Programming Application Framework (IT3030)

2022

ElectroGrid (EG)

Power Grid Management System

Batch: Y3.S1.WD.DS.05.02

Group: 160

Submitted by:

1. IT20273712– (Subasinghe S. S)
2. IT20155698– (Rajapaksha M.T.U.R)
3. IT20161088– (Pathirana D.P.C.H)
4. IT20216900– (De Silva S. R)
5. IT20785192– (Isurika W.B.M.A)

Submitted to:

Mr. Nalaka Dissanayake

Table of Contents

Work Distribution

Member	Web Service	Functions
IT20161088– (Pathirana D.P.C.H)	User Service	• Register users (consumers) to the system. • View user list • Update and Delete users.
IT20216900– (De Silva S. R)	Bill Service	• Generate a new bill • View added bill list • Update and delete added bills
IT20155698– (Rajapaksha M.T.U.R)	Payment Service	• Add a new payment • View payments • Update and delete payment details
IT20273712 – (Subasinghe S. S)	Overdue Payment Service	• Add overdue payments • View overdue payment details • Update and delete overdue payment details
IT20785192– (Isurika W.B.M.A)	Support Service	• Add complaints • View complaint details • Update and delete complaint details

Introduction

ElectroGrid is an online system which manages the power consumption of users. The systems support automatic bill creation, sharing of bills to users, user payment handling, overdue payment handling and complaint handling.

1) **User Service Implementation**
   There are mainly 3 types of users in the system. Consumer, admin, and manager. To check the bill balance, monthly electricity usage, overdue payment information through the system consumer must be registered to the system by the admin. When registering to the system as a valid user consumer should provide name, address, email, phone number. The administrator of the system will check the details and accept their registering request. After that they can login to the system by providing valid credentials and check their electricity usage and do payments. As same the consumer, the manager also must register to the system to calculate bills, accept payments, and manage overdue information. The managers are added by the administrator of the system. Moreover, the administrator of the system can search for users, update details of the users and remove inactive users from the system.

2) **Bill Service Implementation**
   Bill service mainly focuses on the management of bills according to the consumption of the consumers. Bills are handled by the bill handler and this service uses the user service. The monthly payment amount is calculated in this service and added bills can be updated or deleted when a consumer complaint is received.

3) **Payment Service Implementation**
   After confirming the relevant payment, Consumer can add payment details such as payment method(visa/master), card details (card number, name on card, cvc, expire date), bill amount to go ahead with procedure. System will make sure to validate payment details by avoiding null values and inaccurate data formats. Consumer can view all his/her payment details by payment id. Admin can manage payment details by updating payment details and deleting unnecessary payment details of the system. Payment details will be saved after payment get authorized.

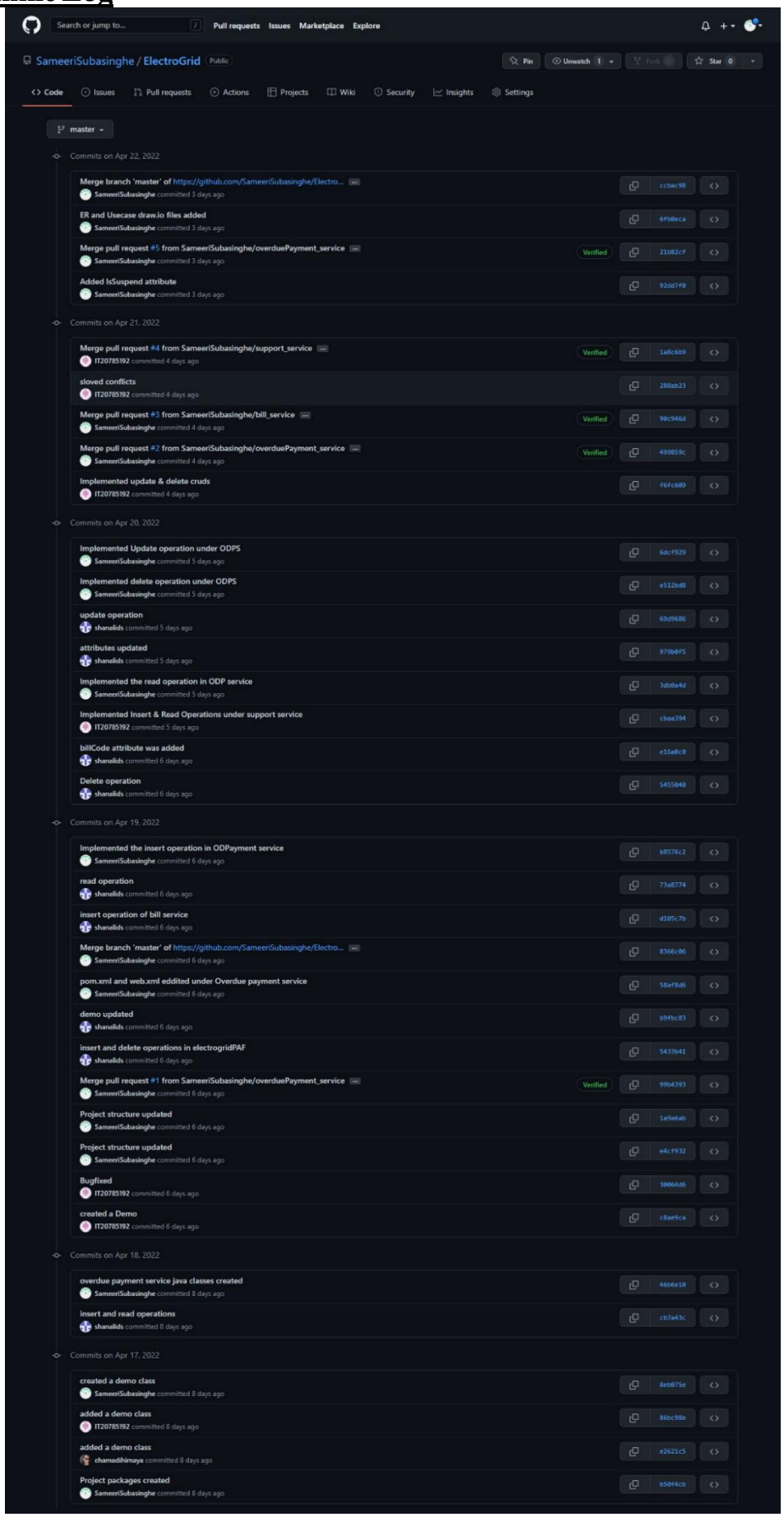
4) **Overdue Payment Service Implementation**
Overdue Payment service is mainly focuses on the unpaid payments, service suspension and service restoration. User service is used by this service. After checking the due payments, the overdue payment handler can add new overdue payments to the system. Updating and deleting an overdue payment can be done if needed.

5) **Support Service Implementation**
Support service is mainly focuses on helping consumers' problems. User service is mainly used by this service. After registering and logging into the system the existing customer can search complaint submission from the system. Then, consumer can complaint their issues directly by filling the form. After adding complaints if they want to change it, they can edit and submit. Consumer can view all complaints and delete their complaints if they need.

GitHub Details

(Database included)
https://github.com/SameeriSubasinghe/ElectroGrid.git

Commit Log


Search or jump to...
Pull requests
Issues
Marketplace
Explore
SameeriSubasinghe / ElectroGrid
Public
Pin
Unwatch 1
Fork
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
Commits on Apr 22, 2022
Merge branch 'master' of https://github.com/SameeriSubasinghe/Electro...
SameeriSubasinghe committed 3 days ago
ccbac98
ER and Usecase draw.io files added
SameeriSubasinghe committed 3 days ago
6fb0eca
Merge pull request #5 from SameeriSubasinghe/overduePayment_service
SameeriSubasinghe committed 3 days ago
Verified
21b82cf
Added IsSuspend attribute
SameeriSubasinghe committed 3 days ago
92dd7f0
Commits on Apr 21, 2022
Merge pull request #4 from SameeriSubasinghe/support_service
IT20785192 committed 4 days ago
Verified
1a8c6b9
sloved conflicts
IT20785192 committed 4 days ago
288ab23
Merge pull request #3 from SameeriSubasinghe/bill_service
SameeriSubasinghe committed 4 days ago
Verified
98c946d
Merge pull request #2 from SameeriSubasinghe/overduePayment_service
SameeriSubasinghe committed 4 days ago
Verified
499859c
Implemented update & delete cruds
IT20785192 committed 4 days ago
f6fc609
Commits on Apr 20, 2022
Implemented Update operation under ODPS
SameeriSubasinghe committed 5 days ago
6dcf929
Implemented delete operation under ODPS
SameeriSubasinghe committed 5 days ago
e512bd8
update operation
shanalids committed 5 days ago
69d9686
attributes updated
shanalids committed 5 days ago
979b0f5
Implemented the read operation in ODP service
SameeriSubasinghe committed 5 days ago
3db0a4d
Implemented Insert & Read Operations under support service
IT20785192 committed 5 days ago
cbaa394
billCode attribute was added
shanalids committed 6 days ago
e16a0c0
Delete operation
shanalids committed 6 days ago
5455040
Commits on Apr 19, 2022
Implemented the insert operation in ODPayment service
SameeriSubasinghe committed 6 days ago
b8576c2
read operation
shanalids committed 6 days ago
73a8774
insert operation of bill service
shanalids committed 6 days ago
d185c7b
Merge branch 'master' of https://github.com/SameeriSubasinghe/Electro...
SameeriSubasinghe committed 6 days ago
8366c06
pom.xml and web.xml eddited under Overdue payment service
SameeriSubasinghe committed 6 days ago
58af8d6
demo updated
shanalids committed 6 days ago
b9fbc83
insert and delete operations in electrogridPAF
shanalids committed 6 days ago
5433b41
Merge pull request #1 from SameeriSubasinghe/overduePayment_service
SameeriSubasinghe committed 6 days ago
Verified
99b4393
Project structure updated
SameeriSubasinghe committed 6 days ago
1a9a6ab
Project structure updated
SameeriSubasinghe committed 6 days ago
e4cf932
Bugfixed
IT20785192 committed 6 days ago
30064d6
created a Demo
IT20785192 committed 6 days ago
c8ae9ca
Commits on Apr 18, 2022
overdue payment service java classes created
SameeriSubasinghe committed 8 days ago
46b8e10
insert and read operations
shanalids committed 8 days ago
cb7a43c
Commits on Apr 17, 2022
created a demo class
SameeriSubasinghe committed 8 days ago
8eb875e
added a demo class
IT20785192 committed 8 days ago
86bc98e
added a demo class
chamadihimaya committed 8 days ago
e2621c5
Project packages created
SameeriSubasinghe committed 8 days ago
b58f4cb

Search or jump to... / Pull requests Issues Marketplace Explore

SameeriSubasinghe / ElectroGrid Public

Pin | Unwatch 1 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master

Commits on Apr 24, 2022

Merge pull request #13 from SameeriSubasinghe/support_service IT20785192 committed 17 hours ago	Verified	f158292
Merge pull request #12 from SameeriSubasinghe/userservice chamadihimaya committed 17 hours ago	Verified	cd0e86a
error fixed chamadihimaya committed 17 hours ago		4330ae7
structure error fixed chamadihimaya committed 17 hours ago		7017ffe
Delete ElectroGrid_Services/UserService directory chamadihimaya committed 17 hours ago	Verified	58546b4
mastetr branch pulled chamadihimaya committed 17 hours ago		eea6908
final commit chamadihimaya committed 17 hours ago		57b9ff7
Merge pull request #11 from SameeriSubasinghe/overduePayment_service SameeriSubasinghe committed yesterday	Verified	a9307d0
Merged SameeriSubasinghe committed yesterday		ba8d725
Merged bill service diagrams SameeriSubasinghe committed yesterday		743373f
Merge pull request #10 from SameeriSubasinghe/newpaymentservice TharinduUdaynga committed yesterday	Verified	1ae813d
payment service added TharinduUdaynga committed yesterday		1d66d17
Delete ElectroGrid_Services/Payment_services directory TharinduUdaynga committed yesterday	Verified	bc561ce
Activity diagram added under ODPS SameeriSubasinghe committed yesterday		16df8d4
Activity diagram added - OPS_update SameeriSubasinghe committed yesterday		30740dd
Merge pull request #9 from SameeriSubasinghe/bill_service shanalids committed yesterday	Verified	217fcde
Merge branch 'master' of https://github.com/SameeriSubasinghe/Electro... shanalids committed yesterday		4f7292b
billService ER diagram error fixed shanalids committed yesterday		d563ac5
billService usecase diagram added shanalids committed yesterday		83980a0
billService ER diagram added shanalids committed yesterday		293f33f
bill class diagram added shanalids committed yesterday		1df47cf
billService activity diagram added shanalids committed yesterday		fc76a1d
Merge pull request #8 from SameeriSubasinghe/overduePayment_service SameeriSubasinghe committed yesterday	Verified	4fa5ea0
Database queries added SameeriSubasinghe committed yesterday		678e9b5
last crud TharinduUdaynga committed 2 days ago		d7ce47c
all curd TharinduUdaynga committed 2 days ago		b273352
curd chamadihimaya committed 2 days ago		e99df36
Delete Payment_Service TharinduUdaynga committed 2 days ago	Verified	f7026e8
Payment_Service TharinduUdaynga committed 2 days ago	Verified	e0eca84

Commits on Apr 23, 2022

Merge branch 'master' of https://github.com/SameeriSubasinghe/Electro... IT20785192 committed 2 days ago	e5c251e
Added Activity Diagram IT20785192 committed 2 days ago	8d4deaf
Design diagrams added SameeriSubasinghe committed 3 days ago	6e8ef8d

Commits on Apr 22, 2022

Merge branch 'master' of https://github.com/SameeriSubasinghe/Electro... IT20785192 committed 3 days ago	2641f56
Design Diagrams IT20785192 committed 3 days ago	10692ec
removed unwanted folder SameeriSubasinghe committed 3 days ago	37c36f4

SE Methodologies

- **Description**
  Agile approach is an iterative software development paradigm. This is a project management methodology that divides a project into stages. The project cycle goes through planning, implementing, executing, and evaluating, and it entails continual development at every stage. As a result, this approach enables to identify software flaws considerably sooner and deliver a viable and workable solution earlier than compared to following other software development approaches.

- **The Usage**
  o The requirements are well defined.
  o The system development is carried out using well known tools.

- **Advantages**
  o Continuous development.
  o Facilitate late changes to requirements.
  o Early detection of bugs.
  o Delivery of a working software at early stages of the project.

- **Disadvantages**
  o Lack of attention to designing.
  o Requires a considerable level of expertise to complete the project implementation.
  o Difficult to scale the scope of the project in early stages.

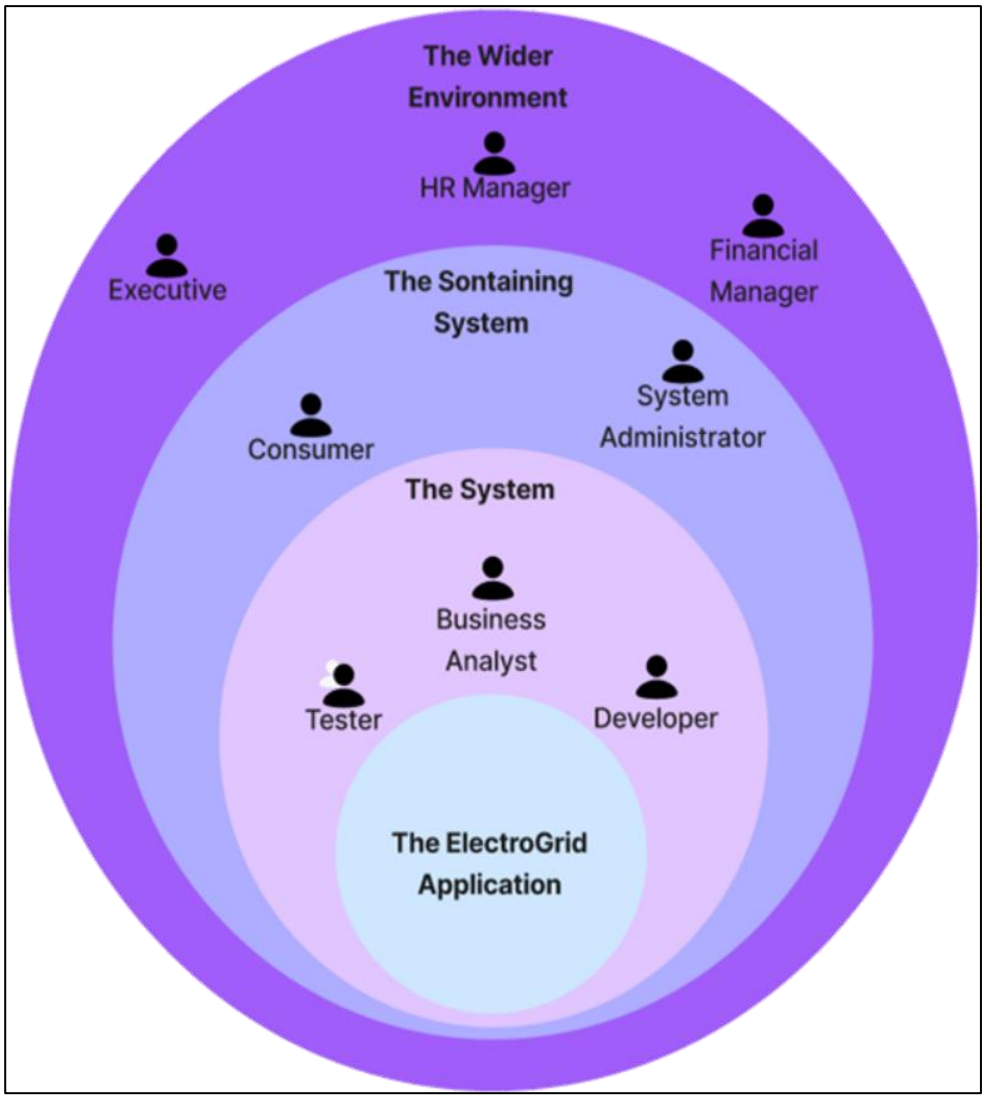
Time schedule (Gantt chart)

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
1) Gathering information & requirements										
2) Divide Functions										
3) Fixing Eclipse and maven										
4) Create ER diagram, Usecase diagram and Activity diagram										
5) Feedback session										
6) Create Database										
7) Individual planning and create GitHub repository										
8) Implementation										
9) Update commits										
10) Finalize the system										
11) Start individual and overall report										
12) Finalize the report										

Requirements Analysis

Requirement analysis focuses on both user and system requirements. Main stakeholders of the ElectroGrid web system are Consumer, System administrator, finance manager etc. This application is a restful web application and application have five services. User service, Bill service, Payment service, Overdue Payment service and Support service.

1) Stakeholder Analysis (onion diagram)



2) **Technical Requirements**

- Technical requirements are the technical points that must be taken into consideration to deliver a successful system.
- User, Bills, Payments, Overdue Payments and Complaint details can be updated, deleted, and viewed as required.

3) **Functional Requirements**

1) **User management**
    - Register users (consumers) to the system.
    - View user list
    - Update and Delete users.

