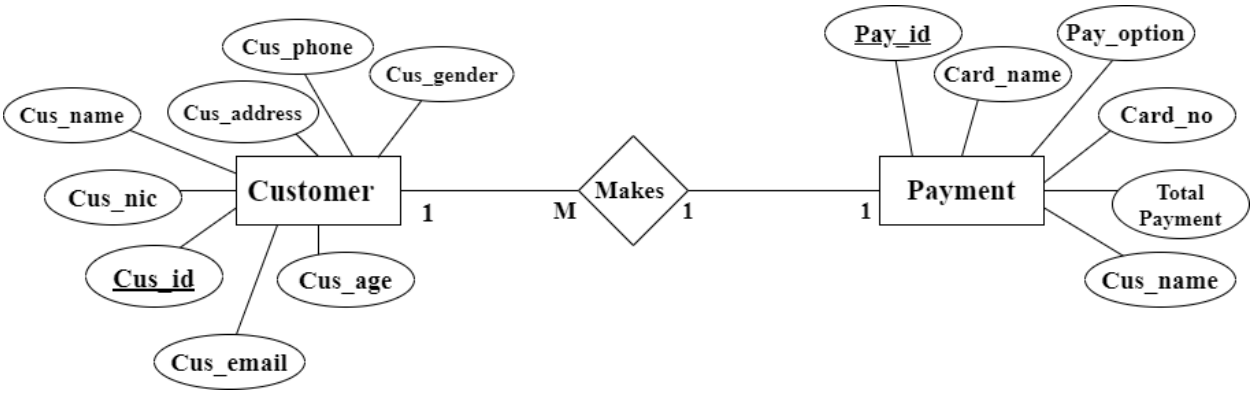
Payment Activity Diagram



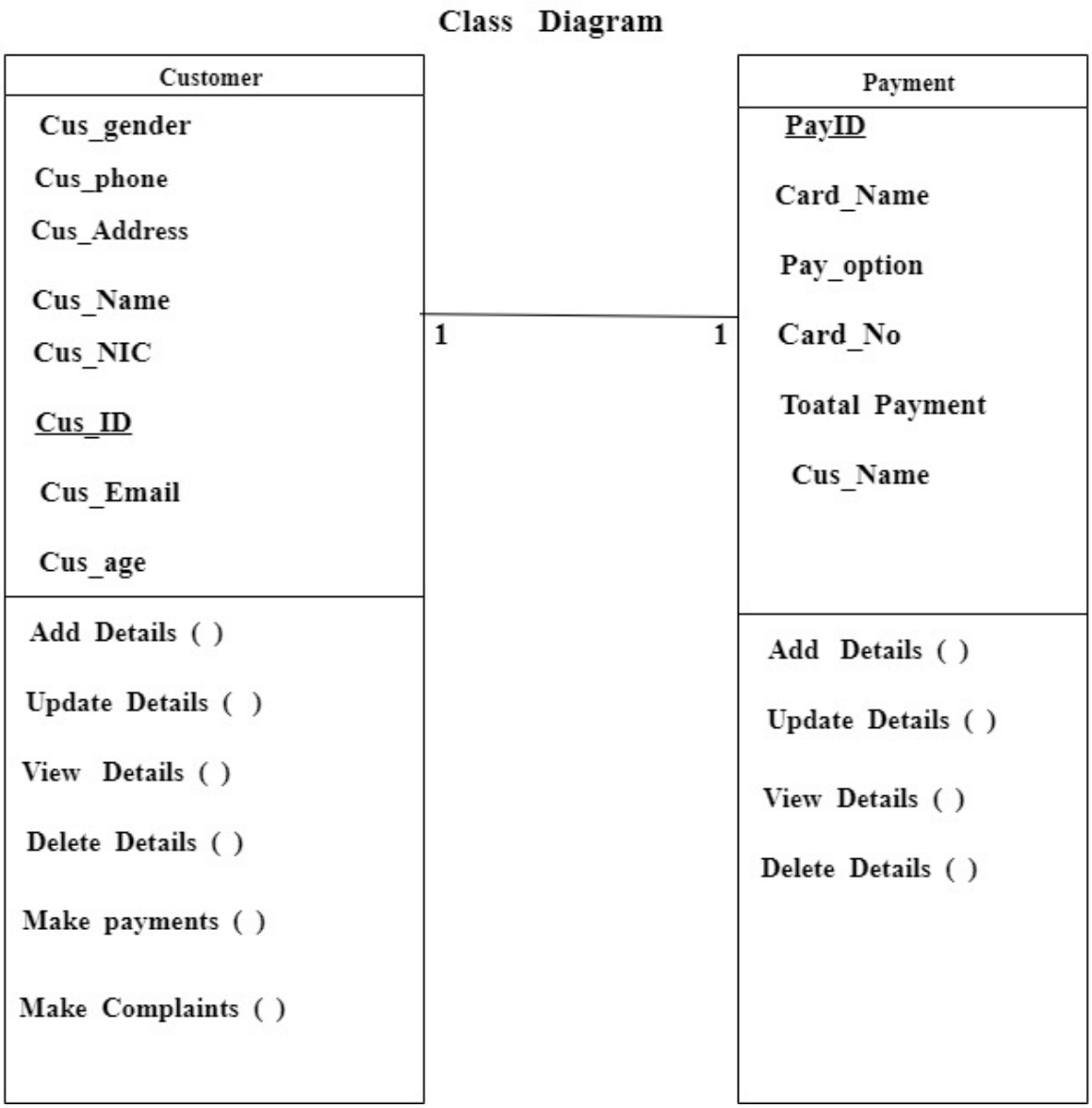
Payment Sequence Diagram



Payment ER diagram



Class diagram



Payment

Create – POST

URL - http://localhost:8080/ElectroGrid/rest/payments

Media raw JSON

```
{
    "payOptions":"Card",
    "totalPayment":"1000",
    "cardName":"HNB",
    "customerName":"Kavindu",
    "customerId":1
}
```

Read – GET

URL - http://localhost:8080/ElectroGrid/rest/payments

Response

```
[
    {
        "paymentId": 1,
