A picture containing text, clipart, vector graphics

Description automatically generated

**Sri Lanka Institute of Information Technology**

**Power Grid Maintenance System – ElectroGrid**

**Project Report**

**Programming Applications and Frameworks – IT3030**

**Group ID – 146**

**Group Members**

Jayasooriya C. A - IT20250942

Gavindya N.A.C - IT20409982

Bandara T.M.Y.M - IT20492052

Rathnaweera R.P.W.G - IT20237554

Amanullath M. U - IT20155520

**Submitted To:** Mr. Nalaka Dissanayake

**2022.04.24**

**Table of Contents**

[**Members details** 3](#_Toc101740466)

[**Git Repository Details** 4](#_Toc101740467)

[Clickable Link to the Remote Repository 4](#_Toc101740468)

[Commit Log 4](#_Toc101740469)

[**Software Engineering Methodology** 6](#_Toc101740470)

[**Time schedule (Gantt chart)** 7](#_Toc101740471)

[**Requirements Analysis** 8](#_Toc101740472)

[Stakeholder Analysis 8](#_Toc101740473)

[Functional Requirements 9](#_Toc101740474)

[Non-functional Requirements 10](#_Toc101740475)

[Technical Requirements [5] 11](#_Toc101740476)

[Use Case Diagram (Requirements Modelling) 12](#_Toc101740477)

[**System’s overall design** 13](#_Toc101740478)

[System’s Overall Architecture 13](#_Toc101740479)

[Overall Database Design – Entity Relationship Diagram 14](#_Toc101740480)

[Class Diagram 15](#_Toc101740481)

[**Individual sections** 16](#_Toc101740482)

[Jayasooriya C.A - IT20250942 16](#_Toc101740483)

[Gavindya N.A.C - IT20409982 18](#_Toc101740484)

[Bandara T.M.Y.M - IT20492052 20](#_Toc101740485)

[Rathnaweera R.P.W.G - IT20237554 22](#_Toc101740486)

[Amanullath M.U - IT20155520 24](#_Toc101740487)

[**System’s Integration Details** 26](#_Toc101740488)

[**References** 27](#_Toc101740489)

[**Appendix** 28](#_Toc101740490)

[Stakeholder Analysis 28](#_Toc101740491)

[Individual Work 29](#_Toc101740492)

[Jayasooriya C.A - IT20250942 29](#_Toc101740493)

[Gavindya N.A.C - IT20409982 32](#_Toc101740494)

[Bandara T.M.Y.M - IT20492052 35](#_Toc101740495)

[Rathnaweera R.P.W.G - IT20237554 38](#_Toc101740496)

[Amanullath M.U - IT20155520 41](#_Toc101740497)

[Testing Methodology and Results 44](#_Toc101740498)

[IT20250942 - Jayasooriya C. A 44](#_Toc101740499)

[IT20409982 - Gavindya N.A.C 48](#_Toc101740500)

[IT20492052 - Bandara T.M.Y.M 52](#_Toc101740501)

[IT20237554 – Rathnaweera R.P.W.G 56](#_Toc101740502)

[IT20155520 - Amanullath M.U 59](#_Toc101740503)

# **Members details**

|  |  |  |
| --- | --- | --- |
| Member Name | IT Number | Contribution |
| Jayasooriya C. A | IT20250942 | Breakdown Information Service   * Add breakdown. * Read all breakdowns. * Read a sector’s breakdowns. * Update breakdown. * Delete breakdown.   Payment Management Service   * Insert payment. |
| Gavindya N.A.C | IT20409982 | User Management Service   * Add a user. * Read all users. * Update user. * Delete user.   Payment Management Service   * Delete payment. |
| Bandara T.M.Y.M | IT20492052 | Bill Management Service   * Add a bill. * Read all bills. * Update bill. * Delete bill.   Payment Management Service   * Update payment. |
| Rathnaweera R.P.W.G | IT20237554 | Inquiry Support Service   * Add an inquiry. * Read all inquiries. * Update inquiry. * Delete inquiry.   Payment Management Service   * Read all payments. |
| Amanullath M. U | IT20155520 | Power Consumption Service   * Add a reading. * Read all reading history. * Update reading. * Delete reading.   Payment Management Service   * Calculate due amount. |

# **Git Repository Details**

## Clickable Link to the Remote Repository

The following is the clickable link which prompts the remote GitHub repository that was created to maintain the collaborative work environment throughout the project’s duration.

<https://github.com/aseljayasooriya/ElectroGrid_PAF.git>

## A screenshot of a computer Description automatically generated with medium confidenceA screenshot of a computer Description automatically generated with medium confidenceCommit Log

Table

Description automatically generatedTable

Description automatically generated

# **Software Engineering Methodology**

To accomplish efficient software development, this project employs the AGILE technique [1]. 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.

Four major principles [2] for agile project are:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The AGILE methodology's typical iteration process flow may be illustrated as follows:

* Requirements – Define the iteration's needs based on the product backlog, sprint backlog, and feedback from customers and stakeholders.
* Development – Create software based on a set of specifications.
* Testing – Internal and external training, as well as documentation development, are all part of the QA (Quality Assurance) process.
* Delivery – Integrate the working iteration into production and deliver it.
* Feedback – Accept consumer and stakeholder input and incorporate it into the next iteration's needs.

