

Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

| INTERNAL VERIFICATION – ASSESSMENT DECISIONS | | | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|--------------------------|-------------|
| Programme title | BTEC HND in Computing | | |
| Assessor | | Internal Verifier | |
| Unit(s) | Unit 04: Database Design & Development | | |
| Assignment title | Database Solution for Polly Pipe | | |
| Student's name | K M G M B ALAHAKOON | | |
| List which assessment criteria the Assessor has awarded. | Pass | Merit | Distinction |
| INTERNAL VERIFIER CHECKLIST | | | |
| Do the assessment criteria awarded match those shown in the assignment brief? | Y/N | | |
| Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work? | Y/N | | |
| Has the work been assessed accurately? | Y/N | | |
| Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions? | Y/N Y/N Y/N Y/N | | |
| Does the assessment decision need | Y/N | | |

| | | |
|------------------------------------------|--|------|
| amending? | | |
| Assessor signature | | Date |
| Internal Verifier signature | | Date |
| Programme Leader signature (if required) | | Date |

| | | | |
|-----------------------------------------------|--|------|--|
| Confirm action completed | | | |
| Remedial action taken Give details: | | | |
| Assessor signature | | Date | |
| Internal Verifier signature | | Date | |
| Programme Leader signature (if required) | | Date | |

Higher Nationals - Summative Assignment Feedback Form

| | | | |
|--------------------|----------------------------------------|---------------------------------|--|
| Student Name/ID | K M G M B Alahakoon | | |
| Unit Title | Unit 04: Database Design & Development | | |
| Assignment Number | 1 | Assessor | |
| Submission Date | | Date Received 1st submission | |
| Re-submission Date | | Date Received 2nd submission | |

Assessor Feedback:

LO1 Use an appropriate design tool to design a relational database system for a substantial problem

Pass, Merit & Distinction P1 M1 D1

Descripts

LO2 Develop a fully functional relational database system, based on an existing system design

Pass, Merit & Distinction P2 P3 M2 M3 D2

Descripts

LO3 Test the system against user and system requirements.

Pass, Merit & Distinction P4 M4 D2

Descripts

LO4 Produce technical and user documentation.

Pass, Merit & Distinction P5 M5 D3

Descripts

| | | |
|-------------------------------|---------------------|-------|
| Grade: | Assessor Signature: | Date: |
| Resubmission Feedback: | | |
| Grade: | Assessor Signature: | Date: |

Internal Verifier's Comments:

Signature & Date:

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.

Assignment Feedback

Formative Feedback: Assessor to Student

Action Plan

Summative feedback

| | | | |
|-------------------------------|---------------------------|-------------|------------|
| Feedback: Student to Assessor | | | |
| Assessor signature | | Date | |
| Student signature | mayuraalahakoon@gmail.com | Date | 24-04-2022 |

Pearson Higher Nationals in Computing

Unit 04: Database Design & Development
Assignment 01

General Guidelines

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Time New Roman**.
2. **Use 1.5 line spacing.** Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each page.** This is useful if individual sheets become detached for any reason.
5. Use word processing application spell check and grammar check function to help editing your assignment.

Important Points:

1. It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.
2. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.

6. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.
10. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct form. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of Edexcel UK.
3. I know what the consequences will be if I plagiarise or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspect of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Pearson, UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the assignment.

mayuraalahakoon@gmail.com

30-04-2022

Student's Signature:

(Provide E-mail ID)

Date:

(Provide Submission Date)

Higher National Diploma in Computing

Assignment Brief

| | |
|-------------------------|--------------------------------------------------|
| Student Name /ID Number | K M G M B ALAHAKOON |
| Unit Number and Title | Unit 4: Database Design & Development |
| Academic Year | 2021/22Database |
| Unit Tutor | Mrs. Ishani |
| Assignment Title | Data base system for Polly Pipe |
| Issue Date | |
| Submission Date | |
| IV Name & Date | |

Submission format

Part 1: The submission should be in the form of an individual written report written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using Harvard referencing system. Please also provide in-text citation and bibliography using Harvard referencing system. The recommended word limit is 3,000–3,500 words, although you will not be penalised for exceeding the total word limit.

Part 2: The submission should be in the form of a fully functional relational database system demonstrated to the Tutor; and an individual written report (please see details in Part 1 above).

Part 3: The submission should be in the form of a witness statement of the testing completed by the Tutor; technical documentation; and a written report (please see details in Part 1 above).

Unit Learning Outcomes:

| | |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | |
| | <p>LO1 Use an appropriate design tool to design a relational database system for a substantial problem.</p> <p>LO2 Develop a fully functional relational database system, based on an existing system design.</p> <p>LO3 Test the system against user and system requirements.</p> <p>LO4 Produce technical and user documentation.</p> |
| | Assignment Brief and Guidance: |

Assignment brief

Polly Pipe is a water sports provider and installer based in Braintree, England. They need you to design and implement a database that meets the data requirements. These necessities are defined in this scenario and below are samples of the paper records that the Polly Pipe preserves.

Polly Pipe is focused in placing aquariums at business customers. Customers can request several installations, but each installation is tailor-made for a specific customer. Facilities are classified by type. One or more employees are assigned to each facility. Because these facilities are often very large, they can include carpenters and masons as well as water installers. The facilities use equipment such as aquariums, air pumps and thermostats. There can be multiple computers in a facility.

Below are examples of paper records that Polly Pipe currently maintains.

Staff Management Record

| Staff Number | Name | Type |
|--------------|---------------|----------------------|
| SHA1 | Dave Clark | Plumber |
| SHA8 | John Smith | Installation Manager |
| SHA2 | Freddy Davies | Aquatics installer |
| SHA11 | McCloud | Aquatics installer |
| SHA23 | Satpal Singh | Plumber |
| SHA66 | Winstn Kodogo | Aquatics installer |
| SHA55 | Alison Smith | Brick Layer |

Equipment Type Table

| Type | Equipment |
|-------------|------------------------------------------------------------------|
| Tanks | 20 gallon tank, 50 gallon tank, 100 gallon tank, 200 gallon tank |
| Thermostats | Standard, Super |
| Air Pumps | Standard, Super |

| | Filters | | Air driven, Undergravel | | | | |
|--|-----------------|-------------------|-------------------------------|----------|-----------|-------------------------|----------------------------|
| | Installation ID | Installation Type | Installation Name and Address | Customer | Equipment | Types of Staff Required | Period of Staff assignment |
| | | | | | | | |

| | | | | | | | |
|--|-----|---------------------|---------------------------------------------------|-------------------|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|-------------------------------|
| | 234 | Freshwater Tropical | Oak House, 17 Wroxton Road, Hertfordshire, H5 667 | Lee sun | A. 2 air pumps 200 gallons fish tank 1 x standard thermostat | 1 x Carpenter 1 x Aquatics installer 1 x Electrician | From 1st September 2012 |
| | 654 | Freshwater Cold | Bayliss House, Orange Street, Kent, K7 988 | Sally Dench | 2 air pumps 200 gallons fish tank Large Gravel Bag 2 x standard thermostat | 5 x Carpenters 1 x Installation Manager 1 x Aquatics installer 1 x Plumber 3 x Labourers | 1st June 2005 – 1st June 2011 |
| | 767 | Marine | Eaglestone Castle, Eaglestone , Kent | Perry Vanderru ne | 2 x 200 gallons fish tanks 500 Wood panels | 10 x Carpenters 2 x Installation Manager 1 x Aquatics installer 1 x Plumber 3 x Labourers | From 30th June 2012 |
| | 943 | Marine | 23 Sackville Street, Wilts. W55 | Eric Mackintosh | 2 air pumps 200 gallons fish tank 1 x standard | No staff required | |

| | | | | | | | |
|-----|---------------------|-------------------------------|------------------|-----------------------------------------------------------|------------------------|-----------------------------------------|--|
| | | | | | thermostat | | |
| 157 | Freshwater Tropical | Humberto son Castle, Kent, K8 | Perry Vanderrune | 2 air pumps 400 gallons fish tank 3 x standard thermostat | 1 x Aquatics installer | 1st September 2005 – 1st September 2012 | |

Instillation Management Form

Activity 1

1.1. Identify the user and system requirements to design a database for the above scenario and design a relational database system using conceptual design (ER Model) by including identifiers (primary Key) of entities and cardinalities, participations of relationships. Convert the ER Model into logical database design using relational database model including primary keys foreign keys and referential Integrities. It should contain at least five interrelated tables. Check whether the provided logical design is normalised. If not, normalize the database by removing the anomalies.

(Note:-It is allowed to have your own assumptions and related attributes within the scope of the case study given)

1.2. Design set of simple interfaces to input and output for the above scenario using Wireframe or any interface-designing tool. Evaluate the effectiveness of the given design (ERD and Logical design) in terms of the identified user and system requirements .

Activity 2

Activity 2.1

- a. Develop a relational database system according to the ER diagram you have created (Use SQL DDL statements). Provide evidence of the use of a suitable IDE to create a simple interface to insert, update and delete data in the database.

Implement proper security mechanisms in the developed database.

Evaluate the database solution developed and its effectiveness with relevant to the user and system requirements identified, system security mechanisms (EX:- User groups, access permissions) and the maintenance of the database.

Activity 2.2

- a. Explain the usage of DML with below mentioned queries by giving at least one single example per each case from the developed database. Assess the usage of the below SQL statements with the examples from the developed database to prove that the data extracted through them are meaningful and relevant to the given scenario.

Select/ Where / Update / Between / In / Group by / Order by / Having

Activity 3

Activity 3.1

Provide a suitable test plan to test the system against user and system requirements. provide relevant test cases for the database you have implemented. Assess how the selected test data can be used to improve the effectiveness of testing.

Note:- Learner needs to give expected results in a tabular format and screenshots of the actual results with the conclusion

Activity 3.2

Get independent feedback on your database solution from the non-technical users and some developers (use surveys, questioners, interviews or any other feedback collecting method) and make recommendations and suggestions for improvements in a separate conclusion/recommendations section.

Activity 4

Produce a technical documentation and a user guide for the developed database system.
Suitable diagrams (Use case diagram, class diagram, flow charts, DFD level 0 and

1) should be included in the technical documentation to show data movement in the system.

Assess the developed database by suggesting future enhancements to ensure the effectiveness of the system.

| Grading Criteria | Achieved | Feedback |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|----------|
| LO1 Use an appropriate design tool to design a relational database system for a substantial problem | | |
| P1 Design a relational database system using appropriate design tools and techniques, containing at least four interrelated tables, with clear statements of user and system requirements. | | |
| M1 Produce a comprehensive design for a fully functional system that includes interface and output designs, data validations and data normalization. | | |
| D1 Evaluate the effectiveness of the design in relation to user and system requirements. | | |

| | | |
|-------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| LO2 Develop a fully functional relational database system, based on an existing system design | | |
| P2 Develop the database system with evidence of user interface, output, and data validations, and querying across multiple tables. | | |
| P3 Implement a query language into the relational database system | | |
| M2 Implement a fully functional database system that includes system security and database maintenance. | | |
| M3 Assess whether meaningful data has been extracted using query tools to produce appropriate management information. | | |

| | | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| LO3 Test the systems against user and system requirements | | |
| P4 Test the system against user and system requirements. | | |
| M4 Assess the effectiveness of the testing, including an explanation of the choice of test data used. | | |
| LO2 & LO3 D2 Evaluate the effectiveness of the database solution in relation to user and system requirements, and suggest improvements. | | |
| LO4 Produce technical and user documentation | | |
| P5 Produce technical and user documentation. | | |
| M5 Produce technical and user documentation for a fully functional system, including diagrams showing movement of data through the system, and flowcharts describing how the system works. | | |
| D3 Evaluate the database in terms of improvements needed to ensure the continued effectiveness of the system. | | |

Table of Contents

| | |
|----------------------------------------------|----|
| Acknowledgments..... | 25 |
| 1.1.0 Requirements of Polly Pipe System..... | 26 |
| 1.1.1 Business Requirements | 26 |
| 1.1.2 User Requirements | 26 |
| 1.1.3 System Requirements..... | 27 |
| 1.1.2 Entity Relationship Diagram..... | 28 |
| 1.1.3 ER Diagram for Polly Pipe system | 28 |
| 1.1.4 Logical Relational Schema | 30 |
| 1.1.5 Data Dictionary | 31 |
| 1.1.6 Normalization..... | 35 |
| 1.1.6.0 First normal form (1NF) –..... | 35 |
| 1.1.6.1 Second normal form (2NF)-..... | 40 |
| 1.1.6.2 Third normal form (3NF)-..... | 49 |
| 1.2.0 GUI for PollyPIPE system | 51 |
| 2.0 Data Definition Language | 55 |
| 2.1 CREATE | 55 |
| 2.2 DROP DDL Statement..... | 61 |
| 2.3 ALTER DDL Statement..... | 61 |
| 2.4 Evaluate the above-mentioned GUI | 62 |
| 2.5 Database security | 62 |
| 2.5.0 Authentication | 62 |
| 2.5.1 Authorization..... | 65 |
| 2.5.2 Data protection | 65 |
| 2.6 Data Manipulation Language (DML) | 66 |
| 2.6.0 INSERT Statement..... | 67 |

| | |
|-----------------------------------------------------------|----|
| 2.6.1 SELECT Statement | 67 |
| 2.6.2 ORDER BY Statement..... | 68 |
| 2.6.3 UPDATE Statement..... | 68 |
| 2.6.4 GROUP BY Statement..... | 69 |
| 2.6.5 IN Statement..... | 69 |
| 2.6.6 BETWEEN Statement..... | 70 |
| 2.6.7 HAVING Statement | 70 |
| 3.0 Testing the developed PollyPIPE system..... | 71 |
| 3.1 Get independent feedback form using Google From | 74 |
| 3.2 Analysing feedback data | 77 |
| 4.0 Technical Documentation | 83 |
| 4.1 User guide for developed Polly PIPE system | 84 |
| 4.2 A use case diagram for the PollyPIPE system | 90 |
| 4.3 Class Diagram for PollyPIPE system..... | 91 |
| 4.4 Flow chart for PollyPIPE system | 91 |
| 4.5 Future enhancements for the PollyPIPE system | 92 |
| Consolations | 93 |
| References | 94 |

Table of Figures

| | |
|---------------------------------------------------|----|
| Figure 1 ER Diagram | 29 |
| Figure 2 Logical Relational Schema | 30 |
| Figure 3 Login GUI..... | 51 |
| Figure 4 Customer Details GUI | 51 |
| Figure 5 Payment & Staff Details GUI | 52 |
| Figure 6 Equipment Details Form..... | 52 |
| Figure 7 Installation Employee Details form | 53 |
| Figure 8 System Main Menu..... | 53 |
| Figure 9 Example of create statement 0 | 55 |
| Figure 10 Example of Create statement 1 | 56 |
| Figure 11 Example of create statement 2 | 56 |
| Figure 12 Example of create statement 3 | 56 |
| Figure 13 Example of create statement 4 | 57 |
| Figure 14 Example of create statement 5 | 57 |
| Figure 15 Example of create statement 6 | 58 |
| Figure 16 Example of create statement 7 | 58 |
| Figure 17 Example of create statement 8 | 59 |
| Figure 18 Example of create statement 9 | 59 |
| Figure 19 Example of create statement 10 | 60 |
| Figure 20 Example of create statement 11 | 60 |
| Figure 21 Example of DROP statement 0..... | 61 |
| Figure 22 Example of ALTER statement 0..... | 61 |
| Figure 23 Login with valid details | 63 |
| Figure 24 Main Menu..... | 63 |
| Figure 25 Login with invalid details | 64 |
| Figure 26 Login details in the database..... | 64 |
| Figure 27 user authorization menu..... | 65 |
| Figure 28 Example of INSERT statement | 67 |
| Figure 29 Example of select statement 0 | 67 |
| Figure 30 Example of select statement 1 | 67 |
| Figure 31 Example of ORDER statement | 68 |

| | |
|----------------------------------------------|----|
| Figure 32 Example of UPDATE statement..... | 68 |
| Figure 33 Example of GROUP statement..... | 69 |
| Figure 34 Example of IN statement | 69 |
| Figure 35 Example of BETWEEN statement | 70 |
| Figure 36 Example of HAVING statement..... | 70 |
| Figure 37 Test case 0..... | 71 |
| Figure 38 Test case 1..... | 71 |
| Figure 39 Test case 2..... | 72 |
| Figure 40 Test case 3..... | 72 |
| Figure 41 Test case 4..... | 73 |
| Figure 42 User Feedback | 76 |
| Figure 43 Analysing feedback data 0..... | 77 |
| Figure 44 Analysing feedback data 1..... | 77 |
| Figure 45 Analysing feedback data 2..... | 78 |
| Figure 46 Analysing feedback data 3..... | 78 |
| Figure 47 Analysing feedback data 4..... | 79 |
| Figure 48 Analysing feedback data 5..... | 79 |
| Figure 49 Analysing feedback data 6..... | 80 |
| Figure 50 Analysing feedback data7..... | 80 |
| Figure 51 Analysing feedback data 8..... | 81 |
| Figure 52 Analysing feedback data 9..... | 81 |
| Figure 53 Analysing feedback data 10..... | 82 |
| Figure 54 Analysing feedback data 11 | 82 |
| Figure 55 User guide 0 | 84 |
| Figure 56 User guide 1 | 85 |
| Figure 57 User guide 2 | 87 |
| Figure 58 User guide 3 | 89 |
| Figure 59 User case diagram..... | 90 |
| Figure 60 Class diagram..... | 91 |
| Figure 61 Flow chart | 91 |
| Figure 62 Gantt chart | 93 |

Acknowledgments

I would like to express my gratitude to Mis. Ishani for her guidance and extreme patience in supervising this thesis. I am indebted to my parents and my sister for their continued support both financially and spiritually throughout my academic year.

Activity 1.1

1.1.0 Requirements of Polly Pipe System

Requirements are functionally expected by the user from the whole system.

There are three types of requirements.

1. Business Requirements
2. User Requirements
3. System Requirements

1.1.1 Business Requirements

These include a high-level statement of goals, objectives, and needs. Business requirements do not include any details or specific features. They just state the problem and the business objective to be achieved such as increased revenue/throughput/ customer each, reduced expenses/ errors, improved customer service, etc. (altexsoft, 2022)

1.1.2 User Requirements

User requirements often referred to as user needs, describe what the user does with the system, such as what activities that user must be able to perform. User requirements are generally documented in User Requirements Documents (URD) using narrative text. (Parker, 2012)

User requirements in Polly Pipe

- I. Inserting, Updating, Deleting, Uploading all the details into the system
- II. Calculation of payments
- III. Printing all the payments details
- IV. Managing all the staff details
- V. Analysing all the details
- VI. Monitoring the system

1.1.3 System Requirements

System requirements describe specific characteristics that a product must have to meet the needs of the users and the business itself. (altexsoft, 2022)

System requirements can divide into two categories. There are,

1. Software requirements – There are two types of software requirements.
 - I. Functional requirements- This defines what a product must do, and what its features and functions are.