        "payOptions": "Card",
        "totalPayment": 1000,
        "cardName": "HNB",
        "customerName": "Kavindu",
        "customerId": 1
    }
]
```

Update – PUT

URL - http://localhost:8080/ElectroGrid/rest/payments/2

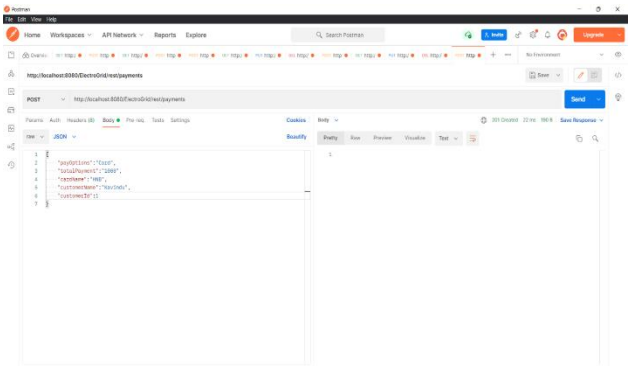
Media raw JSON

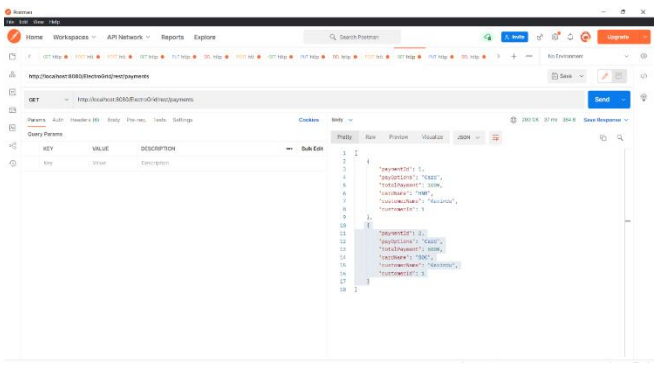
```
 {
```

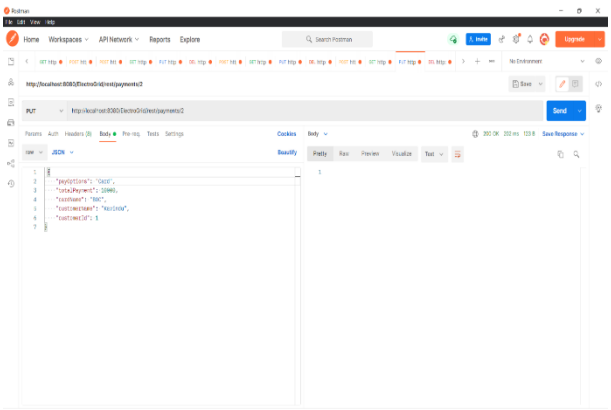
```
    "payOptions": "Card",
    "totalPayment": 10000,
    "cardName": "BOC",
    "customerName": "Kavindu",
    "customerId": 1
}
```