The AGILE methodology supported the development of the system in the following ways:

1. Set up a well-organized project management process.

2. Iterative development and testing are at the heart of the development lifecycle.

3. As a result of cross-team collaboration, requirements and solutions arise.

4. Allows for the rapid development of high-quality software.

5. Can quickly detect flaws, lowering risk.

6. It is more versatile than traditional methods since it can respond to changes faster.

7. It requires less planning and allows for more minor changes.

Restful architecture is used to construct the overall architecture while keeping the system's development in mind. REST web services are used to interact amongst micro services, and each micro service is built using the MVC framework.

# **Time schedule (Gantt chart)**

**Chart, bar chart, waterfall chart

Description automatically generated**

# **Requirements Analysis**

## Stakeholder Analysis

Diagram

Description automatically generated

## Functional Requirements

This establishes the specifications, or what a system should be able to accomplish. When specific conditions are met, it is the system's behavior [3]. The functional requirements of the Electro Grid system are the tasks that it must complete. The functional requirements that emerged from significant inter-team discussions and research are mentioned below.

Registering a new user

* System admin can register a new user to the system. A user can be meter owner and meter reader

Breakdown management

* When a user experiences a technical failure, electricity breakdowns, he/she can inform the breakdown manager via a form.

Bill management

* Usage of a consumers electricity consumptions are detected by the meter reader and the bill of the customer is prepared accordingly.

Payment management

* A meter owner can pay the bill related to their consumptions as the bill manager has finalized.

Support and Inquiry management

* When there are complaints and feedbacks for the consumers, they can use the form and address the admin for clarifications.

Power Consumption management

* The system should have the ability to read the power consumption of the users.

## Non-functional Requirements

These are requirements that aren't tied to a specific feature. Another word for this is quality attributes [4]. These criteria are more crucial to a system than the functional requirements since without them, the system would be unusable.

Performance:

* in the web application performance must be considered heavily since both customers and the system administrators are using this web application to make their task easier so the application must be loaded to the users within a minimum time.
* Web application should support the main available web browsers that commonly used by the users
* There are more than 1000s of record since that when accessing data in the database it should be done with a minimum time.

Security:

* Security is one of the utmost importance in the application as a lot of data and personal information are handle via the application. All the necessary security measures are taken within the application, as it contains payments information, personal contact information.
* To ensure the security adding new users and updating or deleting existing users can be done only using the administrators end.

Availability:

* As this a web application runs via the online platform the application will be used by the users at any time throughout the day. Therefore, the application should be available to accessible at any time. (24\*7)

Safety:

* If a system failure occurs on any time the database of this web application should be backup immediately. Since many useful data are used in this application.

Maintainability:

* The application has been designed and implemented in a very functional and practical manner, where the maintenance of the application needs minimum effort. Proper coding standards and practices have enabled this feature.

Reliability:

* Reliability is very important aspect in a web application since web application must have the ability to perform the provided functions and tasks from the customers’ side and the system administrators’ side with a minimum failure rate.

## Technical Requirements [1]

Browser capability:

* This system should use a web browser like google chrome, Mozilla Firefox, opera, brave to access world wide web. but some older version web browsers like Microsoft edge might not support well for this web application.

Mobile responsive design:

* a website will be responsive if the layout of the web application adjusts to the screen of the viewers who visit this web application. We had developed this to display well across all the devices like mobile, tablet, laptop, and PC.

Operating Systems:

* Since many users are using this web application, we had developed this to support main operating system like Windows, Linux and MACOS. which most of the users use these days.

## Use Case Diagram (Requirements Modelling)

Diagram, schematic

Description automatically generated

# **System’s overall design**

## System’s Overall Architecture

Diagram

Description automatically generated

The RESTful architecture was used in the development of this project. The cornerstone of a RESTful API, also known as a RESTful web service [5] or REST API, is Representational State Transfer (REST).

This architectural style and technique are often used in web services development for communications; this style allows systems to request access to and modify online resources using a standard and predetermined set of rules, hence it was chosen for this project with unanimous agreement within the team [6].

This system is made up of a client who makes requests for resources and a server that provides those resources and replies 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 interacts with the end user, an API differs from a GUI or other user interface [7]. This API only exposes database functionalities such as Create, Read, Update, and Delete (CRUD) activities via those services.

## Overall Database Design – Entity Relationship Diagram

Diagram

Description automatically generated

## Class Diagram

Diagram

Description automatically generated

# **Individual sections**

## Jayasooriya C.A - IT20250942

API description of the Breakdown Information Service.

**POST** -

• Resource: Breakdowns

• Request: POST BreakdownInformationService/BreakdownService/Breakdowns

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: breakdownSector, breakdownDate, startTime, endTime, breakdownType.

• Response: String Status message “Inserted Successfully”

**GET** -

• Resource: Breakdowns

• Request: GET BreakdownInformationService/BreakdownService/Breakdowns

• Response: HTML table with breakdownSector, breakdownDate, startTime, endTime, breakdownType columns.