Functional requirements in Polly Pipe system.

 - A user must have to inserting, updating, and deleting all the details into the database
 - The system must have to GUI
 - A user must have done the authentication process
 - A user must have to do calculations (payments, salaries, etc...)
 - A user must have to manage all the staff and company details
 - A user should be able to check the installing process status
 - A user should be able to get a print outing
 - II. Non-functional requirements – This defines the general properties of a system.
 - Accessibility
 - Security
 - Availability
 - Backup and restore
 - Compatibility of software, tools, standards, platform, database
 - Efficiency
 - Supportability

2. Hardware requirements – These are the requirements of a hardware device.

Hardware requirements in Polly Pipe system.

| | |
|--------------------|--------------------------------------------------------------------------------------|
| Operating System | Windows 10 or any Linux Distributions (E.g., Ubuntu) |
| CPU | Intel or AMD processor with 64-bit support; Recommended: 2.8 GHz or faster processor |
| RAM | Minimum 4GB Ram Maximum 8 GB Ram |
| Monitor Resolution | 1920x1080 |
| GPU | Intel, Nvidia, or AMD; Recommend Intel(R) UHD Graphics 620 or up |
| Disk Storage | 60 GB of free disk space |
| Internet | Wi-Fi or wired (Fiber cables etc.) |

1.1.2 Entity Relationship Diagram

This is a type of flowchart that illustrates how “entities” such as people, objects, or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education, and research. (Lucidchart, 2022)

1.1.3 ER Diagram for Polly Pipe system

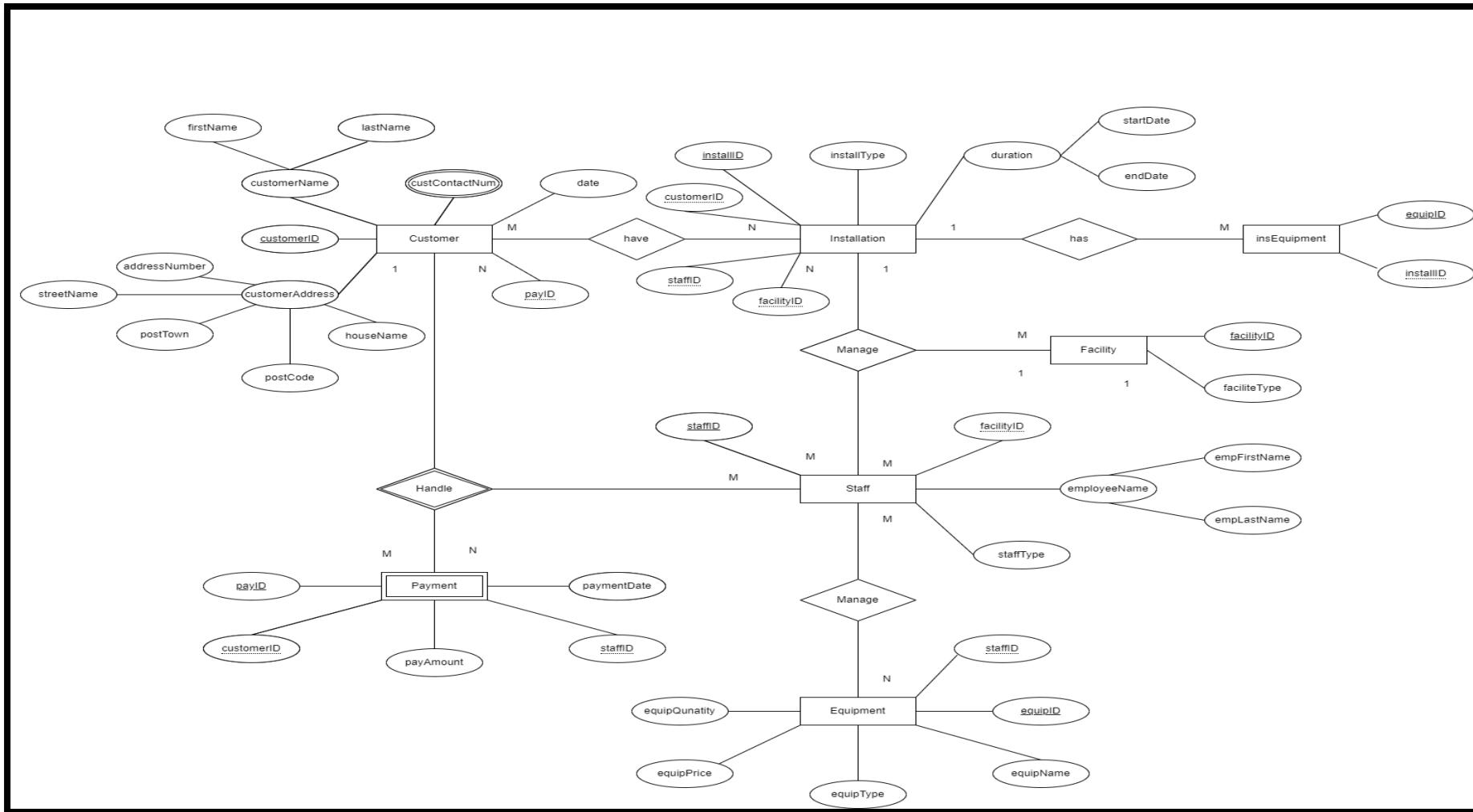


Figure 1 ER Diagram

1.1.4 Logical Relational Schema

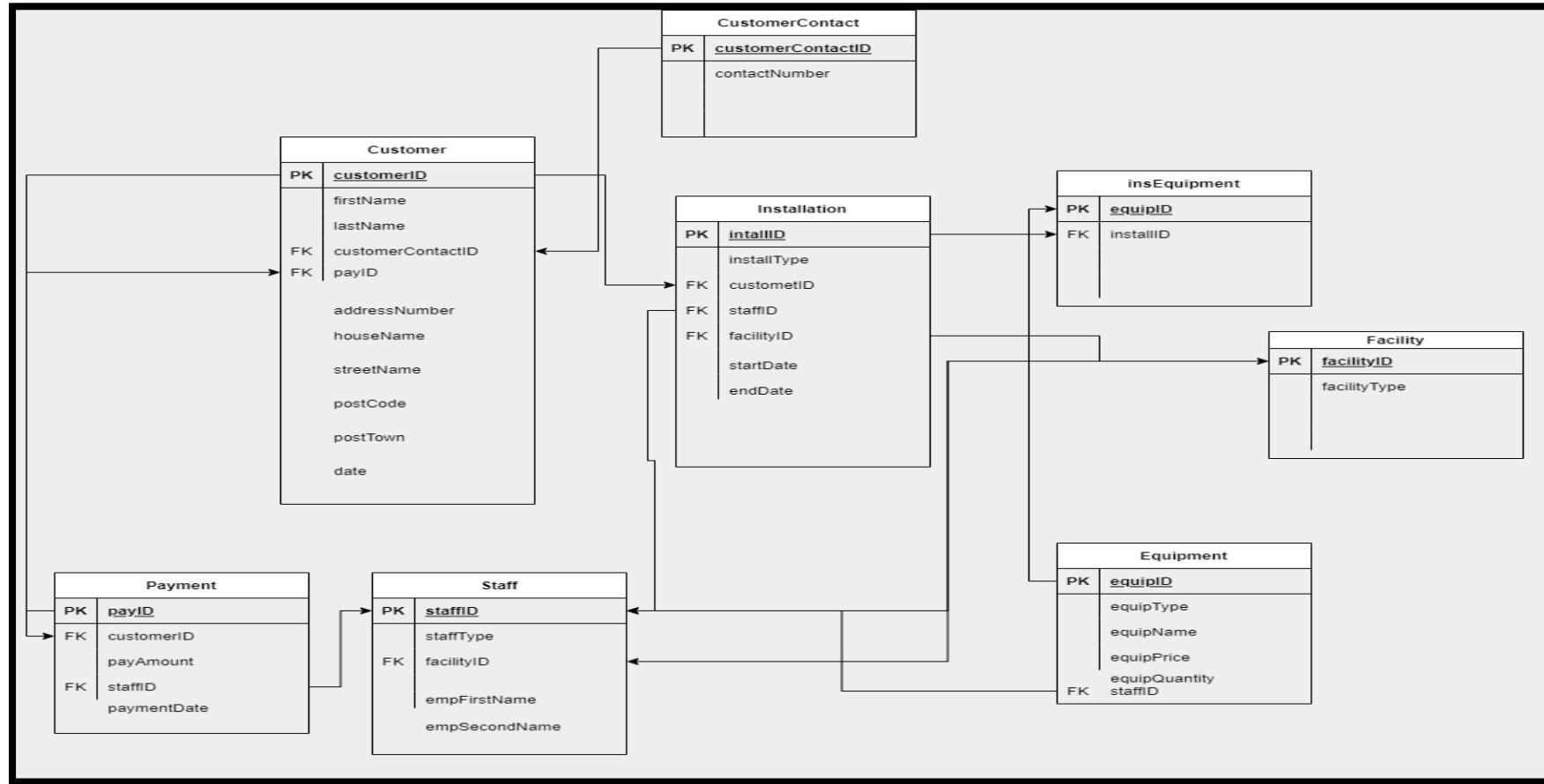


Figure2LogicalRelationalSchema

1.1.5 Data Dictionary

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. It describes the meanings and purposes of data elements within the context of a project and provides guidance on interpretation, accepted meanings, and representation. A Data Dictionary also provides metadata about data elements. (MERCED, 2022)

1. Data Dictionary of the Customer table

| Name | Type | Length | Constraint | Description |
|-------------------|---------|--------|------------|------------------------------------|
| firstName | Varchar | 20 | Not null | First name of customer |
| lastName | Varchar | 20 | Not null | Last name of customer |
| <u>customerID</u> | int | 10 | Not null | Unique number ID for all customers |
| addressNumber | int | 10 | null | Home address number of customers |
| houseName | Varchar | 10 | null | House name of customer |
| streetName | Varchar | 10 | null | The street name of customer |
| postTown | Varchar | 10 | null | Post town of customer |
| postCode | Varchar | 10 | null | Post code of customer |

| | | | | |
|--------------|------|------------|------|---------------------------------|
| date | date | dd/mm/yyyy | null | The date of a customer came |
| <u>payID</u> | int | 10 | null | Unique number ID for Payment ID |

2. Data Dictionary of the Payment table

| Name | Type | Length | Constraint | Description |
|--------------------|---------|------------|------------|-----------------------------------|
| <u>payID</u> | int | 10 | Not null | Unique number ID for all payments |
| <u>payAmount</u> | money | - | Not null | Money of the payment |
| <u>paymentDate</u> | date | dd/mm/yyyy | Not null | Date for all payments |
| <u>staffID</u> | Varchar | 10 | Not null | Unique number ID for staff id |
| <u>customerID</u> | int | 10 | Not null | Unique number ID for the customer |

3. Data Dictionary of the Staff table

| Name | Type | Length | Constraint | Description |
|------------------|---------|--------|------------|---------------------|
| <u>staffID</u> | varchar | 10 | Not null | Unique ID for staff |
| <u>staffType</u> | Varchar | 10 | Not null | Type of staff |

| | | | | |
|-------------------|---------|----|----------|--------------------------------|
| employeeName | Varchar | 20 | null | Name of employee |
| <u>equipID</u> | int | 10 | Not null | Unique number for Equipment ID |
| <u>facilityID</u> | int | 10 | Not null | Unique number of Facility |

4. Data Dictionary of the Facility table

| Name | Type | Length | Constraint | Description |
|-------------------|---------|--------|------------|---------------------------------------|
| <u>facilityID</u> | int | 10 | Not null | Unique number ID for each of facility |
| facilityType | Varchar | 10 | Not null | Name of facility |

5. Data Dictionary of the Equipment table

| Name | Type | Length | Constraint | Description |
|----------------|---------|--------|------------|----------------------------------------|
| <u>equipID</u> | int | 10 | Not null | Unique number ID for each of equipment |
| equipType | Varchar | 10 | Not null | Type of equipment |
| equipName | Varchar | 10 | Not null | Name of equipment |
| equipQuantity | int | MAX | Not null | Quantity of equipment |
| equipPrice | money | - | Not null | Price of |

| | | | | |
|--|--|--|--|----------------|
| | | | | each equipment |
|--|--|--|--|----------------|

6. Data Dictionary of the CustomerContact table

| Name | Type | Length | Constraint | Description |
|-------------------------|------|--------|------------|---------------------------------------------------------|
| <u>customerContacID</u> | int | 10 | Not null | Unique number ID for each of customer's contact numbers |
| contactNumber | int | 1 | Not null | Contact of Customer |

7. Data Dictionary of the Installation table

| Name | Type | Length | Constraint | Description |
|-------------------|---------|------------|------------|-------------------------------------------|
| <u>installID</u> | int | 10 | Not null | Unique number ID for each of installation |
| installType | Varchar | 10 | Not null | Type of installation |
| <u>staffID</u> | int | 10 | Not null | Unique number ID for staff |
| startDate | date | dd/mm/yyyy | Not null | Start Date of installation |
| endDate | date | Dd/mm/yyyy | Not null | Last date of installation |
| <u>facilityID</u> | int | 10 | null | Unique number ID for facility |

8. Data Dictionary of the insEquipment table

| Name | Type | Length | Constraint | Description |
|----------------|------|--------|------------|--------------------------------------------|
| <u>equipID</u> | int | 10 | Not null | Unique number ID for each of the Equipment |
| installID | int | 10 | Not null | InstallID from Installtion Table |

1.1.6 Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency. (Microsoft, 2022)

Normalization is the process of removing potential anomalies from the database.

These are – Insertion, update, and deletion anomalies.

Types of normalization.

1.1.6.0 First normal form (1NF) —

- i. Eliminate repeating groups in individual tables.
- ii. Create a separate table for each set of related data.
- iii. Identify each set of related data with a primary key.

Customer Table

| <u>installcus tomerID</u> | <u>fsrtName</u> | <u>lastName</u> | <u>installI D</u> | <u>customer ContactID</u> | <u>payID</u> | <u>Address Number</u> | <u>houseName</u> | <u>streetName</u> | <u>postTown</u> | <u>postCode</u> | <u>date</u> |
|-------------------------------|-----------------|-----------------|-----------------------|-------------------------------|--------------|---------------------------|-------------------|-------------------|-----------------|-----------------|-------------|
| 01 | Lee | A.sun | 234 | 222 | 333 | 17 | Oak House | Wroxton Road | | H5 667 | 6-5-2012 |
| 02 | Sally | Dench | 654 | 444 | 555 | | Bayliss House | Orange Street | Kent | K7 988 | 7-6-2011 |
| 03 | Perry | Vanderrune | 767 | 666 | 777 | | Eaglestone Castle | | Kent | | 6-5-2012 |
| 04 | Eric | Mackintosh | 943 | 888 | 999 | 23 | | Sackville Steet | Wilts | W55 | |
| 03 | Perry | Vanderrune | 157 | 666 | 111 | | Humbertson Castle | | Kent | K8 | 5-7-2005 |

New Table: Customer Address Table

| <u>CusAddressID</u> | AddressNumber | houseName | streetName | postTown | postCode |
|---------------------|---------------|-------------------|------------------|----------|----------|
| 1 | 17 | Oak House | Wroxton Road | | H5 667 |
| 2 | | Bayliss House | Orange Street | pst1 | K7 988 |
| 3 | | Eaglestone Castle | | pst1 | |
| 4 | 23 | | Sackville Street | pst2 | W55 |
| 5 | | Humbertson Castle | | pst1 | K8 |

New Table: Post Town Table

| <u>postTownID</u> | postTown |
|-------------------|----------|
| Pst '1 | Kent |
| pst2 | Wilts |

Customer Table

| <u>installCustomerID</u> | fisrtName | lastName | <u>customerContactID</u> | <u>payID</u> | <u>CusAddressID</u> |
|--------------------------|-----------|------------|--------------------------|--------------|---------------------|
| 01 | Lee | A.sun | 222 | 333 | 1 |
| 02 | Sally | Dench | 444 | 555 | 2 |
| 03 | Perry | Vanderrune | 666 | 777 | 3 |
| 04 | Eric | Mackintosh | 888 | 999 | 4 |
| 03 | Perry | Vanderrune | 666 | 777 | 5 |

Normalized Customer Table

| <u>customerID</u> | <u>installCustomerID</u> | <u>payID</u> | <u>CusAddressID</u> |
|-------------------|--------------------------|--------------|---------------------|
| 01 | 01 | 333 | 1 |
| 02 | 02 | 555 | 2 |
| 03 | 03 | 777 | 3 |
| 04 | 04 | 999 | 4 |
| 05 | 03 | 777 | 5 |

Installation Equipment Table: UNF

| <u>equipID</u> | <u>installID</u> |
|----------------|------------------|
| 0008 | 234 |
| 0008 | 234 |
| 0005 | 234 |
| 0006 | 234 |
| 0009 | 654 |
| 0009 | 654 |
| 0005 | 654 |
| 0123 | 654 |
| 0006 | 654 |
| 0005 | 654 |
| 0005 | 654 |
| 0124 | 654 |
| 0012 | 943 |
| 0008 | 943 |
| 0008 | 943 |
| 0008 | 943 |
| 0006 | 943 |
| 0008 | 157 |
| 0008 | 157 |
| 0012 | 157 |
| 0006 | 157 |

Normalized Installation Equipment Table

| <u>installEquipID</u> | <u>equipID</u> | <u>installID</u> |
|-----------------------|----------------|------------------|
| insEqui01 | 0008 | 234 |
| insEqui02 | 0008 | 234 |
| insEqui03 | 0005 | 234 |
| insEqui04 | 0006 | 234 |
| insEqui05 | 0009 | 654 |
| insEqui06 | 0009 | 654 |
| insEqui07 | 0005 | 654 |
| insEqui08 | 0123 | 654 |
| insEqui09 | 0006 | 654 |
| insEqui10 | 0005 | 654 |
| insEqui11 | 0005 | 654 |
| insEqui12 | 0124 | 654 |
| insEqui13 | 0012 | 943 |
| insEqui14 | 0008 | 943 |
| insEqui15 | 0008 | 943 |
| insEqui16 | 0008 | 943 |
| insEqui17 | 0006 | 943 |
| insEqui18 | 0008 | 157 |
| insEqui19 | 0008 | 157 |
| insEqui20 | 0012 | 157 |
| insEqui21 | 0006 | 157 |

1.1.6.1 Second normal form (2NF)-

- i. Create separate tables for a set of values that apply to multiple records.
- ii. Relate these tables with a foreign key.