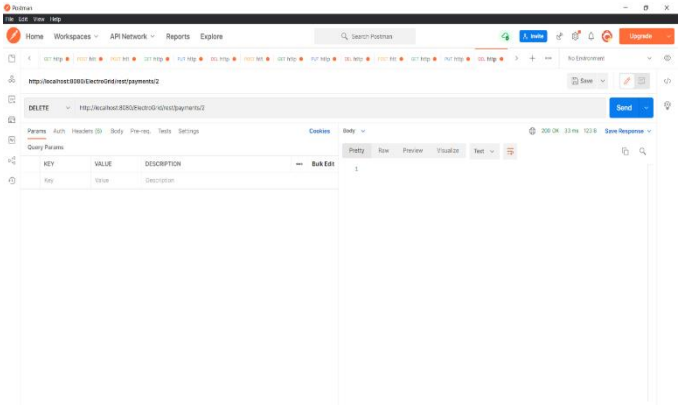

Delete – DELETE

URL - http://localhost:8080/ElectroGrid/rest/payments/2









Test Cases

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

1	Add Payment Details	{
		"payOptions":"Card",
		"totalPayment":"1000",
		"cardName":"HNB",
		"customerName":"Kavindu",
		"customerId":1
		}
			New Payment details are added to the database.	New Payment details are added to the database.	**PASS**

2	Update Payment Details	{
		"payOptions": "Card",
		"totalPayment": 10000,
		"cardName": "BOC",
		"customerName": "Kavindu"
		,
		"customerId": 1
		}
			Payment details are updated according to relevant input payment details.	Payment details are updated according to relevant input payment details.	**PASS**

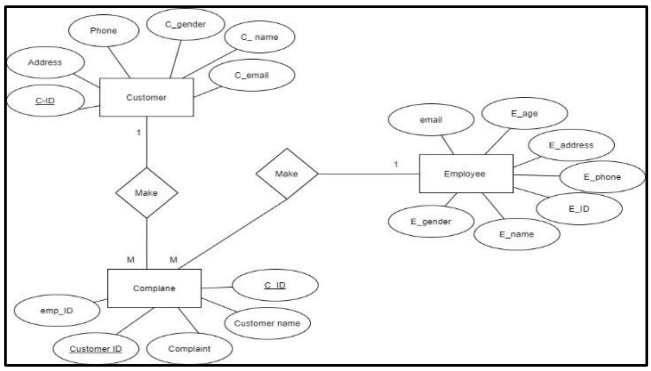
3	Delete Payment Details		Payment details are deleted from the database.	Payment Details are deleted successfully.	**PASS**

IT20018900 (Rathnayaka R.M.A.N) URL: - http://localhost:8080/ElectroGrid/rest/complaints

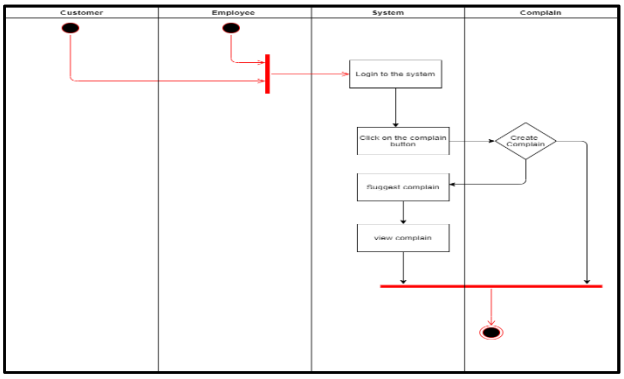
Complain management

Customer and employee can make any complaints regarding any inconvenience they may have. Also one or more complaints can be lodged. It is up to the individual to decide whether or not to make such a complaint. Customer and employee can insert, delete, update and view the complaint details.