**GET** -

Resource: Breakdowns

• Request: GET BreakdownInformationService/BreakdownService/Breakdowns/{breakdownSector}

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: breakdownSector.

• Response: HTML table with breakdownSector, breakdownDate, startTime, endTime, breakdownType columns.

**PUT** -

• Resource: Breakdowns

• Request: PUT BreakdownInformationService/BreakdownService/Breakdowns

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: breakdownID, breakdownSector, breakdownDate, startTime, endTime, breakdownType.

• Response: String Status message “Updated successfully.”

**DELETE**

• Resource: Breakdowns

• Request: DELETE BreakdownInformationService/BreakdownService/Breakdowns

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: breakdownID.

• Response: String Status message “Deleted Successfully”

API description of the Breakdown Payment Service.

**POST** -

• Resource: Payments

• Request: POST PaymentManagementService/PaymentService/Payments

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: accountNo, paymentAmount, paymentMethod, cardNo, email.

• Response: String Status message “Inserted Successfully”

* Please refer the Individual Work Appendix section for:

1. Individual Class diagrams.
2. Individual Activity diagrams.
3. Use case Scenario.
4. Tools used, including justifications for their selection.

* Please refer the Testing methodology Appendix section for:

1. Testing methodology and results

## Gavindya N.A.C - IT20409982

API description of the User Management Service.

**Post-**

* Resource: Users
* Request: POST UserManagementService/UserManagementService/Users
  + Media Type: Form Data- Application\_FORM\_URLENCODED
  + Data: userID, userNIC, userName, userAddress, userType, userSector
* Response: String Status message “Inserted Successfully”

**Get –**

* Resource: Users
* Request: Get UserManagementService/UserManagementService/Users
  + Data: userID, userNIC, userName, userAddress, userType, userSector
* Response: HTML table with userNIC, userName, userAddress, userType, userSector columns

**PUT-**

* Resource: Users
* Request: PUT UserManagementService/UserManagementService/Users
  + Media Type: Application\_JSON
  + Data: userID, userNIC, userName, userAddress, userType, userSector
* Response: String Status message “Updated Successfully”

**DELETE-**

* Resource: Users
* Request: DELETE UserManagementService/UserManagementService/Users
  + Media Type: Application\_XML
  + Data: userID
* Response: String Status message “Deleted Successfully”

API description of the Payment Service.

**POST** -

* Resource: Payments
* Request: DELETE PaymentManagementService/PaymentService/Payments
  + Media Type: Form Data- Application\_FORM\_URLENCODED
  + Data: paymentID
* Response: String Status message “Deleted Successfully”
* Please refer the Individual Work Appendix section for:

1. Individual Class diagrams.

2. Individual Activity diagrams.

3. Use case Scenario.

4. Tools used, including justifications for their selection.

* Please refer the Testing methodology Appendix section for:

1. Testing methodology and results

## Bandara T.M.Y.M - IT20492052

API description of the Bill management service.

**POST** -

• Resource: Bills

• Request: POST BillService/BillService/Bills

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: billCode, accountNo, billmonth, units, meterReader\_name

• Response: String Status message “Inserted Successfully”

**GET** -

• Resource: Bills

• Request: BillService/BillService/Bills

• Response: HTML table with billCode, accountNo, billmonth, units, billAmount, meterReader\_name columns.

**PUT** -

• Resource: Bills

• Request: PUT BillService/BillService/Bills

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: billID, billCode, accountNo, billmonth, units, meterReader\_name

• Response: String Status message “Updated successfully.”

**DELETE**

• Resource: Bills

• Request: DELETE BillService/BillService/Bills

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: billID.

• Response: String Status message “Deleted Successfully”

API description of the Breakdown Payment Service.

**PUT** -

• Resource: Payments

• Request: POST PaymentManagementService/PaymentService/Payments

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: paymentID, accountNo, paymentAmount, paymentMethod, cardNo, email.

• Response: String Status message “Updated Successfully”

* Please refer the Individual Work Appendix section for:

1. Individual Class diagrams.

2. Individual Activity diagrams.

3. Use case Scenario.

4. Tools used, including justifications for their selection.

* Please refer the Testing methodology Appendix section for:

1. Testing methodology and results

## Rathnaweera R.P.W.G - IT20237554

API description of the Support Inquiry Service

**POST-**

* Resource: Inquiry
* Request: POST SupportInquiryService/InquiryService/Inquiry

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: inquiryTitle,inquiryDesc,contactNum

* Response: String Status message “inserted successfully”

**GET-**

* Resource: Inquiry
* Request: GET SupportInquiryService/InquiryService/Inquiry
* Response: HTML table with Inquiry Title, Inquiry Description, Contact Number columns.