2) **Bill management**
    - Generate a new bill
    - View added bill list
    - Update and delete added bills

3) **Payment management**
    - Add a new payment
    - View payments
    - Update and delete payment details

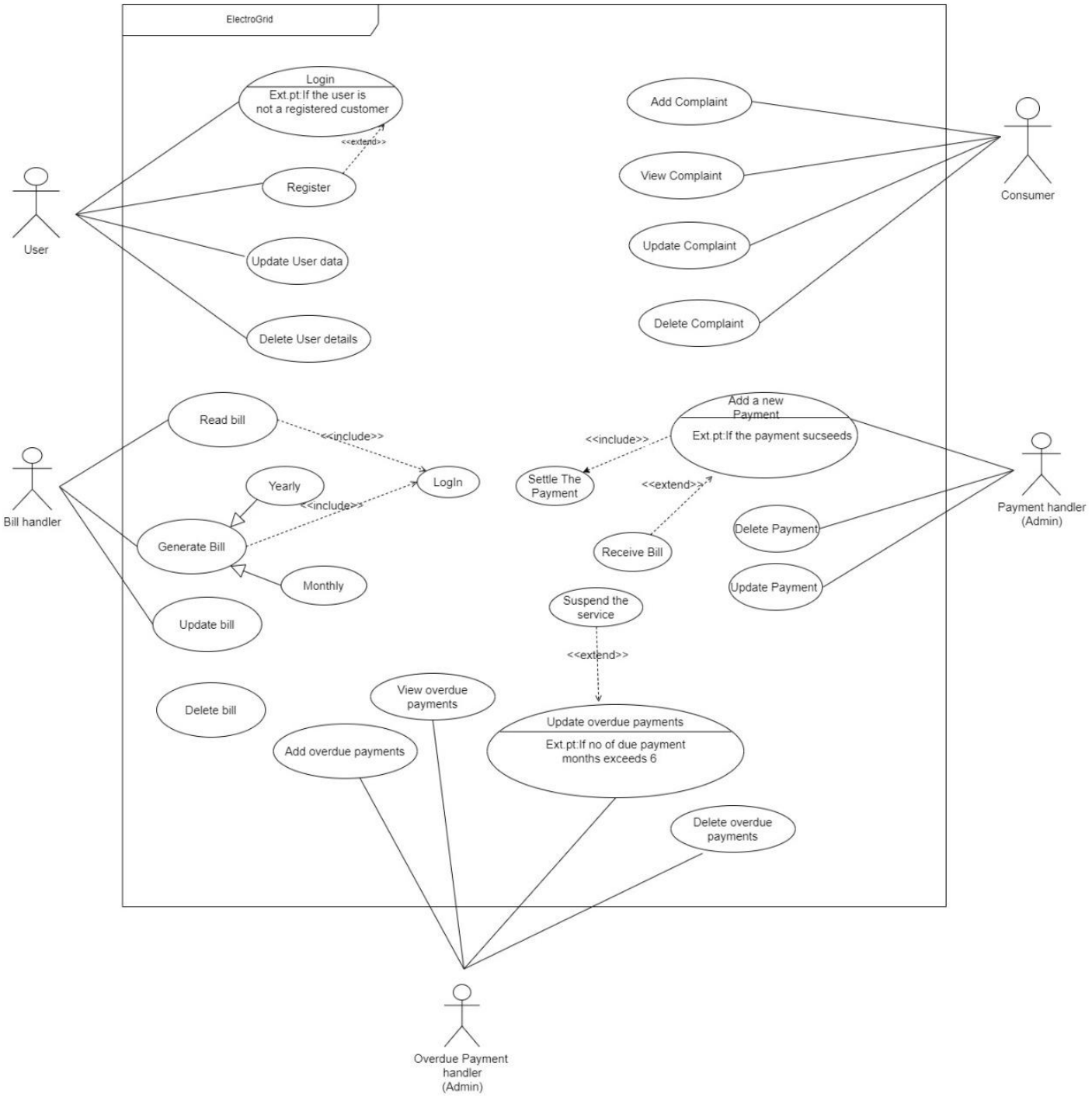
4) **Overdue Payment management**
    - Add overdue payments
    - View overdue payment details
    - Update and delete overdue payment details

5) **Support management**
    - Add complaints
    - View complaint details
    - Update and delete complaint details

4) **Non – Functional Requirements**

- Performance – Response time, Throughput, Utilization, user interface design, Conformity
- Security requirements - All data inside the system, or parts of it, is secured from virus attacks and illegal access, thanks to security regulations. The logins for each user are different. As a result, only the persons involved may modify it.
- Availability – The system accessibility 24 * 7.
- Security – Maintenance of a system database backup.
- Software Quality Attributes
    - Availability
    - Maintainability
    - Usability
    - Accuracy
    - Accessibility
    - Reliability

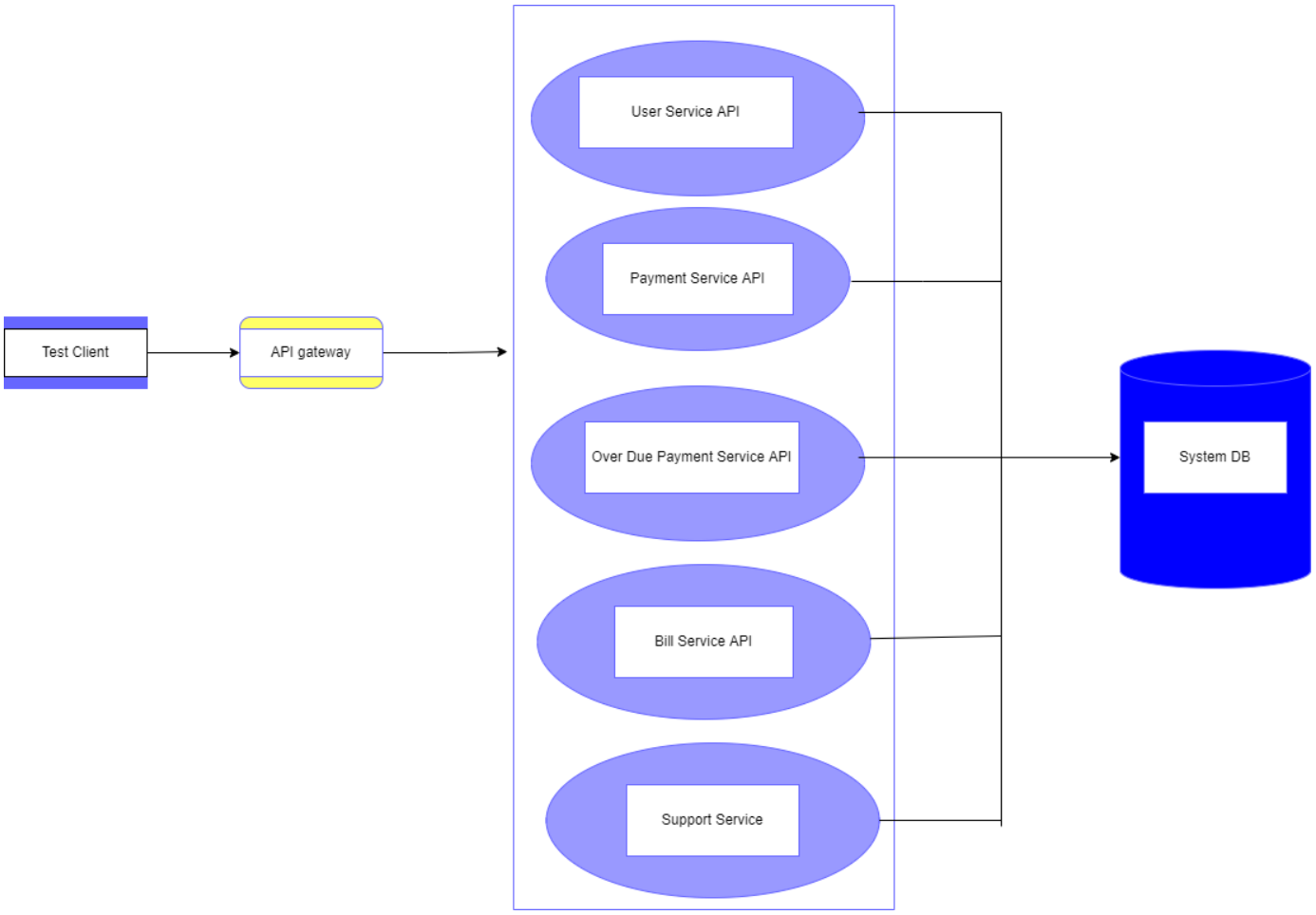
5) Requirements Modelling (Use case Diagram)



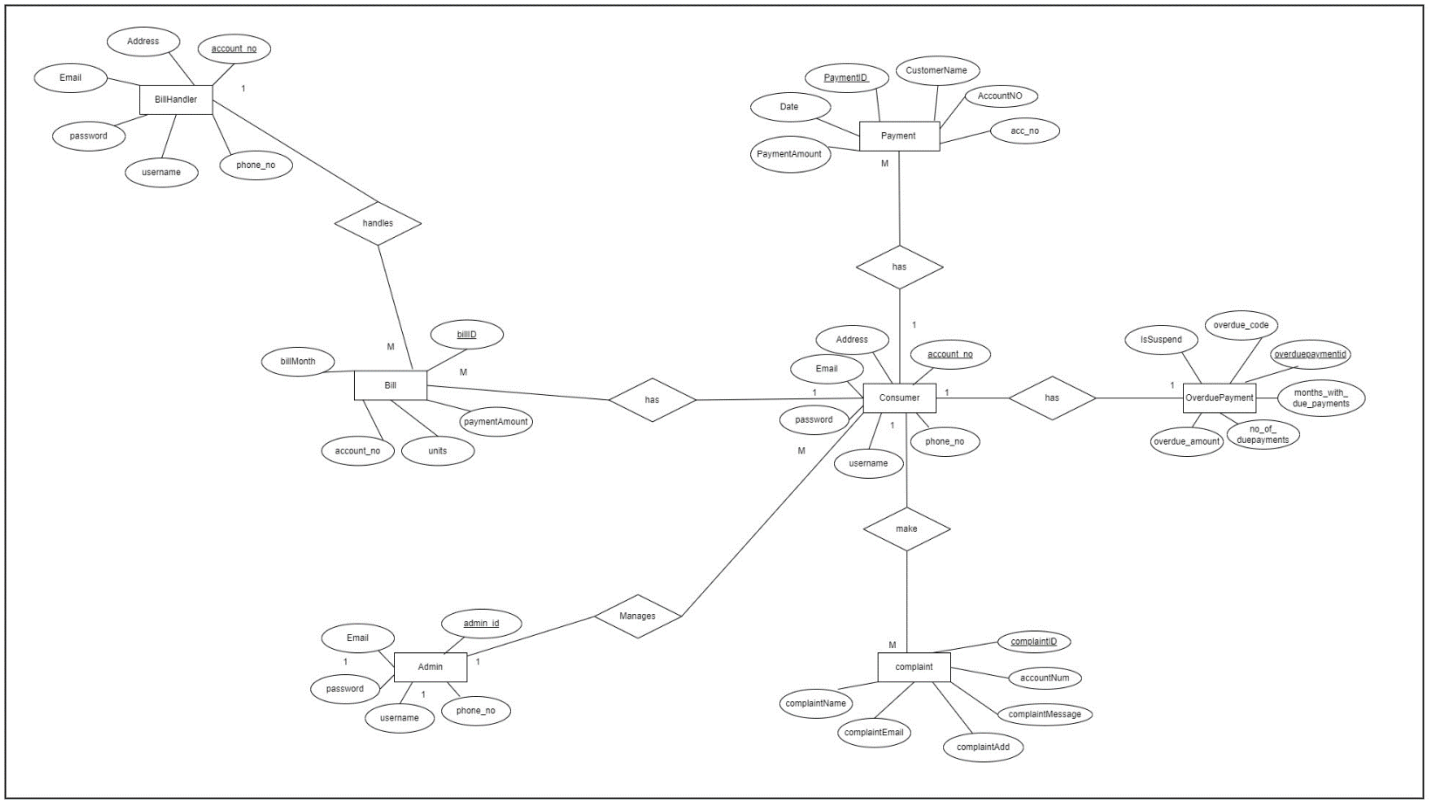
System's overall design

1) Overall Architecture

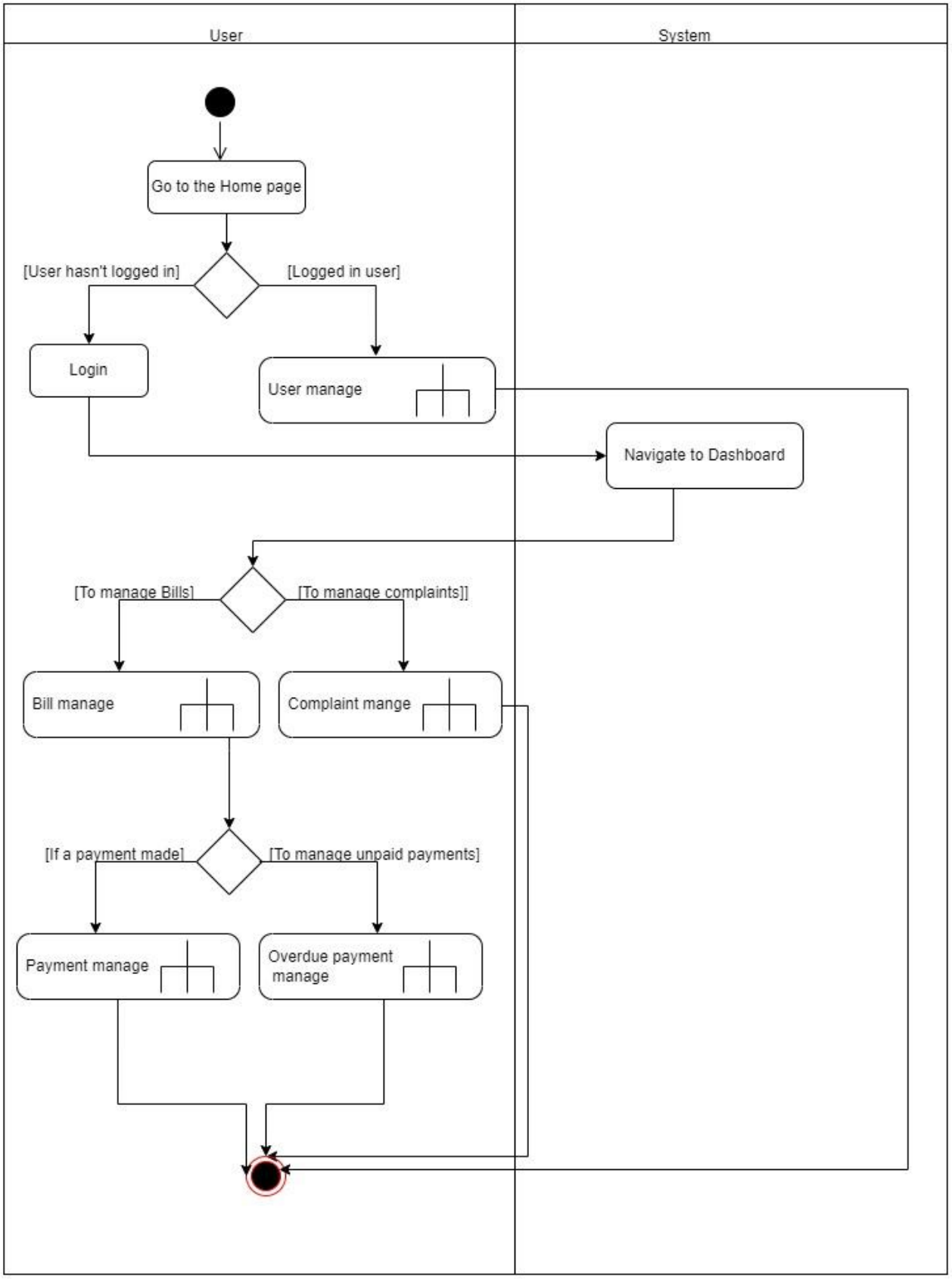
ElectroGrid is a power management system where the registered users can record the user details, power consumption, billing, payments, overdue payments and complaints
This management system consists of five services as User Service, Payment Service, Bill Service, Overdue payment Service, and Support service All of the services share a database. Postman was also used to collect the inputs and test the outputs. When a client makes a request, the data is transferred through a gateway to the relevant service, and the response is provided back to the client in the same way. APIs (Application Programming Interfaces) allow these procedures, such as improving current services, to work independently and more efficiently.



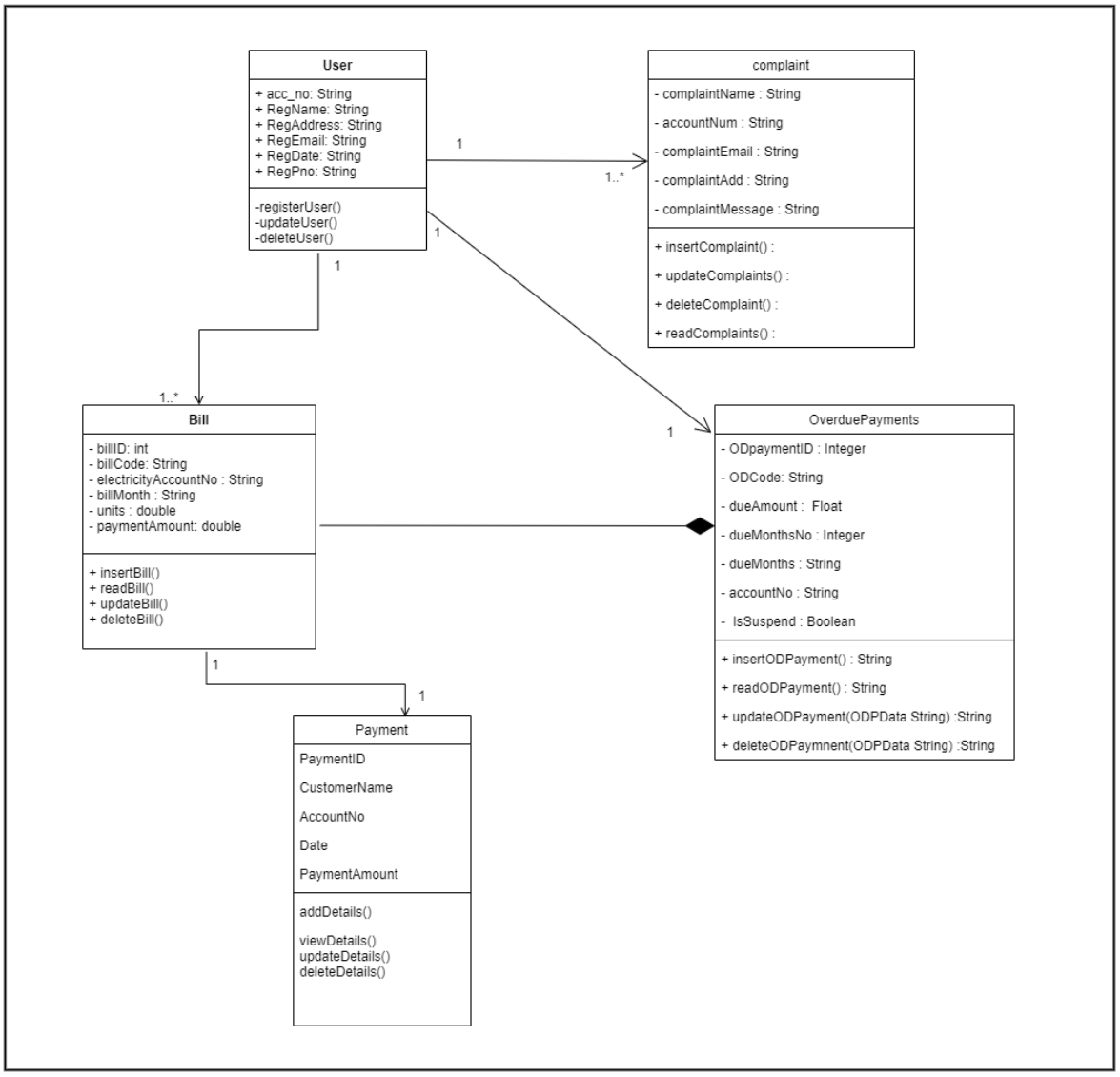
2) Overall DB Design (ER)



1) Activity Diagram



2) Overall Class Diagram



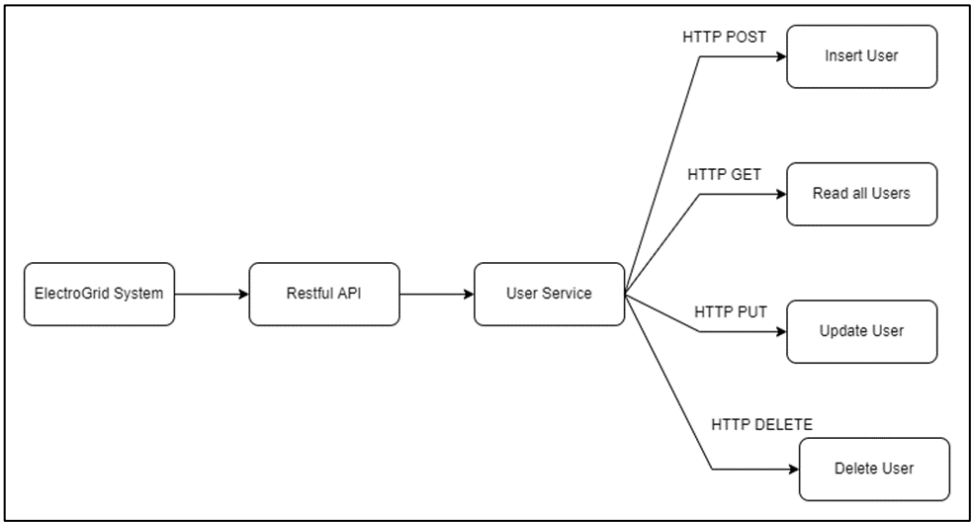
Individual Sections

User Service

1. **Service design**

   Admin can register users to the system using registration form by providing their details. After that user can login to the system as a valid user by providing valid credentials. The access to the system varies according to the user type. Admin can update and delete users if needed.

   a) API of the service



I. Create User (POST)

Resource: Registration
Request: POST User_Service/RegistrationAPI/Registration
Media: Form Data - URL encoded
Data:

regName : “Mia”
regAddress : “Malabe”
regEmail: “mia@gmail.com”
regDate: “2020.04.01”
regPNo: “0777788263”

Response: Inserted successfully
URL: http://localhost:8083/User_Service/RegistrationAPI/Registration

II. Update User (PUT)

Resource: Registration
Request: PUT User_Service/RegistrationAPI/Registration
Media: Form data – Application JSON
Data: {

```
"regName":"Mia",
"regAddress":"kaduwela",
"regEmail":"mia@gmail.com",
"regDate":"20.2.2022",
"regPNo":"0710768124",
"regID":"1"
}
```

Response: Updated successfully

URL: http://localhost:8083/User_Service/RegistrationAPI/Registration

III. View User (GET)

Resource: Registration
Request: GET User_Service/RegistrationAPI/Registration
Media: Form Data
Response: HTML table with all attributes in the User table
URL: http://localhost:8083/User_Service/RegistrationAPI/Registration

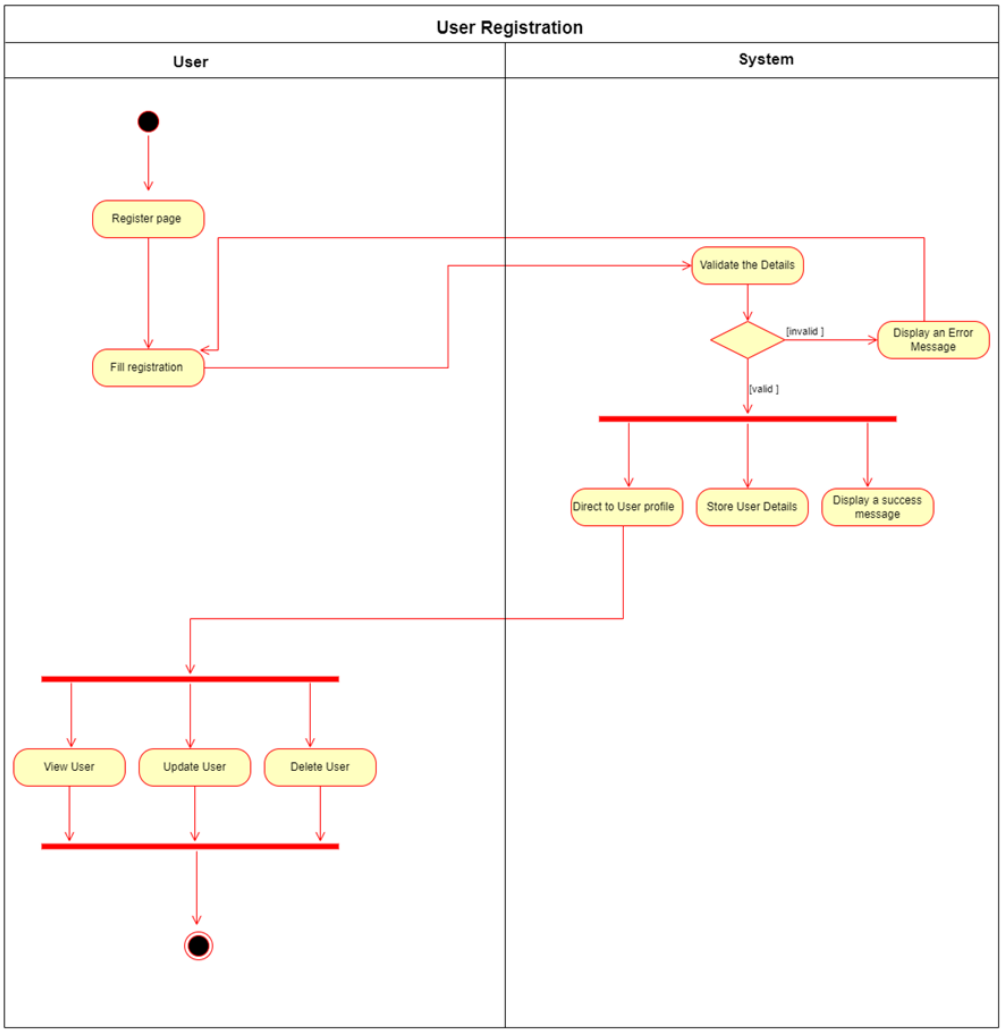
IV. Delete User (DELETE)

Resource: Registration
Request: DELETE User_Service/RegistrationAPI/Registration
Media: Application XML
Data: <regData>
<regID>1</regID>
</regData>
Response: Deleted successfully
URL: http://localhost:8083/User_Service/RegistrationAPI/Registration

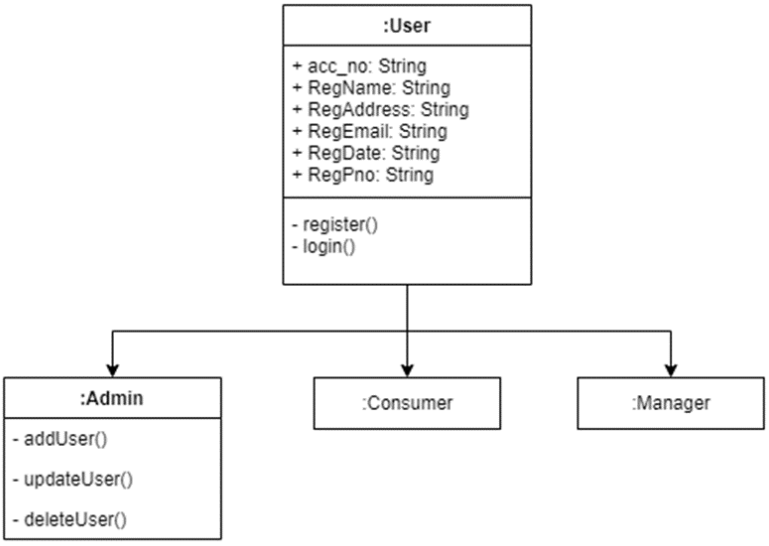
b) Internal logic

The responsibility of the user service is to manage the details of the users who register to the system. Mainly there are 3 types of users: consumer, manager, and admin. When registering to the system as a consumer they should provide name and contact details. The administrator of the system will check and accept their registering request. Then they are given the access to check their electricity usage and do payments. The manager can register to the system and calculate bills, check complains and requests of the consumers and handle overdue payment details.

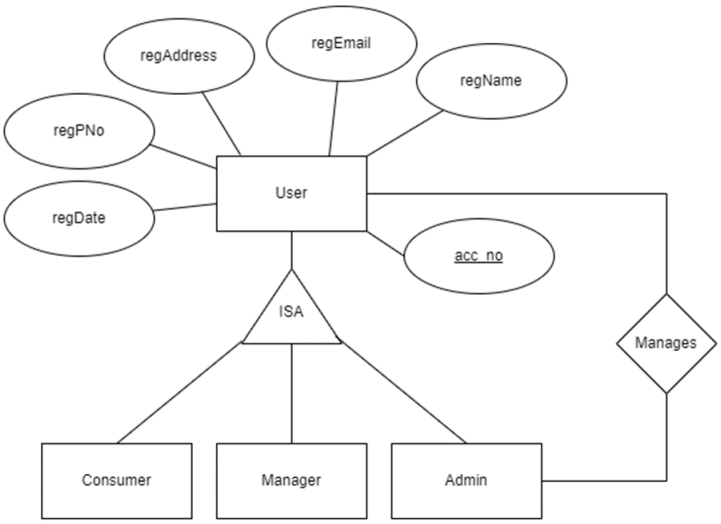
c) Activity Diagram



d) Class Diagram



e) Database for the service (ER)



2. Service development and testing

a) Tools used

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

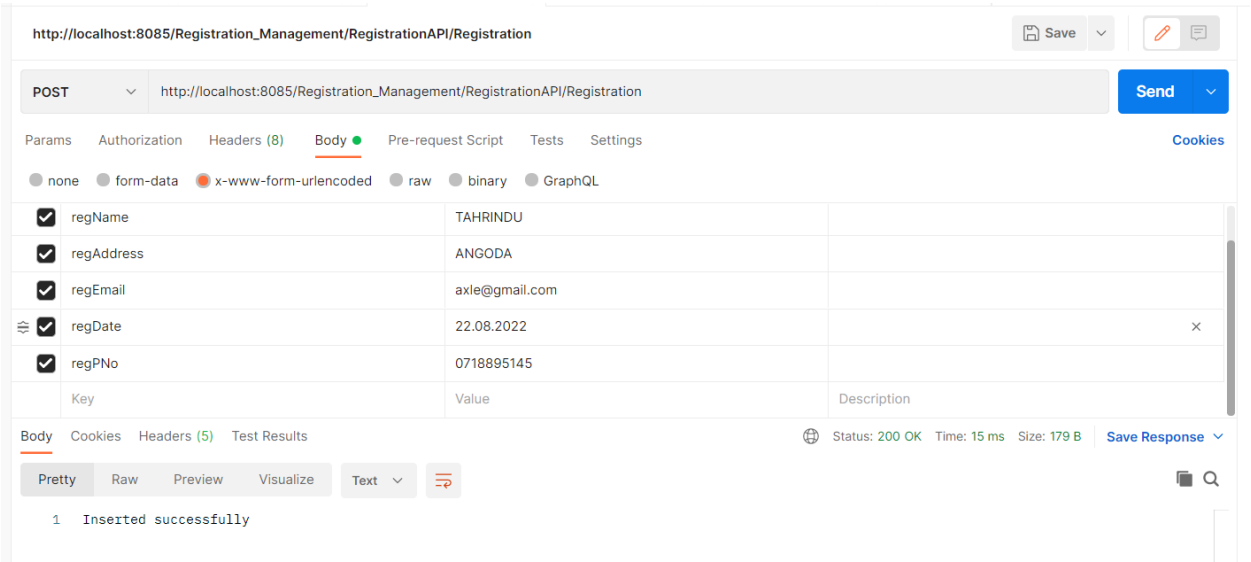
b) Testing methodology and results

Test ID	Description	Input	Expected Output	Actual Output	Result
1	Create user	regName : “Mia” regAddress : “Malabe” regEmail: “mia@gmail.com” regDate: “2020.04.01” regPNo: “0777788263”	Inserted Successfully	Inserted Successfully	Pass
2	View user		Display a HTML table with all the attributes in user table	Display a HTML table with all the attributes in user table	Pass

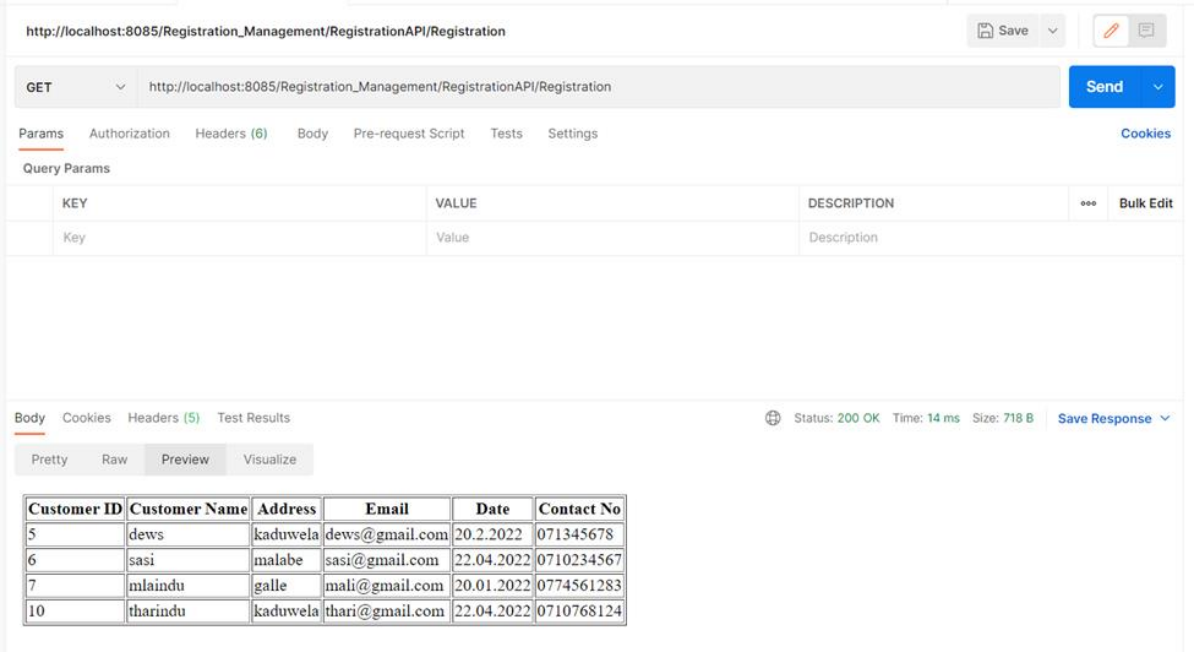
3	Update user	{
		"regName":"Mia",
		"regAddress":"kaduwela",
		"regEmail":"mia@gmail.com",
		"regDate":"20.2.2022",
		"regPNo":"0710768124",
		"regID":"1"
		}
			Updated successfully	Updated successfully	Pass

4	Delete user	<regData> <regID>1</regID> </regData>	Deleted successfully	Deleted successfully	Pass

1) Add User

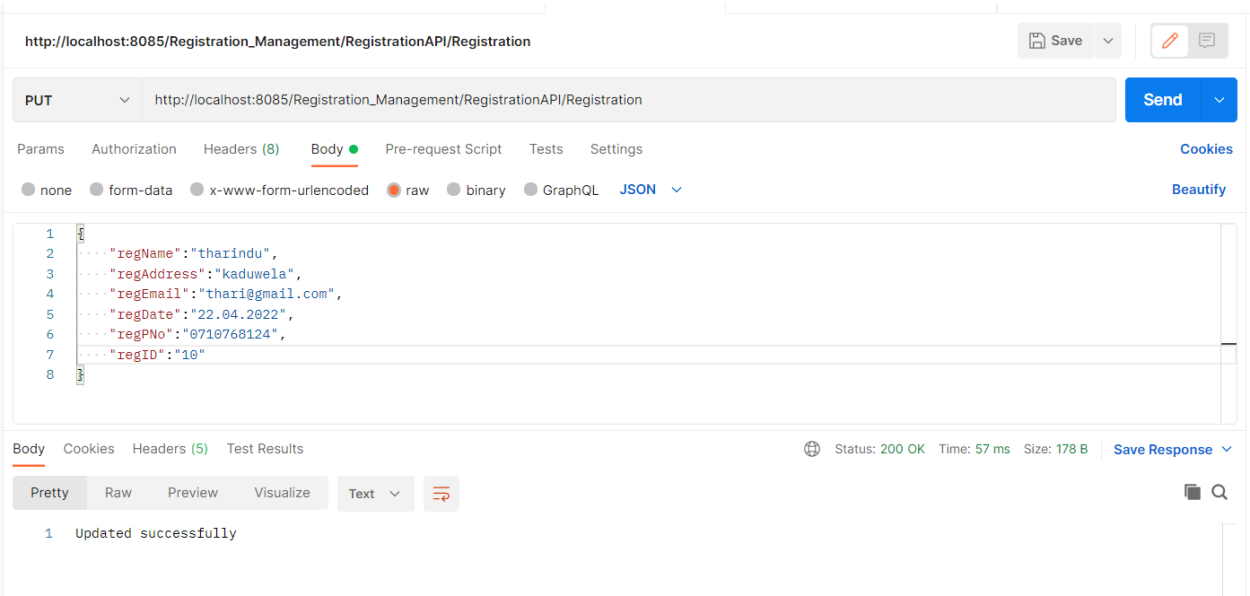


2) View User

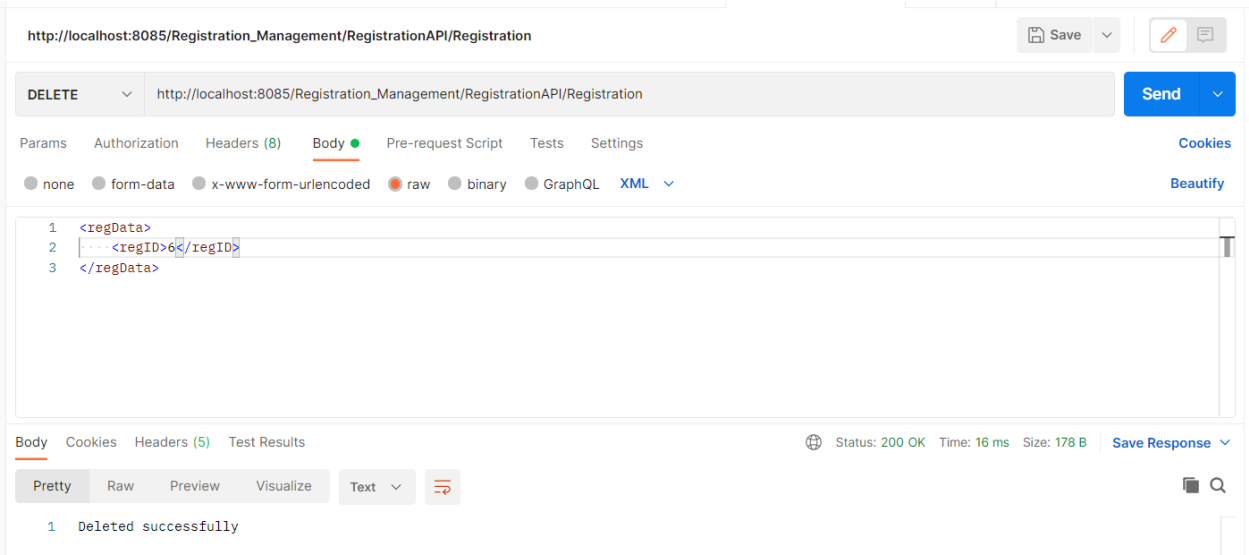


Customer ID	Customer Name	Address	Email	Date	Contact No
5	dews	kaduwela	dews@gmail.com	20.2.2022	071345678
6	sasi	malabe	sasi@gmail.com	22.04.2022	0710234567
7	mlaindu	galle	mali@gmail.com	20.01.2022	0774561283
10	tharindu	kaduwela	thari@gmail.com	22.04.2022	0710768124

3) Update User



4) Delete User



3. Assumptions

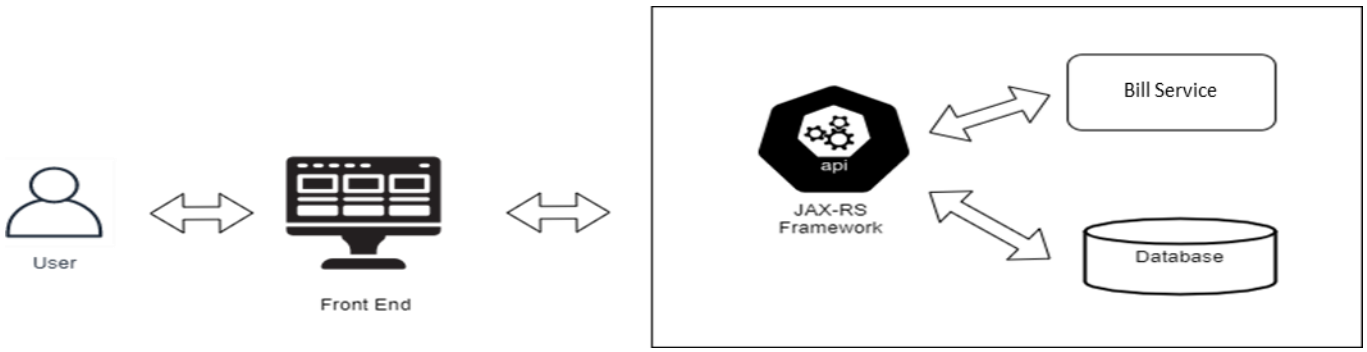
- There are mainly three types of users in the system consumer, manager, and admin
- Admin has the responsibility to manage all the users of the system
- Manager and Consumers are added by the administrator to maintain the transactions of the system.