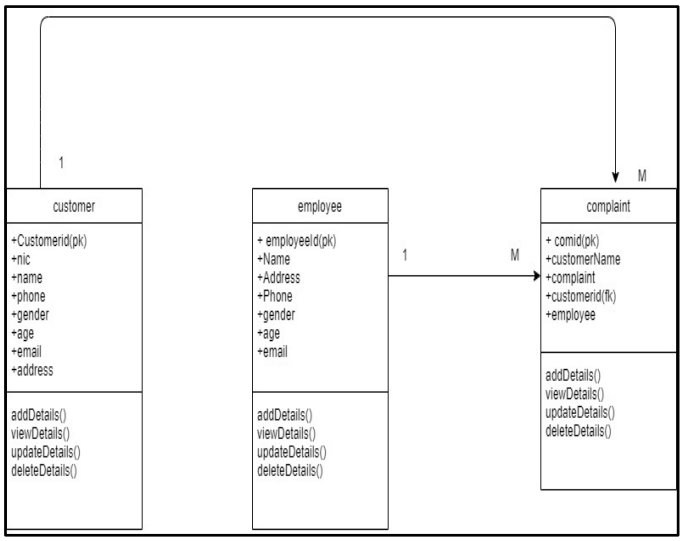
Er Diagram



Activity Diagram



Class Diagram



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

URL: -http://localhost:8080/ElectroGrid/rest/bills/6

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

URL: - http://localhost:8080/ElectroGrid/rest/complaints

Response


```
[
  {
    "comId": 1,
    "customerName": "Kavindu",
    "complaint": "Bill payment not debited",
    "customerId": 1,
    "employeeId": 1
  }
]
```


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

`URL: - http://localhost:8080/ElectroGrid/rest/complaints

Media raw JSON


```
{
   "customerName":"Kavindu",
   "complaint":"Bill not send to me",
   "customerId":1,
   "employeeId":1
}
```


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

`URL: - http://localhost:8090/GadgetBadget/InventorService/Inventors

Media raw JSON


```
{
   "customerName": "Kavindu",
   "complaint": "Monthly Bill payment is done",
  "customerId": 1,
   "employeeId": 1
```

}

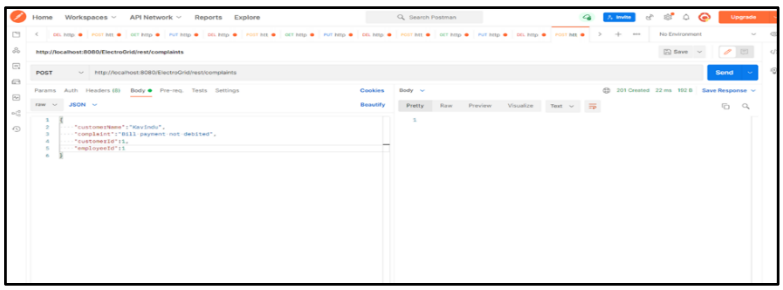
Test Cases

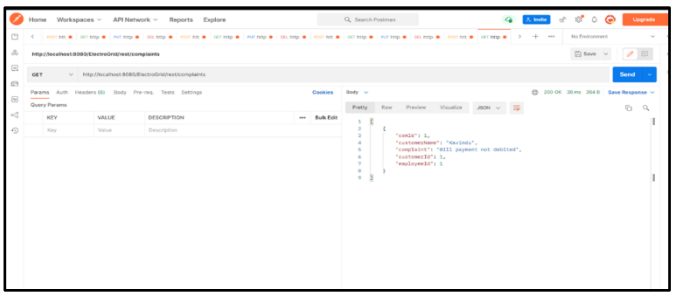
Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)

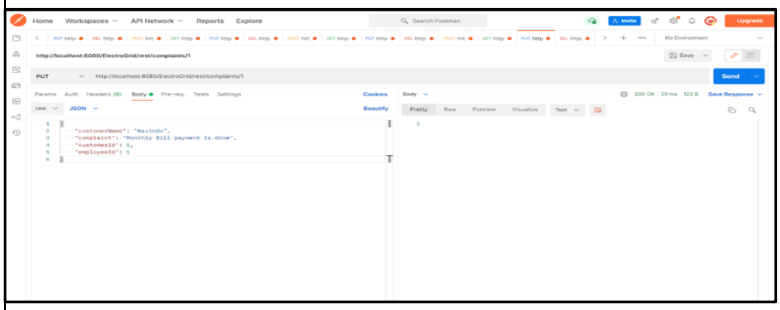
1	Add complain Details	{
		"customerName":"Kavindu",
		"complaint":"Bill not send to me",
		"customerId":1,
		"employeeId":1
		}
			New complain details are added to the database.	New complain details are added to the database.	PASS