**PUT-**

* Resource: Inquiry
* Request: PUT SupportInquiryService/InquiryService/Inquiry

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: inquiryID,inquiryTitle,inquiryDesc,contactNum

* Response: String Status message “Update Sucessfully”

**DELETE-**

* Resource: Inquiry
* Request: DELETE SupportInquiryService/InquiryService/Inquiry

Media Type: Form Data- Application\_FORM\_URLENCODED

Data: inquiryID

* Response: String Status message “Deleted Sucessfully”

API description of the Breakdown Payment Service

**GET-**

* Resource: Payment
* Request: GET PaymentManagementService/PaymentService/Payment
* Response: JSON array with Account Number, Payment Amount, Payment Method, Card Number, email columns.
* Please refer the Individual Work Appendix section for:

1. Individual Class diagrams.

2. Individual Activity diagrams.

3. Use case Scenario.

4. Tools used, including justifications for their selection.

* Please refer the Testing methodology Appendix section for:

1. Testing methodology and results

## Amanullath M.U - IT20155520

API description of the Power Consumption Information Service.

**POST** -

• Resource: MeterReadings

• Request: POST PowerConsumptionService/readings

Media Type: JSON- Application\_Json

Data: meterReaderId, accountNo, year, month, reading.

• Response: JSON object {

"data": {

// inserted data

},

"message": "Meter Reading added successfully",

"status": "success"

}

**GET** -

• Resource: MeterReadings

• Request: GET PowerConsumptionService/readings

• Response: JSON array of meterRaeading objects.

**GET** -

Resource: MeterReadings

• Request: GET PowerConsumptionService/readings/account/:accountNo?year={year}&month={month}

Data: account number, year, month.

• Response: Meter Reading object as JSON which matches parameters.

**PUT** -

• Resource: MeterReadings

• Request: PUT PowerConsumptionService/readings/account/1?year=2022&month=10

Media Type: JSON – Application\_Json

Data: reading.

• Response: JSON with Status “success” and message “Meter Reading Updated successfully.”

**DELETE**

• Resource: MeterReadings

•Request:

DELETE PowerConsumptionService/readings/account/:accountNo?year={year}&month={month}

Data: accountNo, year, month.

• Response: String Status message “Deleted Successfully”

API description of the Breakdown Payment Service.

**POST** -

• Resource: Payments

•Request:

POST PaymentManagementService/PaymentService/Payments/account/:accountNo/due?asOfDate={date}

Data: accountNo, asOfDate (the date which the due amount must be calculated.

• Response: Double calculated due amount as at the given date

* Please refer the Individual Work Appendix section for:

1. Individual Class diagrams.

2. Individual Activity diagrams.

3. Use case Scenario.

4. Tools used, including justifications for their selection.

* Please refer the Testing methodology Appendix section for:

1. Testing methodology and results

# **System’s Integration Details**

SI (system integration) [8] is an IT or engineering process or phase involved with bringing together several subsystems or components into a single huge system. It guarantees that each integrated subsystem performs to specification.

SI may also be used to add value to a system by combining features from multiple systems to create new functionality.

Using an API gateway to integrate a group of online services [5] might be deemed a successful method.

An API organizes the requests that are handled by the microservices architecture in order to provide a more user-friendly experience for the client. With request routing, composition, and protocol translation, it accepts all API calls from clients and directs them to the relevant microservice. It's a translator who takes a customer's various requests and condenses them into a single request, reducing the number of rounds between the client and the application. In most cases, it responds to a request by calling numerous microservices and aggregating the data to find the most efficient approach.

It makes sense to create an API gateway [9] for most microservices-based systems since it operates as a single point of entry into the system. The API gateway is in charge of request routing, composition, and protocol translation, and it may help the system run more smoothly. Each of the application's customers receives a bespoke API when using an API gateway. Some requests are simply sent to the relevant backend service, while others are handled by calling numerous backend services and aggregating the results. If the backend services fail, the API gateway can hide the faults by delivering cached or default data.

# **References**

|  |  |
| --- | --- |
| [1] | "The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.," p. Chapter 7. |
| [2] | J. M. D. Santos, "project-management.com," 21 12 2021. [Online]. Available: https://project-management.com/. |
| [3] | M. Martin, "guru99," 18 02 2022. [Online]. Available: https://www.guru99.com/functional-requirement-specification-example.html. |
| [4] | D. Leffingwell, "Scaled Agile Framework," 10 02 2021. [Online]. Available: https://www.scaledagileframework.com/nonfunctional-requirements/. |
| [5] | Oracle, "What Are RESTful Web Services?," [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html. |
| [6] | "researchgate," researchgate, [Online]. Available: https://www.researchgate.net/figure/Overall-system-architecture. |
| [7] | "redhat," redhat, 31 October 2017. [Online]. Available: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces. |
| [8] | "altexsoft," altexsoft, 10 March 2021. [Online]. Available: https://www.altexsoft.com/blog/system-integration/. |
| [9] | C. Richardson, "microservices," [Online]. Available: https://microservices.io/patterns/apigateway.html. |
| [10] | I. Sacolick, "infoworld," 6 APR 2022. [Online]. Available: https://www.infoworld.com/. |
| [11] | T. Hamilton, "guru99," 16 April 2022. [Online]. Available: https://www.guru99.com/unit-testing-guide.html. |
| [12] | AnandDoijode, "geeksforgeeks," [Online]. Available: https://www.geeksforgeeks.org/system-testing. |
| [13] | R. Awati, "techtarget," [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/integration-testing. |
| [14] | "atlassian," atlassian, [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control. |
| [15] | tomcat.apache.org, "Apache Tomcat 9," apache software foundation, 22 March 2022. [Online]. Available: https://tomcat.apache.org/tomcat-9.0-doc/index.html. |

# **Appendix**

## Stakeholder Analysis

The Stakeholder Onion Diagram is a popular tool for visualizing stakeholder links to a project aim. We can determine the following stakeholder categories in order to create a Stakeholder Onion Diagram:

• Layer 1: Stakeholders who have been directly involved in the development of the product, which might be a new system or process. Software developers, testers, and other stakeholders are examples of stakeholders.