Staff Table

| <u>staffID</u> | staffType | empFirstName | empLastName |
|----------------|----------------------|--------------|-------------|
| SHA1 | Plumber | Dave | Clark |
| SHA8 | Installation Manager | John | Smith |
| SHA9 | Installation Manager | Adam | Tyler |
| SHA2 | Aquatics installer | Freddy | Davies |
| SHA11 | Aquatics installer | McCloud | |
| SHA23 | Plumber | Satpal | Singh |
| SHA66 | Aquatics installer | Winston | Kodagu |
| SHA55 | Brick Layer | Alison | Smith |
| SHA77 | Electrician | Andrew | Green |
| SHA88 | Carpenter | Tom | Bruce |
| SHA99 | Carpenter | Daniel | Christopher |
| SHA10 | Carpenter | Robert | Charles |
| SHA12 | Carpenter | James | Matthew |
| SHA13 | Carpenter | David | Anthony |
| SHA18 | Carpenter | Gary | Jacob |
| SHA19 | Carpenter | Timothy | Nicholas |
| SHA20 | Carpenter | Stephen | Larry |
| SHA21 | Carpenter | Justin | Scott |
| SHA22 | Carpenter | Samuel | Brandon |
| SHA15 | Labourer | Donald | Trump |
| SHA16 | Labourer | Paul | Walker |

| | | | |
|-------|----------|-------|---------|
| SHA17 | Labourer | Kevin | Kenneth |
|-------|----------|-------|---------|

New Table: TypeofStaff Table

| <u>staffTypeID</u> | staffType |
|--------------------|----------------------|
| staty1 | Plumber |
| staty2 | Installation Manager |
| staty3 | Aquatics installer |
| staty4 | Brick Layer |
| staty5 | Carpenter |
| staty6 | Electrician |
| staty7 | Labourer |

Normalized Staff Table

| <u>staffID</u> | <u>staffTypeID</u> | empFirstName | empLastName |
|----------------|--------------------|--------------|-------------|
| SHA1 | staty1 | Dave | Clark |
| SHA8 | staty2 | John | Smith |
| SHA9 | staty2 | Adam | Tyler |
| SHA2 | staty3 | Freddy | Davies |
| SHA11 | staty3 | McCloud | |
| SHA23 | staty1 | Satpal | Singh |
| SHA66 | staty3 | Winston | Kodagu |
| SHA55 | staty5 | Alison | Smith |
| SHA77 | staty7 | Andrew | Green |
| SHA88 | staty6 | Tom | Bruce |
| SHA99 | staty6 | Daniel | Christopher |
| SHA10 | staty6 | Robert | Charles |
| SHA12 | staty6 | James | Matthew |
| SHA13 | staty6 | David | Anthony |
| SHA18 | staty6 | Gary | Jacob |

| | | | |
|-------|--------|---------|----------|
| SHA19 | staty6 | Timothy | Nicholas |
| SHA20 | staty6 | Stephen | Larry |
| SHA21 | staty6 | Justin | Scott |
| SHA22 | staty6 | Samuel | Brandon |
| SHA15 | staty7 | Donald | Trump |
| SHA16 | staty7 | Paul | Walker |
| SHA17 | staty7 | Kevin | Kenneth |

Equipment Table

| <u>equipID</u> | equipName | equipType | equipPrice | equipQuantity |
|----------------|-------------|--------------|------------|---------------|
| 0001 | Tanks | 20-gallon | | |
| 0002 | Tanks | 50-gallon | | |
| 0004 | Tanks | 100-gallon | | |
| 0005 | Tanks | 200-gallon | | |
| 0012 | Tanks | 400-gallon | | |
| 0006 | Thermostats | Standard | | |
| 0007 | Thermostats | Super | | |
| 0008 | Air Pumps | Standard | | |
| 0009 | Air Pumps | Super | | |
| 0010 | Filters | Air driven | | |
| 0011 | Filters | Under-gravel | | |
| 0111 | Gavel Bag | Small | | |
| 0122 | Gavel Bag | Medium | | |
| 0123 | Gavel Bag | Large | | |
| 0124 | Wood Panel | 500-panel | | |

New Table: Equipment Name Table

| <u>equipNameID</u> | equipName |
|--------------------|-------------|
| eq01 | Tanks |
| eq02 | Thermostats |
| eq03 | Air Pumps |
| eq04 | Filters |
| eq05 | Gavel Bag |
| eq06 | Wood Panel |

New Table: Equipment Type Table

| <u>equipTypeID</u> | equipType | equipQuantity | equipPrice |
|--------------------|-------------|---------------|------------|
| eqType01 | 20-gallon | | |
| eqType02 | 50-gallon | | |
| eqType03 | 100-gallon | | |
| eqType04 | 200-gallon | | |
| eqType05 | 400-gallon | | |
| eqType06 | Standard | | |
| eqType07 | Super | | |
| eqType08 | Air-driven | | |
| eqType09 | Undergravel | | |
| eqType10 | Small | | |
| eqType11 | Medium | | |
| eqType12 | Large | | |
| eqType13 | 500-panel | | |

Normalized Equipment Table

| <u>equipID</u> | <u>equipNameID</u> | <u>equipTypeID</u> |
|----------------|--------------------|--------------------|
| 0001 | eq01 | eqType01 |
| 0002 | eq01 | eqType02 |
| 0004 | eq01 | eqType03 |
| 0005 | eq01 | eqType04 |
| 0012 | eq01 | eqType05 |
| 0006 | eq02 | eqType06 |
| 0007 | eq02 | eqType07 |
| 0008 | eq03 | eqType06 |
| 0009 | eq03 | eqType07 |
| 0010 | eq04 | eqType08 |
| 0011 | eq04 | eqType09 |
| 0111 | eq05 | eqType10 |
| 0122 | eq05 | eqType11 |
| 0123 | eq05 | eqType12 |
| 0124 | eq06 | eqType13 |

Install Table

| <u>installID</u> | <u>CustomerID</u> | installType | <u>staffID</u> | <u>facilityID</u> | <u>startDate</u> | <u>endDate</u> |
|------------------|-------------------|------------------------|------------------------|-------------------|------------------|----------------|
| 234 | 01 | Freshwater Tropical | SHA88 SHA2 SHA77 | | 1-09- 2012 | |

| | | | | | | |
|-----|----|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|--|----------------|---------------|
| 654 | 02 | Freshwater Cold | SHA88 SHA99 SHA10 SHA12 SHA13 SHA8 SHA66 SHA23 SHA15 SHA16 SHA17 | | 1-06- 2005 | 1-06- 2012 |
| 767 | 03 | Marine | SHA88 SHA99 SHA10 SHA12 SHA13 SHA18 SHA19 SHA20 SHA21 SHA22 SHA8 SHA9 SHA11 SHA23 SHA15 SHA16 SHA17 | | 30-06- 2012 | |

| | | | | | | |
|-----|----|---------------------|-------|--|-----------|-----------|
| 943 | 04 | Marine | | | 1-09-2005 | 1-09-2012 |
| 157 | 03 | Freshwater Tropical | SHA66 | | | |

New Table: Installation Type Table

| <u>installTypeID</u> | installType |
|----------------------|---------------------|
| insTy01 | Freshwater Tropical |
| insTy02 | Freshwater Cold |
| insTy03 | Marine |

New Table: Installation Employee Table

| <u>installEmployeeID</u> | staffID | instalID |
|--------------------------|---------|----------|
| insEpm01 | SHA88 | 234 |
| insEpm02 | SHA2 | 234 |
| insEpm03 | SHA77 | 234 |
| insEpm04 | SHA88 | 654 |
| insEpm05 | SHA99 | 654 |
| insEpm06 | SHA10 | 654 |
| insEpm07 | SHA12 | 654 |
| insEpm08 | SHA13 | 654 |
| insEpm09 | SHA8 | 654 |
| insEpm10 | SHA66 | 654 |
| insEpm11 | SHA23 | 654 |
| insEpm12 | SHA15 | 654 |

| | | |
|----------|-------|-----|
| insEpm13 | SHA16 | 654 |
| insEpm14 | SHA17 | 654 |
| insEpm15 | SHA88 | 767 |
| insEpm16 | SHA99 | 767 |
| insEpm17 | SHA10 | 767 |
| insEpm18 | SHA12 | 767 |
| insEpm19 | SHA13 | 767 |
| insEpm20 | SHA18 | 767 |
| insEpm21 | SHA19 | 767 |
| insEpm22 | SHA20 | 767 |
| insEpm23 | SHA21 | 767 |
| insEpm24 | SHA22 | 767 |
| insEpm25 | SHA8 | 767 |
| insEpm26 | SHA9 | 767 |
| insEpm27 | SHA11 | 767 |
| insEpm28 | SHA23 | 767 |
| insEpm29 | SHA15 | 767 |
| insEpm30 | SHA16 | 767 |
| insEpm31 | SHA17 | 767 |
| insEmp32 | SHA66 | 157 |

New Table: InstallID Table

| <u>installID</u> | <u>installTypeID</u> |
|------------------|----------------------|
| 234 | insTy01 |
| 654 | insTy02 |
| 767 | insTy03 |
| 943 | insTy03 |
| 157 | insTy01 |

New Table: FinalInstallation Table

| <u>insFinalID</u> | <u>customerID</u> | <u>installEmployeeID</u> | <u>instStartDate</u> | <u>instEndDate</u> |
|-------------------|-------------------|--------------------------|----------------------|--------------------|
| ins01 | 01 | insEpm01 | stdt01 | |
| ins02 | 01 | insEpm02 | stdt01 | |
| ins03 | 01 | insEpm03 | stdt01 | |
| ins04 | 02 | insEpm04 | stdt02 | endt01 |
| ins05 | 02 | insEpm05 | stdt02 | endt01 |
| ins06 | 02 | insEpm06 | stdt02 | endt01 |
| ins07 | 02 | insEpm07 | stdt02 | endt01 |
| ins08 | 02 | insEpm08 | stdt02 | endt01 |
| ins09 | 02 | insEpm09 | stdt02 | endt01 |
| ins10 | 02 | insEpm10 | stdt02 | endt01 |
| ins11 | 02 | insEpm11 | stdt02 | endt01 |
| ins12 | 02 | insEpm12 | stdt02 | endt01 |
| ins13 | 02 | insEpm13 | stdt02 | endt01 |
| ins14 | 02 | insEpm14 | stdt02 | endt01 |
| ins15 | 03 | insEpm15 | stdt03 | |
| ins16 | 03 | insEpm16 | stdt03 | |
| ins17 | 03 | insEpm17 | stdt03 | |
| ins18 | 03 | insEpm18 | stdt03 | |
| ins19 | 03 | insEpm19 | stdt03 | |
| ins20 | 03 | insEpm20 | stdt03 | |
| ins21 | 03 | insEpm21 | stdt03 | |
| ins22 | 03 | insEpm22 | stdt03 | |
| ins23 | 03 | insEpm23 | stdt03 | |
| ins24 | 03 | insEpm24 | stdt03 | |
| ins25 | 03 | insEpm25 | stdt03 | |
| ins26 | 03 | insEpm26 | stdt03 | |
| ins27 | 03 | insEpm27 | stdt03 | |
| ins28 | 03 | insEpm28 | stdt03 | |
| ins29 | 03 | insEpm29 | stdt03 | |

| | | | | |
|-------|----|----------|--------|--------|
| ins30 | 03 | insEpm30 | stdt03 | |
| ins31 | 03 | insEpm31 | stdt03 | |
| ins32 | 03 | insEmp32 | | |
| ins33 | 04 | | stdt04 | endt02 |

1.1.6.2 Third normal form (3NF)-

- i. Eliminate fields that do not depend on the key
- ii. Remove transitive dependencies into a new relation.

New Table: Installation Start date table

| <u>instStartDate</u> | <u>startDate</u> |
|----------------------|------------------|
| stdt01 | 1-09- 2012 |
| stdt02 | 1-06-2005 |
| stdt03 | 30-06-2012 |
| stdt04 | 1-09-2005 |

New Table: Installation End date table

| <u>instEndDate</u> | <u>endDate</u> |
|--------------------|----------------|
| endt01 | 1-06-2012 |
| endt02 | 1-09-2012 |

Already Normalized Tables:

Payment table

| <u>payID</u> | <u>payAmount</u> | <u>paymentDate</u> | <u>customerID</u> | <u>staffID</u> |
|--------------|------------------|--------------------|-------------------|----------------|
| 333 | | | 01 | |
| 555 | | | 02 | |
| 777 | | | 03 | |
| 999 | | | 04 | |
| 111 | | | 03 | |

CustomerContact

| <u>customerContactID</u> | contactNumber |
|--------------------------|---------------|
| | |
| | |
| | |

Facility Table

| <u>facilityID</u> | facilityType |
|-------------------|--------------|
| | |
| | |

Activity 1.2

1.2.0 GUI for PollyPIPE system

Login Form of PollyPIPE System



Figure 3 Login GUI

Customer Details Form

A screenshot of a Windows-style application window titled "CUSTOMER DETAILS". The title bar has standard minimize, maximize, and close buttons. The main area is titled "Edit Customer Details" and contains a grid of form fields. The fields are grouped into two columns. The left column includes "Customer ID" (dropdown), "First Name" (text), "Last Name" (text), "Contact Number ID" (dropdown), "Contact Number" (text), and "Payment ID" (dropdown). The right column includes "Address ID" (dropdown), "Address Number" (text), "House Name" (text), "Street Name" (text), "Post Code" (text), "Post Town ID" (dropdown), and "Post Town" (text). At the bottom are four buttons: "INSERT", "UPDATE", "DELETE", and "BACK".

Figure 4 Customer Details GUI

Payment & Staff Details Form

The screenshot shows a Windows application window titled "DETAILS". The interface is divided into two main sections: "Payment Details" and "Staff Details".

- Payment Details:** Contains fields for "Payment ID" (dropdown), "Pay Amount" (text input), "Payment Date" (date picker showing "4/15/2022"), "Customer ID" (text input), and "Staff ID" (text input).
- Staff Details:** Contains fields for "Staff ID" (dropdown), "Staff TypeID" (text input), "Employee First Name" (text input), "Employee Last Name" (text input), "Staff TypeID" (text input), and "Staff Type" (dropdown).
- Buttons:** At the bottom are four green buttons labeled "INSERT", "UPDATE", "DELETE", and "BACK".

Figure 5 Payment & Staff Details GUI

Equipment Details Form

The screenshot shows a Windows application window titled "Equipment Details". The interface is divided into several sections:

- Equipment Name and Type Details:** Contains fields for "Equipment NameID" (dropdown), "Equipment Name" (text input), "Equipment TypeID" (text input), "Equipment Type" (dropdown), "Equipment Quantity" (text input), and "Price" (text input).
- EquipmentDetails:** Contains fields for "EquipID" (text input), "Equipment NameID" (text input), and "Equipment TypeID" (text input).
- Installation Equipment Details:** Contains fields for "Install Equipment ID" (text input), "EquipID" (text input), and "Installation ID" (text input).
- Installation Date Details:** Contains fields for "Start Date ID" (text input), "Start Date" (date picker showing "4/15/2022"), "End DateID" (text input), and "Start Date" (date picker showing "4/15/2022").
- Type of Installions Details:** Contains fields for "Install Type ID" (text input), "Name of Type" (text input), "Installation ID" (text input), and "Install Type ID" (text input).
- Buttons:** At the bottom are four green buttons labeled "INSERT", "UPDATE", "DELETE", and "BACK".

Figure 6 Equipment Details From

Installation Employee Details Form

| Installation Employee Details | |
|-------------------------------|----------------------|
| Installation Employee ID | <input type="text"/> |
| Staff ID | <input type="text"/> |
| Installation ID | <input type="text"/> |

| Final Installation Details | |
|----------------------------|----------------------|
| finalInstall ID | <input type="text"/> |
| Customer ID | <input type="text"/> |
| Install Employee ID | <input type="text"/> |
| Install Start Date ID | <input type="text"/> |
| Install End Date ID | <input type="text"/> |

Buttons: INSERT, UPDATE, DELETE, BACK

Figure 7 Installation Employee Details form

Main Menu Form



Figure 8 System Main Menu

In the given scenario Polly Pipe company must collect customer data, and their requirements, staff details, equipment details, and payment details. Above ER Diagram and Logical database schema is the first step to identifying user and system requirements like this. Therefore, the above design mentioned all the database tables and data dictionaries for this system. In ERD mentioned the Customer table, Staff table, Installation table, Payment table, and Equipment table for the store of all the data saving it in the database. According to the design of the database, all information is stored in one place, and it will be the easiest way to control.

For example, Customers can give their first name, last name, address details, contact number, and installation requirements to Polly Pipe. Then the system can store all the customer details under the Customer Details table. In the installation, the system can store the installation start date, and the end date, using equipment details, and employee details under the Installation table. It has particular fields for that. This database can manage all the equipment details like their prices, and quantities under the Equipment table. Above mentioned GUI can control all functions and it is very user-friendly for the company.

The above design mentioned describes and represents the system requirements to retrieve the effectiveness and the efficiency of the database.

Activity 2

2.0 Data Definition Language

Data definition language (DDL) statements in SQL are used to create schema and to define the type and structure of the data that will be stored in a database.

1. CREATE – This query is used to create a database, tables
1. ALTER – This command is used to modify the structure of an already existing table.
2. DROP – This command is used to delete an existing database or an object within a database

2.1 CREATE

Creating a database for PollyPIPE

The screenshot shows the SSMS interface. On the left is the Object Explorer pane, which displays the 'DATABASES' node under 'DESKTOP-UO6IKBR, <default>'. Inside 'DATABASES', there are several databases listed: System Databases, Ayubo, DWDiagnostics, Esoft, and PollyPipe. Under 'PollyPipe', a new database named 'PollyPIPEcompany' is shown, highlighted with a red box. On the right is the 'SQLQuery_1.sql' query editor window. It contains the following SQL code:

```
CREATE DATABASE PollyPIPEcompany;
```

The line 'CREATE DATABASE PollyPIPEcompany;' is highlighted with a red box. Below the code, the 'Messages' pane shows the execution results:

```
12:37:01 AM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.530
```

Figure 9 Example of create statement 0

Creating Customer Details data table for PollyPipe

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the 'Tables' node under the 'PollyPipe' database. A red arrow points from the 'Tables' node towards the query window. The query window contains the following SQL code:

```

USE PollyPipe;
GO
CREATE TABLE Customer(
    customerID varchar (20) PRIMARY KEY,
    firstName varchar (20),
    lastName varchar (20),
    customerContactID int FOREIGN KEY REFERENCES CustomerContact (customerContactID),
    cusAddressID int FOREIGN KEY REFERENCES CustomerAddress (cusAddressID),
    payID int FOREIGN KEY REFERENCES Payment (payID)
);
GO
CREATE * DDL TRIGGER.

```

The results pane shows the table structure with columns: customerID, firstName, lastName, customerContactID, cusAddressID, and payID.

Figure 10 Example of Create statement 1

Creating Customer Contact Details table for PollyPipe

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the 'Tables' node under the 'PollyPipe' database. A red arrow points from the 'Tables' node towards the query window. The query window contains the following SQL code:

```

66
67
68
69 CREATE TABLE CustomerContact(
70     customerContactID int PRIMARY KEY,
71     contactNumber int
72 );
73
74
75
76
77

```

The results pane shows the table structure with columns: customerContactID and contactNumber.

Figure 11 Example of create statement 2

Creating Customer Address Details table for PollyPipe

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the 'Tables' node under the 'PollyPipe' database. A red arrow points from the 'Tables' node towards the query window. The query window contains the following SQL code:

```

91
92 CREATE TABLE CustomerAddress(
93     cusAddressID int PRIMARY KEY,
94     addressNumber varchar (10),
95     houseName varchar (50),
96     streetName varchar (50),
97     postCode varchar (20),
98     postTownID int FOREIGN KEY REFERENCES PostTown (postTownID)
99 );
100
101

```

The results pane shows the table structure with columns: cusAddressID, addressNumber, houseName, streetName, postCode, and postTownID.

Figure 12 Example of create statement 3

Creating Customer Post Town Details table for PollyPipe

The screenshot shows the Object Explorer on the left with a tree view of tables under 'Tables'. A red arrow points from the 'Tables' node towards the 'PostTown' table. The 'PostTown' table is highlighted with a red box. The 'Script Selection' context menu is open over the table, with the 'CREATE TABLE' option selected and highlighted with a blue box. The 'Results' tab is active in the bottom pane, showing the generated CREATE TABLE script:

```

111  CREATE TABLE PostTown(
112      postTownID int PRIMARY KEY,
113      postTown varchar(50)
114  );
115
116
117
118
119
120
121
122
123
124
125
126

```

The results pane shows the table structure with columns: postTownID and postTown.

Figure 13 Example of create statement 4

Creating Payment Details table for PollyPipe

The screenshot shows the Object Explorer on the left with a tree view of tables. A red arrow points from the 'Tables' node towards the 'Payment' table. The 'Payment' table is highlighted with a red box. The 'Script Selection' context menu is open over the table, with the 'CREATE TABLE' option selected and highlighted with a blue box. The 'Results' tab is active in the bottom pane, showing the generated CREATE TABLE script:

```

125
126
127
128  CREATE TABLE Payment(
129      payID int PRIMARY KEY,
130      payAmount money,
131      paymentDate date,
132      staffID varchar(20) FOREIGN KEY REFERENCES Staff (sfattID)
133  );
134
135
136
137
138
139
140

```

The results pane shows the table structure with columns: payID, payAmount, paymentDate, and staffID.

Figure 14 Example of create statement 5

Creating Staff Details table for PollyPipe

The screenshot shows the Object Explorer on the left with a red arrow pointing to the 'dbo.Staff' node. The Results pane on the right displays the CREATE TABLE statement and its results. The results table has columns: staffID, empFirstName, empSecondName, and staffTypeID.

```

139  CREATE TABLE Staff(
140  staffID varchar (20) PRIMARY KEY,
141  empFirstName varchar (50) NOT NULL,
142  empSecondName varchar (50) NOT NULL,
143  staffTypeID int FOREIGN KEY REFERENCES StaffType (staffTypeID)
144  );
145
146
147
148
149
150
151
152

```

| staffID | empFirstName | empSecondName | staffTypeID |
|---------|--------------|---------------|-------------|
| | | | |

Figure 15 Example of create statement 6

Creating Staff Type Details table for PollyPipe

The screenshot shows the Object Explorer on the left with a red arrow pointing to the 'dbo.StaffType' node. The Results pane on the right displays the CREATE TABLE statement and its results. The results table has columns: staffTypeID and staffType.

```

153  CREATE TABLE StaffType(
154  staffTypeID int PRIMARY KEY,
155  staffType varchar (50) NOT NULL
156  );
157
158
159
160
161
162
163
164
165
166

```

| staffTypeID | staffType |
|-------------|-----------|
| | |

Figure 16 Example of create statement 7

Creating EquipmentName, EquipmentType, Equipment tables for PollyPipe

The screenshot shows the Object Explorer on the left with a red box highlighting the 'Tables' node. Inside the 'Tables' node, several tables are listed, with three specifically highlighted by a red box: 'dbo.EquipmentName', 'dbo.EquipmentType', and 'dbo.Equipment'. To the right, the 'Results' tab of the query editor displays the T-SQL code for creating these three tables. The code includes primary key definitions, foreign key references to other tables like 'EquipmentType' and 'Equipment', and data types like varchar(20) and int.

```

165 CREATE TABLE EquipmentName (
166     equipNameID varchar(20) PRIMARY KEY,
167     equipName varchar (50) NOT NULL
168 );
169 CREATE TABLE EquipmentType (
170     equipTypeID varchar (20) PRIMARY KEY,
171     equipType varchar (50) NOT NULL,
172     equipQuantity int
173 );
174 CREATE TABLE Equipment(
175     equipID int PRIMARY KEY,
176     equipPrice money NOT NULL,
177     equipTypeID varchar (20) FOREIGN KEY REFERENCES EquipmentType (equipTypeID),
178     equipNameID varchar (20) FOREIGN KEY REFERENCES EquipmentName (equipNameID)
179 );
180
  
```

Figure 17 Example of create statement 8

Creating InstallEquipment, Facility, InstallationType tables for PollyPipe

The screenshot shows the Object Explorer on the left with a red box highlighting the 'Tables' node. Inside the 'Tables' node, several tables are listed, with three specifically highlighted by a red box: 'dbo.InstallEquipment', 'dbo.Facility', and 'dbo.InstallationType'. To the right, the 'Results' tab of the query editor displays the T-SQL code for creating these three tables. The code includes primary key definitions, foreign key references to other tables like 'Equipment' and 'Install', and data types like varchar(20) and int.

```

181 CREATE TABLE InstallEquipment(
182     installEquipID varchar (20) PRIMARY KEY,
183     equipID int FOREIGN KEY REFERENCES Equipment(equipID),
184     installID int FOREIGN KEY REFERENCES Install (installID)
185 );
186
187
188 CREATE TABLE Facility(
189     facilityID varchar (20) PRIMARY KEY,
190     facilityType varchar (50)
191 );
192
193 CREATE TABLE InstallationType(
194     installTypeID int PRIMARY KEY,
195     installType varchar (50)
196 );
197
198
199
  
```

Figure 18 Example of create statement 9

Creating InstallationEmployee, InstallationStartDate, InstallationEndDate tables for PollyPipe

The screenshot shows the Object Explorer on the left with a list of database tables. On the right, the Results pane displays the T-SQL code for creating three tables:

```

202 CREATE TABLE InstallationEmployee(
203     installEmpID varchar(20) PRIMARY KEY,
204     staffID varchar(20) FOREIGN KEY REFERENCES Staff (staffID)
205     installID int FOREIGN KEY REFERENCES Install (installID)
206 );
207
208 CREATE TABLE InstallStartDate (
209     instStartDate int PRIMARY KEY,
210     startDate date
211 );
212
213 CREATE TABLE InstallEndDate (
214     instEndDate int PRIMARY KEY,
215     EndDate date
216 );
217
218 );
219
220

```

The Results pane also shows a preview of the table structure with columns: installEmpID, staffID, and installID.

Figure 19 Example of create statement 10

Creating Install, FinalInstallation tables for PollyPipe

The screenshot shows the Object Explorer on the left with a list of database tables. A red box highlights the 'FinalInstallationTable' entry. On the right, the Results pane displays the T-SQL code for creating two tables:

```

223 CREATE TABLE Install(
224     installID int PRIMARY KEY,
225     installTypeID int FOREIGN KEY REFERENCES InstallationType(installTypeID)
226 );
227
228 CREATE TABLE FinalInstallationTable(
229     insFinalID varchar(20) PRIMARY KEY,
230     customerID varchar(20) FOREIGN KEY REFERENCES Customer (customerID),
231     installEmpID varchar(20) FOREIGN KEY REFERENCES InstallationEmployee (installEmpID),
232     installID int FOREIGN KEY REFERENCES Install (installID),
233     instStartDate int FOREIGN KEY REFERENCES InstallStartDate (instStartDate),
234     instEndDate int FOREIGN KEY REFERENCES InstallEndDate (instEndDate)
235 );
236
237

```

The Results pane also shows a preview of the table structure with columns: insFinalID, customerID, installEmpID, installID, instStartDate, and instEndDate.

Figure 20 Example of create statement 11

2.2 DROP DDL Statement

The screenshot shows a SQL Server Management Studio window. On the left is a tree view of databases under 'DESKTOP-UO6IKBR, <default> (Inte...'. The 'PollyPipe' database is selected and highlighted with a blue border. In the main pane, there is a code editor with the following SQL script:

```

240
241
242
243
244 DROP DATABASE PollyPIPEcompany;
245
246
247
248
249
250
251
252
253
254
255
256
257

```

Below the code editor is a 'Messages' section with the following log entries:

- 2:03:29 AM Started executing query at Line 244
- Commands completed successfully.
- Total execution time: 00:00:00.092

Figure 21 Example of DROP statement 0

2.3 ALTER DDL Statement

The screenshot shows a SQL Server Management Studio window. On the left is a tree view of tables under 'PollyPipe'. The 'dbo.CustomerAddress' table is selected and highlighted with a blue border. In the main pane, there is a code editor with the following SQL script:

```

90 CREATE TABLE CustomerAddress(
91   cusAddressID int PRIMARY KEY,
92   addressNumber varchar (10),
93   houseName varchar (50),
94   streetName varchar (50),
95   postCode varchar (20),
96
97 );
98
99
100
101
102
103 ALTER TABLE CustomerAddress
104 ADD FOREIGN KEY (postTownID) REFERENCES PostTown;
105
106
107

```

Below the code editor is a 'Results' tab showing the structure of the 'CustomerAddress' table:

| cusAddressID | addressNumber | houseName | streetName | postCode | postTownID |
|--------------|---------------|-----------|------------|----------|------------|
|--------------|---------------|-----------|------------|----------|------------|

A red arrow points to the 'Keys' node in the tree view, indicating the target for the ALTER statement.

Figure 22 Example of ALTER statement 0

2.4 Evaluate the above-mentioned GUI

Polypipe can manage this system using above mentioned GUI interfaces and these are very user-friendly and simple. In ER Diagram it displayed all data tables in the database. Therefore, above created GUI interfaces can handle all the user requirements. For example, collecting customer details and their installation requirements can be input into the system easily. It has a Login interface and when the admin needs control of this system firstly, he/she needs to use their unique username and password. If the username and password match, he/she can handle the system.

In the company, they have to store their staff details, equipment details, installation details, and customer details. This system has GUI interfaces for these particular requirements. For example, if a company needs to store their staff details, they can use GUI for that. It will be more efficient and time-saving. The above already mentioned main menu GUI has function buttons for controlling all the system requirements and user requirements and it is a centralized GUI.

2.5 Database security

Database security is where the company or owners secure databases and their objects at various levels by adopting adequate access control measures inside the database system. It has 4 categories. These are,

1. Authentication
2. Authorization
3. Data protection
4. SQL Server configurations

2.5.0 Authentication

This is the first step to connecting to the databases. It is a process to recognize that some authentic person is using a valid principal or login to connect to the database. If someone using an invalid principal, then we can say your authentication is failed and you cannot connect to the database. (Singh, 2022)

Login with valid details



Figure 23 Login with valid details



Figure 24 Main Menu

Login with invalid details

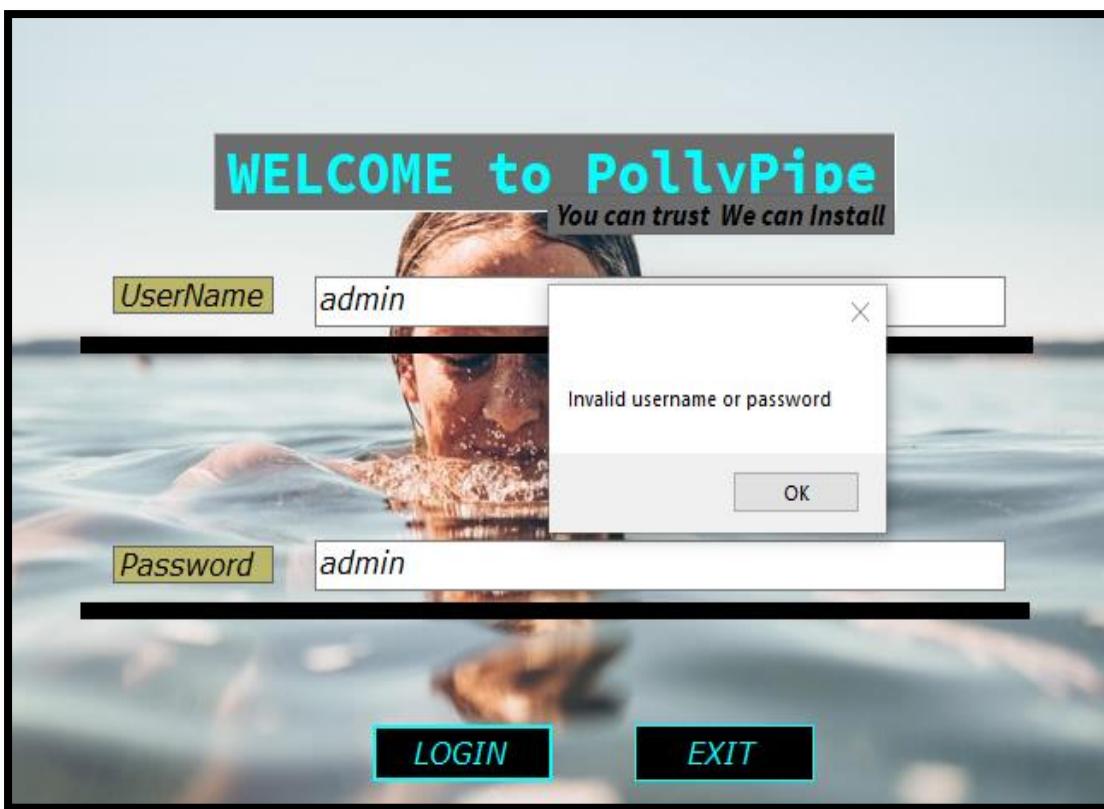


Figure 25 Login with invalid details

Login details in the database

A screenshot of a SQL Server Management Studio (SSMS) results grid. The left pane shows a tree view of database objects under 'dbo.facility'. The right pane displays a results grid with two columns: 'Username' and 'Password'. A single row is present with the values 'admin' and 'admin1234'. This grid is highlighted with a red border.

| | Username | Password |
|---|----------|-----------|
| 1 | admin | admin1234 |

Figure 26 Login details in the database

2.5.1 Authorization

This is a process to grant access to objects. Owners can restrict or authorize somebody on his principal to access that database object. Even owners can control the type of access if they don't want them to write or update anything in the database object then owners/companies can do that by denying these rights on that specified object. (Singh, 2022)

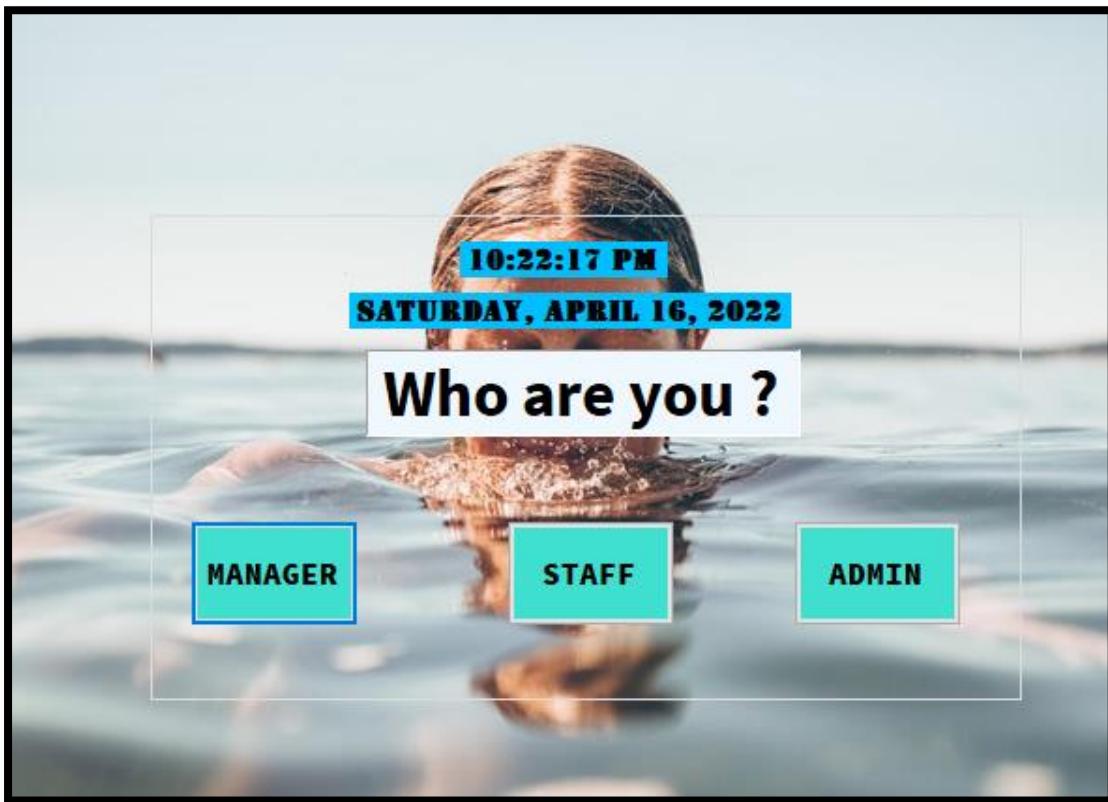


Figure 27 user authorization menu

2.5.2 Data protection

This is very popular these days to protect data using encryption from unauthorized use. It is used to encrypt our data at rest or on transit for SQL Server databases. SQL Server offers multiple options to apply encryption or masking for various types of needs like Transparent Data Encryption, Always Encryption, Data masking, etc. Database backups can also be encrypted to prevent any unauthorized access to its data. (Singh, 2022)

2.6 Data Manipulation Language (DML)

A data manipulation language (DML) is a family computer language including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables, and modifying existing data. DML is mostly incorporated in SQL databases. (techopedia, 2014)

- **SELECT** – This command is used to retrieve rows from a table. The syntax is `SELECT [column name] from [table name] where [conditions]`. This is the most widely used DML command in SQL.
- **UPDATE** – This command modifies data of one or more records. An update command syntax is `UPDATE [table name] SET [column name = value] where [condition]`.
- **INSERT** – This command adds one or more records to a database table. The insert command syntax is `INSERT INTO [table name] [column] VALUES [value(s)]`.
- **DELETE** – This command removes one or more records from a table according to specified conditions. Delete command syntax is `DELETE FROM [table name] where [condition]`. (techopedia, 2014)

2.6.0 INSERT Statement

The screenshot shows the SQL Server Management Studio interface. On the left, there's a tree view of database objects under 'PolyPIPECompany'. In the center, a query window displays the following code:

```

376 INSERT INTO CustomerAddress (cusAddressID, addressNumber, houseName, streetName, postCode, postTownID)
377 VALUES (1, 17, 'Oak House', 'Wroxton Road', 'H5667', 12),
378 (2, '', 'Bayliss House', 'Orange Street', '', 12),
379 (3, '', 'Eagirstone Castle', '', 12),
380 (4, 23, '', 'Sackville Street', 'W55', 27),
381 (5, '', 'Humberston Castle', '', 'K8', 12)
382
383

```

A red box highlights the 'VALUES' part of the statement. Below the query window, a 'Messages' pane shows the execution results:

Started executing query at Line 372
(5 rows affected)
Total execution time: 00:00:00.003

Figure 28 Example of INSERT statement

2.6.1 SELECT Statement

The screenshot shows the SQL Server Management Studio interface. On the left, the object browser shows 'Tables' under 'PolyPIPECompany'. In the center, a query window displays the following code:

```

1 SELECT * from CustomerAddress;
2
3
4
5
6
7

```

A red box highlights the 'SELECT * from CustomerAddress;' statement. Below the query window, a 'Results' pane shows the data:

| | cusAddressID | addressNumber | houseName | streetName | postCode | postTownID |
|---|--------------|---------------|-------------------|------------------|----------|------------|
| 1 | 1002 | 17 | Oak House | Wroxton Road | H5667 | 4 |
| 2 | 1003 | 0 | Bayliss House | Orange Street | | 3 |
| 3 | 1004 | 0 | Eagirstone Castle | | | 3 |
| 4 | 1005 | 23 | | Sackville Street | W55 | 4 |
| 5 | 1006 | 0 | Humberston Castle | | K8 | 3 |

Figure 29 Example of select statement 0

The screenshot shows the SQL Server Management Studio interface. On the left, the object browser shows 'Tables' under 'PolyPIPECompany'. In the center, a query window displays the following code:

```

3
4
5 SELECT * From Payment
6 WHERE staffID = 'staff3';
7
8

```

A red box highlights the 'SELECT * From Payment' statement. Below the query window, a 'Results' pane shows the data:

| | payID | payAmount | paymentDate | staffID |
|---|-------|-----------|-------------|---------|
| 1 | 10 | 50000.00 | 2012-01-09 | staff3 |
| 2 | 50 | 24560.00 | 2005-09-01 | staff3 |

Figure 30 Example of select statement 1

2.6.2 ORDER BY Statement

The screenshot shows the Object Explorer on the left with the database structure. In the center, a query window displays the following code:

```

13
14    SELECT * FROM StaffType
15    ORDER By StaffTypeID DESC;
16
17
18
19
20

```

A red box highlights the ORDER BY clause. To the right, a results grid shows the data from the StaffType table ordered by StaffTypeID in descending order:

| | staffTypeID | staffType |
|---|-------------|--------------------|
| 1 | 108 | Labourer |
| 2 | 107 | Manager |
| 3 | 106 | Brick layer |
| 4 | 105 | Carpenter |
| 5 | 104 | Aquatics installer |
| 6 | 103 | Plumber |
| 7 | 102 | cashier |
| 8 | 101 | Electrician |
| 9 | 100 | Cleaner |

Figure 31 Example of ORDER statement

2.6.3 UPDATE Statement

The screenshot shows the Object Explorer on the left with the database structure. In the center, a query window displays the following code:

```

/ 
8    UPDATE Payment
9      SET payAmount = payAmount - 1000
10     WHERE payAmount > 30000
11
12
13
14

```

A red box highlights the UPDATE statement. To the right, a results grid shows the data from the Payment table after the update:

| | payID | payAmount | paymentDate | staffID |
|---|-------|-----------|-------------|---------|
| 1 | 10 | 49000.00 | 2012-01-09 | staff3 |
| 2 | 47 | 33500.00 | 2005-01-06 | staff1 |
| 3 | 48 | 66000.00 | 2012-06-30 | staff1 |
| 4 | 50 | 24560.00 | 2005-09-01 | staff3 |
| 5 | 6 | 5555.00 | 2000-03-07 | staff1 |

Figure 32 Example of UPDATE statement

2.6.4 GROUP BY Statement

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer tree shows a database structure with several tables like InstallationType, InstallEndDate, InstallEquipment, InstallStartDate, and Payment. The 'Payment' table is selected and expanded, showing its columns (Columns), keys (Keys), constraints (Constraints), triggers (Triggers), indexes (Indexes), and statistics (Statistics). In the center, the query window contains the following SQL code:

```

11
12
13
14   SELECT payID, SUM(payAmount)
15   FROM Payment
16   GROUP BY payID
17
18

```

A red box highlights the last three lines of the query. To the right, the results pane displays a table with five rows, each showing a payID and its corresponding sum of payAmount:

| | payID | (No column name) |
|---|-------|------------------|
| 1 | 10 | 49000.00 |
| 2 | 47 | 33500.00 |
| 3 | 48 | 66000.00 |
| 4 | 50 | 24560.00 |
| 5 | 6 | 5555.00 |

Figure 33 Example of GROUP statement

2.6.5 IN Statement

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer tree shows a database structure with tables like InstallationType, InstallEndDate, InstallEquipment, InstallStartDate, and Payment. The 'Payment' table is selected and expanded, showing its columns (Columns), keys (Keys), constraints (Constraints), triggers (Triggers), indexes (Indexes), and statistics (Statistics). In the center, the query window contains the following SQL code:

```

4
5
6   SELECT * FROM
7   Staff
8   WHERE staffTypeID IN (109)
9
10
11

```

A red box highlights the last three lines of the query. To the right, the results pane displays a table with two rows, each showing staffID, empFirstName, empSecondName, and staffTypeID:

| | staffID | empFirstName | empSecondName | staffTypeID |
|---|---------|--------------|---------------|-------------|
| 1 | staff5 | John | Smith | 109 |
| 2 | staff6 | Adam | Tyler | 109 |

Figure 34 Example of IN statement

2.6.6 BETWEEN Statement

The screenshot shows the SQL Server Management Studio interface. The left pane displays a tree view of database objects under 'Servers' > 'dbo'. The 'Payment' object is selected. The right pane contains a query window and a results grid.

```

30
31
32
33
34 SELECT * FROM Payment
35 WHERE payAmount BETWEEN 40000 and 60000;
36
37

```

Results

| | payID | payAmount | paymentDate | staffID |
|---|-------|-----------|-------------|---------|
| 1 | 10 | 49000.00 | 2012-01-09 | staff3 |
| 2 | 68 | 45900.00 | 2015-09-07 | staff3 |
| 3 | 70 | 40000.00 | 2015-08-10 | staff3 |

Figure 35 Example of BETWEEN statement

2.6.7 HAVING Statement

The screenshot shows the SQL Server Management Studio interface. The left pane displays a tree view of database objects under 'Servers' > 'dbo'. The 'Payment' object is selected. The right pane contains a query window and a results grid.

```

17
18 SELECT COUNT(staffID), staffTypeID
19 FROM Staff
20 GROUP BY staffTypeID
21 HAVING COUNT(staffID) > 1;
22
23
24

```

Results

| | (No column name) | staffTypeID |
|---|------------------|-------------|
| 1 | 2 | 102 |
| 2 | 2 | 103 |
| 3 | 3 | 104 |
| 4 | 2 | 109 |

Figure 36 Example of HAVING statement

Activity 3.1

3.0 Testing the developed PollyPIPE system

| | | | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| TEST NO : 01 | TESTER : K. M. G. M. B. ALAHAKOON | DATE : 18-04-2022 | TIME : 8.57 PM |
| TEST : Login as a admin | | | |
| OUTPUT : | | | |
|  <p>WELCOME to PollyPipe You can trust We can install</p> <p>UserName : admin</p> <p>Password : admin1234</p> <p>LOGIN EXIT</p> | |  <p>Main Menu</p> <p>9:02:31 PM MONDAY, APRIL 18, 2022</p> <p>MENU</p> <ul style="list-style-type: none"> CUSTOMER DETAILS PAYMENT DETAILS STAFF DETAILS EQUIPMENT DETAILS INSTALLATION EMPLOYEE DETAILS LOG OUT <p>PollyPIPE You can trust We can install www.pollypipe.com</p> | |
| Expected Result : Logging form Correct User name and Password and Load Main menu form | | Status : Success | |

Figure 37 Test case 0

| | | | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|-------------------|----------------|
| TEST NO : 02 | TESTER : K. M. G. M. B. ALAHAKOON | DATE : 18-04-2022 | TIME : 9:05 PM |
| TEST : Login with wrong details (wrong Username and Password) | | | |
| OUTPUT : | | | |
|  <p>WELCOME to PollyPipe You can trust We can install</p> <p>UserName : andrew</p> <p>Password : andrew12345</p> <p>LOGIN EXIT</p> <p>invalid username or password</p> | | <p>OK</p> | |
| Expected Result : Display "Invalid Username or Passowrd" Message | | Status : Success | |

Figure 38 Test case 1

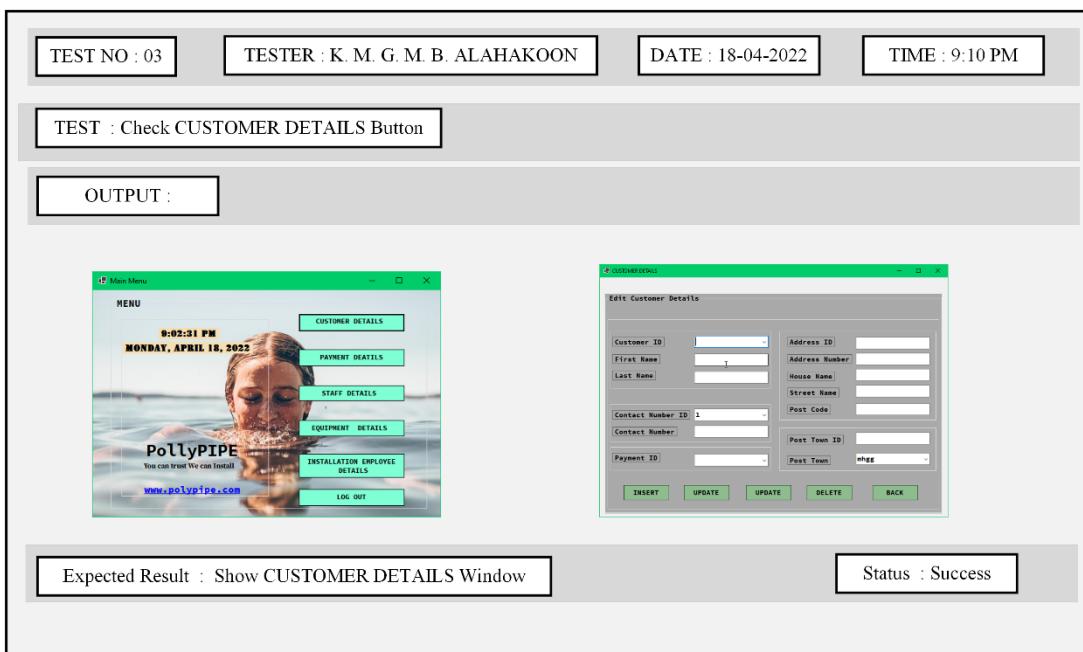


Figure 39 Test case 2

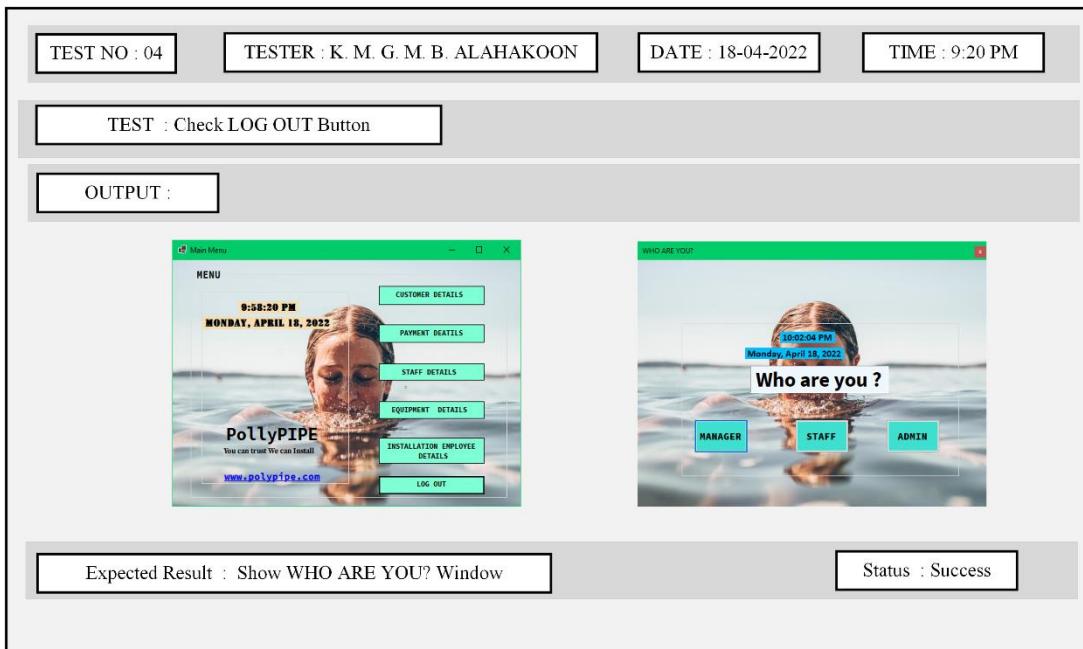


Figure 40 Test case 3

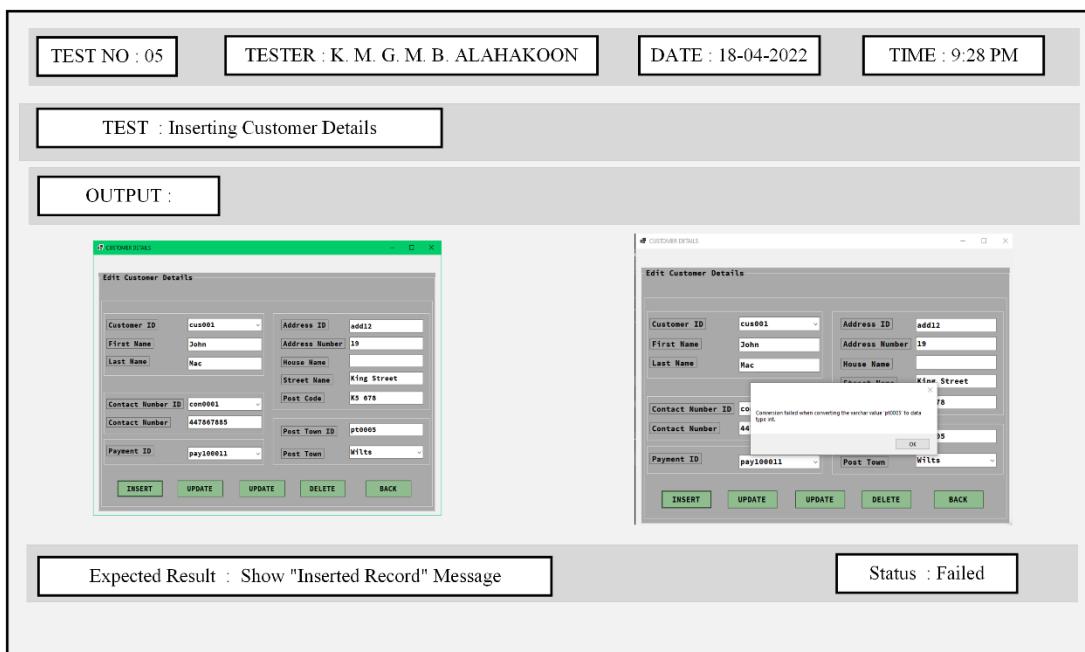


Figure 41 Test case 4

In test case 1, this test case shows how to Login into that system. The test case was passed, and it shows the Login function is working fine. It means that code has not any bugs. In test case 2, it was tested how if authorized person trying to log that system. The output was showing expected results. Therefore, we can assure that code has not any bugs and it's working correctly.

In test case 3and 4 has checked CUSTOMER DETAILS button and Log Out Button. The outputs was expected result. Therefore, also we can assure these buttons are working correctly. But in Test case 5 has failed. That case shows the output was not expected result. Then we can say it has bugs and we must do debugging that code. These test cases are very effective for final product. Without doing test cases it will be occurs lot of problems in developed systems.

These test cases results can be defined as how effectively run developed systems and it will show how that system can achieved customer requirements. Test effectiveness and test efficiency are very important to count for a software product on the market value or an asset to the customer or end user. And this a process of finding bugs in the software and make the software bug free.

Activity 3.2

3.1 Get independent feedback form using Google From

User Feedback

We would love to hear your thoughts or feedback on how we can improve your experience!

mayuraalahakoon@gmail.com [Switch accounts](#) Draft restored

*Required

Email *

Your email address _____

Considering your complete experience with our software, how likely would you be to recommend our company? 0 Very Unlikely to 10 Very Likely *

Considering your complete experience with our software, how likely would you be to recommend our company? 0 Very Unlikely to 10 Very Likely *

0
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10

Ease of installation *

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Look and feel *

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Hardware compatibility *

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Overall reliability

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Operating system compatibility *

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Collaborate with team

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Accessibility of product support

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Security *

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

The figure consists of two screenshots of a Google Form. The top screenshot shows two sections: 'Ease of use' and 'Overall performance', each with five rating options: Very Satisfied, Satisfied, Neutral, Dissatisfied, and Very Dissatisfied. The bottom screenshot shows the same two sections, followed by a comment section asking for additional comments and suggestions, a text input field for 'Your answer', and a footer with 'Submit' button, a progress bar showing 'Page 1 of 1', and links for 'Clear form', 'Never submit passwords through Google Forms.', 'Report Abuse - Terms of Service - Privacy Policy', and the 'Google Forms' logo.

Ease of use *

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Overall performance

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Overall performance

Very Satisfied
 Satisfied
 Neutral
 Dissatisfied
 Very Dissatisfied

Please use the space below for any additional comments and/or suggestions:

Your answer

Submit Page 1 of 1 Clear form

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms

Figure 42 User Feedback

3.2 Analysing feedback data

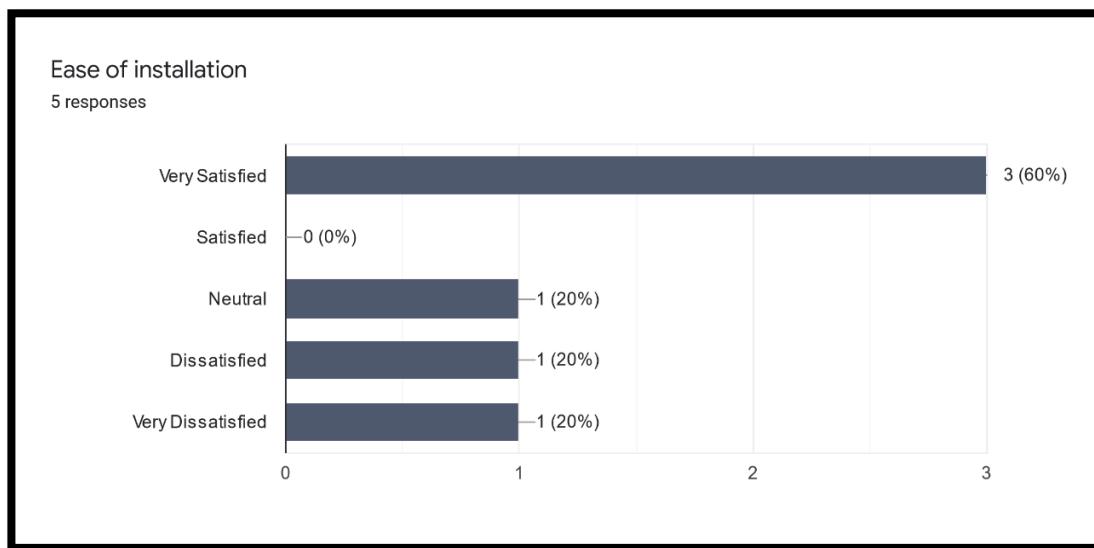


Figure 43 Analysing feedback data 0

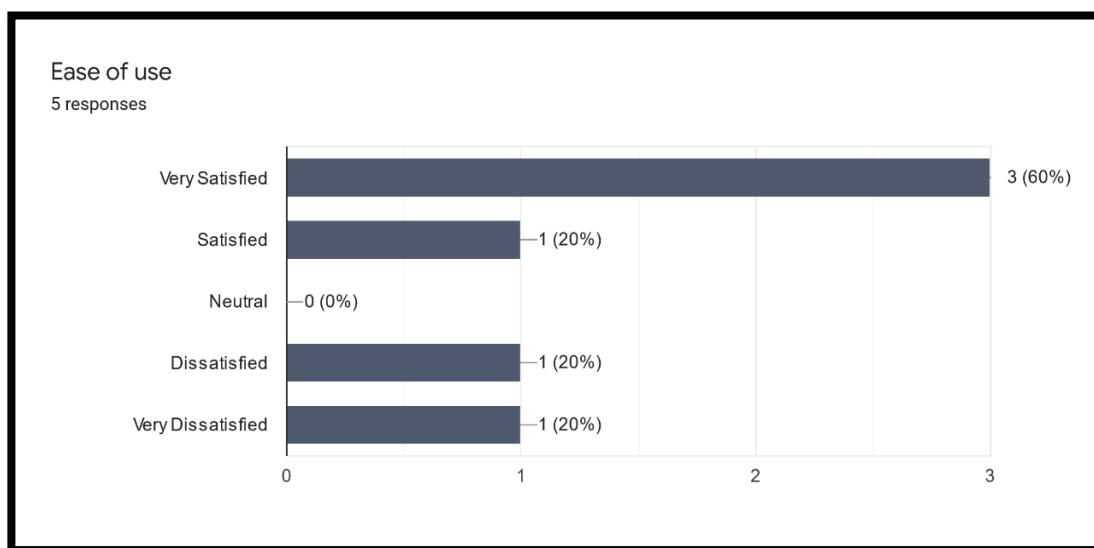


Figure 44 Analysing feedback data 1

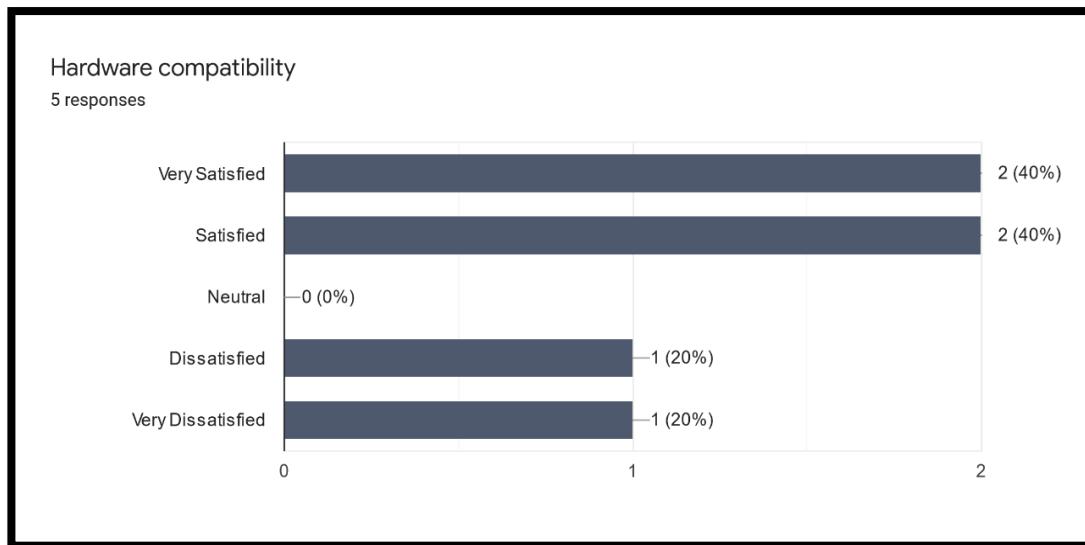


Figure 45 Analysing feedback data 2

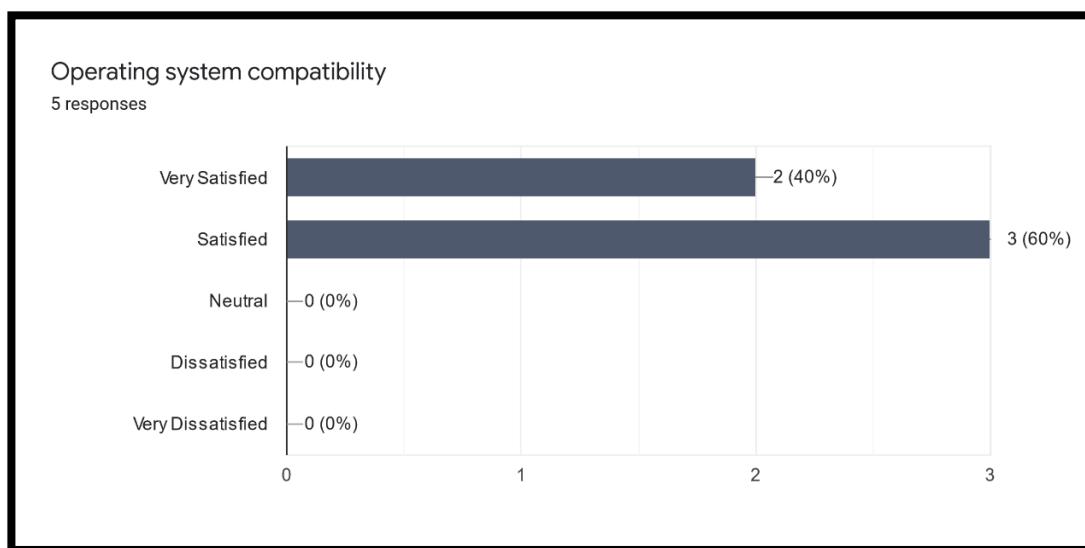


Figure 46 Analysing feedback data 3

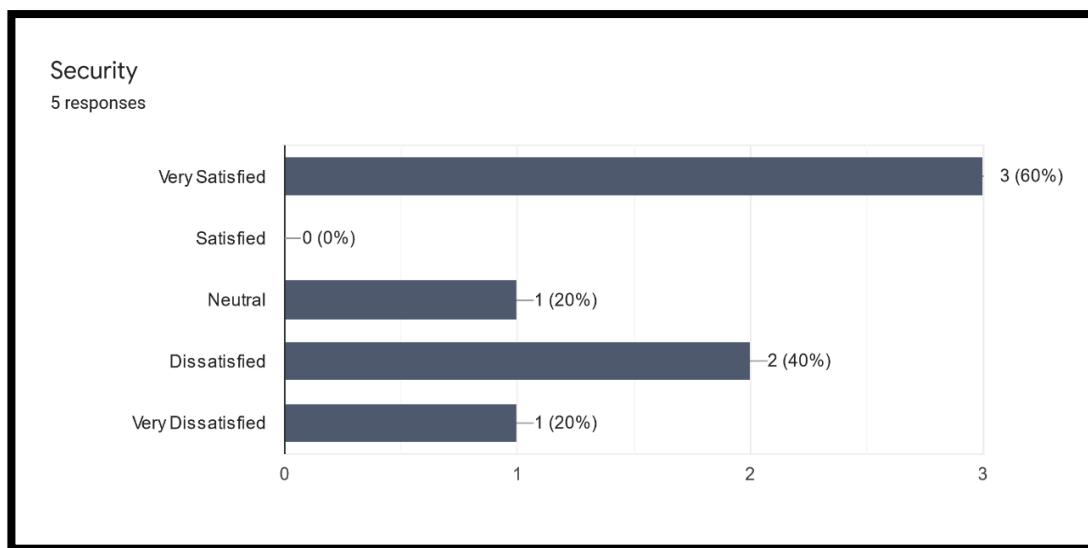


Figure 47 Analysing feedback data 4

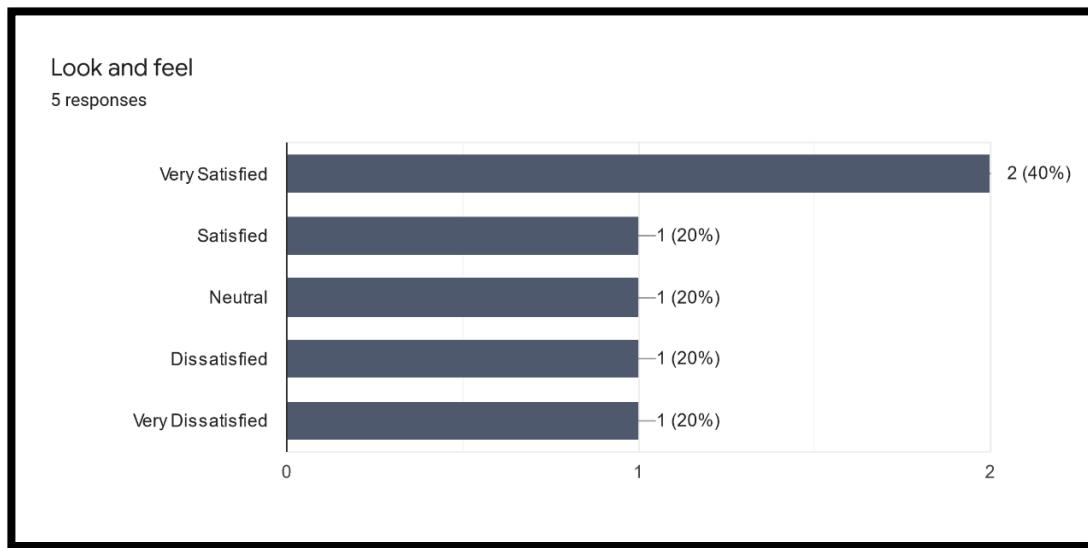


Figure 48 Analysing feedback data 5

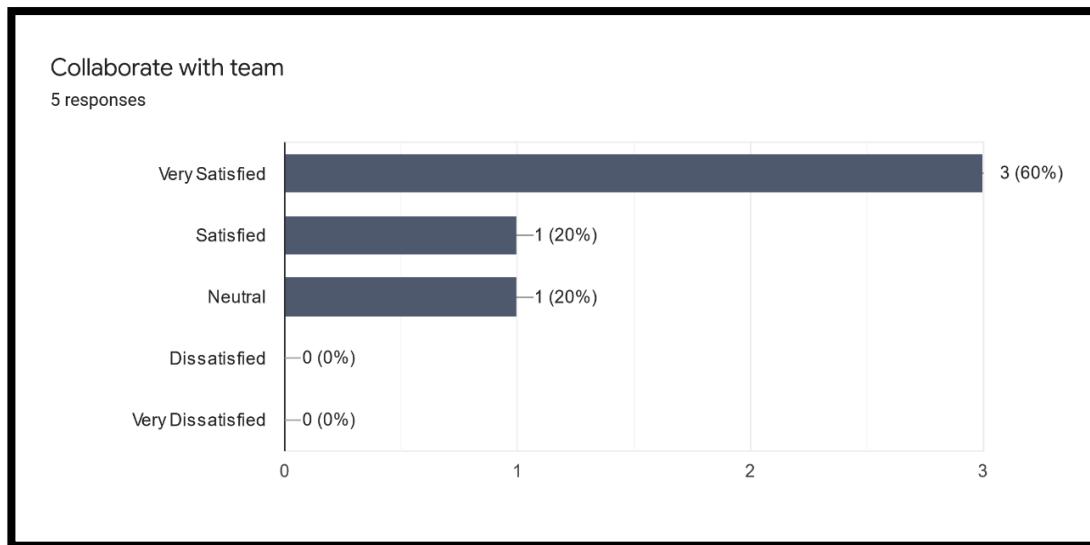


Figure 49 Analysing feedback data 6

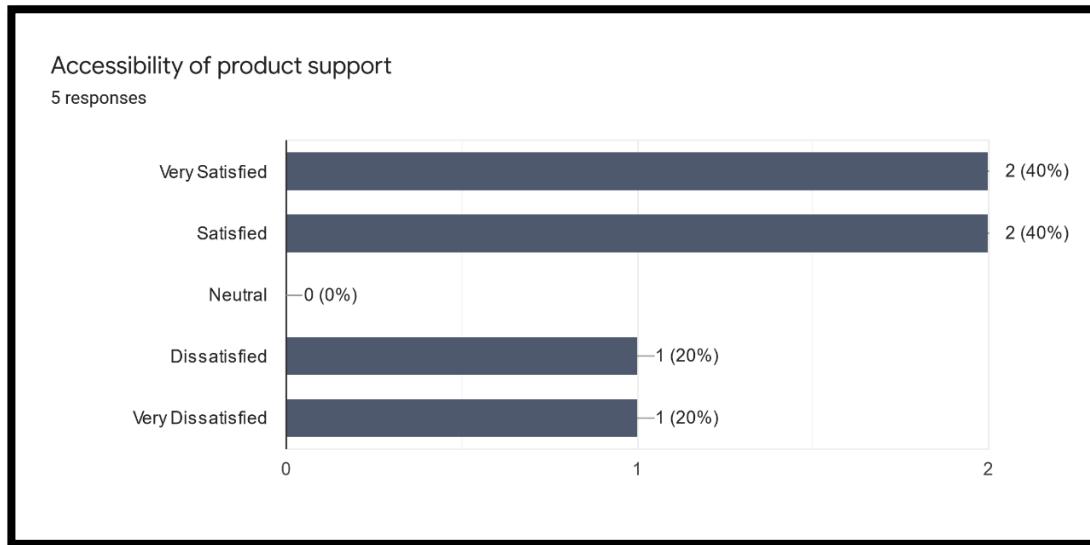


Figure 50 Analysing feedback data7

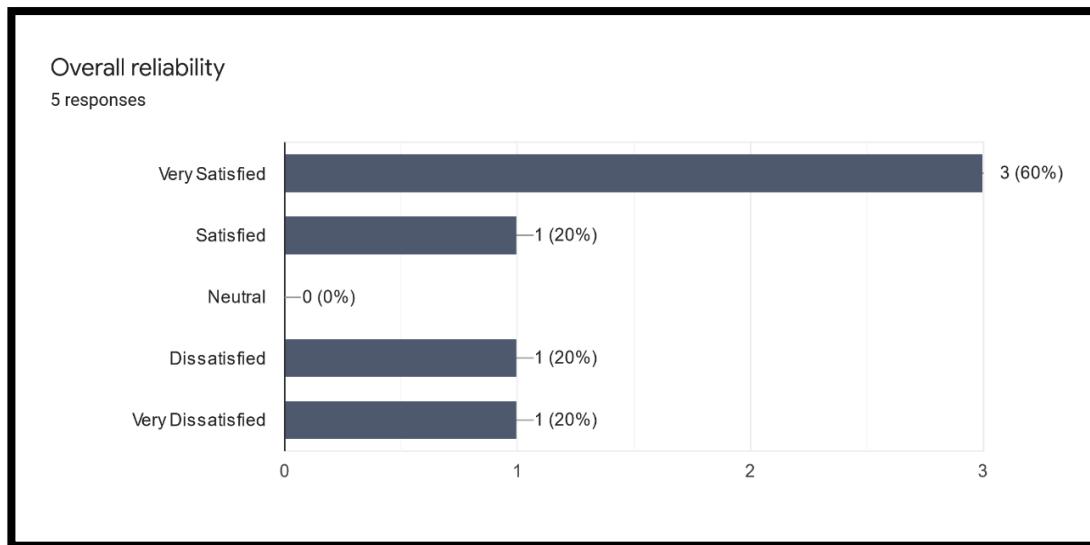


Figure 51 Analysing feedback data 8

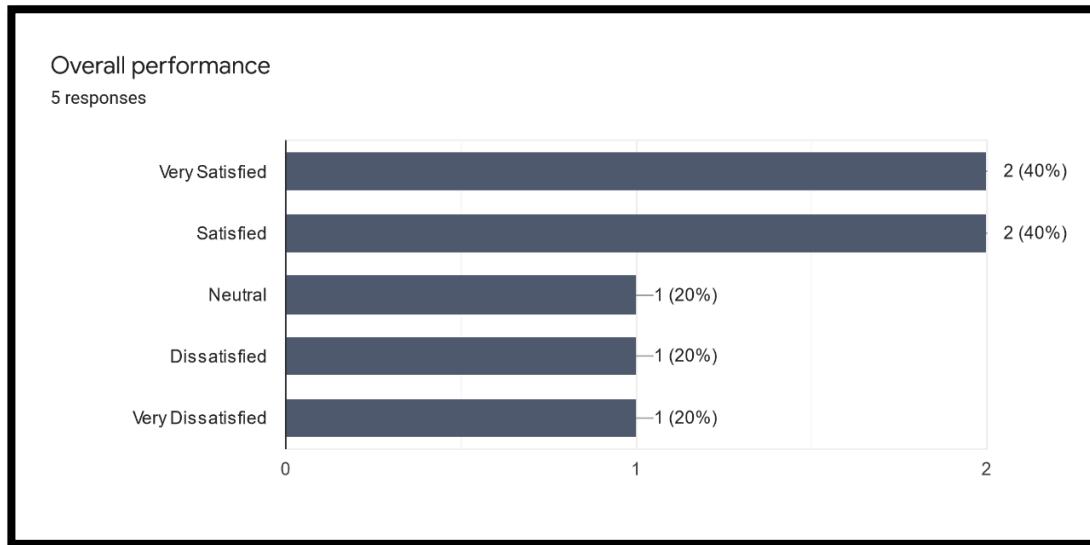


Figure 52 Analysing feedback data 9

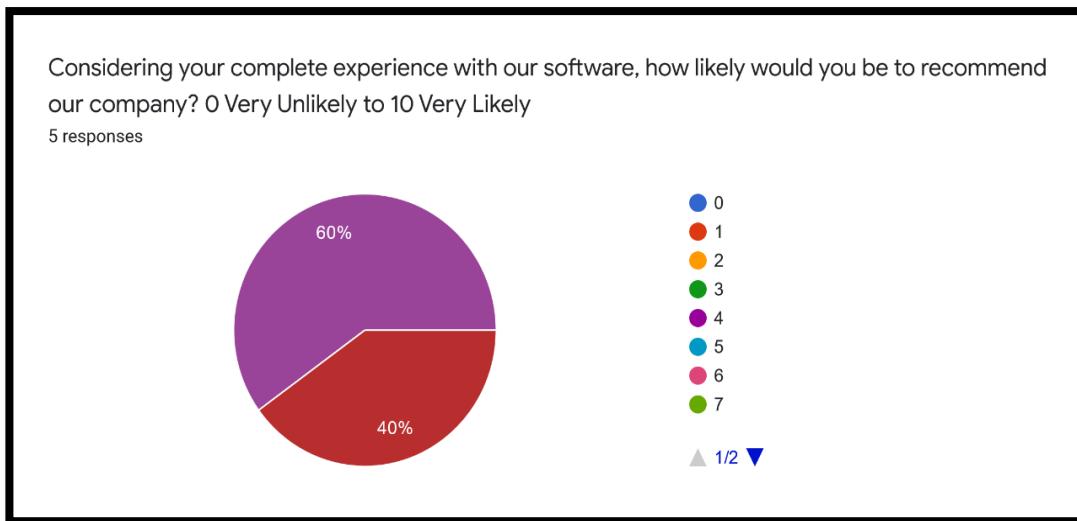


Figure 53 Analysing feedback data 10

Please use the space below for any additional comments and/or suggestions:

2 responses

Great

Improving the design will be good

Figure 54 Analysing feedback data 11

Customer feedback is information provided by clients about whether they are satisfied or dissatisfied with a product or service and the general experience they had with a product. Customer opinion is a resource for improving customer experience and adjusting your actions to their need. (Niepewna, n.d.)

In the above feedback form, one person said our system design needs to improve. As a developer of this PollyPIPE system, a developer must consider that feedback and all the other improvements that come with the next version of that system.

Activity 4

4.0 Technical Documentation

Technical documentation refers to any document that explains the use, functionality, creation, or architecture of a product. (MacKay, 2018)

The main system functional requirements of the Polly Pipe system.

- A user must have to inserting, updating, and deleting all the details into the database
- The system must have to GUI
- A user must have done the authentication process