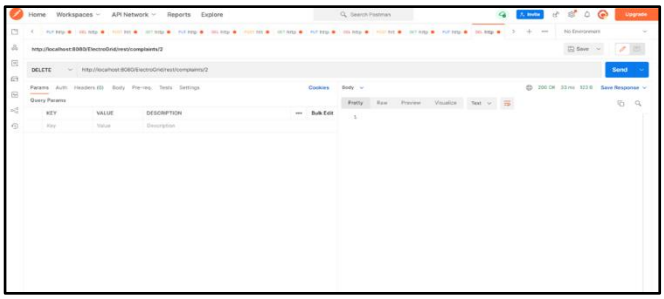
2	Update complain Details	{
		"customerName": "Kavindu",
		"complaint": "Monthly Bill payment is done",
		"customerId": 1,
		"employeeId": 1
		}
			complain details are updated according to relevant input complain details.	complain details are updated according to relevant input complain details.	PASS

3	Delete complain Details		Complain Details are deleted successfully.	Complain Details are deleted successfully.	PASS





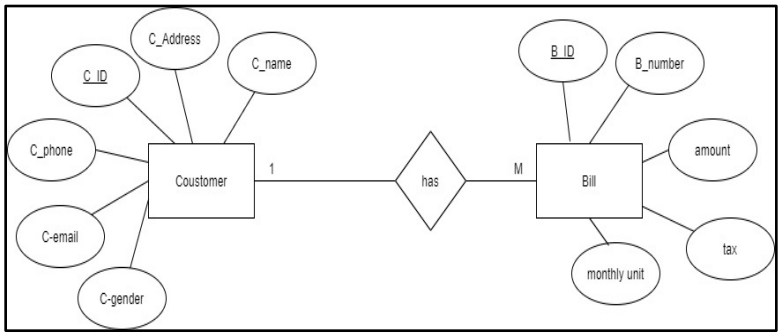




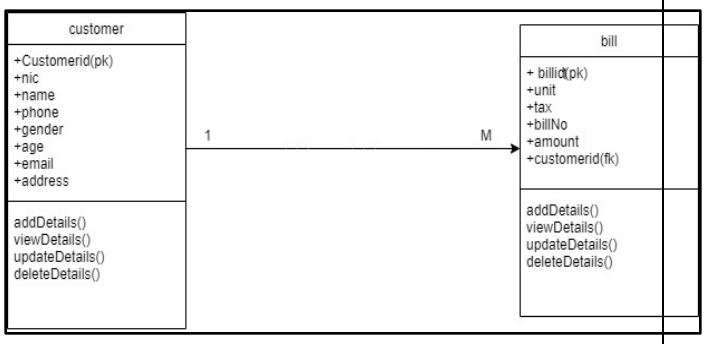
Bill management system URL-http://localhost:8080/ElectroGrid/rest/bills

Monthly bills are issued according to the amount of electricity consumed by the customer. According to this system, the bill is issued automatically after the customer enters the amount of watts he has consumed.

ER diagram



Class diagram



API for get all the Inventor details in the database (GET Request) URL: - http://localhost:8080/ElectroGrid/rest/bills

Response


```
[
  {
    "billId": 5,
    "unit": 150,
    "tax": 1940,
    "billNo": "B0001",
    "amount": 4500,
    "customerId": 1
  }
]
```


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

Media raw JSON


```
{
 "unit":450,
  "tax":5000,
  "billNo":"B0002",
  "amount":10000,
  "customerId":1
}
```


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

`URL: - http://localhost:8080/ElectroGrid/rest/bills/5

Media raw JSON


```
{
  "unit": 150,
  "tax": 1940,
  "billNo": "B0001",
```

"amount": 4500,

"customerId": 1

}

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

URL: - http://localhost:8080/ElectroGrid/rest/bills/6

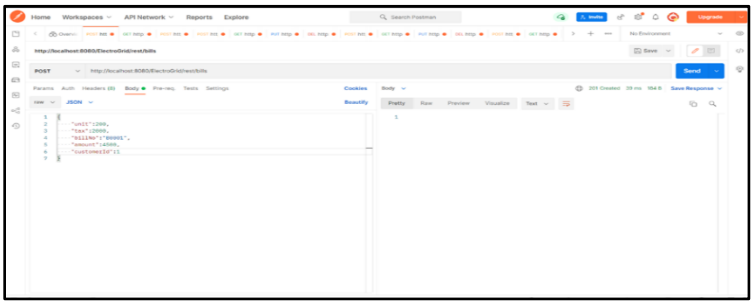
Test Cases

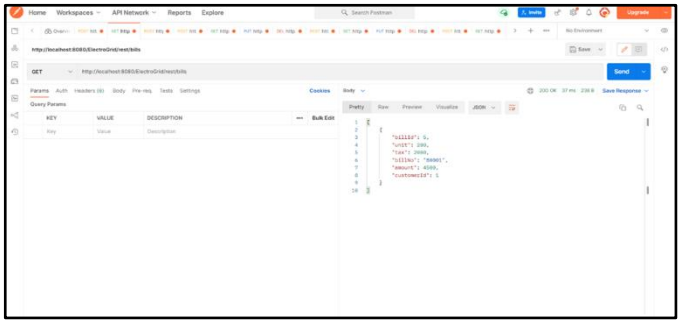
Test ID	Test Description/ Test Steps	Test Input(s)	Expected Output(s)	Actual Output(s)	Result (Pass/Fail)

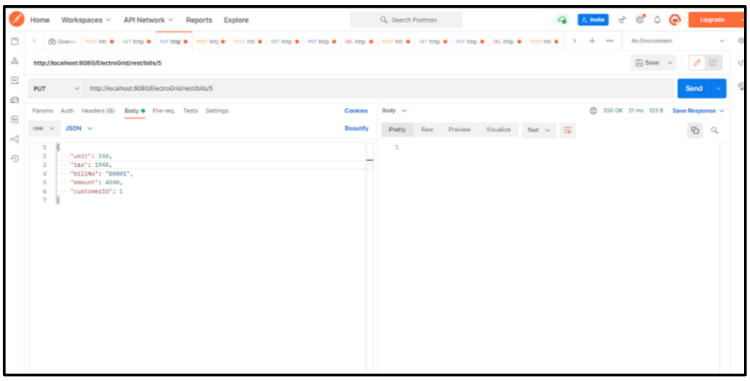
1	Add bill Details	{
		"unit":450,
		"tax":5000,
		"billNo":"B0002",
		"amount":10000,
		"customerId":1
		}
			New bill details are added to the database.	New bill details are added to the database.	PASS

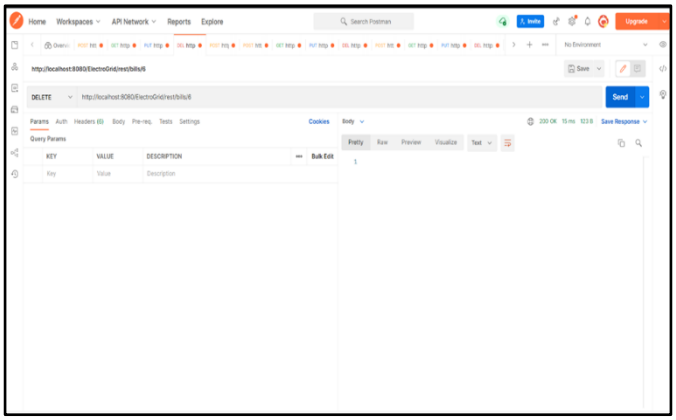
2	Update bill Details	{
		"unit": 150,
		"tax": 1940,
		"billNo": "B0001",
		"amount": 4500,
		"customerId": 1
		}
			bill details are updated according to relevant input bill details.	bill details are updated according to relevant input bill details.	PASS

3	Delete bill Details		bill Details are deleted successfully.	bill Details are deleted successfully.	PASS









Tools Used

• Java-JAX-RS (Jersey): Used for backend development

• Eclipse: Use as IDE

• Apache Tomcat V.10: Used as the server

• My SQL Used as our database

References:

- https://www.youtube.com/watch?v=N1UzS9ojuUY
- https://developer.ibm.com/articles/wa-aj-tomcat/
- https://www.vogella.com/tutorials/REST/article.html
- https://www.codejava.net/java-ee/web-services/java-restful-web-services-tutorial-for-beginner-with-jersey-and-tomcat
- https://www.youtube.com/watch?v=-_VPzhKJPfE

Appendix:

Use case and overall architecture diagram, ER diagrams, class Diagram have been added in earlier slides, each and every individual user have separately added their appendix in their respected part