• Layer 2: Stakeholders whose job is affected by the solution. End users, for example.

• Layer 3: Investors, buyers, viewers, and subject matter experts who engage with the system on a regular basis.

According to this system, the onion diagram layers must be following:

1. Layer 1: Tester, Developer, Business Analyst
2. Layer 2: Sys Admin, Meter owner, Meter reader
3. Layer 3: Funders, Buyers, Viewers

## Individual Work

### Jayasooriya C.A - IT20250942

#### Table Description automatically generatedClass Diagram.

Text

Description automatically generated

#### Activity Diagram.

Diagram

Description automatically generated

#### Use Case Scenario Diagram.

|  |  |  |
| --- | --- | --- |
| **Number** | **EG003** | |
| **Name** | Add breakdown announcement | |
| **Summary** | System Administrator can add a breakdown notice into the system. | |
| **Priority (0-5)** | 5 | |
| **Preconditions** | The user must be logged into the system | |
| **Postconditions** | The added breakdown should be displayed in the Breakdowns table. | |
| **Primary Actor(s)** | System Administrator | |
| **Trigger** | System administrator clicks the add breakdown button. | |
| **Main Scenario** | **Step** | **Action** |
| 1 | The Sys Admin is directed to the portal home page |
| 2 | The Sys Admin presses the “Add Breakdown” button |
| 3 | The system displays the form related to the adding a new breakdown |
| 4 | The Sys Admin enters the relevant details in the text fields |
| 5 | The Sys Admin presses the “Add button” |
| 6 | The system clears the entered details of the form |
| 7 | The system sends the filled data to the database |
| 8 | The system redirects the Sys Admin to the updated Breakdowns table |
| **Extensions** | **Step** | **Action** |
| 4a | System displays error message if the required fields are not filled. |
| 5a | System disables the “Add Breakdown” button unless the checkbox is checked. |

#### Tools Used in the Development Process and the Justifications for Using Them

1. Dependency management tools – Maven.

Maven has been selected and used as the dependency management tool in this project as Maven minimizes the need to study and declare the libraries that your own dependencies require by automatically adding transitive dependencies. It makes managing project dependencies a lot easier. It ensures that the same source code is used across several environments. Dependence Management combines all dependence information into a single POM file, making the references in the child POM file easier to understand. The section on dependency management provides a way for centralizing dependence data.

1. JAX-RS (Jersey) –

It makes creating a RESTful service that can be deployed to any Java application server much easier.

1. Testing tools – Postman.

Postman has been used as the testing tool for this project as Postman is the most suitable tool for testing since it supports a wide range of test types, including unit, functional, integration, regression, mock, and end-to-end tests, all of which can be automated. Creating test suites that will run repeatedly until the application is error-free is one way to automate testing. The advantage is that human errors are extremely unlikely, if not impossible, to occur.

1. Code quality checking tools - Eclipse Check style Plugin.

The project's integrated development environment is Eclipse. As a result, it would be better to use an Eclipse-compatible tool. The Eclipse Checkstyle Plugin inspects Java source code on a regular basis and alerts you if any deviations from standard coding practices are detected. These alarm signals are delivered to the developer via the Eclipse Problems View. This enables for speedier development and saves time.

1. Version Controlling System - Git

We require collaboration among our group members because this is a group project, so it's simple to work on the same shared folder and comprehend all that has been done to the project, hence Git was used to facilitate this.

Git is a free, open-source, cross-platform distributed version control system that allows non-linear development and can manage everything from small to huge projects swiftly and effectively. Local branching, convenient staging areas, and various workflows are just a few of Git's capabilities. It also includes a variety of tools to help us navigate through the history, and each instance of the source has the whole history tree, which is quite useful during development even if we don't have Internet connection.

### Gavindya N.A.C - IT20409982

#### Class Diagram.

Table

Description automatically generatedText

Description automatically generated

#### Activity Diagram.

Diagram

Description automatically generated

#### Use Case Scenario Diagram.

|  |  |  |
| --- | --- | --- |
| **Number** | **EG004** | |
| **Name** | Update an existing user | |
| **Summary** | System Administrator can update an existing user into the system. | |
| **Priority (0-5)** | 3 | |
| **Preconditions** | The user must be logged into the system | |
| **Postconditions** | The added user should be displayed in the user table. | |
| **Primary Actor(s)** | System Administrator | |
| **Trigger** | System administrator clicks the update button. | |
| **Main Scenario** | **Step** | **Action** |
| 1 | The Sys Admin is directed to the portal home page |
| 2 | The Sys Admin presses the “Update User” button |
| 3 | The system displays the form related to the updating a new user |
| 4 | The Sys Admin enters the relevant details in the text fields |
| 5 | The Sys Admin presses the “Update button” |
| 6 | The system clears the entered details of the form |
| 7 | The system sends the filled data to the database |
| 8 | The system redirects the Sys Admin to the updated User table |
| **Extensions** | **Step** | **Action** |
| 4a | System displays error message if the required fields are not filled. |
| 5a | System disables the “Update” button unless the checkbox is checked. |

#### Tools Used in the Development Process and the Justifications for Using Them

1. Dependency Management tools– Maven

Dependency Management consolidates all dependency information into a single POM file, reducing the number of references in child POM files. Maven includes transitive dependencies automatically, so there's no need for it to figure out which libraries are needed. This is accomplished by accessing the dependents' project files from remote repositories.

