

Programming Applications and Frameworks

3rd Year, 1st Semester

Group Assignment

HealthCare – Hospital Management System

Group ID :

Y3.S1.WD.IT.01.02

GID:28

**Table of contents**

1. Member Details
2. Clickable link to the public VCS repository
3. SE Methodology
4. Time Schedule (Gantt Chart)
5. Stakeholder Analysis (Onion Diagram)
6. Requirement Analysis
7. Use case Diagram
8. Overall Architecture
9. overall DB Design (ER Diagram)
10. Class Diagram
11. Individual Sections
12. . System Integration Methodology
13. References

Appendix

**Member details and workload distribution**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Member ID** | **Member Name** | **Workload** |
| 1 |  |  | **Power consumption unit management**   * View unit price * Insert unit price * Update unit price * Delete unit price |
| 2 |  |  | **User / Customer management**   * View customer details * Add customer details * Update customer details * Delete customer details |
| 3 |  |  | **Bill management**   * View bill details * Add bill details * Update bill details * Delete bill details |
| 4 |  |  | **Payment management**   * View payment details * Add payment details * Update payment details * Delete payment details |

**SE Methodology**

In order to accomplish efficient software development, this project employs the AGILE methodology. The AGILE technique is a process that encourages continuous development and testing throughout the project's software development lifecycle. Unlike the Waterfall methodology, development and testing are done simultaneously. First, requirements are gathered and analyzed. The users are always defined and a vision statement on the scope of challenges, opportunities, and values to be addressed is always documented in an agile software development process.

Individuals and interactions take precedence over procedures and tools, functional software takes precedence over thorough documentation, customer collaboration takes precedence over contract negotiation, and adapting to change takes precedence over following a plan.

The Agile typical iteration process flow can be visualized as follows:

Requirements - Define the requirements for the iteration based on the product backlog, sprint backlog, customer, and stakeholder feedback Development - Design and develop software based on defined requirements Testing - QA (Quality Assurance) testing, internal and external training, documentation development Delivery - Integrate and deliver the working iteration into production Feedback - Accept customer and stakeholder feedback and work it into the requirements of the next iteration

**Time Schedule (Gantt Chart)**

Chart

Description automatically generated

**Requirements**

* **Stakeholder analysis (Onion Diagram)**

**Diagram

Description automatically generated**

**Nonfunctional Requirements**

1. Performance

//The system should be able to handle and support usage of at least 2000 users, as most users may prefer to do

payments at the comfort of their home, especially during the pandemic and curfew-imposed times. //

Moreover, it is also crucial to provide uninterrupted accessibility throughout the system at all times.

Accessibility of the system should be available to at least 90% of all the available browsers.

Loading times for the system must not exceed 25 seconds as key tasks are being conducted throughout the system.

The database system of EG must also support 6 million records as calculating accrued previous payments are key

in the billing process, moreover as new users utilizing the system is expected to increase each year.

2. Security

Sensitive data such as bank account details, addresses will be encrypted and stored in the secure system database.

Only the system administrator is able to issue bills and is able to add new users into the system.

3.Safety.

The database of the system will have a backup of all records.

Availability

The system accessibility must be available 24\*7 to the users irrespective of the task carried out.

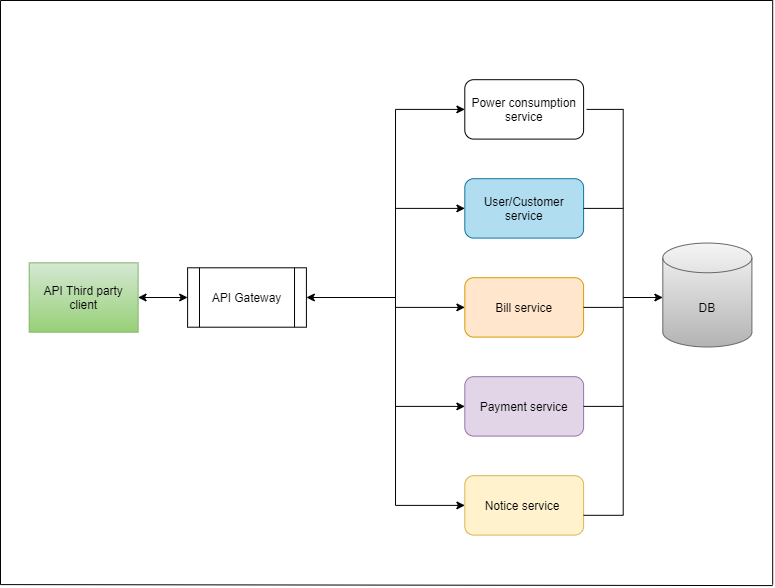
Technical Requirements

The system will be able to be accessed by a computer of minimum 4GB RAM and throughout operating systems such

as Windows, MacOS or Linux. Furthermore, web browsers such as Google Chrome, Mozilla Firefox, Windows Edge

and Samsung Internet will be much more efficient than less known browsers.

**System’s overall design**

**** • Overall Architecture

The REST architectural style is followed by these RESTful web services. It provides a consistent and predetermined set of rules for requesting systems to access and change online resources.

This system is made up of a client who requests resources and a server that provides those resources and answers to those requests. Gateways can be used to call various back-end services and aggregate the results, in addition to accepting direct queries. An API, like practically all software, must cater to the demands of the people who use it. Because it communicates with the end user, an API differs from a GUI or other user interface. This API simply exposes database functionality such as Create, Read, Update, and Delete (CRUD) activities by those services whose valuable APIs go beyond that and represent the functionalities required by the 3rd party user.

Individual Section

API Description of the Power consumption service

* POST
* Resource : PowerConsumption
* Request : POST PowerConsumptionService/PowerConsumption
* Media Type : Form Data – APPLICATION\_FORM\_URLENCODED
* Data : powId, unitDescription, unitPrice
* Response : Display a message – “Inserted successfully” / “Error while inserting data”
* GET
* Resource : PowerConsumption
* Request : GET PowerConsumptionService/PowerConsumption
* Media Type: TEXT\_HTML
* Data : {}
* Response: Read Power Consumption (units-price per units) details
* PUT
* Resource : PowerConsumption
* Request: PUT PowerConsumptionService/PowerConsumption
* Media Type: Application JSON
* Data: powId, unitDescription, unitPrice
* Response: Display a message – “Updated successfully” / “Error while updating data”
* DELETE
* Resource: PowerConsumption
* Request: DELETE PowerConsumptionService/PowerConsumption
* Media Type: Application XML
* Data: powId
* Response: Display a message – “Deleted successfully” / “Error while deleting data”

Development Tools used

* Dependency management tool used - Maven
* Testing tools used – Postman
* Checking code quality tool used - Eclipse check style plug-in

API Description of the User/Customer service

* POST
* Resource : Customer
* Request : POST CustomerService/Customer
* Media Type : Form Data – APPLICATION\_FORM\_URLENCODED
* Data : accountNumber, firstName, lastName, emailId, contactNumber
* Response : Display a message – “Inserted successfully” / “Error while inserting data”
* GET
* Resource : Customer
* Request : GET CustomerService/Customer
* Media Type: TEXT\_HTML
* Data: {}
* Response: Read Customer(user) details
* PUT
* Resource : Customer
* Request: PUT CustomerService/Customer
* Media Type: Application JSON
* Data: accountNumber, firstName, lastName, emailId, contactNumber
* Response: Display a message – “Updated successfully” / “Error while updating data”
* DELETE
* Resource : Customer
* Request: DELETE CustomerService/Customer
* Media Type: Application XML
* Data: accountNumber
* Response: Display a message – “Deleted successfully” / “Error while deleting data”

Development Tools used

* Dependency management tool used - Maven
* Testing tools used – Postman
* Checking code quality tool used - Eclipse check style plug-in

API Description of the Bill service

* POST
* Resource : Bill
* Request : POST /BillService/Bill
* Media Type : Form Data – APPLICATION\_FORM\_URLENCODED
* Data : billId, year, month, totalUnits, powId, dueAmount, billAmount, accountNumber
* Response : Display a message – “Inserted successfully” / “Error while inserting data”
* GET
* Resource : Bill
* Request : GET /BillService/Bill
* Media Type: TEXT\_HTML
* Data: {}
* Response: Read Bill details of the Customers
* PUT
* Resource : Bill
* Request: PUT /BillService/Bill
* Media Type: Application JSON
* Data: billId, year, month, totalUnits, powId, dueAmount, billAmount, accountNumber
* Response: Display a message – “Updated successfully” / “Error while updating data”
* DELETE
* Resource: Bill
* Request: DELETE /BillService/Bill
* Media Type: Application XML
* Data: billId
* Response: Response: Display a message – “Deleted successfully” / “Error while deleting data”

Development Tools used

* Dependency management tool used - Maven
* Testing tools used – Postman
* Checking code quality tool used - Eclipse check style plug-in

API Description of the Payment service

* POST
* Resource : Payment
* Request : POST PaymentService/Payment
* Media Type : Form Data – APPLICATION\_FORM\_URLENCODED
* Data : paymentId, billId, cardHolderName, cardType, bank, payAmount, status
* Response : Display a message – “Inserted successfully” / “Error while inserting data”
* GET
* Resource : Payment
* Request : GET PaymentService/Payment
* Media Type: TEXT\_HTML
* Data; {}
* Response: Read Payment Details of the Customer
* PUT
* Resource : Payment
* Request: PUT PaymentService/Payment
* Media Type: Application JSON
* Data: paymentId, billId, cardHolderName, cardType, bank, payAmount, status
* Response: Display a message – “Updated successfully” / “Error while updating data”
* DELETE
* Resource: Payment
* Request: DELETE PaymentService/Payment
* Media Type: Application XML
* Data: paymentId
* Response: Display a message – “Deleted successfully” / “Error while deleting data”

Development Tools used

* Dependency management tool used - Maven
* Testing tools used – Postman
* Checking code quality tool used - Eclipse check style plug-in

API Description of the notice service

* POST
* Resource :
* Request :
* Media Type :
* Data :
* Response :
* GET
* Resource :
* Request :
* Media Type
* Data
* Response
* PUT
* Resource :
* Request
* Media Type
* Data
* Response
* DELETE
* Resource
* Request
* Media Type
* Data
* Response

Development Tools used

* Dependency management tool used - Maven
* Testing tools used – Postman
* Checking code quality tool used - Eclipse check style plug-in

**System’s Integration Details**

API gateway is a successful way to integrate a set of web services. API gateway is a management tool, which is sits between the frontend and backend.

The main feature of the API gateway is security, for example authentication and authorization. Connectivity with a range of different protocols, virtualization, scalability, elasticity, high available, manageability and development simplicity are features of the API gateway.

The basic functionality of an API gateway, it gets request from the web app redirect them to the micro service then get back the resonance to the web application. Basic feature the reverse proxy solves the problem of security. The API gateway could aggregate data at this level so that is why here instead of having the latency between frontend and the backend we can use aggregate pattern in order to reduce the latency of the requests.

Mainly, for Web API Testing, we need to check response code, a response message, and response body. API request for the first time to the API Gateway, the request is received by the HTTP/HTTPS transports. carrying messages that are in a specific format is the responsibility of the transport. Receiver is used to receive message, which is provide by the transport and a sender used to send messages. Then the receiving transport selects a message builder based on the message's content type. It uses the builder to process the message's raw payload data and convert it into a common XML, which the Gateway mediation engine then can read and understand. The Gateway includes message builders for text-based and binary content. Then the request is sent through a set of handlers that applies the quality of services on the request message. further, it enforces security, rate-limiting, and transformations if any on API requests while feeding valuable information of these requests to API Analytics. After all the request is routed to the backend endpoint.

**Reference**

[1] What is Agile methodology. Available: [SDLC (Software Development Life Cycle): What is, Phases & Models (guru99.com)](https://www.guru99.com/software-development-life-cycle-tutorial.html#:~:text=Summary%201%20The%20Software%20Development%20Life%20Cycle%20%28SDLC%29,set%20of%20activities%20and%20deliverables%20More%20items...%20)

[2] ConceptDraw “Stakeholders Onion Diagram: A Practical Guidance”. [Online] Available: [Stakeholder Onion Diagrams | How to Create a Stakeholder Onion Diagram | System stakeholders diagram | Diagram Of The Stakeholders (conceptdraw.com)](https://www.conceptdraw.com/examples/diagram-of-the-stakeholders)

[3] THE APP SOLUTION “"Functional and Nonfunctional Requirements: Specification”

[online] Available: [Functional vs Non-functional Requirements: List & Examples of systems engineering best practices and quality attribute (theappsolutions.com)](https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/)

[4] javatpoint, “Advantage of RESTful Web Services” [online] Available: [RESTful Web Services - javatpoint](https://www.javatpoint.com/restful-web-services)

[5] turtorialspoint “JSON Tutorial” [online] Available: [JSON Tutorial (tutorialspoint.com)](https://www.tutorialspoint.com/json/index.htm)

[6] JSON Turtorial for beginners “What is JSON and Learn JSON” Available: [JSON Tutorial for Beginners | What is JSON | Learn JSON - Bing video](https://www.bing.com/videos/search?q=learn+json+tutorial&&view=detail&mid=764217C79C8B1F292E37764217C79C8B1F292E37&&FORM=VRDGAR&ru=%2Fvideos%2Fsearch%3Fq%3Dlearn%2520json%2520tutorial%26qs%3Dn%26form%3DQBVR%26%3D%2525eManage%2520Your%2520Search%2520History%2525E%26sp%3D-1%26pq%3Dlearn%2520json%2520tutorial%26sc%3D0-19%26sk%3D%26cvid%3D04CD44E0974B4075AC1D069A2F7278F1)

[7] turtorialspoint “learn eclipse” Available: [Eclipse Tutorial (tutorialspoint.com)](https://www.tutorialspoint.com/eclipse/index.htm)

[8] java T point “"How to Implement GET and POST Requests With Java” available: [Java Get Post - Javatpoint](https://www.javatpoint.com/java-get-post)

[9]GeeksforGeeks “API Testing Using Postman” Available: [Basics of API Testing Using Postman - GeeksforGeeks](https://www.geeksforgeeks.org/basics-of-api-testing-using-postman/)

[10] CodeJava “Java CRUD RESTful web Services Example with jursey and Tomcat” Available: [Java CRUD RESTful Web Services Examples with Jersey and Tomcat (codejava.net)](https://www.codejava.net/java-ee/web-services/java-crud-restful-web-services-examples-with-jersey-and-tomcat)

[11] YouTube video about “API Manager documentation” Available: [What is WSO2 API Manager ? - YouTube](https://www.youtube.com/watch?v=nr1cFyxVdDw&t=2s)

[12] API Gateway Explained. Available: [API Gateway explained - Bing video](https://www.bing.com/videos/search?q=api+gateway&&view=detail&mid=69E72232C445983990DC69E72232C445983990DC&&FORM=VRDGAR&ru=%2Fvideos%2Fsearch%3Fq%3Dapi%2Bgateway%26FORM%3DHDRSC4)

[13] Difference API testing and the unit testing. [online] Available: [automated tests - What's difference between UI level testing and API level testing? - Stack Overflow](https://stackoverflow.com/questions/9525814/whats-difference-between-ui-level-testing-and-api-level-testing)

[14] Guru99“SDLC (Software Development Life Cycle): What is, Phases & Models” [online] Available: [SDLC (Software Development Life Cycle): What is, Phases & Models (guru99.com)](https://www.guru99.com/software-development-life-cycle-tutorial.html#:~:text=Summary%201%20The%20Software%20Development%20Life%20Cycle%20%28SDLC%29,set%20of%20activities%20and%20deliverables%20More%20items...%20)

**1. Stakeholder Analysis**

The Stakeholder Onion Diagram is a way of visualizing the relationship of stakeholders to a project goal.

To develop a Stakeholder Onion Diagram, we must identify the following categories

of stakeholders:

• Layer 1: This represents the Business System and entails not just the final product, but those stakeholders who interact

directly with it.

• Layer 2:This layer represents the business or organization that hosts, controls, or generally controls the project solution.

These are the other stakeholders within the organization who may not interact directly with the solution who benefit from it.

• Layer 3:This layer represents the wider environment in which the firm operates. This layer is populated with stakeholders

who are outside the firm but who are still important.

• Layer 4: External Stakeholders such as customers, buyers, government and suppliers.

According to this system, the onion diagram layers includes:

• Layer 1: Tester, Developer, Business Analyst

• Layer 2: System Administrator, User Customer, Billing Specialist

• Layer 3: Head Electrician, HR Manager, Financial Manager

• Layer 4: Supplier, Competitor, Politician

1. **Tools for Power Consumption service - IT**

**Reason to use Maven** – Transitive dependencies are automatically being added, eliminating the need to explore and specify the libraries that your own dependencies need by adding.

Usage - It makes project dependencies easier to handle. It must make certain that the identical source code is utilized in all situations. Dependence Management is used to consolidate all dependency information into a single POM file, simplifying the child POM file's references.

This feature is supported by looking through the project files of your dependencies from the previously mentioned external repositories. In general, when the project inherits from its parents or via its dependents, all of those projects' dependencies are utilized in your project.

The dependency management section serves as a central repository for dependent information. It is possible to place all information about the dependence in the common POM and have uncomplicated references to the artifact in the child POMs when you have a collection of projects that inherit from the same parent. It's especially useful when you have a lot of qualities and don't want to re-enter them under several kid projects. Finally, dependency management may be used to establish a basic version of an asset that can be shared across several projects.

1. **Tools for User/Customer service - IT**

**Reason to use Maven** - By automatically introducing transitive dependencies, Maven reduces the need to investigate and declare the libraries that your own dependencies require.

**Usage** - It makes project dependencies easier to handle. It must make certain that the identical source code is utilized in all situations. Dependence Management is used to consolidate all dependency information into a single POM file, simplifying the child POM file's references.

This feature is supported by looking through the project files of your dependencies from the previously mentioned external repositories. In general, when the project inherits from its parents or via its dependents, all of those projects' dependencies are utilized in your project.

The dependency management section serves as a central repository for dependent data. It is possible to put all information about the dependence in the common POM and have uncomplex references to the artifact in the child POMs when you have a collection of projects that inherit from a common parent. It's especially useful when you have a lot of qualities and don't want to re-enter them for each of your children's projects. Finally, dependency management may be used to establish a basic version of an asset that can be shared across several projects.

1. **Tools for Bill service – IT**

**Reason to use Maven -** By automatically introducing transitive dependencies, Maven reduces the need to investigate and declare the libraries that your own dependencies require.

**Usage -** It facilitates project dependency management. It needs to make certain that the same source code is utilized in all situations. Dependence Management is used to consolidate all dependency information into a single POM file, making the child POM file's references easier to understand.

This capability is made possible by looking through the project files of your dependencies from the previously mentioned external repositories. In general, when the project inherits from its parents or through its dependents, all of those projects' dependencies are utilized in yours.

The section on dependency management provides a way for centralizing dependence data. It is possible to include all information about the dependence in the common POM and have uncomplex references to the artifact in the child POMs when you have a collection of projects that all inherit from the same parent. When you have a lot of qualities and don't want to re-enter them under various offspring projects, this becomes incredibly useful. Finally, dependency management may be used to create a simple version of an asset that can be reused across several projects.

1. **Tools for Payment Service – IT**

**Reason to use Maven -** By adding transitive dependencies automatically, Maven eliminates the need to find and define the libraries that your own dependencies require. Reading the project files of your dependencies from the remote repository indicated simplifies this capability. In general, all dependencies from those projects, as well as those that the project inherits from its parents or dependents, are utilized in your project. The dependency management section serves as a central repository for dependent data. It's feasible to include all information about the dependence in the common POM and have easier references to the artifacts in the child POMs when you have a group of projects that inherit from a common parent.

**Testing, Reason to choose Postman-** Tests are automated by building test suites that can be performed repeatedly. Many other sorts of tests may be automated with Postman, including unit tests, functional tests, integration tests, end-to-end tests, regression tests, mock tests, and so on. Automated testing reduces testing time and eliminates human mistake.

**Checking code quality, Reason to use Checkstyle plug-in –** Because this project was developed using Eclipse, a plugin that can be utilized in the same environment is beneficial.

The Checkstyle Plugin (eclipse-cs) brings Checkstyle, a popular source code analyzer, into the Eclipse IDE. Checkstyle is a development tool that helps you make sure your Java code follows a set of coding guidelines. Checkstyle does this by examining your Java source code and highlighting elements that break a set of coding principles. Your code is regularly reviewed for errors using the Checkstyle Plugin. Problems are reported in the Eclipse workbench via the Eclipse Problems View and source code annotations, exactly as they are with compiler errors or warnings. A user-friendly Checkstyle configuration editor aids in the creation and maintenance of audit rule settings.

1. **Tools for Notice Service – IT**

**Dependency Management – Maven**

Dependency Management collects all dependency information into a single POM file, simplifying references in child POM files. Maven adds transitive dependencies automatically, thus it does not need to discover the libraries necessary. This is accomplished by accessing the project files of the dependencies from remote repositories.

Dependence Management is the process of centralizing dependency information. Maven allows projects that share a common parent and a shared POM to have easier references in the child POM. Furthermore, dependency management establishes a standard version of an asset for usage across many projects.

**Testing tools – Postman**

Unit tests, functional tests, integration tests, regression tests, mock tests, and end-to-end tests are all available in Postman, making it the greatest tool for testing.

**Code quality tool – Eclipse Checkstyle Plugin**

Eclipse is the project's integrated development environment. As a result, using a tool that supports Eclipse would be preferable. The Eclipse Checkstyle Plugin inspects Java source code on a regular basis and notifies you if there are any deviations from the conventional coding norms. The Eclipse Problems View is used to deliver these alert signals to the developer. This saves time and allows for faster development. The developer can also use the Checkstyle Configuration editor to design and manage audit rule setups.

1. Diagram

   Description automatically generatedPower consumption unit price management activity diagram
2. User / Customer Management Activity diagram

Diagram

Description automatically generated

1. Diagram

   Description automatically generatedBill Management Activity diagram
2. Payment Management Activity diagram

Diagram

Description automatically generated

1. Notice management Activity diagram

Diagram

Description automatically generated

**Appendix D**

**Individual DB tables**

* Power Consumption database

CREATE TABLE power\_consumption

(

powId int(10) NOT NULL PRIMARY KEY,

unitDescription varchar(200) NOT NULL,

unitPrice decimal(5,2) NOT NULL,

CONSTRAINT ck\_unitPrice1 CHECK (unitPrice > 0.00)

);

* Customer Database

CREATE TABLE customer

(

`accountNumber` int(10) NOT NULL PRIMARY KEY,

`firstName` varchar(20) NOT NULL,

`lastName` varchar(20) NOT NULL,

`address` varchar(100) NOT NULL,

`emailId` varchar(50) //CHECK (emailId LIKE '%\_@\_%.\_%'),

`contactNumber` char(10) NOT NULL //CHECK (contactNumber like'[0-9]{10}')

);

* Monthly Bill Database

CREATE TABLE monthly\_bill

(

billId int(10) NOT NULL AUTO\_INCREMENT,

year int(4) NOT NULL,

month varchar(15) NOT NULL,

totalUnits int(10) NOT NULL,

powId int(10) NOT NULL,

dueAmount decimal(10,2) NOT NULL,

billAmount decimal(10,2) NOT NULL,

accountNumber int(10) NOT NULL,

PRIMARY KEY(billId),

FOREIGN KEY(powId) REFERENCES power\_consumption(powId),

)

* Payment Database

CREATE TABLE payment

(

paymentId int(10) NOT NULL AUTO\_INCREMENT,

billId int(10) NOT NULL,

cardHolderName varchar(40) NOT NULL,

cardType int(10) NOT NULL,

bank varchar(40)) NOT NULL,

payAmount decimal(10,2) NOT NULL,

status varchar(20) NOT NULL,

PRIMARY KEY(paymentId),

FOREIGN KEY(billId) REFERENCES monthly\_bill(billId),

);

**Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | Test Description | Test Input | Expected Output | Actual Output | Result |
| 01 | Insert PowerConsumption | powId="4"  unitDescription="Units Above 300"  unitPrice="60.00" | Display a message as “Inserted successfully” | Display a message as “Inserted successfully” | Pass |
| 02 | Insert PowerConsumption | powId="4"  unitDescription="Units Above 300"  unitPrice="0.00" | Display a message as “Error while inserting data” | Display a message as “Error while inserting data” | Pass |
| 03 | Update PowerConsumption | powId="4"  unitDescription="Units Above 300"  unitPrice="65.00" | Display a message as “Updated successfully” | Display a message as “Updated successfully” | Pass |
| 04 | Update PowerConsumption | powId="4"  unitDescription="Units Above 300"  unitPrice="" / unitPrice=null | Display a message as “Error while updating data” | Display a message as “Error while updating data” | Pass |
| 05 | Delete PowerConsumption | powId="4" | Display a message as “Deleted Successfully” | Display a message as “Deleted Successfully” | Pass |
| 06 | Delete PowerConsumption  (Assume there are no records in power\_consumption table with powId="6") | powId="6" | Display a message as “Error while deleting data” | Display a message as “Error while deleting data” | Pass |
| 07 | Insert Customer | accountNumber="189054"  firstName="Dinura"  lastName="Perera"  address="75/A, Woxhell Street, Colombo-6"  emailId="dinuraf2000@gmail.com"  contactNumber="0724534780" | Display a message as “Inserted successfully” | Display a message as “Inserted successfully” | Pass |
| 08 | Insert Customer  (Enter contact Number more than 10 digits/characters) | accountNumber="189054"  firstName="Dinura"  lastName="Perera"  address="75/A, Woxhell Street, Colombo-6"  emailId="dinuraf2000@gmail.com"  contactNumber="072453478010" | Display a message as “Error while inserting data” | Display a message as “Error while inserting data” | Pass |
| 09 | Update Customer | accountNumber="189054"  firstName="Amal"  lastName="Perera"  address="75/A, Woxhell Street, Colombo-6"  emailId="amalp1996@gmail.com"  contactNumber="0778790890" | Display a message as “Updated successfully” | Display a message as “Updated successfully” | Pass |
| 10 | Update Customer | accountNumber="189054"  firstName="Amal"  lastName="Perera"  address="75/A, Woxhell Street, Colombo-6"  emailId="amalp1996@gmail.com"  contactNumber="" / contactNumber=null | Display a message as “Error while updating data” | Display a message as “Error while updating data” | Pass |
| 11 | Delete Customer | accountNumber="189054" | Display a message as “Deleted Successfully” | Display a message as “Deleted Successfully” | Pass |
| 12 | Generate(Add) Bill | year="2022"  month="February"  totalUnits="110"  powId="2"  dueAmount="300.00"  billAmount="3600.00"  accountNumber="189054" | Display a message as “Inserted successfully” | Display a message as “Inserted successfully” | Pass |
| 13 | Generate(Add) Bill  (Generate(Add) bill which contain same year,month,accountNumber entered previously  Violate the constraint - un\_mothly\_bill) | year="2022"  month="February"  totalUnits="130"  powId="2"  dueAmount="300.00"  billAmount="4200.00"  accountNumber="189054" | Display a message as “Error while inserting data” | Display a message as “Error while inserting data” | Pass |
| 14 | Update Bill | billId="14"  year="2022"  month="February"  totalUnits="120"  powId="2"  dueAmount="300.00"  billAmount="3900.00"  accountNumber="189054" | Display a message as “Updated successfully” | Display a message as “Updated successfully” | Pass |
| 15 | Update Bill  (Enter(Select) invalid accountNumber - There is no customer with particular account number.)  (violate the foreign key constraint) | billId="14"  year="2022"  month="February"  totalUnits="120"  powId="2"  dueAmount="300.00"  billAmount="3900.00"  accountNumber="189080" | Display a message as “Error while updating data” | Display a message as “Error while updating data” | Pass |
| 16 | Update Bill | billId="14"  year="" / year=null  month="February"  totalUnits="120"  powId="2"  dueAmount="300.00"  billAmount="3900.00"  accountNumber="189080" | Display a message as “Error while updating data” | Display a message as “Error while updating data” | Pass |
| 17 | Delete the Bill | billId="14" | Display a message as “Deleted Successfully” | Display a message as “Deleted Successfully” | Pass |
| 18 | Make(Add) Payment | billId="9"  cardHolderName="Dulanjani F"  cardType="VISA"  bank="BOC"  payAmount="3500.00"  status="Paid" | Display a message as “Inserted successfully” | Display a message as “Inserted successfully” | Pass |
| 19 | Make(Add) Payment | billId="9"  cardHolderName="Dulanjani F"  cardType="VISA"  bank="BOC"  payAmount="" / payAmount=null  status="Paid" | Display a message as “Error while inserting data” | Display a message as “Error while inserting data” | Pass |
| 20 | Update Payment | paymentId="3"  billId = "7"  cardHolderName="Abdulla MIM"  cardType="MASTER"  bank="COMMERCIAL"  payAmount="1600.00"  status="Paid" | Display a message as “Updated successfully” | Display a message as “Updated successfully” | Pass |
| 21 | Remove the Payment | paymentId="4" | Display a message as “Deleted Successfully” | Display a message as “Deleted Successfully” | Pass |
| 22 |  |  |  |  |  |
| 23 |  |  |  |  |  |
| 24 |  |  |  |  |  |