- A user must have to do calculations (payments, salaries, etc...)
- A user must have to manage all the staff and company details
- A user should be able to check the installing process status
- A user should be able to get a print outing

These main requirements are included in this system. The company can easily install this system on their devices and control it. Below mentioned user guide can use for training their employees. Because all the employees are not technical people.

4.1 User guide for developed Polly PIPE system

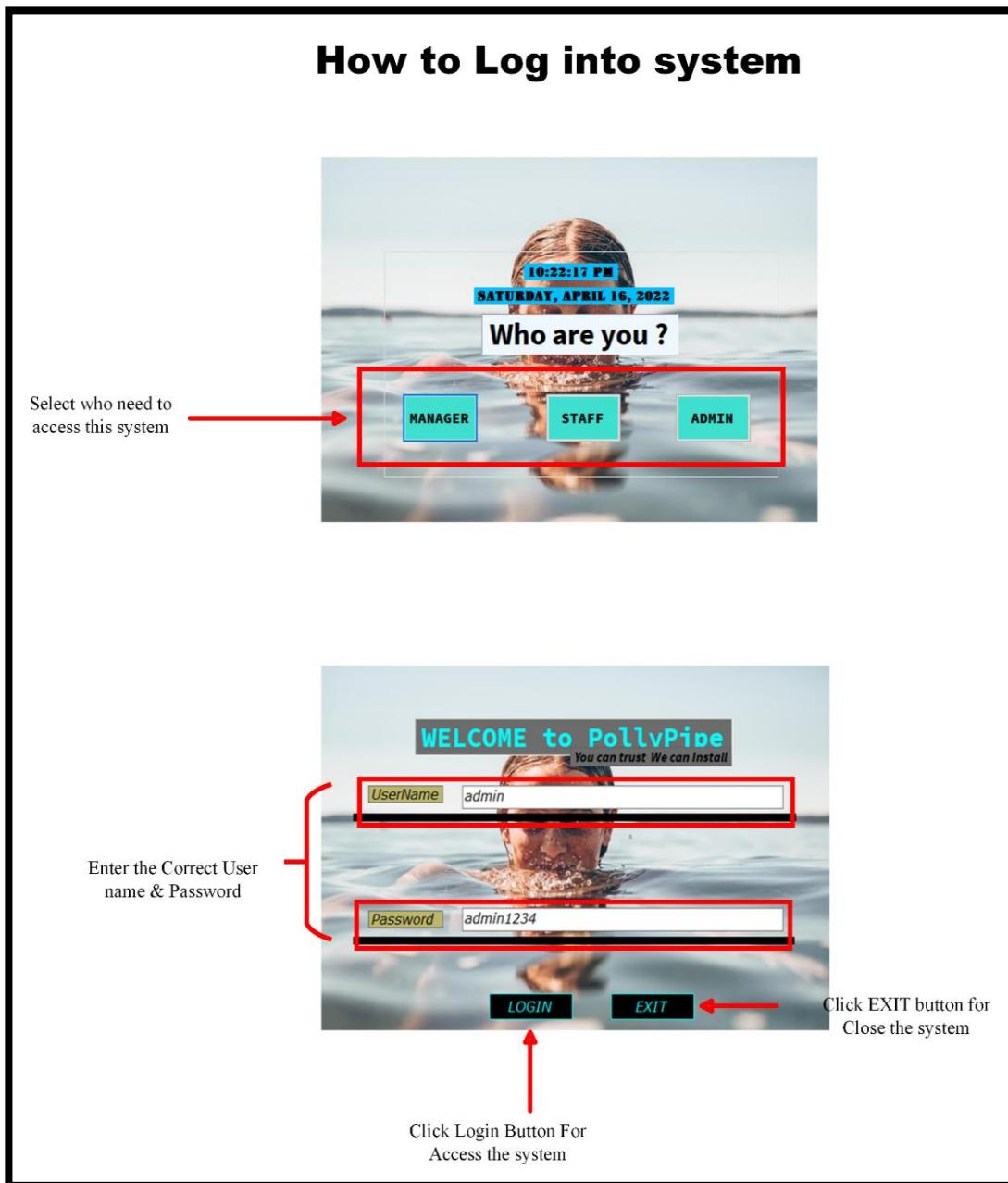


Figure 55 User guide 0

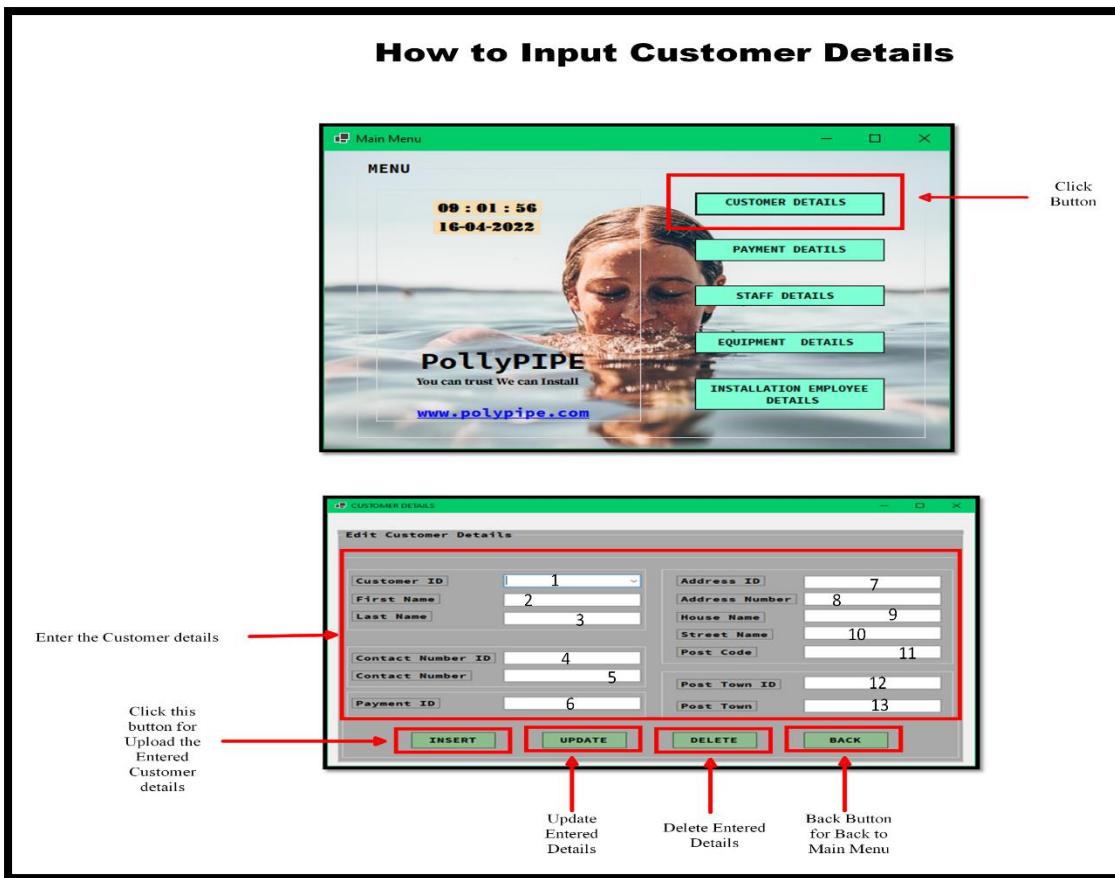


Figure 56 User guide 1

- 1 – Input the customer ID (Every customer has unique Customer ID)
- 2- Input the customer first name
- 3- Input the customer second name
- 4- Input the customer Contact Number ID (Every customer has unique customer contact id)
- 5- Input the customer Contact Number
- 6- Input the Customer Payment ID (Every customer has unique customer Payment ID)
- 7- Input the Customer Address ID (Every customer has unique Customer AddressID)
- 8- Input the Customer Address Number
- 9- Input the Customer House Name
- 10 – Input the Customer Street Name
- 11- Input the Customer Post Code
- 12- Input the Customer Post Town ID (Every customer has unique Post town ID)
- 13- Input the customer Post Town

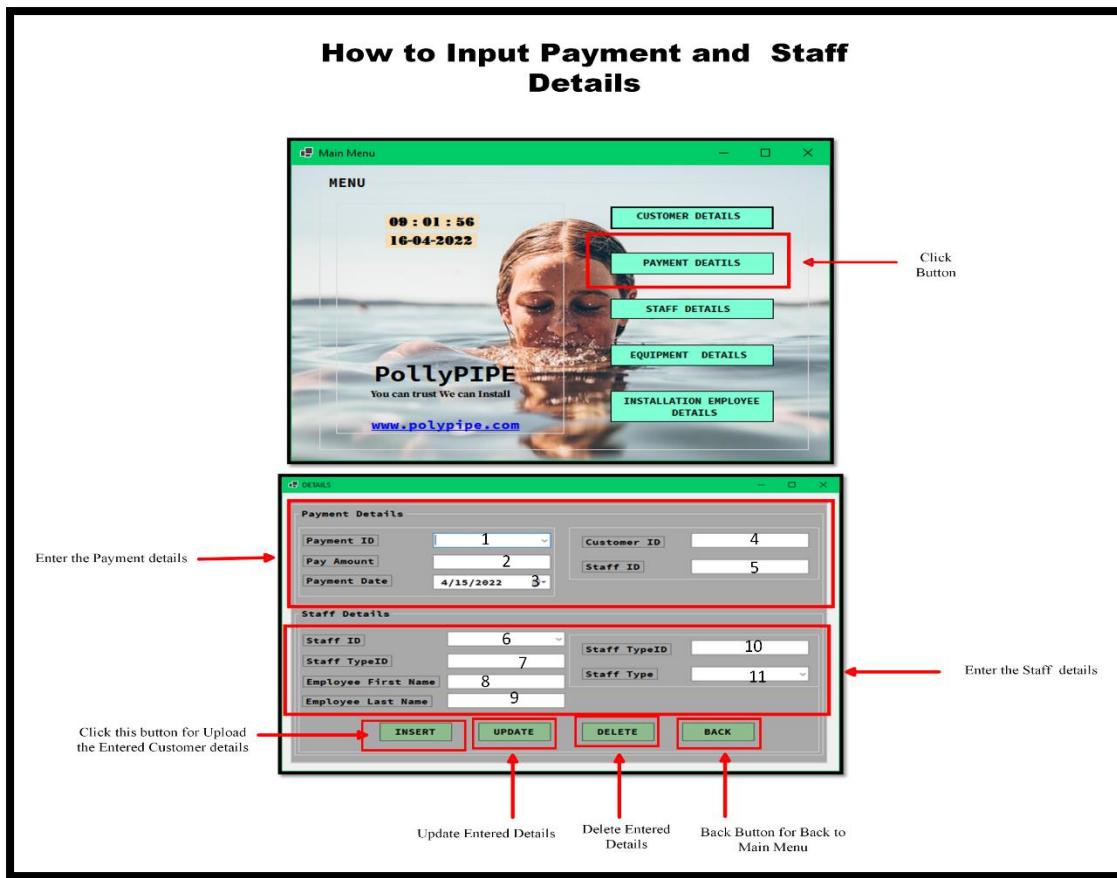


Figure 57 User guide 2

- 1 – Input the Payment ID (Every Payment has unique ID)
- 2- Input the Pay amount
- 3- Input the Payment Date
- 4- Input the Customer ID
- 5- Input the involved staff ID
- 6- Input the Staff ID (Every Staff has Unique ID)
- 7- Input the Staff Type ID
- 8- Input the Staff Employee first name
- 9- Input the Staff Employee last name
10. Input the Staff Type ID
11. Input the Staff Type name

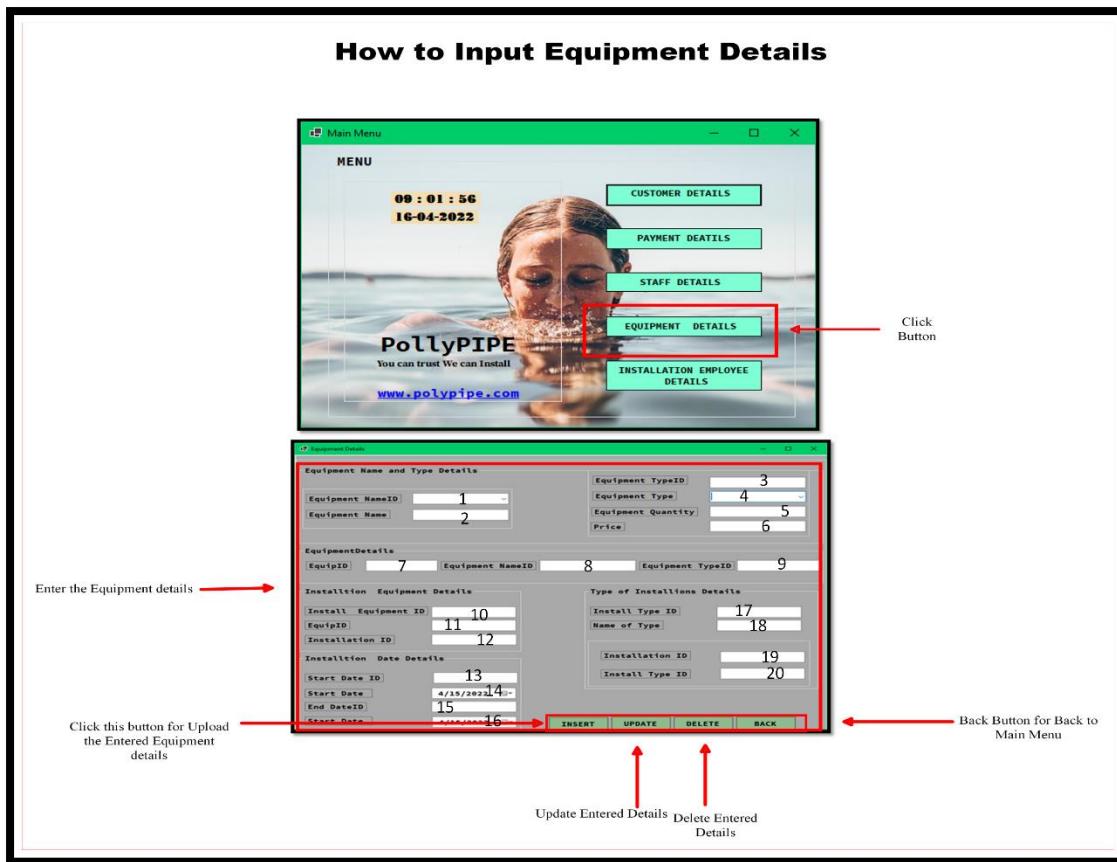


Figure 58 User guide 2

- 1- Input the Equipment Name Id (Every Equipment name has Unique ID)
- 2- Input the Equipment Name
- 3- Input the Equipment Type ID (Every Equipment Type has Unique ID)
- 4- Input the Equipment Type
- 5- Input the Equipment Quantity
- 6- Input the Price of each equipment
- 7- Input the EquipID (Each of Equipment has Unique ID)
- 8- Input the Equipment Name ID
- 9- Input the Equipment Type ID
- 10- Input the Installation Equipment ID (Unique ID)
- 11- Input the Equip ID
- 12- Input the Installation ID
- 13- Input the Installation Start Date ID (Unique ID)
- 14- Select Starting Date
- 15- Input the Installation Ending Date ID (Unique ID)