The centralization of dependence information is what Dependency Management is all about. For projects that have a common parent and a shared POM, Maven enables for simplified references in the child POM. Furthermore, dependency management establishes a common version of an asset that may be used across various projects.

1. Version Controlling tools– Git

Git is a free, cross-platform, and open-source distributed version control solution that supports non-linear development and can handle everything from tiny to extremely large projects quickly and efficiently. Git has a number of useful features, like local branching, easy staging spaces, and different workflows. It also provides a number of tools to assist us in navigating around the history, and each instance of the source contains the whole history tree, which is quite useful during development even if we don't have access to the Internet.

1. Testing tools – Postman

Postman is the best tool for testing since it allows you to automate a wide range of test types, including unit, functional, integration, regression, mock, and end-to-end tests. One technique to automate testing is to create test suites that will run repeatedly until the application is error-free. Human mistakes are highly rare, if not impossible, to occur, which is a benefit.

### Bandara T.M.Y.M - IT20492052

#### Class Diagram.

Table

Description automatically generatedText

Description automatically generated

#### Activity Diagram.

Diagram, schematic

Description automatically generated

#### Use Case Scenario Diagram.

|  |  |  |
| --- | --- | --- |
| **Number** | **EG004** | |
| **Name** | Add bill | |
| **Summary** | Users can register a new item to the system | |
| **Priority (0-5)** | 5 | |
| **Preconditions** | The user must be logged into the system | |
| **Postconditions** | The added bill should be in the added bill list | |
| **Primary Actor(s)** | Meter reader | |
| **Trigger** | Meter reader clicks the add bill option | |
| **Main Scenario** | **Step** | **Action** |
| 1 | User logs into the system using staff credentials |
| 2 | The system redirects to the bill management portal |
| 3 | The user presses the “Add bill” button |
| 4 | The system displays the form related to the adding a new bill |
| 5 | The user enters the relevant details in the text fields |
| 6 | The user presses the “Add button” |
| 7 | The system clears the entered details of the form. |
| 8 | The system sends the filled data to the database and shows them in the added bill list |
| **Extensions** | **Step** | **Action** |
| 1a | The system displays an error message |
| 6a | The system displays alert messages |
| 6b | The user presses the ok button |
| 6c | The user enters correct data to the fields |

#### Tools Used in the Development Process and the Justifications for Using Them

1. Dependency management tools – Maven.

By introducing transitive dependencies automatically, Maven reduces the need to investigate and specify the libraries that your own dependencies require. It makes project dependencies easier to manage. It ensures that the same source code is used in different settings. Dependence Management is used to consolidate all dependency information into a single POM file, simplifying the child POM file's references. The section on dependency management is a way for centralizing dependence data.

1. Testing tools – Postman.

Postman is the finest testing tool since it can automate a variety of test kinds, including unit, functional, integration, regression, mock, and end-to-end tests. Creating test suites that run repeatedly until the program is error-free is one method for automating testing. Human errors are extremely rare, if not non-existent, which is a plus.

1. Code quality checking tools - Eclipse Check style Plugin.

Eclipse is the project's integrated development environment. As a result, choosing 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.

### Rathnaweera R.P.W.G - IT20237554

#### Class Diagram.

Table

Description automatically generatedText

Description automatically generated

#### Activity Diagram.

Diagram

Description automatically generated

#### Use Case Scenario Diagram.

|  |  |  |
| --- | --- | --- |
| **Number** | **EG004** | |
| **Name** | Make inquiry | |
| **Summary** | Meter owner(user) makes inquiry through the system | |
| **Priority (0-5)** | 5 | |
| **Preconditions** | The meter owner must be logged into the system | |
| **Postconditions** | Submitted inquiry should be posted on the inquiry table | |
| **Primary Actor(s)** | Meter owner | |
| **Trigger** | Meter owner clicks “Inquiry’ button to add an inquiry | |
| **Main Scenario** | **Step** | **Action** |
| 1 | The meter owner will be directed to the portal home page |
| 2 | The Meter owner clicks on the “Add Inquiry” button |
| 3 | The system will display all the related form details to the meter owner to add an inquiry. |
| 4 | The Meter owner add all the details to the relevant fields shown. |
| 5 | The Meter owner press the “Submit Inquiry” button |
| 6 | The system will clear details in the form. |
| 7 | The system sends the data which was filled by the Meter owner to the database |
| 8 | The system redirects the Meter owner back to the home page |
| **Extensions** | **Step** | **Action** |
| 4a | System displays error message if the Meter owner feed wrong details in the form. |
| 5a | System will disable the “Submit Inquiry” button unless the check box is ticked |

#### Tools Used in the Development Process and the justification for using them

1. Dependency management tools- Maven

The reason to use dependency management tool like maven is to its easy to manage project dependencies. Maven eliminates the need to explore and specify the libraries that your own dependencies need by adding transitive dependencies automatically. The dependency management section is a mechanism for centralizing dependency information, and it had ensured that the same source code will the used across all the environments and using this dependency management we are getting all the dependencies and collection them into a common POM file and simplifying the references in the child POM.