Bill Service

1. **Service design**

   Bill handler is the one who mainly manages this service. This service is the one which oversees the bill creation by taking the consumption of the consumers.

   a) **API of the service**



I. Create Bill (POST)

Resource: Bills
Request: POST -http://localhost:8080/BillService/ElectroService/Bills
Media Type: Form Data - APPLICATION_FORM_URLENCODED
Data: billCode, electricityAccountNo, billMonth , units , paymentAmount
Response: String status message "Inserted successfully"
URL: http://localhost:8080/BillService/ElectroService/Bills

II. View Bill (GET)

Resource: Bills
Request: GET - http://localhost:8080/BillService/ElectroService/Bills
Media Type: TEXT_HTML
Data: billCode, electricityAccountNo, billMonth , units , paymentAmount
Response: HTML table with billCode, electricityAccountNo, billMonth , units , paymentAmount
URL : http://localhost:8080/BillService/ElectroService/Bills

III. Update Bill (PUT)

Resource: Bills
Request: PUT http://localhost:8080/BillService/ElectroService/Bills
Media Type: APPLICATION_JSON, TEXT_PLAIN
Data: billCode, electricityAccountNo, billMonth , units , paymentAmount
Response: String status message "Updated successfully"
URL: http://localhost:8080/BillService/ElectroService/Bills

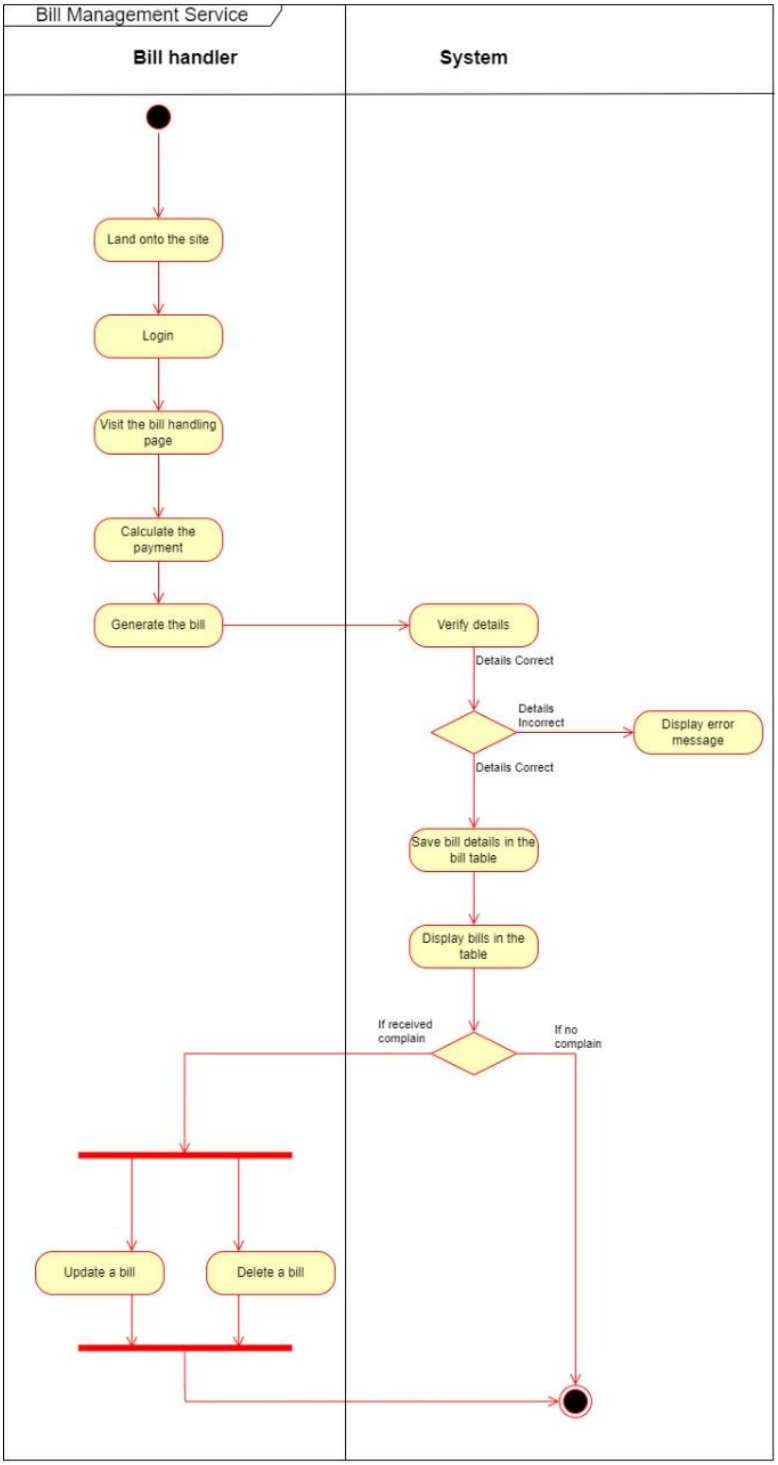
IV. Delete Bill (DELETE)

Resource: Bills
Request: DELETE http://localhost:8080/BillService/ElectroService/Bills
Media Type: APPLICATION_XML
Data: <BillData >
<billID> 3 <billID>
<BillData >
Response: String status message "Deleted successfully"
URL: http://localhost:8080/BillService/ElectroService/Bills

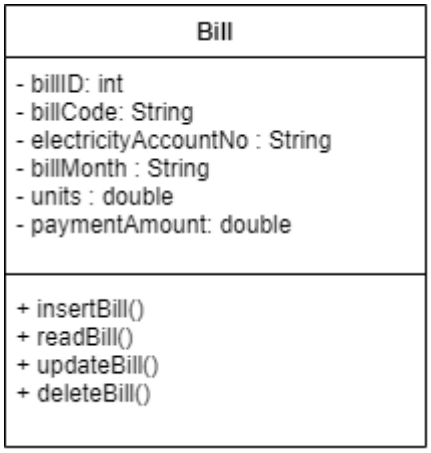
b) Internal logic

The bills are handled by the bill handler. The bill is generated by calculating the monthly payment amount. If a consumer complaint is received, an added bill can be updated or deleted accordingly.

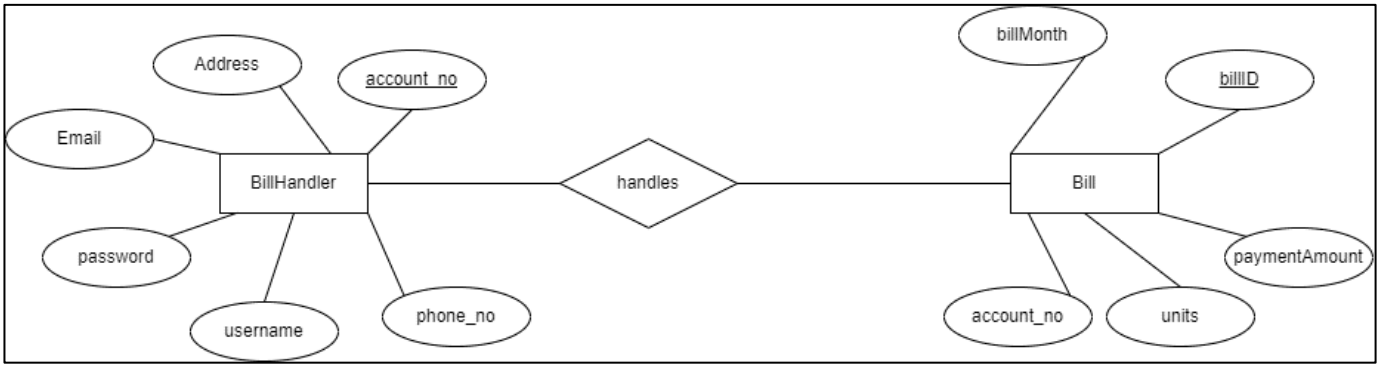
c) Activity Diagram



d) Class Diagram



e) Database for the service (ER)



2. Service development and testing

a) Tools used

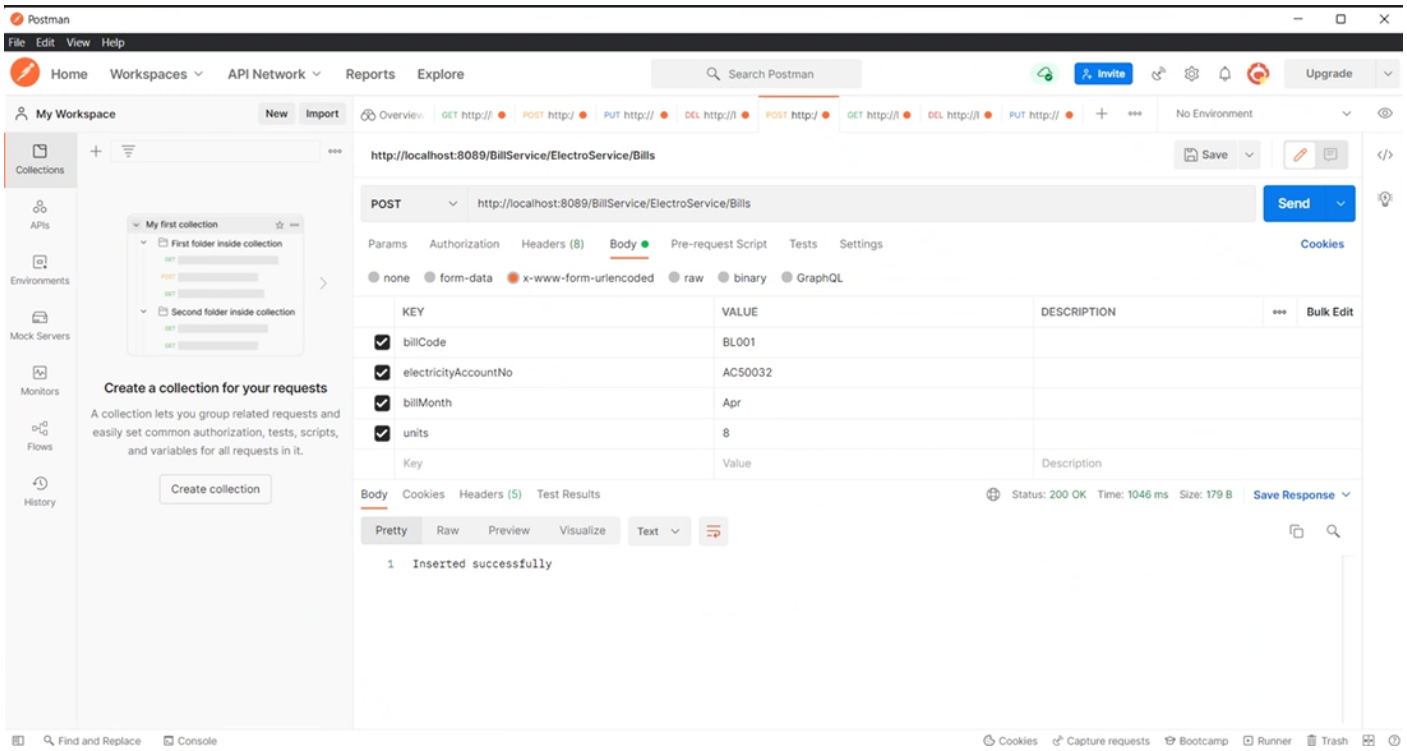
- Dependency Management Tool: Maven
- Testing Tool: Postman
- Version Control System: Git
- IDE: eclipse
- Programming Language: Jersey framework (JAX-RS)
- Programming Language: Java
- Database: phpMyAdmin (MySQL)
- Server: Apache Tomcat Server
- Code quality checking tool: Sonar Lint

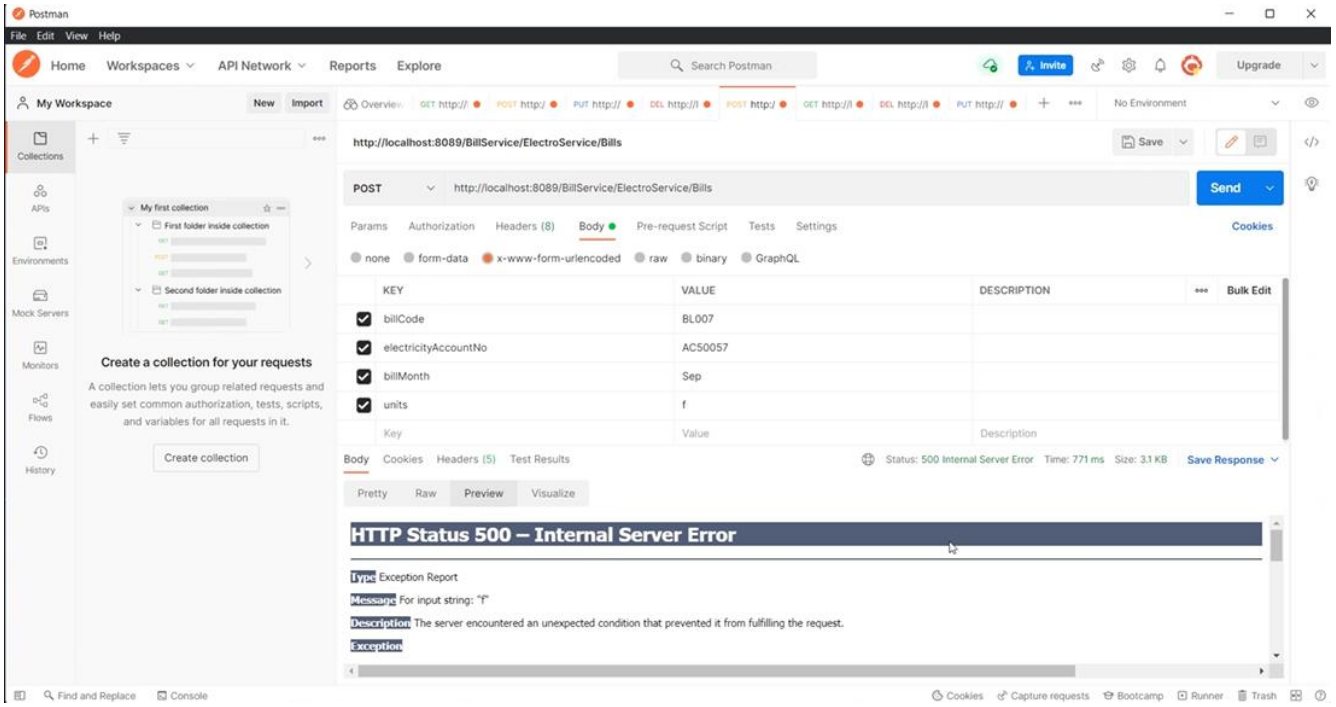
b) Testing methodology and results

Test ID	Description	Input	Expected Output	Actual Output	Result
1	Add Bill details	billCode = “BL003” electricityAccountNo = “AC50038” billMonth = “May” units = “15.0” paymentAmount = “750.0”	Display message “Inserted successfully”	Display message “Inserted successfully”	Pass
1	Add Bill details	billCode = “BL003” electricityAccountNo = “AC50038” billMonth = “May” units = “f” paymentAmount = “750.0”	Display message “Error while inserting”	Display message “Error while inserting”	Pass
2	Update Bill	billCode = “BL004” electricityAccountNo = “AC50042” billMonth = “June” units = “14.0” paymentAmount = “700.0”	Display message “Updated successfully”	Display message “Updated successfully”	Pass
3	View Bills		Display Bill details.”	Display Bill details.	Pass
4	Delete Bill	<BillData > <billID> 3 <billID> <BillData >	Display message “Deleted successfully”	Display message “Deleted successfully”	Pass

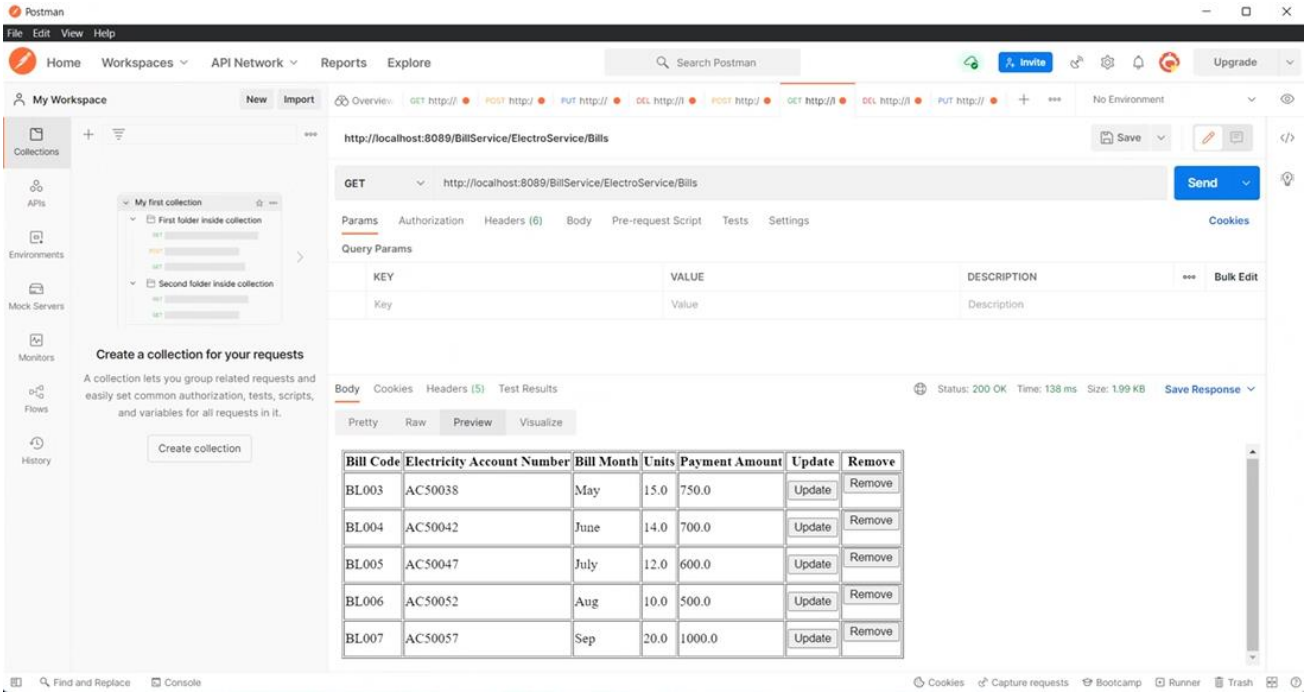
c) Postman Test Results

1) Add Bill

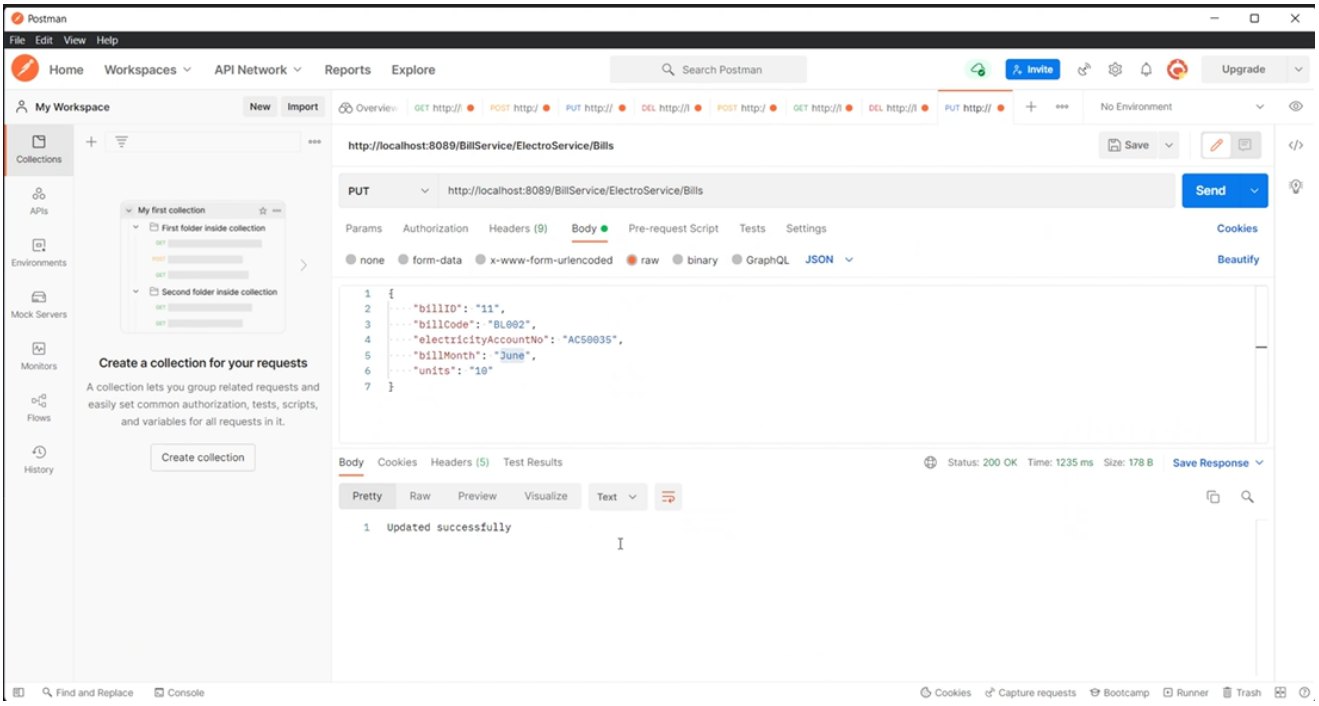




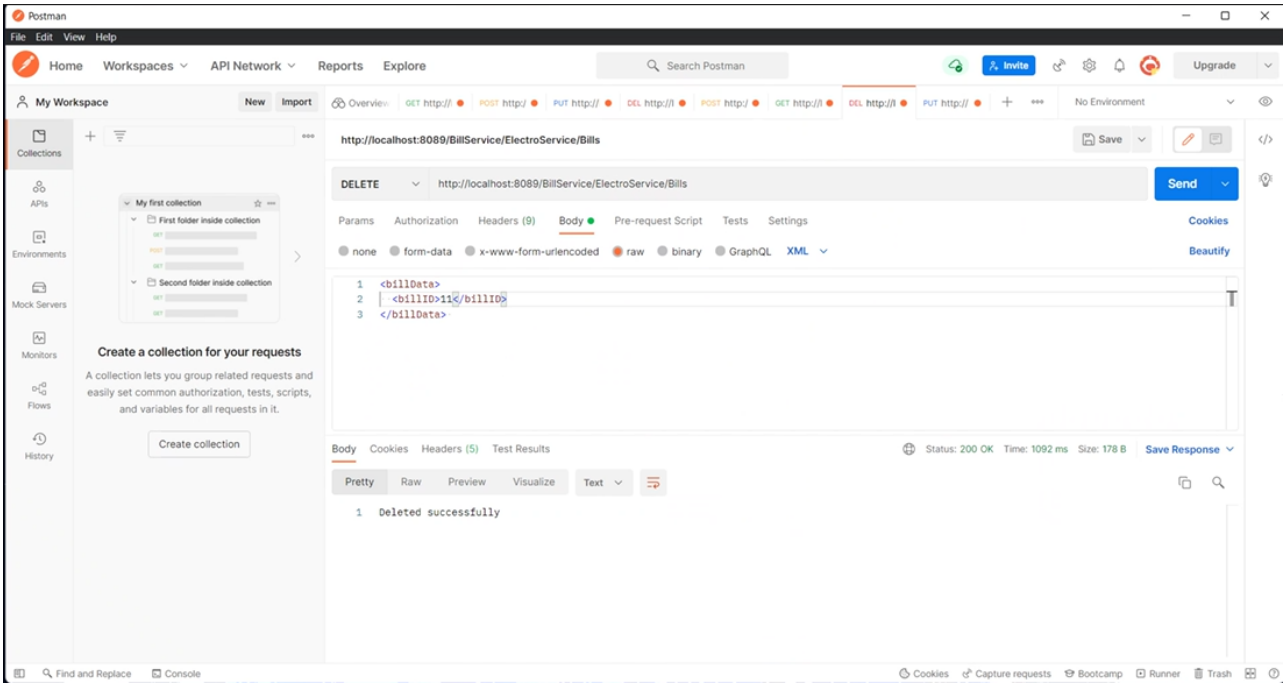
2) View Bill



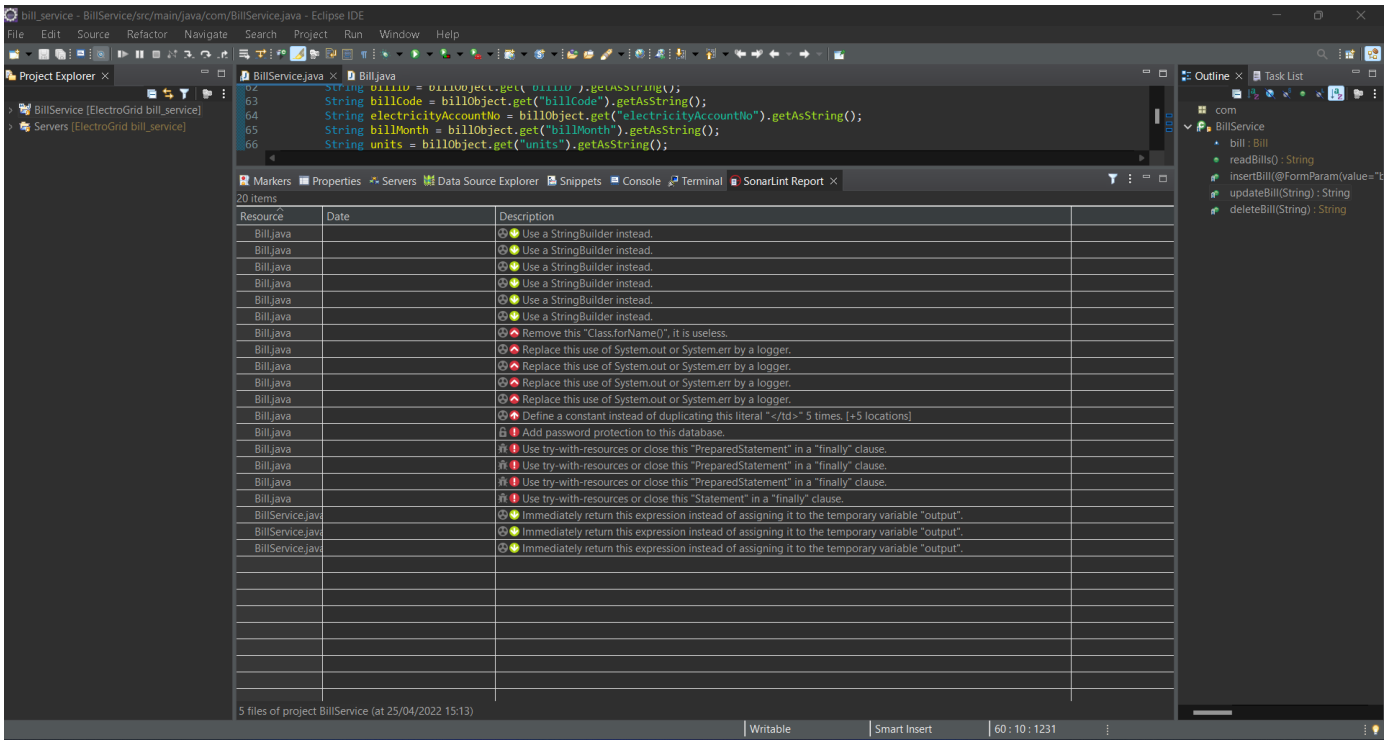
3) Update Bill



4) Delete Bill



d) Code quality check SonarLint Results



e) Assumptions

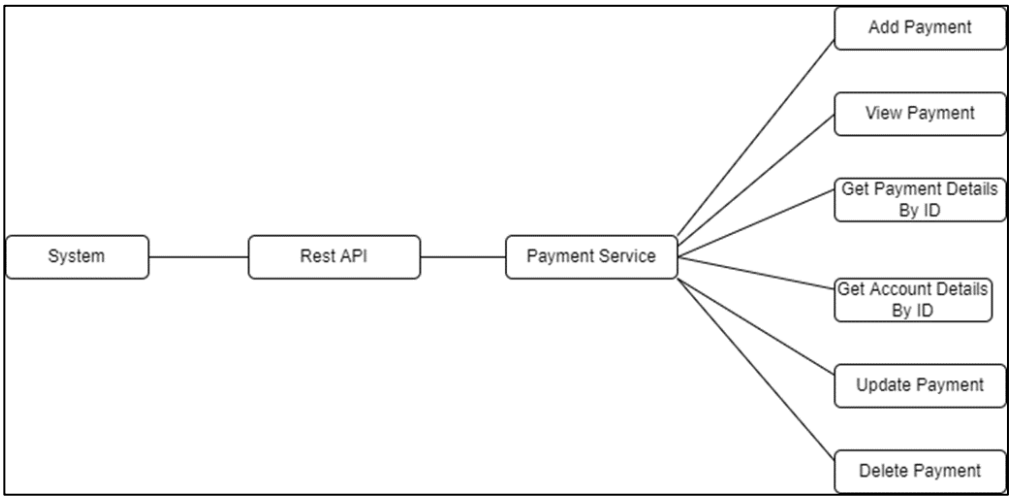
- When a consumer makes a complaint, bill details can be updated and deleted
- Bills are managed by the Bill handler which is a role inherited from the system administrator.

Payment Service

1. **Service design**

   Admin will approve customer's payment after confirming payment details. After payment get authorized system will automatically save each customer's payment details. Admin can add, update, or delete any transaction details.

   a) **API of the service**



I. **Create Payment (POST)**

Resource: Payment
Request: POST
Media: Form data – URL encoded
Data: PaymentID,Customer Name,Data,Acccount number
Response: Inserted successfully
URL: http://localhost:8080/PaymentManagement/PaymentAPI/Payment

II. **Update Payment (PUT)**

Resource: Payment
Request: PUT
Media: form data – application JASON
Data: PaymentID,Customer Name,Data,Acccount numbers
Response: Inserted successfully
URL: http://localhost:8080/PaymentManagement/PaymentAPI/Payment

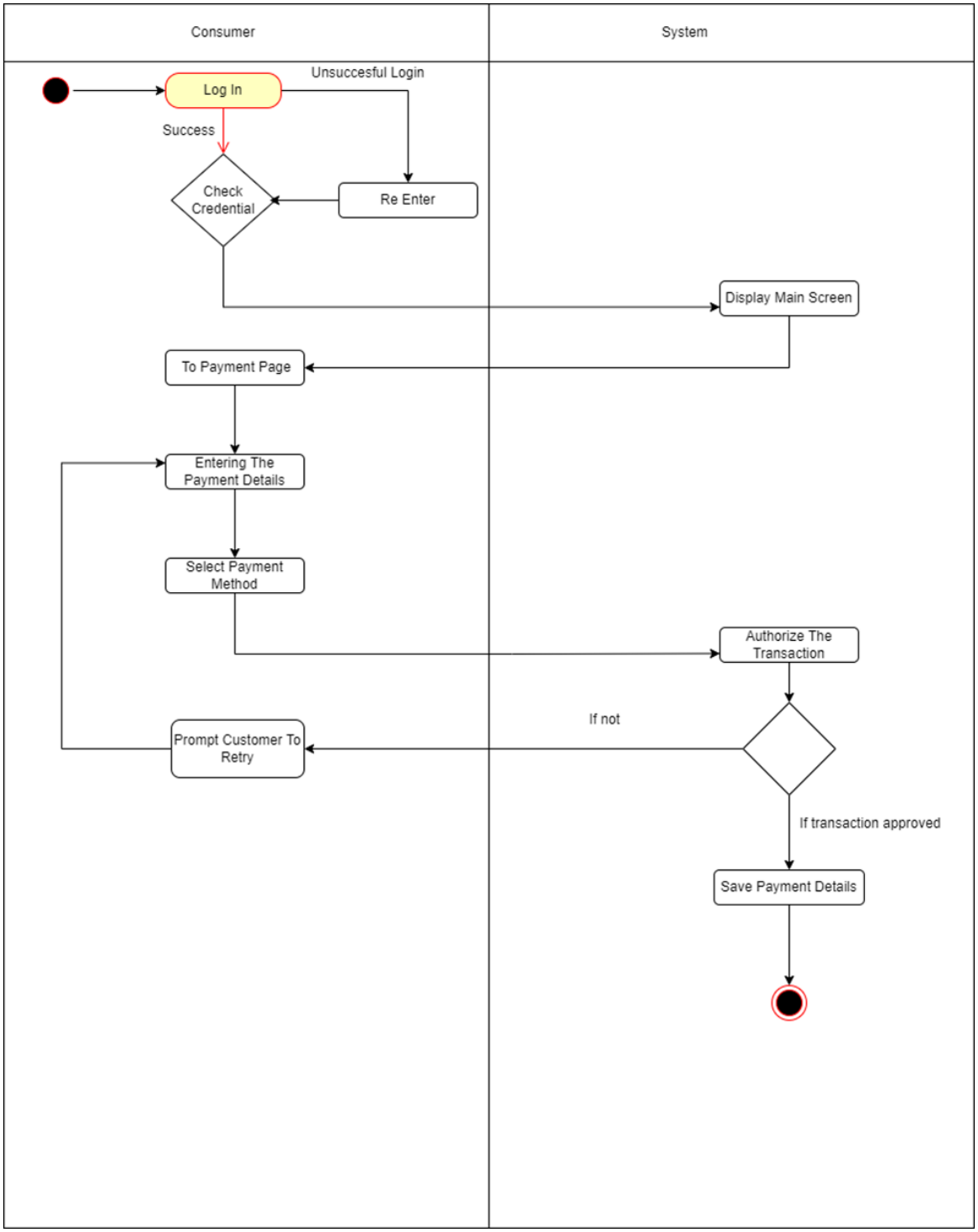
III. **View Payment (GET)**

Resource: Payment
Request: GET
Media: form data
Response: Updated successfully
URL: http://localhost:8080/PaymentManagement/PaymentAPI/Payment

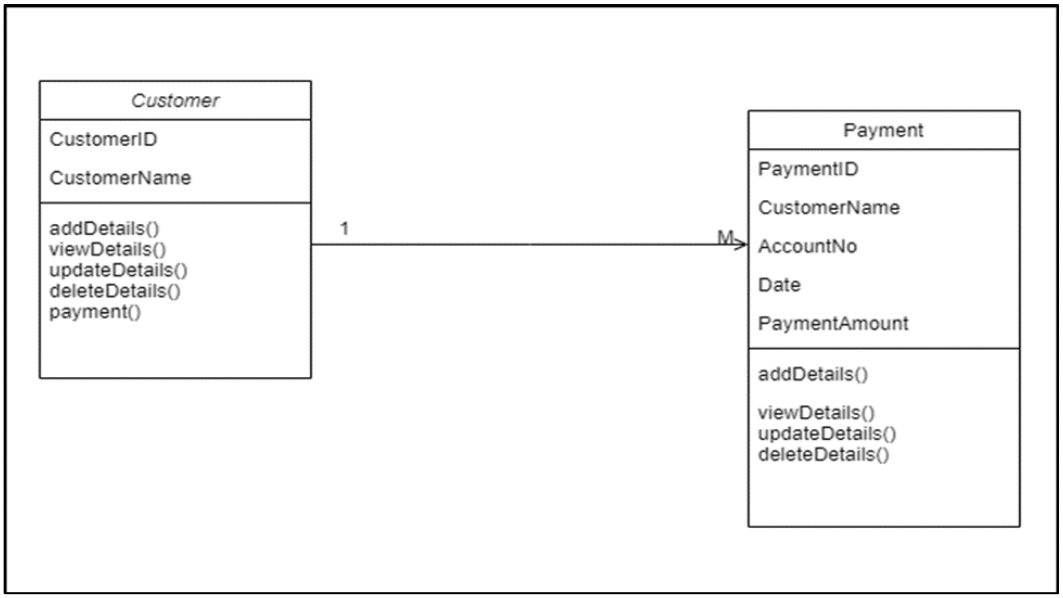
IV. Delete Payment (DELETE)

Resource: Payment
Request: DELETE
Media: application XML
Data: PaymentID
Response: Deleted successfully
URL: http://localhost:8080/PaymentManagement/PaymentAPI/Payment

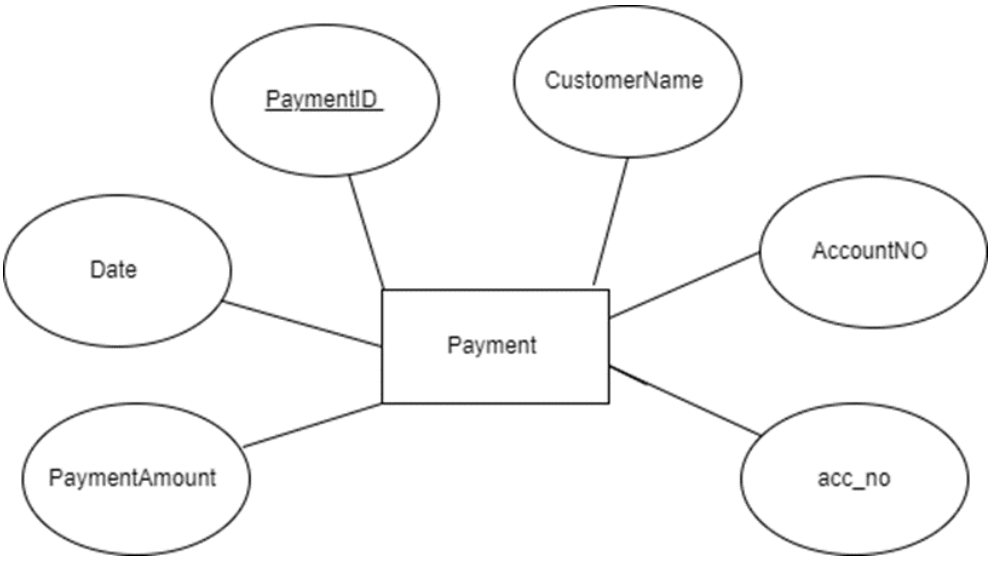
b) Internal logic (Activity Diagram)



c) Class Diagram



d) Database for the service (ER)



2. Service development and testing

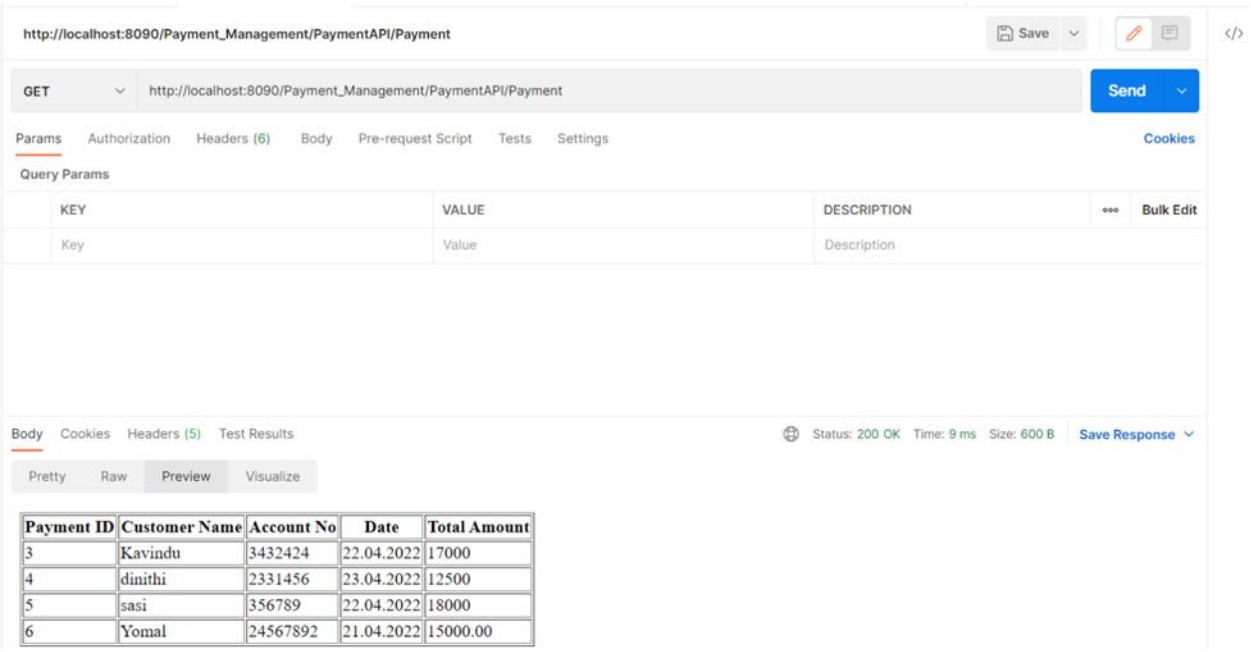
f) Tools used

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

g) Testing methodology and results

Test ID	Description	Input	Expected Output	Actual Output	Result
1	Create Payment	PaymentID CustomerName Payment Amount AccountNo date	Display Message "Payment Done successfully"	"Payment Done Successfully "Message displayed	Pass
2	View Payment	PaymentID	Display Payment details for the given ID	Relevant details are displayed	Pass
3	Update Payme nt	PaymentID CustomerName Payment Amount AccountNo date	Display Message "Payment Details Updated successfully"	"Payment Details Updated successfully" Message is displayed.	Pass
4	Delete Payment	PaymentID	Display message "Payment Details deleted successfully"	"Payment Details deleted successfully" message displayed.	pass

5) View Payment



Payment ID	Customer Name	Account No	Date	Total Amount
3	Kavindu	3432424	22.04.2022	17000
4	dinithi	2331456	23.04.2022	12500
5	sasi	356789	22.04.2022	18000
6	Yomal	24567892	21.04.2022	15000.00

6) Add Payment

http://localhost:8090/Payment_Management/PaymentAPI/Payment

Save

POST http://localhost:8090/Payment_Management/PaymentAPI/Payment Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

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

KEY	VALUE	DESCRIPTION
CustomerName	Yomal	
AccountNO	24567892	
Date	21.04.2022	
PaymentAmount	15000.00	
Key	Value	Description

Bulk Edit

Body Cookies Headers (5) Test Results Status: 200 OK Time: 17 ms Size: 179 B Save Response

Pretty Raw Preview Visualize Text

```
1  Inserted successfully
```

7) Update Payment

http://localhost:8090/Payment_Management/PaymentAPI/Payment

Save

PUT http://localhost:8090/Payment_Management/PaymentAPI/Payment Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

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

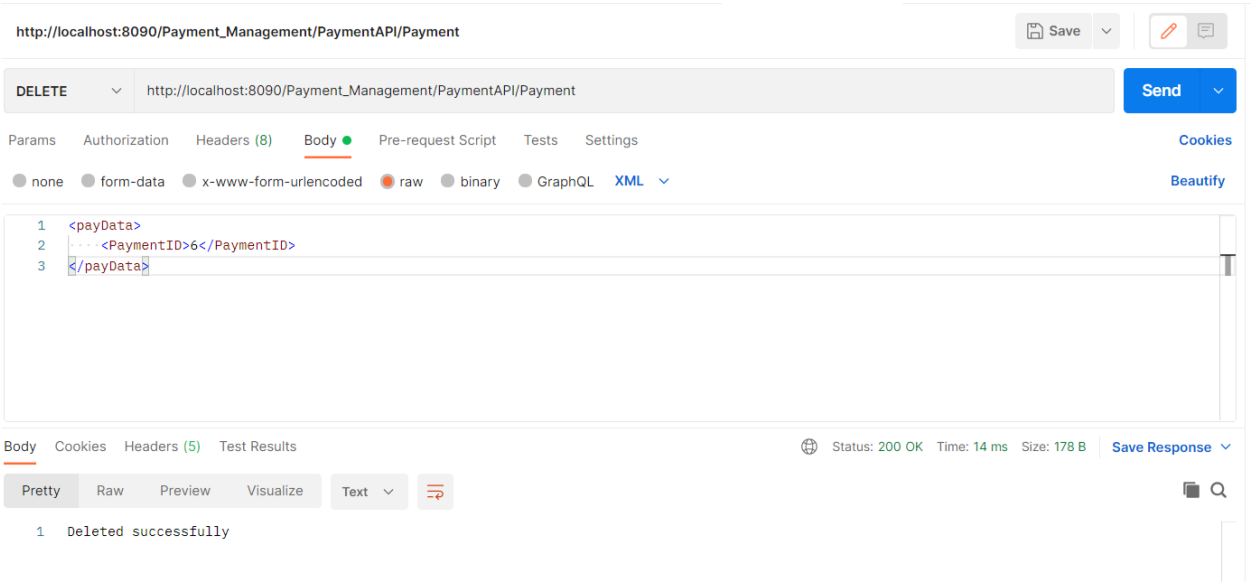
```
1  {
2      "CustomerName":"Dineth",
3      "AccountNO":"4556",
4      "Date":"22.04.2022",
5      "PaymentAmount":"10000",
6      "PaymentID":"6"
7  }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 15 ms Size: 178 B Save Response

Pretty Raw Preview Visualize Text

```
1  Updated successfully
```

8) Delete Payment



3. Assumptions

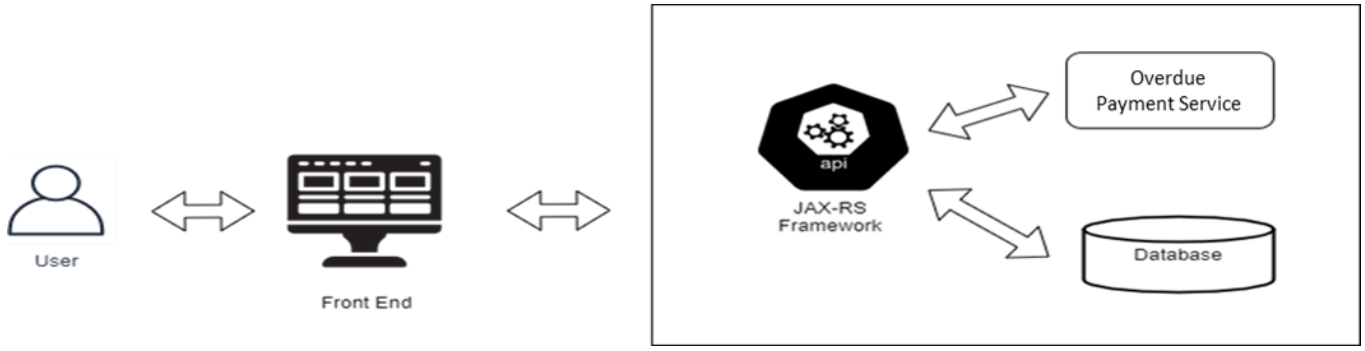
- Only admin can add new payment, update, and Delete payment entries.
- System will calculate the bill amount for relevant power usage.
- Only admin can View all payment details.
- User can delete Payment details within 7 days after they make the payment.

Overdue Payment Service

1. **Service design**

   Overdue payment service manages the overdue payments and service suspension. Overdue payment handler facilitates the management of due payments, service suspension due to overdue payments and service restorations.

   a) **API of the service**



I. Read Overdue Payments (GET)

Resource: ODPayments
Request: GET
http://localhost:8083/OverduePaymentService/ODPayService/ODPayments
Media Type: TEXT_HTML
Response: HTML table with all attributes in overdue payments(odspayments) table
URL: http://localhost:8083/OverduePaymentService/ODPayService/ODPayments

II. Create Overdue Payments (POST)

Resource: ODPayments
Request: POST
http://localhost:8083/OverduePaymentService/ODPayService/ODPayments
Media Type: Form Data - APPLICATION_FORM_URLENCODED
Data: ODCode, dueAmount, dueMonthsNo, dueMonths, accountNo, IsSuspend
Response: String message displayed as "Inserted Successfully" or "Error while inserting the Overdue Payment"
URL: http://localhost:8083/OverduePaymentService/ODPayService/ODPayments

III. Update Overdue Payments (PUT)

Resource: ODPayments
Request: PUT
http://localhost:8083/OverduePaymentService/ODPayService/ODPayments
Media Type: Form Data - APPLICATION_JSON
Data: ODPaymentID, ODCode, dueAmount, dueMonthsNo, dueMonths, accountNo, IsSuspend
Response: String message displayed as "Updated Successfully" or "Error while updating the Overdue Payment."
URL: http://localhost:8083/OverduePaymentService/ODPayService/ODPayments

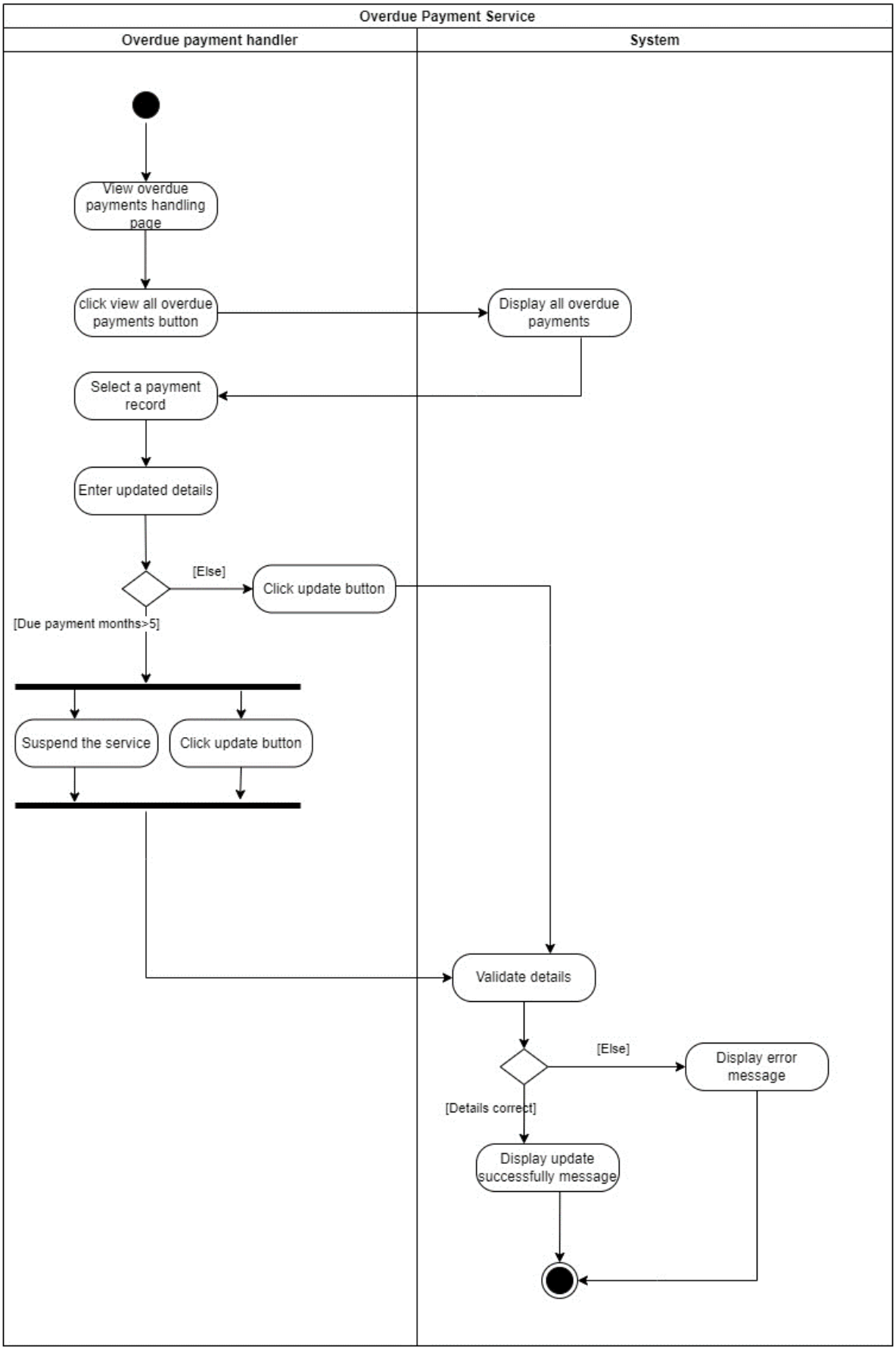
IV. Delete Overdue Payments (DELETE)

Resource: ODPayments
Request: DELETE
http://localhost:8083/OverduePaymentService/ODPayService/ODPayments
Media Type: APPLICATION_XML
Data: ODPaymentID
Response: String message displayed as "Deleted Successfully" or "Error while updating the Overdue Payment."
URL: http://localhost:8083/OverduePaymentService/ODPayService/ODPayments

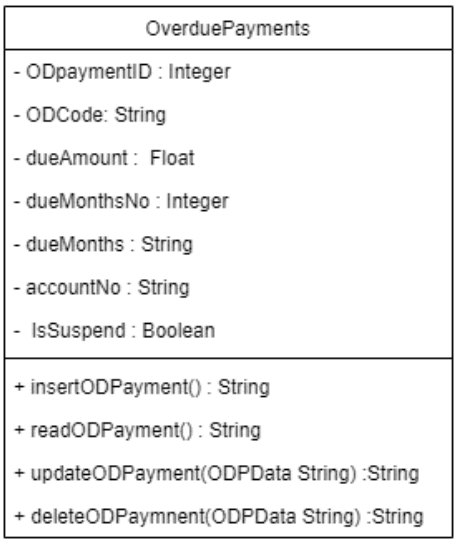
b) Internal logic

Overdue Payments are handled by the Overdue payment handler. In this service, overdue payments are created by considering the unpaid payments for three months. Added overdue payments can be updated if the relevant user is having more due payments. If the no of due months is six or exceeds six months, the overdue payment handler is responsible for the service suspension for the relevant consumer. If the consumer pays a full amount or a part of the total due amount overdue payments can be removed and if there's a service suspension it can be restored.

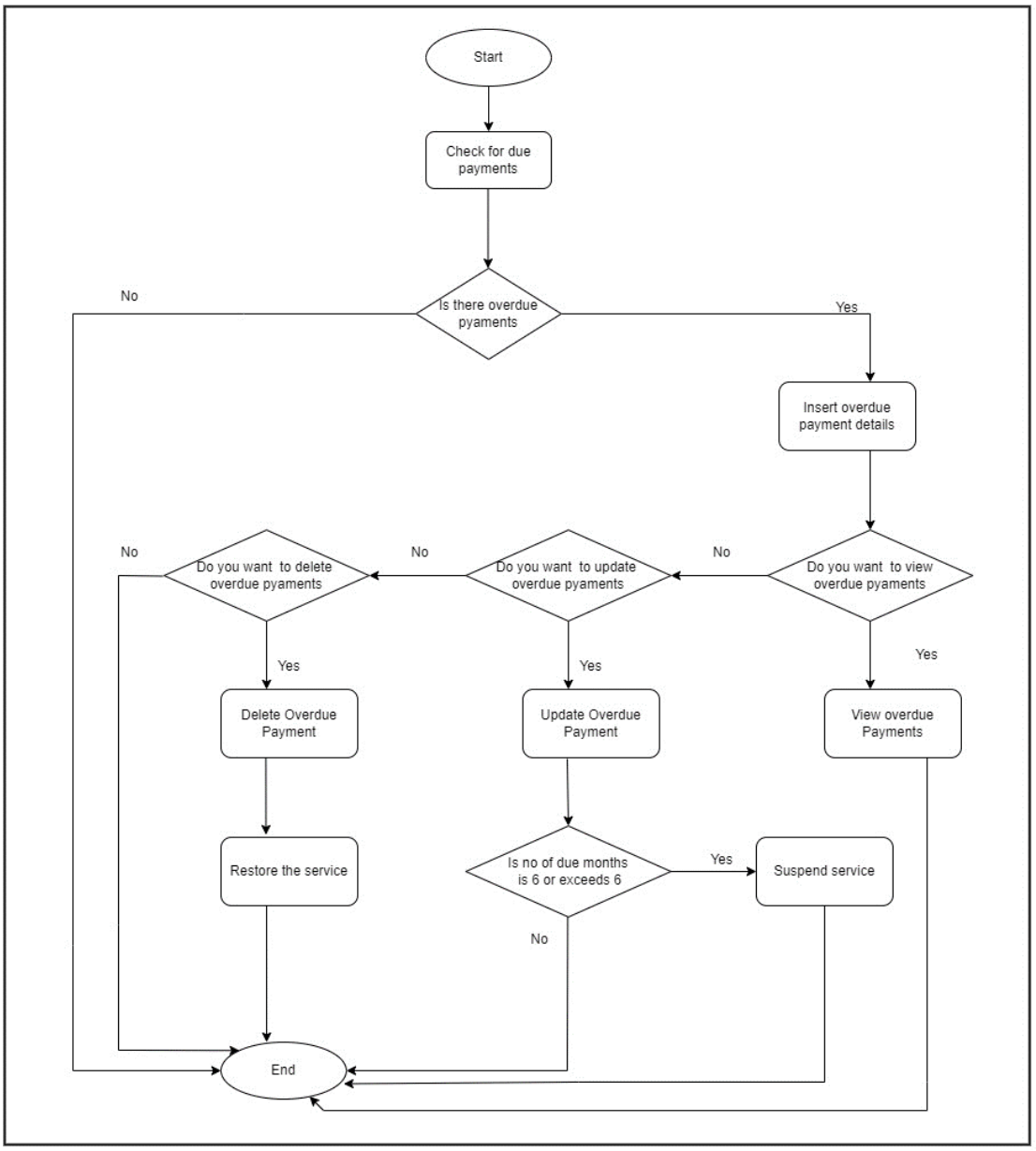
c) Activity Diagram



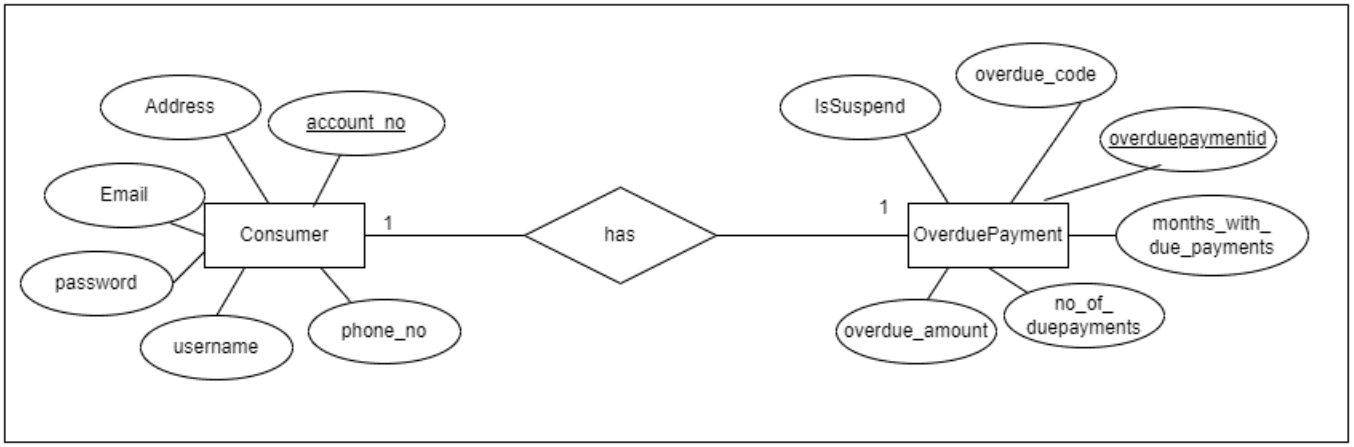
d) Class Diagram



e) Flow chart



f) Database for the service (ER)



2. Service development and testing

a) Tools used

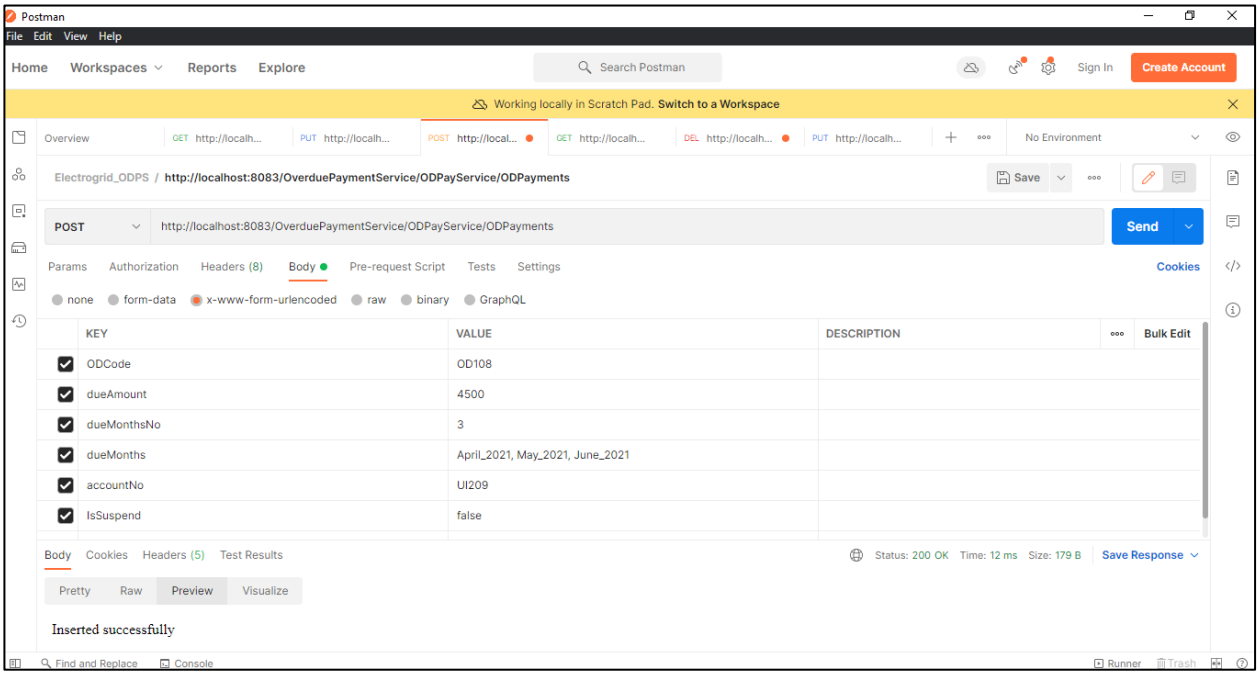
- Dependency Management Tool: Maven
- Testing Tool: Postman
- Version Control System: Git (To integrate the system)
- IDE: eclipse
- Programming Language: Jersey framework (JAX-RS) / Java
- Database: phpMyAdmin (MySQL)
- Server: Apache Tomcat Server
- Code quality checking tool: Sonar Lint

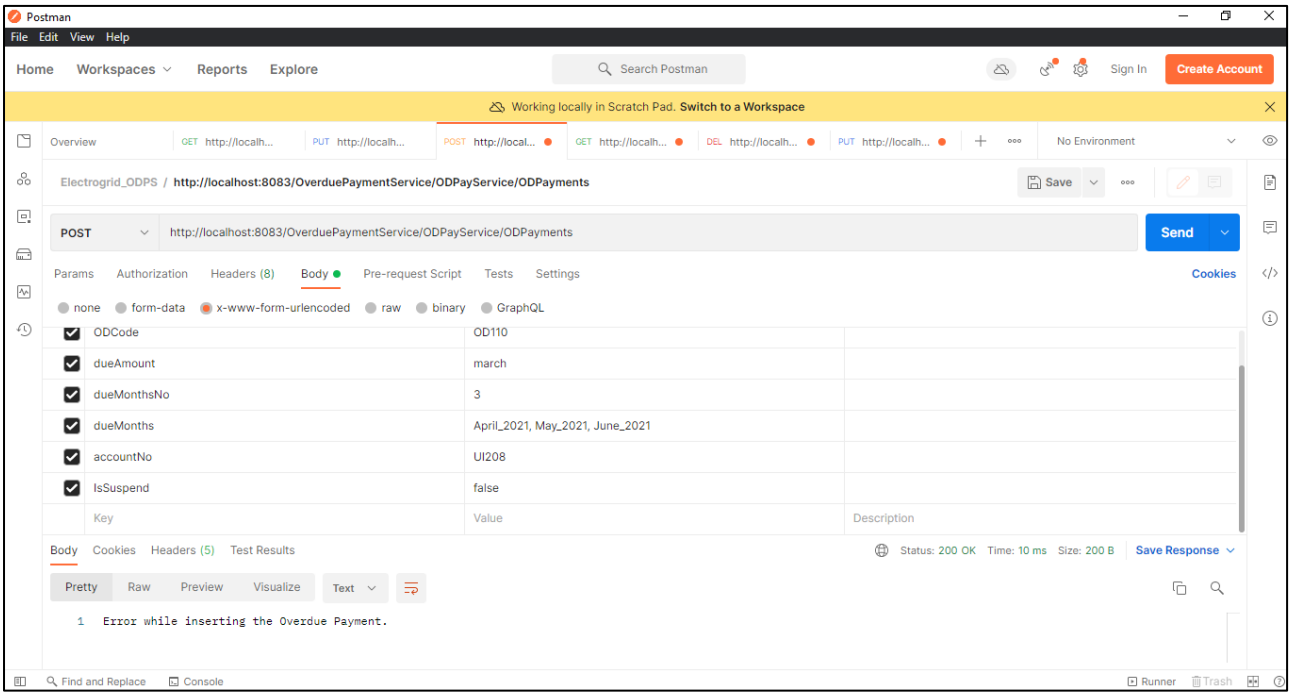
b) Testing methodology and results

Test ID	Description	Input	Expected output	Actual output	Result
1	Add Overdue payment	ODCode: “OD109” dueAmount:”4520” dueMonthsNo:”3” dueMonths:” April_2021, May_2021, June_2021” accountNo:”UI208” IsSuspend: “false”	Displays “Inserted Successfully” message	Displays “Inserted Successfully” message	Pass
2	Add Overdue payment	ODCode: “OD110” dueAmount:” march” dueMonthsNo:”3” dueMonths: “April_2021, May_2021, June_2021” accountNo:”UI208” IsSuspend:” false”	Displays “Error while inserting the overdue Payment” message	Displays “Error while inserting the overdue Payment” message	Pass
3	View Overdue payments		Displays HTML Table	Displays HTML Table	Pass
4	Update Overdue payments	ODPaymentID: "1" ODCode: "OD104" dueAmount:"8500.00" dueMonthsNo: “6", dueMonths:"March_2021, April_2021, May_2021, June_2021, July_2021, August_2021", accountNo: "U126", IsSuspend": "true"	Displays “Updated Successfully” message	Displays “Updated Successfully” message	Pass
5	Update Overdue payments	ODPaymentID: "1" ODCode: "OD104" dueAmount:"8500.00" dueMonthsNo: “march", dueMonths:"March_2021, April_2021, May_2021, June_2021, July_2021, August_2021", accountNo: "U126", IsSuspend": "true"	Displays “Error while updating the Overdue Payment” message	Displays “Error while updating the Overdue Payment” message	Pass
6	Delete Overdue payment	ODPaymentID: "8"	Displays “Deleted Successfully” message	Displays “Deleted Successfully” message	Pass

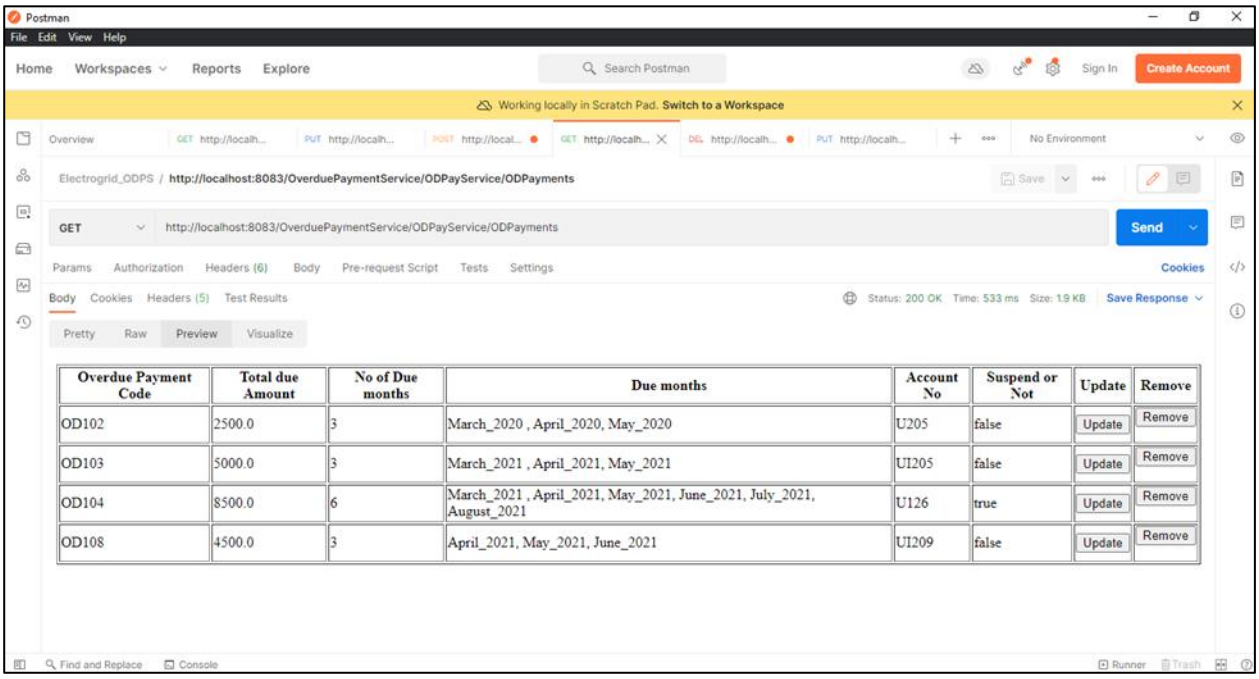
c) Postman Test Results

1) Add Overdue Payment



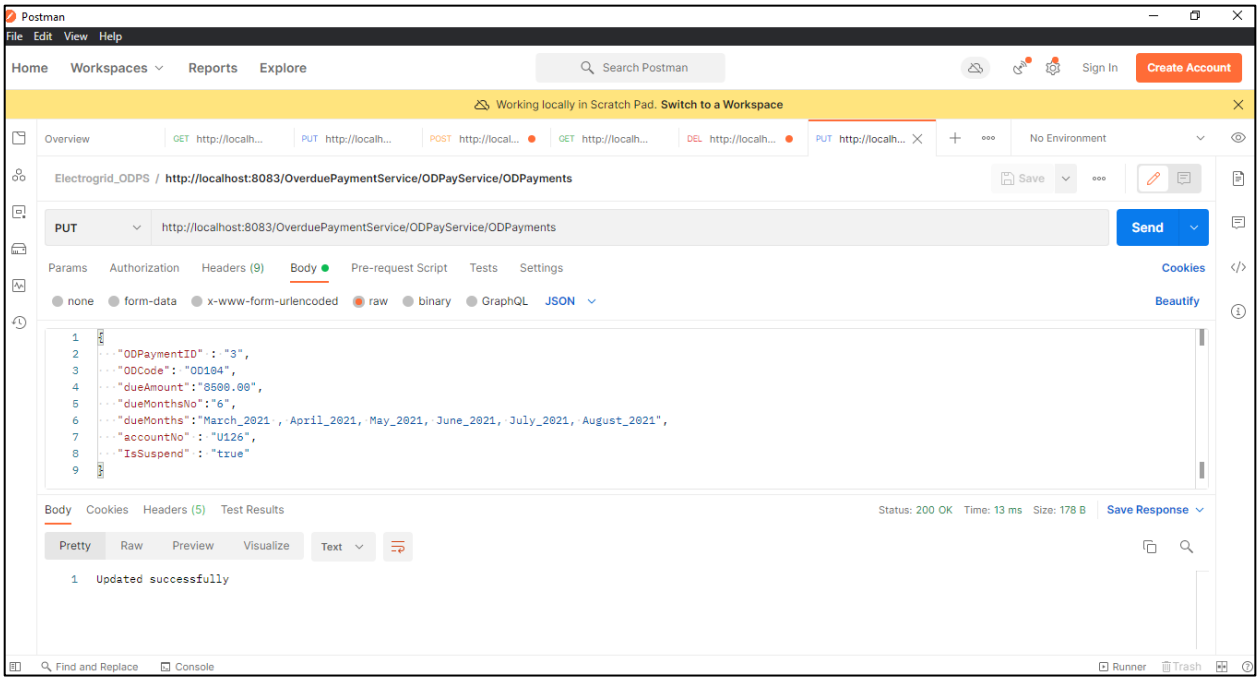


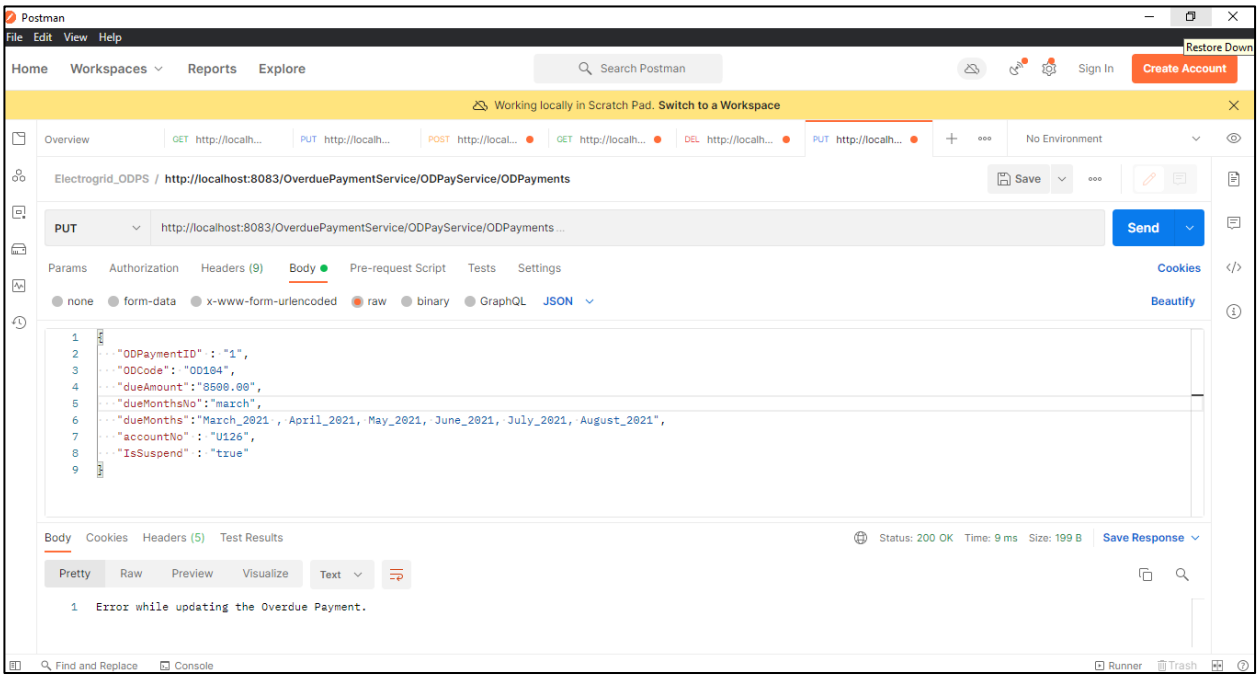
2) View Overdue Payments



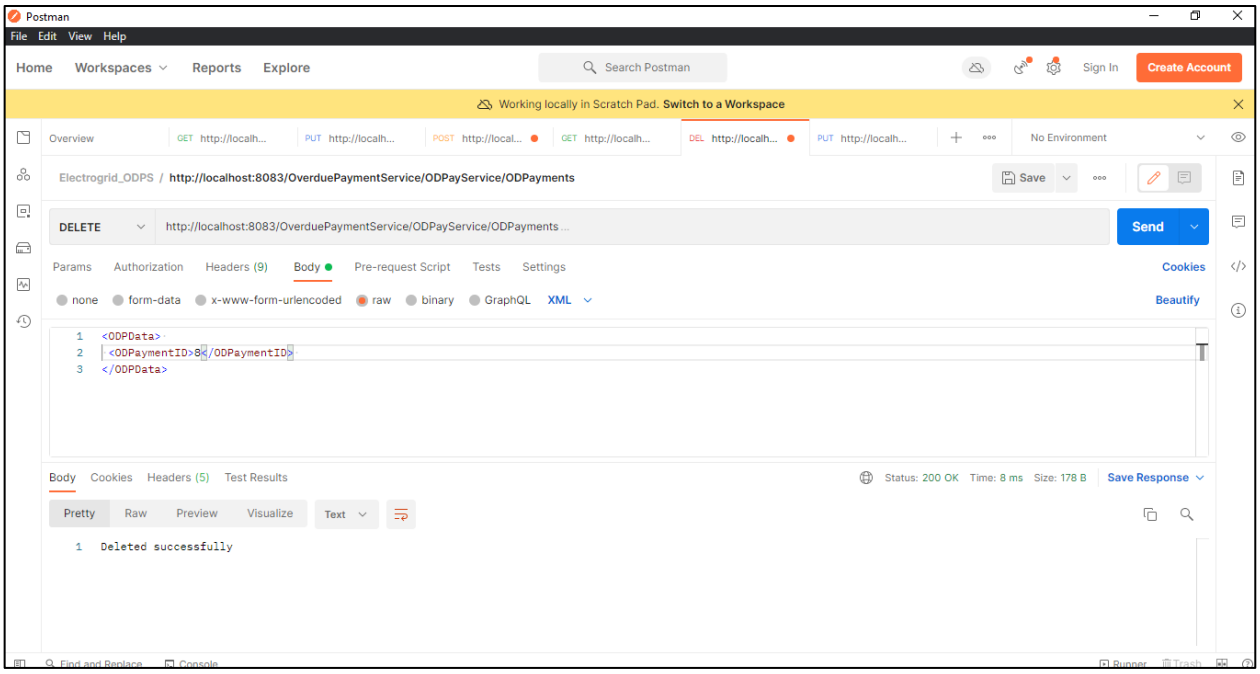
Overdue Payment Code	Total due Amount	No of Due months	Due months	Account No	Suspend or Not	Update	Remove
OD102	2500.0	3	March_2020 , April_2020, May_2020	U205	false	Update	Remove
OD103	5000.0	3	March_2021 , April_2021, May_2021	UI205	false	Update	Remove
OD104	8500.0	6	March_2021 , April_2021, May_2021, June_2021, July_2021, August_2021	U126	true	Update	Remove
OD108	4500.0	3	April_2021, May_2021, June_2021	UI209	false	Update	Remove

3) Update Overdue Payments

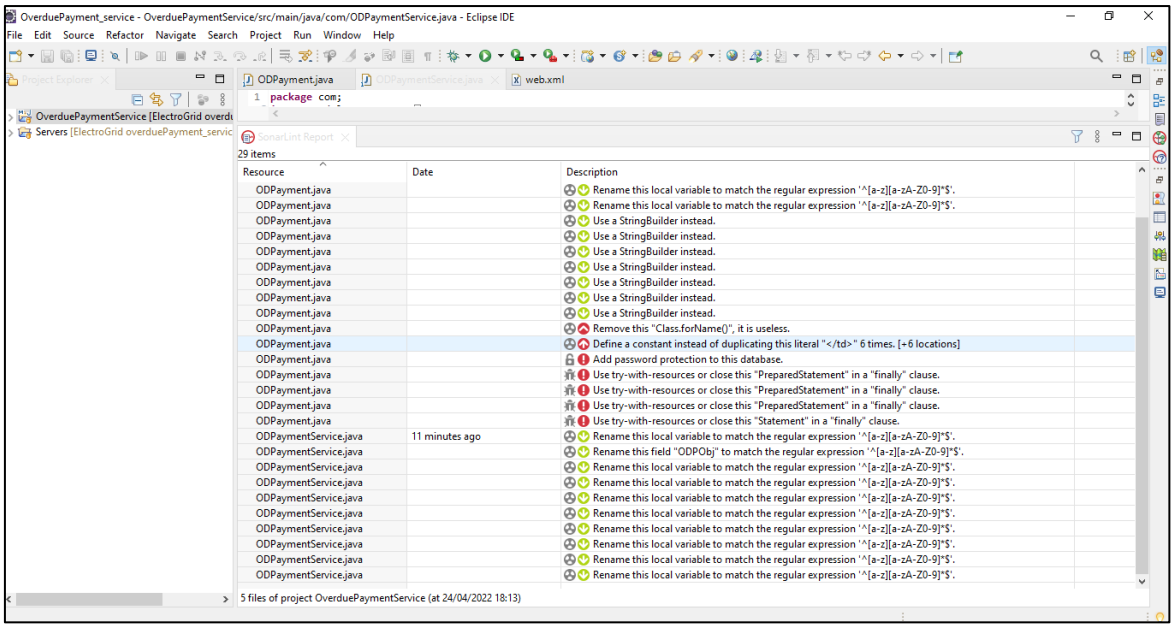




4) Delete Overdue Payments



d) Code Quality Check Sonar Lint Results



-

3. Assumptions

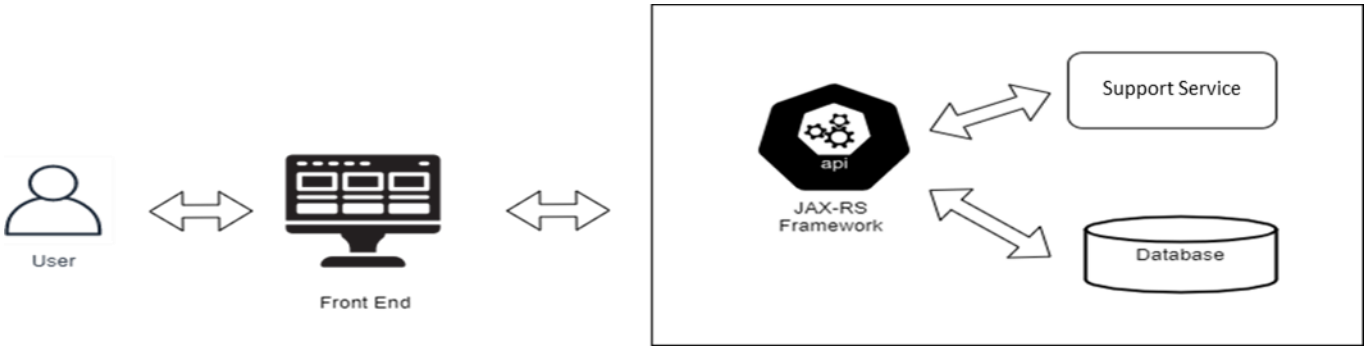
- Overdue payments are managed by the Overdue payment handler which is a role inherited from the system administrator.
- Overdue payments are added to the system by checking the due payments which exceeds three months.
- Added overdue payments can be updated every month for payments made.
- If the number of due payment months exceeds five months, then the service will be suspended.
- Overdue payments will be deleted when the consumer makes a payment.

Support Service

1. **Service design**

   Support service is meant to manage all the complaints of consumers. This service is operated by the admin. Consumers can complaint the issue by entering their details. Also, they will update and delete the added complaints as needed. All consumers and admins can read complaints in detail.

   a) **API of the service**



I. **Create Complaint (POST)**
Resource: SupportS
Request: POST - http://localhost:8080/SupportService/supportService/SupportS
Media Type: Form Data - APPLICATION_FORM_URLENCODED
Data: accountNum, complaintName , complaintAdd, complaintPhone, complaintEmail, complaintMessage
Response: String status message "Inserted successfully"
URL: http://localhost:8080/SupportService/supportService/SupportS

II. **View Complaint (GET)**
Resource: SupportS
Request: GET - http://localhost:8080/SupportService/supportService/SupportS
Media Type: Form Data - TEXT_HTML
Data: complaintID , accountNum, complaintName , complaintAdd, complaintPhone, complaintEmail, complaintMessage
Response: : HTML table with complaintID , accountNum, complaintName , complaintAdd, complaintPhone, complaintEmail, complaintMessage
URL: http://localhost:8080/SupportService/supportService/SupportS

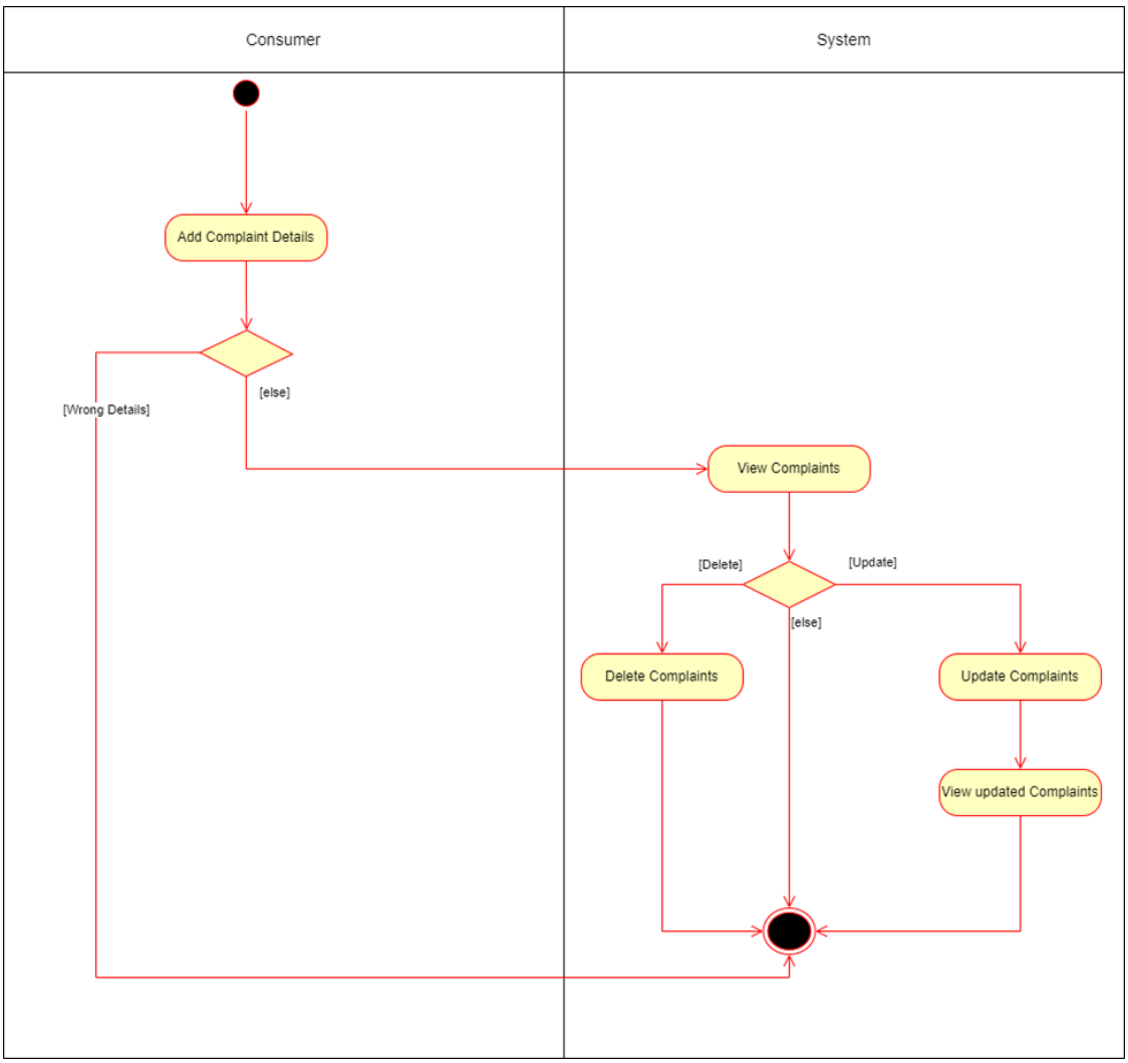
III. **Update Complaint (GET)**
Resource: SupportS
Request: http://localhost:8080/SupportService/supportService/SupportS
Media Type: APPLICATION_JSON, TEXT_PLAIN
Data: complaintID , accountNum, complaintName , complaintAdd, complaintPhone, complaintEmail, complaintMessage
Response: String status message "Updated successfully"
URL: http://localhost:8080/SupportService/supportService/SupportS

IV. Delete Complaint (DELETE)

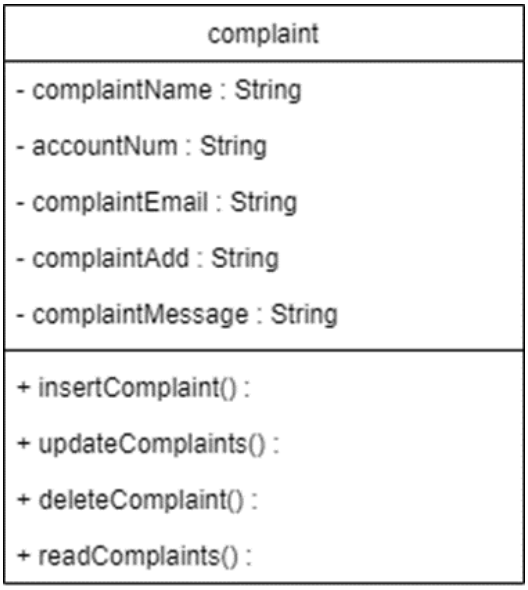
Resource: SupportS
Request: http://localhost:8080/SupportService/supportService/SupportS
Media Type: APPLICATION_XML
Data: <ComplaintDetails>
<complaintID> 3 </complaintID>
</ComplaintDetails>
Response: String status message "Deleted successfully"
URL: http://localhost:8080/SupportService/supportService/SupportS

b) Internal logic (Activity Diagram)

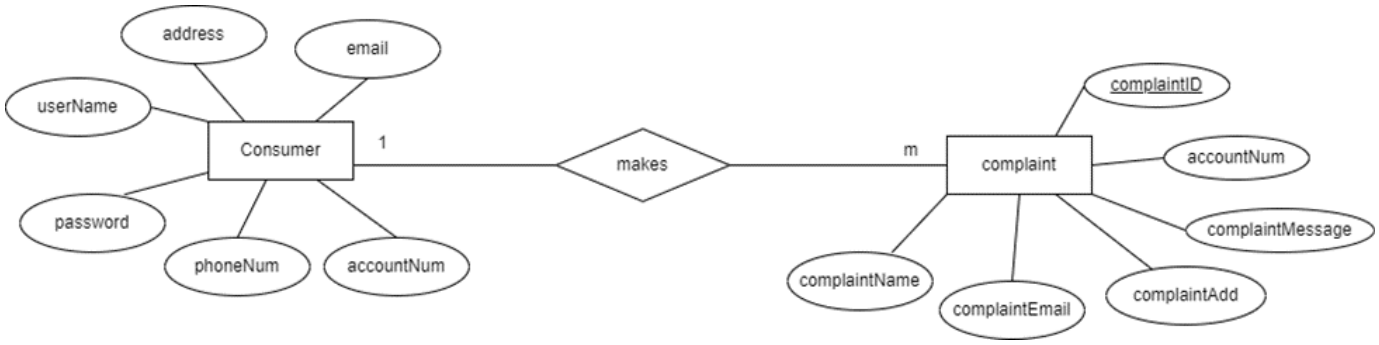
Consumer can directly add their complaints. Consumer can add their details into form and add complaints. After adding they can delete or edit their complaints.



c) Class Diagram



d) Database for the service (ER)



2. Service development and testing

a) Tools used

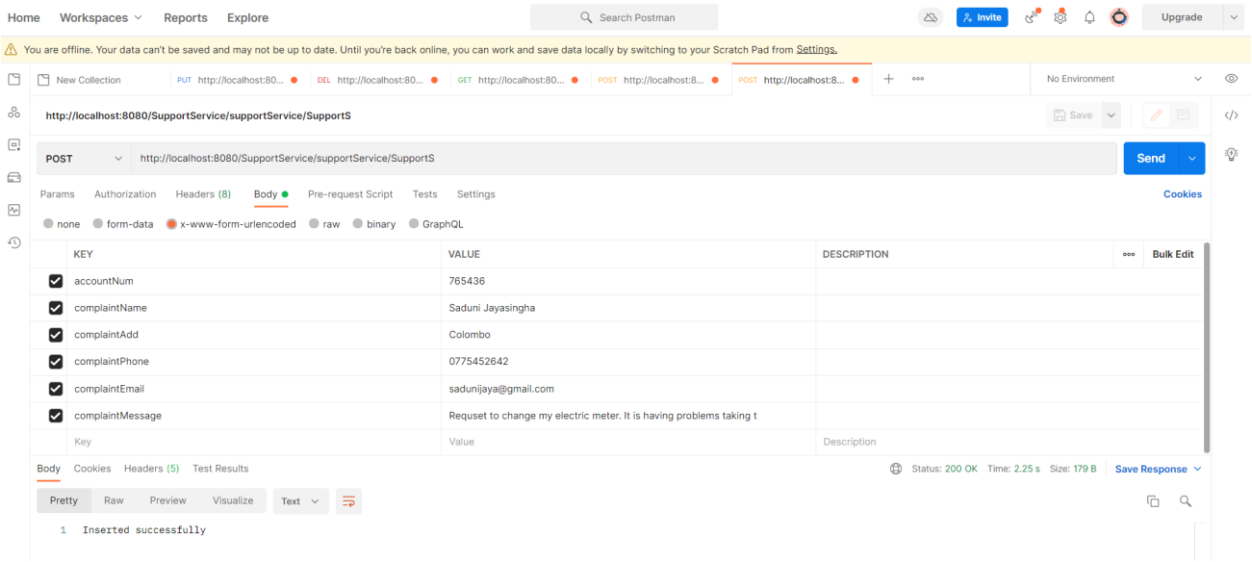
- Dependency Management Tool: Maven
- Testing Tool: Postman
- Version Control System: Git
- IDE: eclipse
- Programming Language: Jersey framework (JAX-RS)
- Programming Language: Java
- Database: phpMyAdmin (MySQL)
- Server: Apache Tomcat Server
- Code quality checking tool: Sonar Lint

b) Testing methodology and results

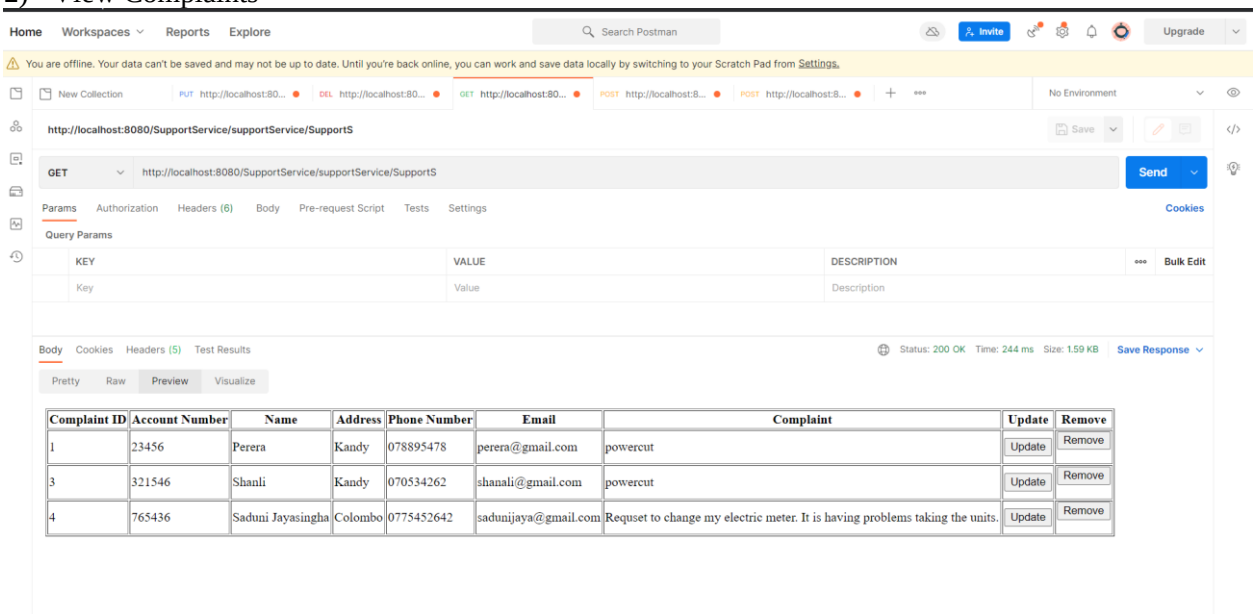
Test ID	Description	Input	Expected output	Actual output	Result
1	Add complaint	accountNum= “765432” complaintName = “Saduni Jayasingha” complaintAdd = “Colombo” complaintPhone = 0775452642 complaintEmail = “sadunijaya@gmail.com” complaintMessage = “Requset to change my electric meter. It is having problems taking the units.”	Display message “Inserted successfully”	Display message “Inserted successfully”	Pass
2	Update complaint	complaintID =”1” accountNum= “764932” complaintName = “Perera complaintAdd = “NuwaraEliya” complaintPhone = 0775452642 complaintEmail = “perera@gmail.com” complaintMessage = “Requset to change my electric meter. It is having problems taking the units.”	Display message “Updated successfully”	Display message “Updated successfully”	Pass
3	View complaints		Display complaint details.	Display complaint details.	Pass
4	Delete complaint	<ComplaintDetails> <complaintID> 3 </complaintID> </ComplaintDetails>	Display message “Deleted successfully”	Display message “Deleted successfully”	Pass

c) Postman Test Results

1) Add Complaints

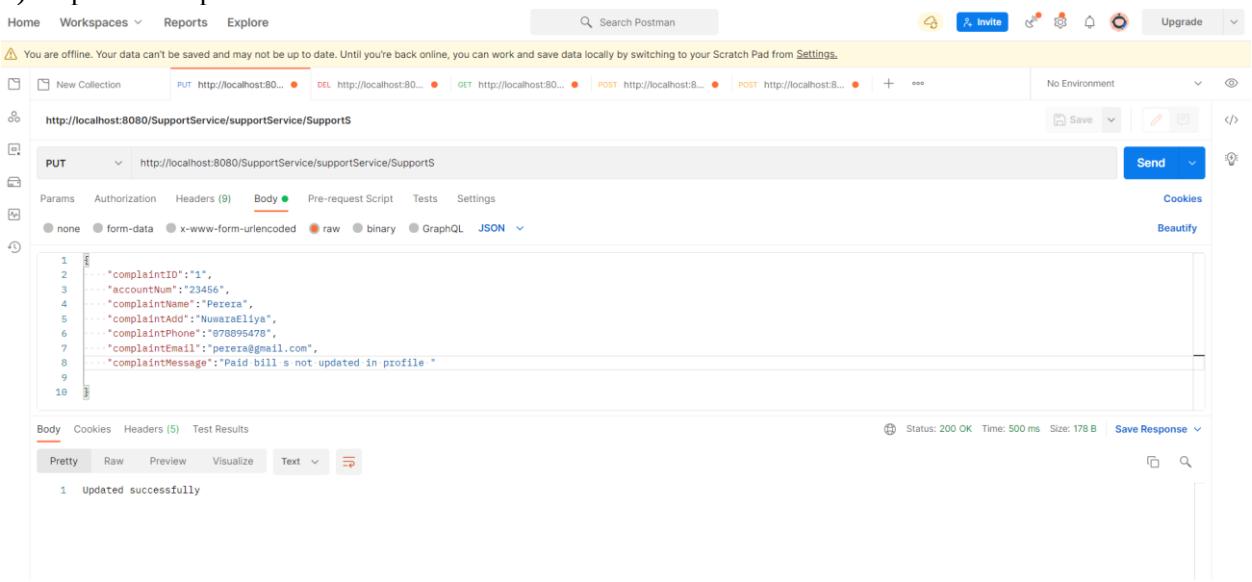


2) View Complaints

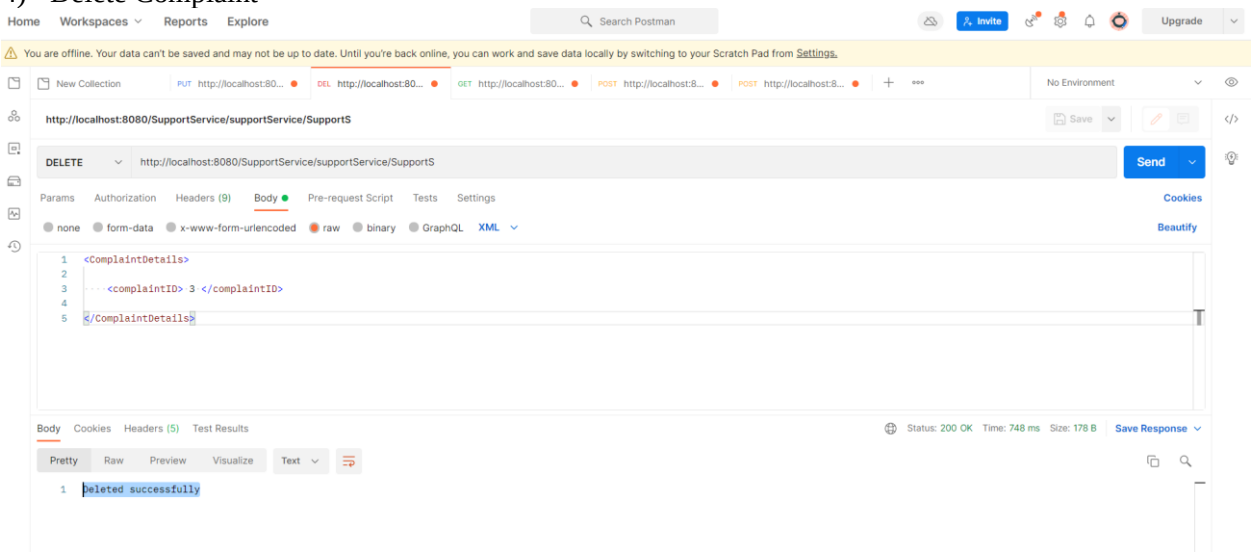


Complaint ID	Account Number	Name	Address	Phone Number	Email	Complaint	Update	Remove
1	23456	Perera	Kandy	078895478	perera@gmail.com	powercut	Update	Remove
3	321546	Shanli	Kandy	070534262	shanali@gmail.com	powercut	Update	Remove
4	765436	Saduni Jayasingha	Colombo	0775452642	sadunijaya@gmail.com	Requset to change my electric meter. It is having problems taking the units.	Update	Remove

3) Update Complaint



4) Delete Complaint



d) Code Quality Check Sonar Lint Results

SupportService.java | SupportS.java

```
13        Class.forName("com.mysql.jdbc.Driver");
14    //Provide the correct details: DBServer/DBName, username, password
```

Console | SonarLint Rule Description | SonarLint On-The-Fly

20 items

Date	Description	Resource
	Rename this local variable to match the regular expression '^[a-z][a-zA-Z0-9]*$'.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Use a StringBuilder instead.	SupportS.java
	Remove this "Class.forName()", it is useless.	SupportS.java
	Replace this use of System.out or System.err by a logger.	SupportS.java
	Replace this use of System.out or System.err by a logger.	SupportS.java
	Replace this use of System.out or System.err by a logger.	SupportS.java
	Replace this use of System.out or System.err by a logger.	SupportS.java
	Define a constant instead of duplicating this literal "</td>" 7 times. [+7 locations]	SupportS.java
	Add password protection to this database.	SupportS.java
	Use try-with-resources or close this "PreparedStatement" in a "finally" clause.	SupportS.java
	Use try-with-resources or close this "PreparedStatement" in a "finally" clause.	SupportS.java
	Use try-with-resources or close this "PreparedStatement" in a "finally" clause.	SupportS.java
	Use try-with-resources or close this "Statement" in a "finally" clause.	SupportS.java

3. Assumptions

- Admin has the responsibility to reply and update the complaints.
- Consumer can add any number of complaints using same name.

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
  - Version 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://howtodoinjava.com/
- SE Methodologies
  https://acodez.in/12-best-software-development-methodologies-pros-cons/
- Maven Documentation
  https://maven.apache.org/guides/
- Java Development
  http://www.java2novice.com/