- 16- Select Installation Ending Date
- 17- Input the Installation type ID (Unique Id)
- 18- Input the Name of Type
- 19- Input the Installation ID
- 20- Input the Installation Type ID

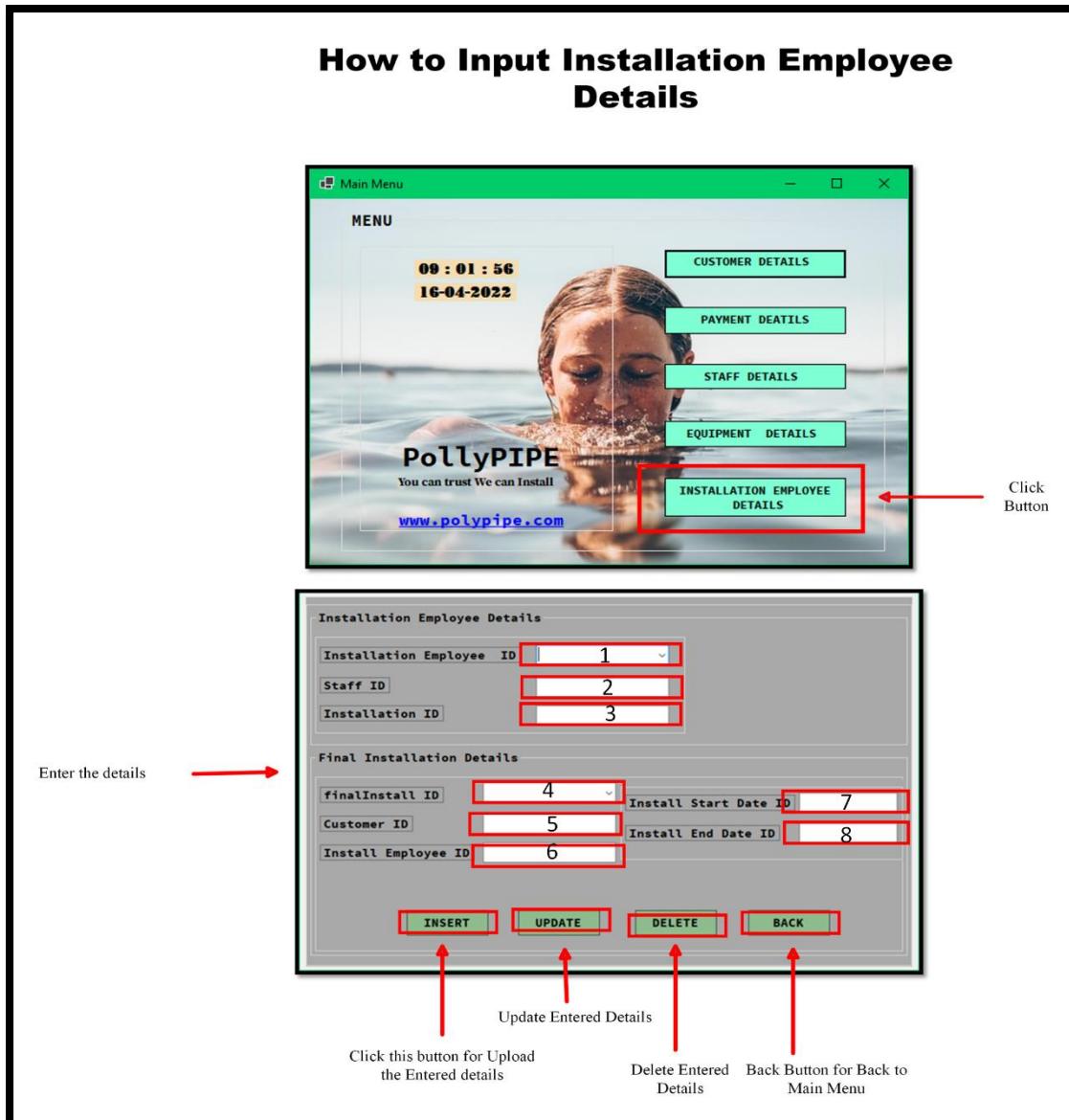


Figure 59 User guide 3

- 1 – Input the Installation Employee ID (Every Employee has unique Install Employee ID)
- 2- Input the Staff ID that Employee has
- 3- Input the Installation ID that Employee must work
- 4- Input the final Installation ID
- 5- Input the Customer ID
- 6- Input the Install Employee ID
- 7- Input the Installation Start Date ID
- 8- Input the Installation End Date ID

4.2 A use case diagram for the PollyPIPE system

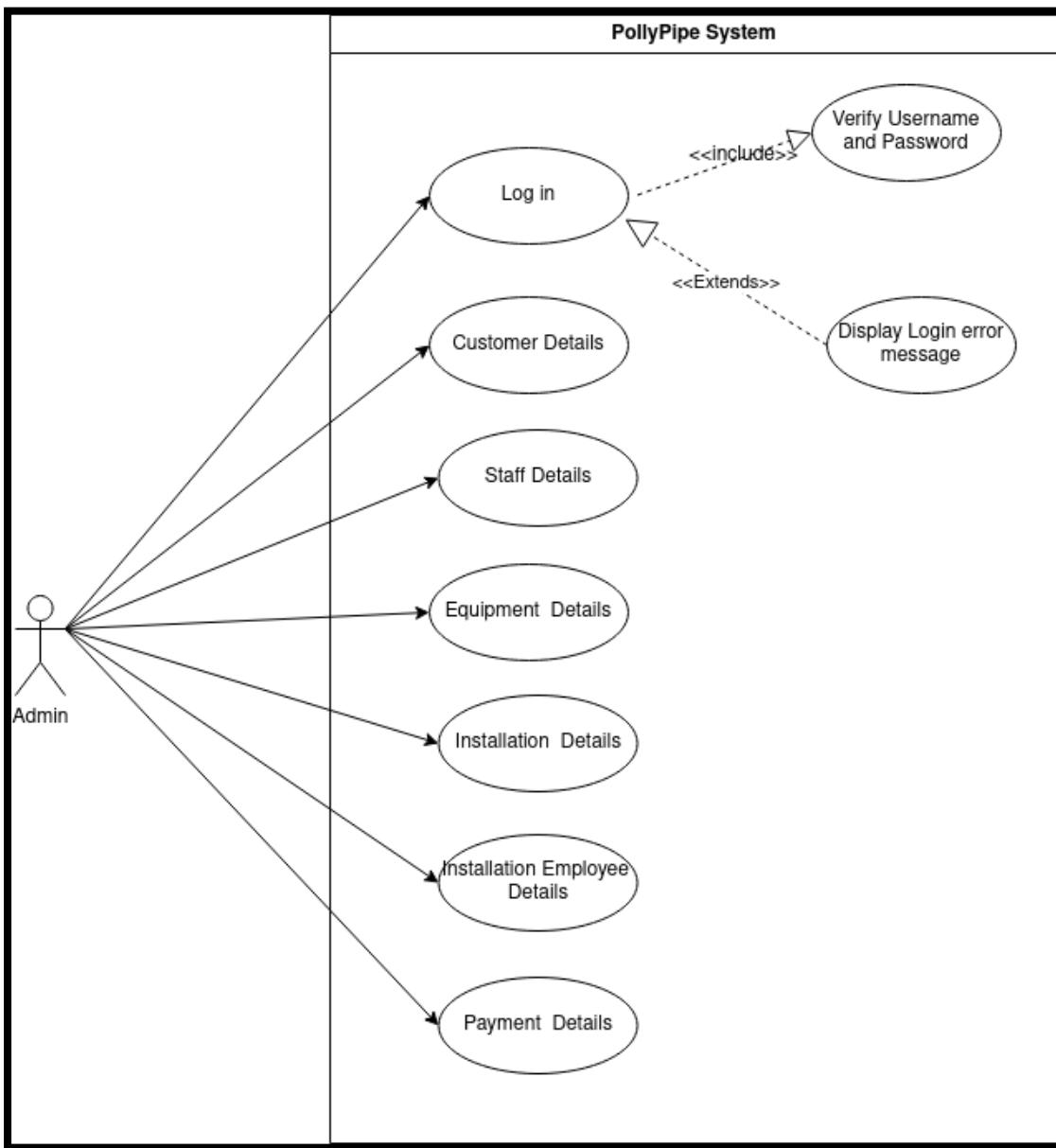


Figure 60 User case diagram

4.3 Class Diagram for PollyPIPE system

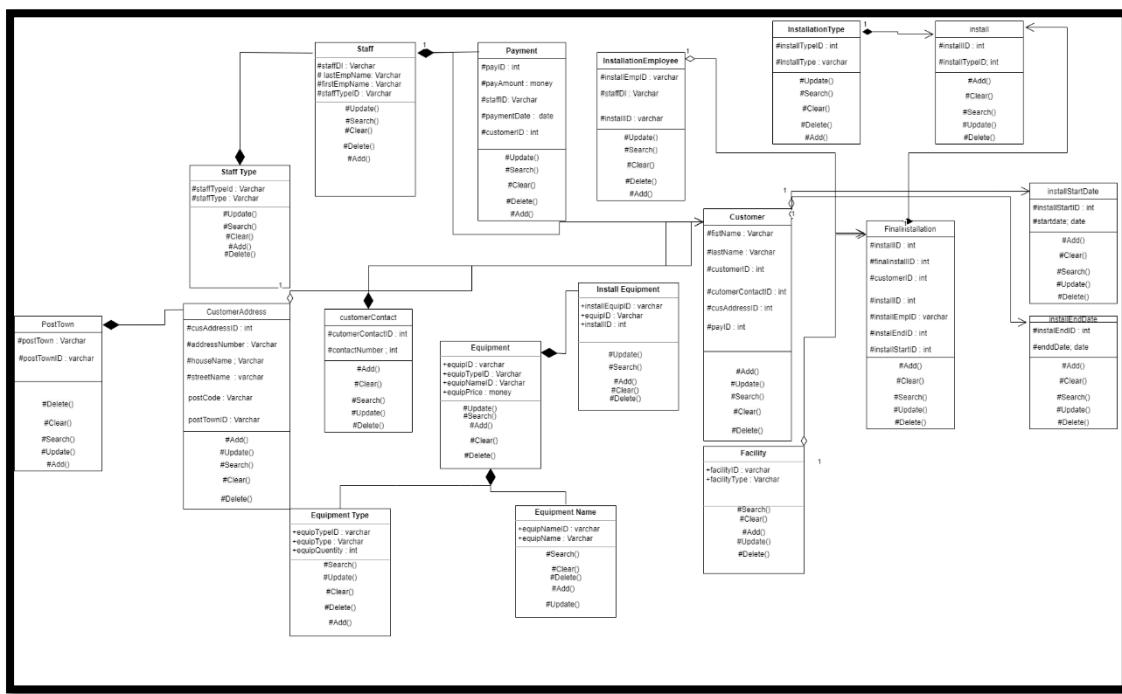


Figure 61 Class diagram

4.4 Flow chart for PollyPIPE system

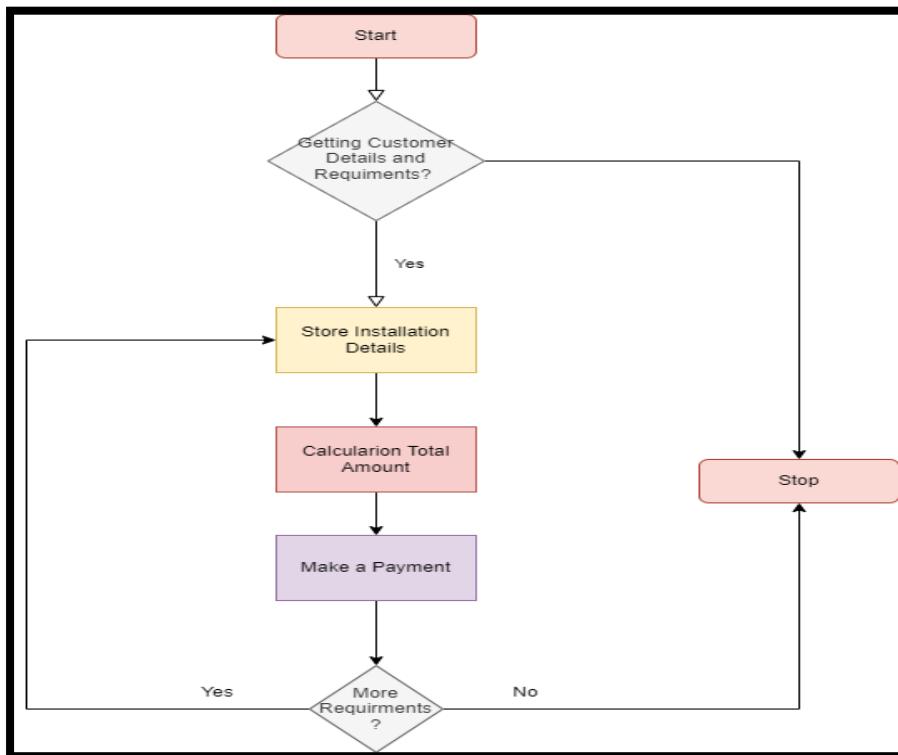


Figure 62 Flow chart

4.5 Future enhancements for the PollyPIPE system

1. Updating current Database version – The major factor in database performance is the version of SQL server that system currently deploying. Staying up to date with the latest version of that system database can have a significant impact on overall database performance. It's possible that one query may perform better in older versions of SQL than in new ones, but when looking at overall performance, new versions tend to perform better.
2. Upgrading Disk Types – The type of disks in this PollyPIPE system server can greatly impact the performance of that SQL query. Therefore, adding SSD disks into the system and working with SSD disks can significantly improve overall database performance and specifically SQL query performance.
3. Go to Cloud service – It will reduce the physical storage that system has.
Examples: AW Cloud, Oracle, Microsoft Azure, Google Cloud Platform
4. Allocating more memory in the system- Basically, having more memory available will help to boost the system's efficiency and overall performance.
5. Improving GUI for getting a good user experience.
6. Adding Data Encryption methods to the system for protecting customer details.

Consolations

In conclusion, this report gives you the fundamental knowledge of Database Design and Development. In this scenario, I had a build system using SQL server and C# Windows application for a company called Polypipe (Pvt) Ltd, and their main requirements are mentioned. This report gives you all the information about SQL, the type of SQL language, examples, GUI for systems, how to collect user feedback and analyze, a user guide for the developed system, testing process, and future enhancements for the developed system.

Gantt Chart

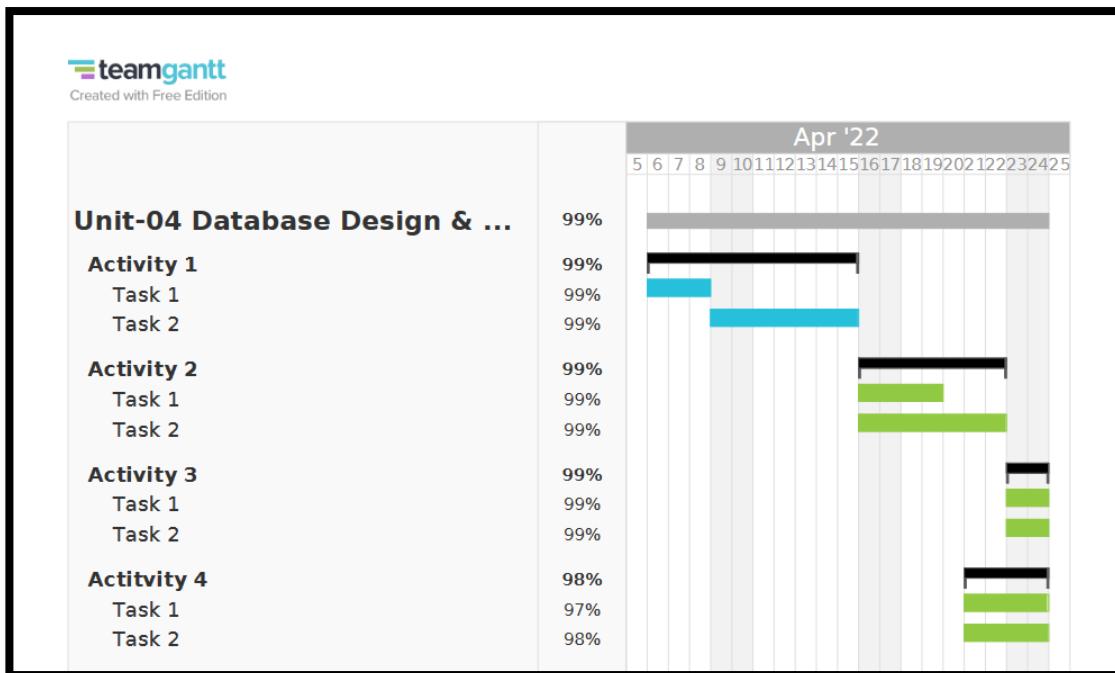


Figure 63 Gantt chart

References

- altexsoft, 2022. *altexsoft*. [Online] Available at: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/> [Accessed 6 April 2022].
- Lucidchart, 2022. *Lucidchart*. [Online] Available at: <https://www.lucidchart.com/pages/er-diagrams> [Accessed 6 April 2022].
- MacKay, J., 2018. *planio*. [Online] Available at: <https://plan.io/blog/technical-documentation/> [Accessed 21 April 2022].
- MERCED, U., 2022. UC MERCED LIBRARY. [Online] Available at: <https://library.ucmerced.edu/data-dictionaries> [Accessed 6 April 2022].
- Microsoft, 2022. *Microsoft*. [Online] Available at: <https://docs.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description> [Accessed 7 April 2022].
- Niepewna, U., n.d. *Startquestion*. [Online] Available at: <https://blog.startquestion.com/7-reasons-why-customer-feedback-is-important-to-your-business/> [Accessed 24 April 2022].
- Parker, J., 2012. *enFOCUS SOLUTION*. [Online] Available at: <https://enfocussolutions.com/business-user-and-system-requirements/> [Accessed 6 April 2022].
- Singh, M., 2022. *SQLShack*. [Online] Available at: <https://www.sqlshack.com/understanding-security-testing-for-sql-server-environments/> [Accessed 16 April 2022].
- techopedia, 2014. *techopedia*. [Online] Available at: <https://www.techopedia.com/definition/1179/data-manipulation->

language-dml

[Accessed 16 April 2022].