1. Testing Tools – Postman

Reason for selecting postman as a testing tool is that the tests can be done automated by developing test suites where the can-do tests repeatedly. such as unit tests, functional tests, integration tests, end-to-end tests, regression tests, mock tests, and other sorts of testing can all be automated with Postman. By using this automated testing method, it will lead us to reduces the risk of the human mistake and streamlines the testing process.

1. JAX-RS (Jersey) Framework

By using this jersey framework, it makes easy to create a RESTful service that can be deployed to any Java application server

1. Checking code quality- Checkstyle plug-in

To implement this project, we had used JAVA as the programming language and Eclipse as the IDE, so when using a plugin which can be used in the same coding environment is effective since that I had chosen the checkstyle plugin. The Checkstyle Plugin (eclipse-cs) integrates the well-known source code analyzer Checkstyle into the Eclipse IDE. Checkstyle is a development tool to help our ensure that your Java code adheres to a set of coding standards.

1. Version Control System – Git

Reason to use git as our version control system is that when we are developing we may come across to several type of errors since that it’s easy to use a version control system to undo the changes which we had used to develop the project and restore to the previous versions, Since this a group project we need a collaboration among our group members so it’s easy to work on the same shared folder which is easy to understand all the things that which have been done to the project.

### Amanullath M.U - IT20155520

#### Class Diagram.

Table

Description automatically generatedTable

Description automatically generated

#### Activity Diagram.

Diagram

Description automatically generated

#### Use Case Scenario Diagram.

|  |  |  |
| --- | --- | --- |
| **Number** | **EG005** | |
| **Name** | Add power consumption. | |
| **Summary** | Meter reader adds consumed units into the system. | |
| **Priority (0-5)** | 5 | |
| **Preconditions** | The user must be logged into the system | |
| **Postconditions** |  | |
| **Primary Actor(s)** | Meter reader | |
| **Trigger** | Meter reader wanting to add a reading. | |
| **Main Scenario** | **Step** | **Action** |
| 1 | The Meter reader is directed to the portal home page |
| 2 | The Meter reader presses the “Add Reading” button |
| 3 | The system displays add account number form. |
| 4 | The Meter reader enters the account no. |
| 5 | The system displays add year and date form. |
| 6 | The Meter reader enters the year and date. |
| 7 | The system displays add used units form. |
| 8 | The Meter reader enters the used units no. and submits the form. |
| 9 | System enters the power consumption details to the database. |
| **Extensions** | **Step** | **Action** |
| 3a | System displays proper error message if fields are not filled. |
| 8a | System displays error message if the required fields are not filled. |

#### Tools Used in the Development Process and the justification for using them

1. Dependency management tools – Maven.

Maven eliminates the need to research and define the libraries that your own dependencies require by automatically creating transitive dependencies. It facilitates the management of project dependencies. It ensures that the same source code is used across several environments. Dependence Management consolidates all dependence information into a single POM file, making the references in the child POM file easier to understand. The dependency management section is a method of centralizing dependency information.

1. Testing tools – Postman.

Postman is the best testing tool since it can automate a wide variety of test types, including unit, functional, integration, regression, mock, and end-to-end tests. One method for automating testing is to create test suites that will run until the application is error-free. Human errors are highly rare, if not impossible, to occur.

1. Code quality checking tools - Eclipse Check style Plugin.

The project's integrated development environment is Eclipse. As a result, it would be preferable to use an Eclipse-compatible tool. The Eclipse Checkstyle Plugin inspects Java source code on a regular basis and alerts you if any deviations from standard coding practices are detected. These alert signals are delivered to the developer via the Eclipse Problems View. This enables for speedier development and saves time.

## Testing Methodology and Results

### IT20250942 - Jayasooriya C. A

Testing – Breakdown Information Service and Payment Management Service (Create Payment function)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Test Inputs | Expected Outputs | Actual Outputs | Pass / Fail |
| T1001 | Insert a breakdown | breakdownSector = “F”  breakdownDate = “2022-04-22”  startTime = “11:00:00”  endTime = “17:00:00”  breakdownType = “Repair and cut” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |
| T1002 | Read sector breakdowns | breakdownSector = “A” | Display all breakdowns of sector A | Display all breakdowns of sector A | Pass |
| T1003 | Update a breakdown | breakdownID = “8”  breakdownSector = “F”  breakdownDate = “2022-04-25”  startTime = “11:00:00”  endTime = “17:00:00”  breakdownType = “Power cut canceled” | Display message as “Updated successfully” | Display message as “Updated successfully” | Pass |
| T1004 | Delete a breakdown | breakdownID = “8” | Display message as “Deleted successfully” | Display message as “Deleted successfully” | Pass |
| T1005 | Insert a payment | accountNo = “EG1122”  paymentAmount = “1220.00”  paymentMethod = “VISA”  cardNo = “111-222-231-332”  email= “aseljay@gmail.com” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, email, website

Description automatically generatedTable

Description automatically generated

Graphical user interface, text, application, email, website

Description automatically generated

Table

Description automatically generated

Graphical user interface, text, email, website

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, email, website

Description automatically generated

### IT20409982 - Gavindya N.A.C

Testing – User Management Service and Payment Management Service (Delete Payment function)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Test Inputs | Expected Outputs | Actual Outputs | Pass / Fail |
| T2001 | Insert a User | userID = “6”  userNIC = “678v”  userName = “wishwa”  userAddress = “kurunagala”  userType = “O”  userSector = “R” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |
| T2002 | Read Users |  | Display all users | Display all users | Pass |
| T2003 | Update a user | userID = “6”  userNIC = “999v”  userName = “kamal”  userAddress = “galle”  userType = “O”  userSector = “F” | Display message as “Updated successfully” | Display message as “Updated successfully” | Pass |
| T2004 | Delete a user | userID = “6” | Display message as “Deleted successfully” | Display message as “Deleted successfully” | Pass |
| T2005 | Delete a payment | paymentID = “3” | Display message as “Deleted successfully” | Display message as “Deleted successfully” | Pass |

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### IT20492052 - Bandara T.M.Y.M

Testing – Bill Service and Payment Management Service (Update Payment function)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Test Inputs | Expected Outputs | Actual Outputs | Pass / Fail |
| TST1001 | Insert a bill | billCode = “B020”  accountNo = “B700001”  billMonth = “Apr”  units = 5  meterReader\_name = “Saman” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |
| TST1002 | Insert a bill | billCode = “B030”  accountNo = “B700002”  billMonth = “Mar”  units = 7  meterReader\_name = “Jagath” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |
| TST1003 | Update a bill | billID = 8  billCode = “B021”  accountNo= “B700003”  billMonth = “Jun”  units = 10  meterReader\_name = “Jane” | Display message as “Updated successfully” | Display message as “Updated successfully” | Pass |
| TST1004 | Delete a bill | billID = 9 | Display message as “Deleted successfully” | Display message as “Deleted successfully” | Pass |
| TST1005 | Update a payment | paymentID = 2  accountNo = “EG2233”  paymentAmount = “12000.00”  paymentMethod = “VISA”  cardNo = “456-978-963-214”  email= “abs@gmail.com” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

### IT20237554 – Rathnaweera R.P.W.G

Testing – Support Inquiry Service and Payment Management Service (Read Payment function) –

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Test Inputs | Expected Outputs | Actual Outputs | Pass / Fail |
| T1010 | Insert an inquiry | inquiryTitle=”Refund”  inquiryDesc=”Need a payment refund for the bill”  contactNum=”0767990025” | Display message as “Inserted successfully” | Display message as “Inserted successfully” | Pass |
| T1011 | Update an inquiry | inquiryID=”12”  inquiryTitle=”Refund update”  inquiryDesc=”Updated Description”  contactNum=”0711812925” | Display message as “Update successfully” | Display message as “Update successfully” | Pass |
| T1012 | Delete an Inquiry | inquiryID=”12” | Display message as “Deleted successfully” | Display message as “Deleted successfully” | Pass |
| T1013 | Delete an Inquiry without passing inquiryID | inquiryTitle=”Refund update”  inquiryDesc=”Updated Description”  contactNum=”0711812925” | Display message as “Error while deleting the inquiry” | Display message as “Error while deleting the inquiry” | Pass |

**Graphical user interface, text, application, email

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

### IT20155520 - Amanullath M.U

Testing – Power Consumption Information Service and Payment Management Service (Calculate Due Amount function) – IT20155520 Amanullath M. U

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test Description | Test Inputs | Expected Outputs | Actual Outputs | Pass / Fail |
| T1001 | Insert a meter reading | "meterReaderId": 2,  "accountNo": "1",  "year": 2022,  "month": 12,  "reading": 20000 | Display message as “Meter Reading added successfully” with inserted data | Display message as “Meter Reading added successfully” with inserted data | Pass |
| T1002 | Get all meter readings by account no | accountNo = “1” | Display all meter reading of account number 1 | Display all meter reading of account number 1 | Pass |
| T1003 | Update a meter reading | Reading=10000  accountNo=”1”  month=12  year=2022 | Display message as “Updated successfully” | Display message as “Updated successfully” | Pass |
| T1004 | Delete a meter reading | accountNo=”1”  month=12  year=2022 | Display message as “Deleted successfully” | Display message as “Deleted successfully” | Pass |
| T1005 | Calculate due amount | accountNo=”1”  month=12  year=2022 | Display message as display the due amount as of 2022 december | Display message as display the due amount as of 2022 december | Pass |

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, email, website

Description automatically generatedTable

Description automatically generated

Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence