American Journal of Interdisciplinary Research in Engineering and Sciences
ISSN: 2994-3175 | Impact Factor: 5.58
Volume. 12, Number 1; January-March, 2025;
Published By: Scientific and Academic Development Institute (SADI)
8933 Willis Ave Los Angeles, California
https://sadijournals.org/index.php/AJIRES|editorial@sadipub.com

DESIGN AND IMPLEMENTATION OF A MYSQL BASED DATABASE FOR AN ONLINE FAULT REPORTING SYSTEM.

Iyidobi Jonathan Chijindu
Department of Mechatronic Engineering, Enugu State University of Science and Technology, Nigeria
DOI: https://doi.org/10.5281/zenodo.15176120

Abstract: In this paper, a Mysql-based database was developed for the implementation of an online fault reporting system for a utility company. The Ikeja zone Power Holding Company of Nigeria, now Ikeja Electric Company was used as the case study utility company. A use case diagram was used to depict major requirements of the system and their roles in the system being created. An activity diagram showed the direction of flow of information in the database while an entity relation diagram depicted the relationship between the key tables that formed the database. The database tables resulted from a progressive normalization of the unnormalized table from first normal form to third normal form tables. The designed database was implemented in Mysql platform. Result of test conducted on the system showed that the developed database is able to accept inputted data, store accepted data in the right tables and allow retrieval of the data with the correct queries. In conclusion, the developed system was found to be functional, effective and with the ability to support a dynamic online fault reporting system. It is recommended that future researchers in this area should consider equipping the database with security features that will guarantee the integrity of the data it is holding.

Keywords: Database, Mysql-Based, Online, Reporting, System

1. Introduction

The need for an online fault reporting system in any utility company can never be over stressed. A web-based fault reporting system enables users of critical social services like electrical power to timely report faults or outages in their area to facilitate immediate restoration of services. Such fault reporting platforms if designed to be effective should be web-based, dynamic and user friendly. To ensure that users are able to interact with the system in a manner that allows them to input and retrieve information from the system; there is need to equip the system with a functional database. The role of a database in a web-based developments can never be over stressed. Doga et al (2023) sees database as a system where data needed for the operation of a web application is inserted, saved, read or altered on a daily basis. The author noted that one of the key merits and values of databases is the ability to accumulation and protect large amounts of data. Database has been found to play a critical role in facilitating the operation of Web programming.

For an effective model such a database, there is need to use Unified Modelling Language (UML). Fowler (2004) defined UML as set of visual or diagrammatic representations; supported by meta-model, and which is used in the design and description of software systems. To ensure that all user requirements are taken into account and that the system function in the right and logical sequence; two UML diagrams: the Use Case diagram and the Activity diagram are recommended for such modelling and design.

Eriksson and Penker (1998) defined Use Case as a style in modelling which depicts either what functions an already existing system performs or the functions that a new system will perform, and which is targeted at generating a tangible outcome for a user. Fowler (2004) noted that Use case is a good tool employed to capture a system's functional requirements and operates by illustrating the interactions between the system and the system users. Eriksson and Penker (1998) noted that Use case diagrams are used to depict Use case models and that a Use case diagram shows the system, the use cases and the actor and the interaction between them.

Eriksson and Penker (1998) defined an actor as an object that communicates with the system either by sending or receiving information from the system.

The expected actors in such a model are the users and the system administrator. The users include the customers who need to report faults that occur in their properties (residential homes, offices, shops, etc.) and the fault monitoring staff who patrol the utility facilities/equipment (e.g. transformers, circuit breakers, power lines, insulators) to observe and report faults for speedy attention.

2 System Modeling

2.1 The Use Case Diagram

As required by this type of system, users will need to login to this system before being able to use this system so as to help regulate users' activities, so "login to the system", is a use case. Consequently, "register with the system" is also a use case as users must register to obtain login details. The system will need to authenticate or validate the login details of the user, so "authenticate user" is also a use case. After registration, users will need to know whether their registration was successful so "confirm registration" is also a use case. User details and fault details are also very useful information that need to be in the database. "Enter user details" and "enter fault details" are also use cases. Moreso, user and fault details need to be confirmed; as a result, "confirm success in entering user details" and "confirm success in entering fault details" are also identified as use cases. There is strong need for users to be able to track the progress of fault rectification. Based on this; "track fault status" was identified as a use case. The system should be able to allow the utility company to manage fault rectification process. To do this, the system administrator needs to constantly interact with the system while allocating rectification duties, updating the status of fault reported and managing the database in general. For this reason, "update the system" is a use case. There is also a strong need for the system to be a source of information regarding how best to use and handle the utility facilities, best safety practices with electricity and how best to manage the facilities to reduce fault occurrences." Learn more" is therefore identified as a use case. "Contact us is also identified as a use case. Majority of the above use cases were adopted from related primary research carried out in Iyidobi et al (2023)

In figure 2.1 below, we observe that the use case "login to the system" includes the use case "register with the system" this is because, no user can login to the system unless he/she registers to obtain login details. Also use cases "enter user details", "update the system" and "check fault status" includes "login to the" system since a user must login to be able to access these functions in the system.

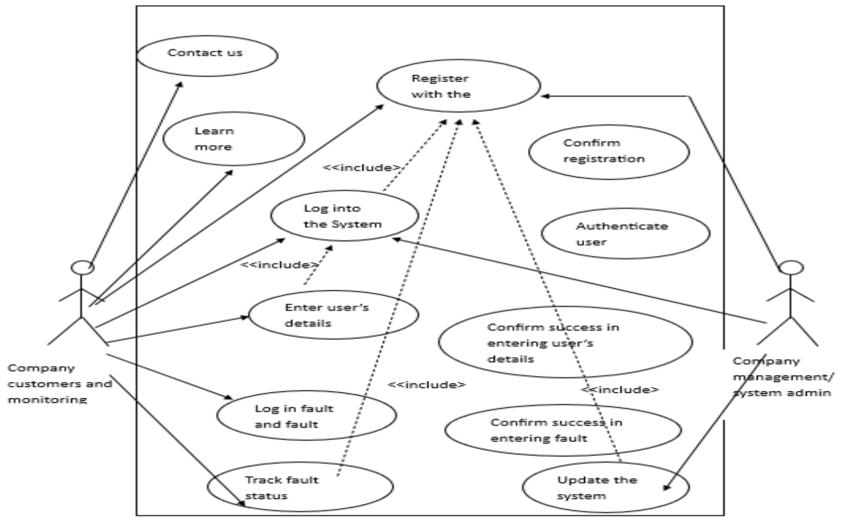
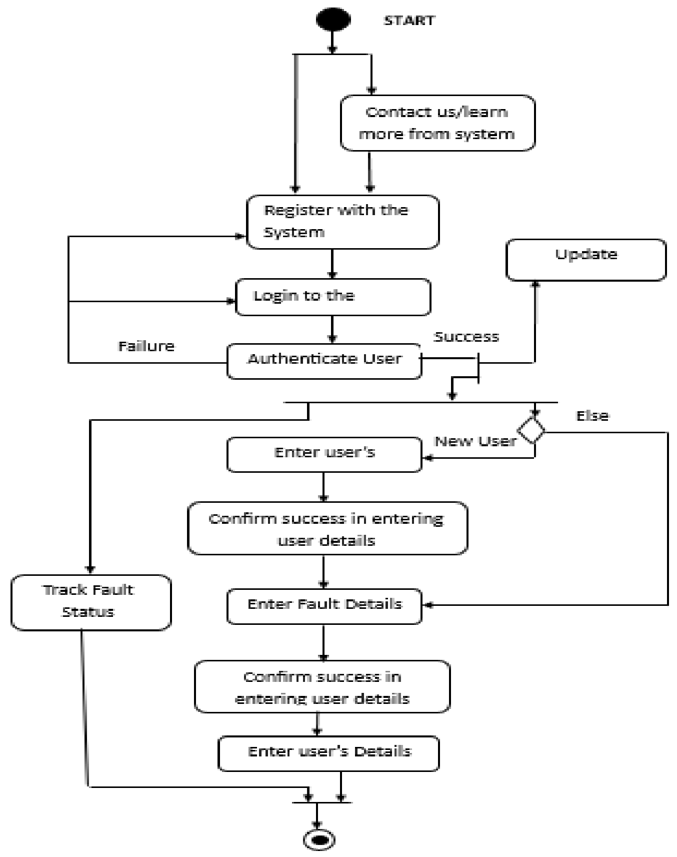


Figure 2.1: Use Case diagram

2.2 ACTIVITY DIAGRAM

According to Fowler (2004), Activity diagrams are tools used to depict the flow of work, processes of business and logical procedures. In this design, activity diagram will be used to show the sequence of flow of the activities to be performed with the system being designed. This will ensure that the system is implemented in a sequence that guarantees that the system performs its intended functions effectively and efficiently.
In the activity diagram shown in figure 2.2 below; on starting the process, users have the option of learning more on how to use the system and then viewing the contact us page first before moving ahead to register, or moving directly to register with the system. After a successful registration, the system confirms the success of registration to the user. After registering successfully, the user proceeds to log into the system. The system validates the user's login details and either denies him access or allows him access to the next stage depending on whether his details match any of the ones in the database or not. If the user's details match any of the ones in the database, the user is allowed to move to the next stage. On the other hand, if the user's login details do not match any of the ones in the database, the user is denied access to the next stage and re-directed back to the login stage. After a successful login, the user is allowed to enter his details. On the other hand, if the user has previously logged in a fault, he can move to check the status of the fault reported. From figure 2.2, we observe that new users are compelled to enter their details before entering any fault details. However, old users must move directly to entered fault detail after a successful login. This has been done to ensure that while users are not allowed to report a fault unless their details are in the database, and also to prevent multiple entering of users' details that will cause redundancy in the system. After entering personal details, the user receives a confirmation that his details have been successfully entered and then directed to enter the details of the fault that is being reported. On entering the fault details, the system confirms to the user through a displayed message that the fault details were successfully received by the system. The user ends the fault reporting process by logging out of the system.



2.3 DATABASE DESIGN AND BLUE PRINT

One the major objectives behind this project is to build a database that will capture and store fault reports logged in by an Electricity Utility Company in Nigeria customers and its fault monitoring staff. The case study company used is the old Power Holding Company of Nigeria (PHCN) Ikeja Zone now Ikeja Electric. These reports will help PHCN management to expeditiously send its staff to rectify the faults so as to reduce outage duration to the lowest possible level. This will increase customer satisfaction for PHCN customers in general and reduce cost of production for industrial consumers of electricity. Also, an accumulation of these reports over a period of time will help PHCN to device an effective plan against re-occurrence of the reported faults. Considering that the success of this project depends very much on how effective this database is, care was taken to ensure that the tables were correctly normalized and that the columns are structured to accept only the right data. Mysql database will be used for this design. This is because it is an open source and provides an easy and effective connection with server end language that may be employed during web design.

TABLE NORMALISATION

Bower (1993) defined normalization as a technique employed in database design to eliminate redundancy in a database. According to Bower, redundancy in a database is the undesirable duplication of data in a way that causes inconsistency in the database. Bower maintained that the consequences of redundancy in a database range from poor performance (speed) to creating confusion when the database is being updated (Bower, 1993). The above points were reechoed by when they observed that normalization is a database development design strategy for arranging data in an organized and consistent manner. The authors added that normalization minimizes redundancy and preserve the integrity of the database. It also removes unwanted characteristics relating to data

Figure 2.2: Activity Diagram for the system

Insertion, data deletion, and updating. Normalization is to reduces complexities, removes duplicates, and forms data in a consistent way. To avoid any undesirable redundancies, care was taken to ensure that the tables are well normalized. Table 2.1 shows the unnormalized test data that is used to make this normalization process easier and clearer. (Note: These data are not the data used for the real testing of the database after implementation). The attributes in table 2.1 are interrelated. The first set of attributes tends to identify a particular customer; the next set of attributes tends to identify a particular fault while the last set of attributes tends to identify the PHCN rectification team. There is an obvious relationship between these set of attributes. A customer reports a fault and a reported fault rectified by a PHCN repair team. Gillenson (2005) identified three stages of normalization. They include First normal form, second normal form and third normal form.

UNNORMALILED DATA (with attributes)

Table 2.3: **Unnormalized data (with attributes).**

Customer ID	First name	Sur name	e-mail address	P/Address	Phone	B/Unit Name	B/Unit Mgr	B/Unit Phone	Fault Type	Fault Description	Fault Date	Fault Time	status	R/Team ID	R/Team Name	R/Team Phone	R/Team Leader
001	John	Igwe	Igwe @	5, Kpaja	1234	Ojodu	Luke	9876	Open cct	Black out	2/3/09	10pm	fixed	601	A	1348	okey
	John	Igwe	Igwe @	5, Kpaja	1234	Ojodu	Luke	9876	Short cct	Low volt	4/10/10	3pm	unfixed	602	B	6531	uko
002	Okeke	Ugbo	Ugbo @	2, church	4532	Ikeja	Ugo	4567	Open cct	Damaged pole	7/10/10	8am	In progress	601	A	1348	okey

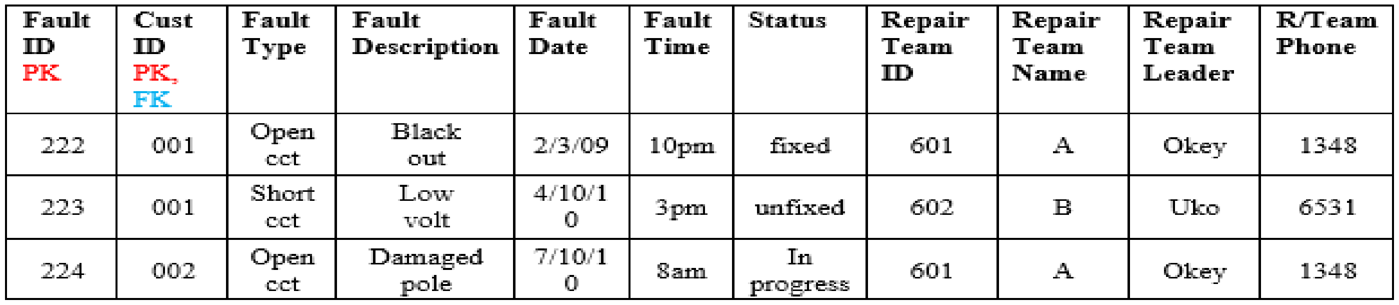
FIRST NORMAL FORM

Bower (1993) noted that repeating groups are not allowed in first normal form. The task involved in converting the unnormalized table to first normal form is to form a new table with columns that have repeating values along with the column(s) they depend on. As can be seen from table 2.3 the repeating columns that have been separated from the original table together with Customer ID form another table. The columns that make up this table include: fault ID, Customer ID, fault type, fault description, fault date, fault time, status, repair team ID, repair team name, repair team leader, and repair team phone number. The primary key of this table is a combination of the fault ID and the customer ID. Customer ID is also a foreign key to this table. The original table now contains columns that are fully dependent on customer ID. As can be seen from table 2.2 the table contains the following columns: customer ID, Customer firstname, Customer surname, e-mail address, property address, phone number, Business unit name and Business Unit manager. The customer ID is made the primary key as all the other columns are dependent on it while.

Table 2.2: FIRST AND SECOND NORMAL FORM (A)

Customer ID PK	Customer Firstname	Customer Surname	e-mail address	Property Address	Customer Phone	Business Unit Name	B/unit Phone	B/ Unit Mgr
001	John	igwe	igwe@	5, Kpaja	1234	Ojodu	9876	Luke
002	Okeke	ugbo	ubgo@	2, church	4532	Ikeja	4567	Ugo

Table 2.3: IN FIRST AND SECOND NORMAL FORM (B)



Fault ID PK	Cust ID PK, FK	Fault Type	Fault Description	Fault Date	Fault Time	Status	Repair Team ID	Repair Team Name	Repair Team Leader	R/Team Phone
222	001	Open cct	Black out	2/3/09	10pm	fixed	601	A	Okey	1348
223	001	Short cct	Low volt	4/10/10	3pm	unfixed	602	B	Uko	6531
224	002	Open cct	Damaged pole	7/10/10	8am	In progress	601	A	Okey	1348

SECOND NORMAL FORM

Resolution to second normal form demands that all columns depend on the primary key. Looking at table 2.2, we observe that all non-key columns are dependent on the table's primary key, customer ID. This is so since for any particular customer ID, there can be one and only one first name, surname, business unit, unit manager, etc. The table of table 3.2 is therefore already in its second normal form. Similarly, in table 2.3, all the non-key columns are dependent on the table's primary key, fault ID/customer ID. This is so since for every pair of customer ID and fault ID, there is one and only one fault type, property address, repair team, repair team leader, etc. This table is therefore also already resolved into second normal form.

3.2.4 THIRD NORMAL FORM

The third normal form does not allow any column to depend on any non-key column. Observe from table 2.2 that though Business Unit manager, and Business Unit phone number depends on the primary key, customer ID, they also depend on the Business Unit name, a non-key column. To resolve table 2.2 into third normal form, Business Unit name, Business Unit manager, Business Unit phone number are moved from table 2.2 to form a new table 2.4 called Business Unit table. The primary key of this table is Business Unit name. The columns left in table of figure 2.2 (customer ID, first name, surname, e-mail address and property address) together with Business Unit name, form another table called the cdetails table. Business Unit name exists in this table as a foreign key (see table 2.5). Observe from table 2.3 that repair team name, repair team leader and repair team phone number are dependent on a non-key column repair team ID. These columns are removed from table 2.3 to form a new table called Repair Team table. Repair team ID is the primary key. (See table 2.6) The columns left in table 2.3 together with Repair Team ID form another table called the cFault table. Repair team ID exists in the cFault table as a foreign key. (See table 2.7). With the successful resolution into third normal form, the normalization process is completed and we arrived at four tables namely: cdetails table, Business Unit Table, cfault table, and crepair Team table.

THIRD NORMAL FORM TABLES FOR CUSTOMER PART OF THE DATABASE

TABLE 2.4: BUSINESS UNIT TABLE

Business Unit Name	B/ Unit Phone	B/ Unit Mgr
Ojodu	9876	Luke
Ojodu	9876	Luke

Ikeja	4567	Ugo

TABLE 2.5: CDETAILS (CUSTOMER DETAILS) TABLE

Customer ID PK	Customer Firstname	CustomerSurname	e-mail address	PropertyAddress	Customer Phone	Business Unit Name FK
001	John	Igwe	igwe@	5, Kpaja	1234	Ojodu
002	okeke	Ugbo	ubgo@	2, church	4532	Ikeja

TABLE 2.6: CREPAIR TEAM (CUSTOMER REPAIR TEAM) TABLE

Repair Team ID	Repair Team Name	Repair Team Leader	R/Team Phone
601	A	okey	1348
603	B	uko	6531
602	C	Agbak	1348

TABLE 2.7: CFAULT (CUSTOMER FAULT) TABLE

Fault ID PK	Cust ID PK FK	Fault Type	Fault Description	Fault Date	Fault Time	Status	Repair Team ID FK
222	001	Open cct	Black out	2/3/09	10pm	fixed	601
223	001	Short cct	Low volt	4/10/10	3pm	unfixed	602
224	002	Open cct	Damaged pole	7/10/10	8am	In progress	601

THIRD NORMAL FORM OF TABLES FOR STAFF SECTION OF THE DATABASE

TABLE 2.8: SECTION TABLE

Section name	section Phone	Section head
Transmission	33345	Tony

Distribution	89771	George
Sub-Station	12567	Chidi

TABLE 2.9: SDETAILS (STAFF DETAILS) TABLE

Staff ID PK	**Firstname**	**Surname**	**e-mail address**	**staff Phone**	**Section name FK**
08	Peace	kene	kene@	4567	Transmission
09	Adu	tobe	tobe@	7654	Distribution

TABLE 2.10: SREPAIR TEAM (STAFF REPAIR TEAM) TABLE

Repair Team ID	**Repair Team Name**	**Repair Team Leader**	**R/Team Phone**
501	X	Ime	8652
502	Y	Ifeoma	98437
503	Z	Dom	13567

TABLE 2.11: SFAULT (STAFF FAULT) TABLE

Fault ID PK	**staff ID** PK FK	**Fault Type**	**Fault Description**	**Fault Date**	**Fault Time**	**Fault Location**	**Status**	**Repair Team ID** FK
12	08	Sub-satation	Bad transformer	3/3/09	11pm	Maryland	unfixed	501
13	08	Xmission line	Damaged insulator	6/10/10	4pm	Lagos Island	fixed	502
14	09	Distribution	Cable damaged	10/10/10	9am	Surulere	In progress	503

3 DATABASE MODELING

3.1 ENTITY RELATIONSHIP

The blue print of this database is depicted by an entity relationship diagram that shows the relationships between the four final tables arrived at. The boxes represent the tables and the underlined attribute(s) is the primary key of the table represented by the box. From figure 3.1, we observe that a customer belonging to a business unit reports

a fault and the fault reported by a customer is rectified by PHCN repair team.
Observe that there is a one-to many relationships between customer table and Business unit table. A Business Unit can house many customers but a customer can only belong to one Business unit. For this reason, in accordance with the rules of one-to-many relationship, Business Unit name, the primary key of Business Unit table (positioned in the one side) will be posted to the cdetails table (the many side) as a foreign key. This corresponds to the result obtained from normalization. This can be seen from table 2.5.
Note also that there is a one-to-many relationship between customer table and Fault table. A customer can report many faults but a particular fault can be reported by just one customer. In line with the rules of one-to-many relationship, Customer ID, the primary key of the cdetails table (positioned on the one side) is posted to the cfault table (the many side) as a foreign key. This is also confirmed from the result obtained from normalization as can be observed from table 2.7.
It can be seen that a one-to-many relationship also exist between the cfault table and the crepair Team table. This is so because a repair team can be assigned to rectify many faults but a fault can be rectified by one Repair team. In accordance with the rules of one-to-many relationship, the primary key of the repair team table (positioned on the one side) is posted as a foreign key to the cfault table (the many side). This corresponds to the result obtained from normalization. This can be seen from table 2.7.

CASE STUDY DATABASE MODEL

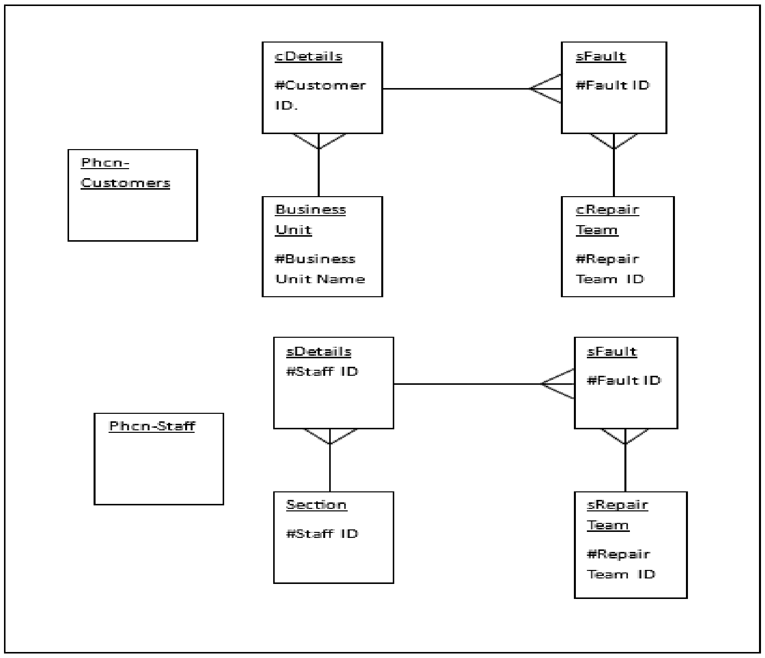


Figure 3.1: Comprehensive Database blue print showing entity relationship between tables.

3.2 STRUCTURE OF THE TABLES

The tables will be structured in such a way as to ensure that enough spaces are provided for data being expected in each column, and that adequate checks and constraints are imposed where necessary to ensure that the database only accepts data entered in the specified format.
Some of the important structures and constraints put in place in this design are hereby highlighted. In the cdetails table, the Customer ID has been constrained to accept only numbers since the PHCN numbers which is

represented by the Customer ID is made of figures only. Longer memory spaces have been allocated to columns like property address to ensure that long customer addresses are accommodated. All columns have been structured not to allow null values. This is to ensure that users enter all their details required.

In the cfault table, unlike in the cdetails table where the primary key, Customer ID (PHCN N0) has been issued to all customers on connection to PHCN mains; Customers do not know the fault ID of the fault they are about to report before hand and as such they cannot fill it in. To resolve this, the fault ID is structured to be generated automatically in an incremental order. In this table, customer ID and fault description are made not null. This is to ensure that users enter this information before they can successfully submit the fault details. Customer ID is needed to help in selecting and displaying fault status to users. On the other hand, since the customer would have already entered his details, PHCN management now need at least a description of the fault to be able to mobilize a well-equipped repair team to fix the fault. Columns like fault date, fault type and time of fault will be allowed to be null since sometimes users may not be able to keep track of these information and may not be able to enter accurate values.

The crepair team table and the Business Unit table are to be completed by the system administrator. Since all the columns are very essential, null values will not be allowed in any of the columns.

The above is the design and blue print of the four related tables that will be used in capturing, storing and managing the faults reported by PHCN Customers in their properties that are connected to PHCN services. However, this system is meant to also allow PHCN fault monitoring team to report faults on PHCN facilities located mainly outside the domain of consumers. To make the management of the database easier and avoid any confusion and complications both on the users’ side and the management side, a similar set of four related tables will be generated for the capturing, storing and management of faults reported by PHCN fault monitoring team. The structure and design of these tables is just the same with the ones above. The major difference is just the change in the names of the tables and some of the columns. There are also some columns that are absent in some of the tables because they do not apply to reports made by monitoring staff. Details of the columns in these tables are shown in tables 2.8., 2.9, 2.10 and 2.11. The tables are in their third normal form and the normalization processes described above was also used to arrive at this result.

As shown in figure 3.1, there are two other separate tables that are neither related to each other nor the ones discussed earlier. They are the tables that will bear the login details of users. One of the tables will serve the PHCN Customers while the other will serve the PHCN staff that will need to use the system either to report a fault, update the system or perform ant managerial tasks on the database. The table holding the login data of the PHCN customers (phcn-customers) will have the following columns: First name, Surname, e-mail address and password. This will also apply to the table that will bear the login details of PHCN staff (Phcn-staff). Phcn-customers will be fed with data coming from the registration form. However, the phcn_staff will bear the same login details that PHCN staff usually use to login to other PHCN systems. It is important to note that customers' e-mail address will be their login user ID. This is to make it easier to restore password to users who may forget their password. Users who forget their password will be required to provide their first name, their surname and their e-mail address. Their password will then be sent to their e-mail where they have exclusive access to it.

4.2 DATABASE IMPLEMENTATION

The database was implemented in a computer system. The login details of the computer system (root and password) were used to connect to mysql server of the computer. The database was then created and named phcn.

Having created the database, all the needed tables were then created using the appropriate mysql commands. A total of ten tables were created. The mysql commands for creating the database and the tables were first typed in and saved in a text format before being copied into the mysql environment. In the cases where the command failed to run due to an error, the error is corrected in the text format and then copied to mysql environment again for running. This style makes error correction easier while at the same time saving the development file in text format.

The first two tables named phcn_customers and phcn_staff are the tables containing the login details (e-mail address and password) of phcn customers and phcn staff respectively. Null values were not allowed and the data type was set as VARCHAR. The tables were standing on their own and had no entity relationship with any other table.

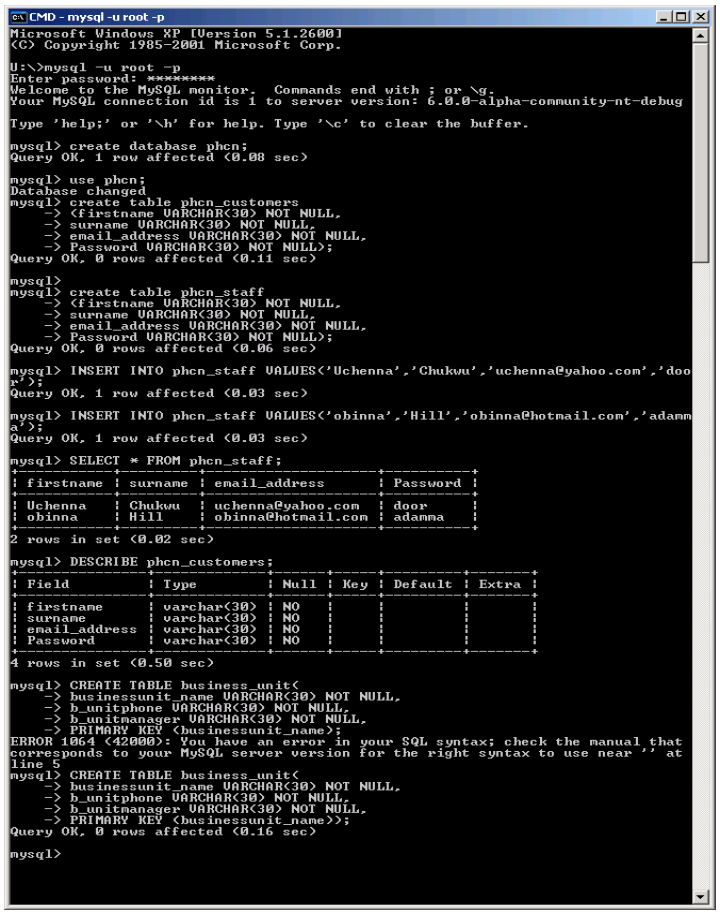
The next set of tables created had to do with data relating to phcn customers, the faults they reported, the repair team dealing with faults. Four tables were created here. The tables are: Cdetails table, Cfault table, Repair team table and the Business unit table. As pointed out earlier at the design stage, these four tables have entity relationships.

In the cdetails table, null values were not allowed and data type for all columns was set as VARCHAR. Memory spaces were allocated to the columns in accordance with the length of the data being expected. Care was taken to ensure that enough space was provided for each column. Customer ID was set as the primary key of this table and Business unit name set as the foreign key using appropriate commands.

In the cfault table, null values were only disallowed in the Customer ID and fault description column. Null values were allowed in the rest of the columns since users sometimes may not have accurate values for them or they may even not have them at all. The Auto increment command was used to progressively generate fault ID's for reported faults in increasing order. As a result, the data type for fault ID was set as INT. Data type for the rest of the columns was set as VARCHAR. A long memory space was allocated for fault description because of the length of data expected. Spaces allocated to other columns were proportional to the data expected. Fault ID/Customer ID was set as the primary key while Customer ID was made the foreign key.

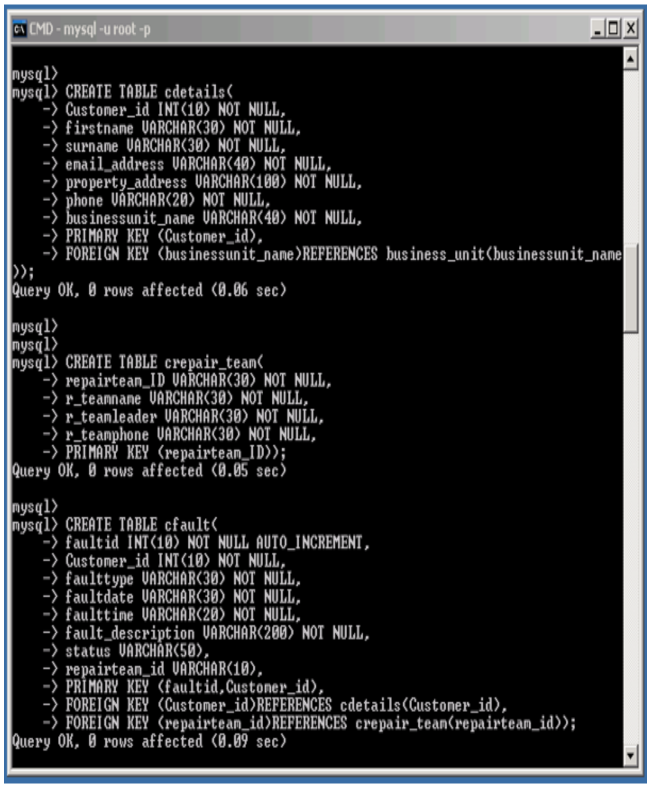
For the crepair team table and the Business unit table, null values were not permitted and the data type for all columns were set as VARCHAR. Memory spaces were allocated in proportion to the length of data expected. Crepair team ID was set as primary key for rectification team table while Business unit name was set as the primary key for the Business unit table.

The remaining four tables have the same structure as the four described above and as such they were developed with the same procedure. The names of the tables are: sdetails table, sfault table, srepair team table and section table. These tables also have entity relationship as described at the design stage. Figures 4.1, 4.2 and 4.3 show screen shots of mysql commands in mysql environment during the development of this database.


CMD - mysql -u root -p

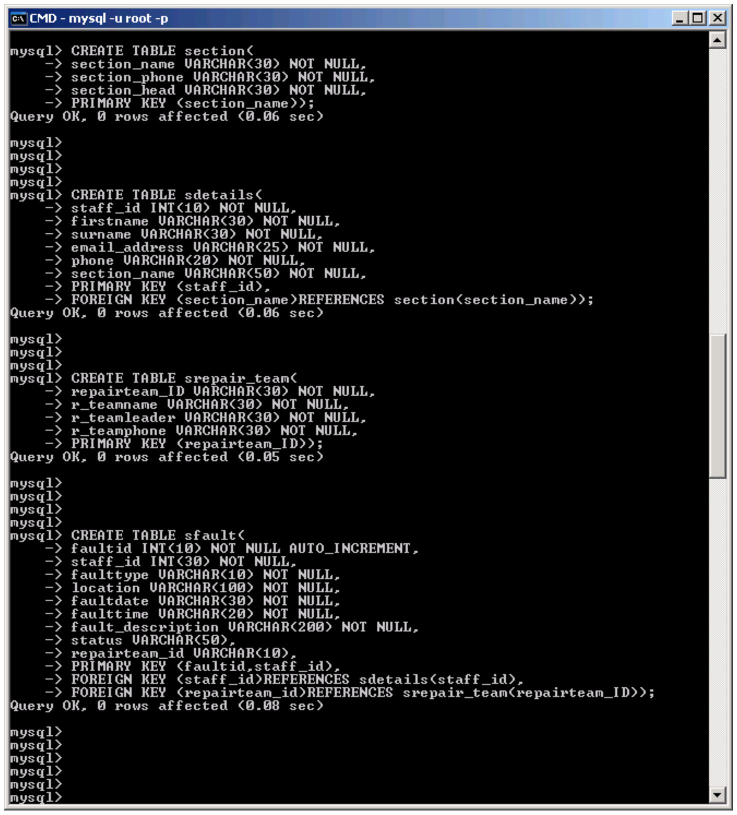
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

U:\>mysql -u root -p
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 6.0.0-alpha-community-nt-debug

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database phcn;
Query OK, 1 row affected (0.08 sec)

mysql> use phcn;
Database changed
mysql> create table phcn_customers
    -> (firstname VARCHAR(30) NOT NULL,
    -> surname VARCHAR(30) NOT NULL,
    -> email_address VARCHAR(30) NOT NULL,
    -> Password VARCHAR(30) NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql>
mysql> create table phcn_staff
    -> (firstname VARCHAR(30) NOT NULL,
    -> surname VARCHAR(30) NOT NULL,
    -> email_address VARCHAR(30) NOT NULL,
    -> Password VARCHAR(30) NOT NULL);
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO phcn_staff VALUES('Uchenna','Chukwu','uchenna@yahoo.com','doo
r');
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO phcn_staff VALUES('obinna','Hill','obinna@hotmail.com','adamm
a');
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM phcn_staff;
+-----------+---------+--------------------+----------+
| firstname | surname | email_address      | Password |
+-----------+---------+--------------------+----------+
| Uchenna   | Chukwu  | uchenna@yahoo.com  | door     |
| obinna    | Hill    | obinna@hotmail.com | adamma   |
+-----------+---------+--------------------+----------+
2 rows in set (0.02 sec)

mysql> DESCRIBE phcn_customers;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| firstname     | varchar(30) | NO   |     |         |       |
| surname       | varchar(30) | NO   |     |         |       |
| email_address | varchar(30) | NO   |     |         |       |
| Password      | varchar(30) | NO   |     |         |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.50 sec)

mysql> CREATE TABLE business_unit(
    -> businessunit_name VARCHAR(30) NOT NULL,
    -> b_unitphone VARCHAR(30) NOT NULL,
    -> b_unitmanager VARCHAR(30) NOT NULL,
    -> PRIMARY KEY (businessunit_name);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near '' at
line 5
mysql> CREATE TABLE business_unit(
    -> businessunit_name VARCHAR(30) NOT NULL,
    -> b_unitphone VARCHAR(30) NOT NULL,
    -> b_unitmanager VARCHAR(30) NOT NULL,
    -> PRIMARY KEY (businessunit_name));
Query OK, 0 rows affected (0.16 sec)

mysql>
```

Figure 4.1: Screen shot showing the creation of Phcn database, phcn_customers table, Phcn staff table and business_unit table.



```
mysql>
mysql> CREATE TABLE cdetails(
    -> Customer_id INT(10) NOT NULL,
    -> firstname VARCHAR(30) NOT NULL,
    -> surname VARCHAR(30) NOT NULL,
    -> email_address VARCHAR(40) NOT NULL,
    -> property_address VARCHAR(100) NOT NULL,
    -> phone VARCHAR(20) NOT NULL,
    -> businessunit_name VARCHAR(40) NOT NULL,
    -> PRIMARY KEY (Customer_id),
    -> FOREIGN KEY (businessunit_name)REFERENCES business_unit(businessunit_name
));
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql>
mysql> CREATE TABLE crepair_team(
    -> repairteam_ID VARCHAR(30) NOT NULL,
    -> r_teamname VARCHAR(30) NOT NULL,
    -> r_teamleader VARCHAR(30) NOT NULL,
    -> r_teamphone VARCHAR(30) NOT NULL,
    -> PRIMARY KEY (repairteam_ID));
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> CREATE TABLE cfault(
    -> faultid INT(10) NOT NULL AUTO_INCREMENT,
    -> Customer_id INT(10) NOT NULL,
    -> faulttype VARCHAR(30) NOT NULL,
    -> faultdate VARCHAR(30) NOT NULL,
    -> faulttime VARCHAR(20) NOT NULL,
    -> fault_description VARCHAR(200) NOT NULL,
    -> status VARCHAR(50),
    -> repairteam_id VARCHAR(10),
    -> PRIMARY KEY (faultid,Customer_id),
    -> FOREIGN KEY (Customer_id)REFERENCES cdetails(Customer_id),
    -> FOREIGN KEY (repairteam_id)REFERENCES crepair_team(repairteam_id));
Query OK, 0 rows affected (0.09 sec)
```

Figure 4.2: Screen shot showing the creation of the cdetails table, crepair_team, and cfault tables.

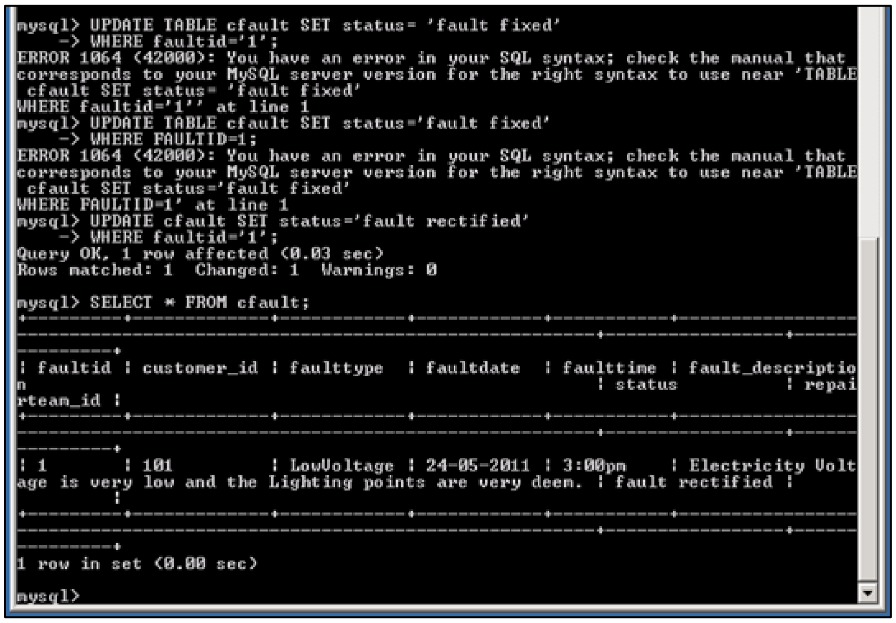


```
mysql> CREATE TABLE section(
    -> section_name VARCHAR(30) NOT NULL,
    -> section_phone VARCHAR(30) NOT NULL,
    -> section_head VARCHAR(30) NOT NULL,
    -> PRIMARY KEY (section_name));
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql>
mysql>
mysql>
mysql> CREATE TABLE sdetails(
    -> staff_id INT(10) NOT NULL,
    -> firstname VARCHAR(30) NOT NULL,
    -> surname VARCHAR(30) NOT NULL,
    -> email_address VARCHAR(25) NOT NULL,
    -> phone VARCHAR(20) NOT NULL,
    -> section_name VARCHAR(50) NOT NULL,
    -> PRIMARY KEY (staff_id),
    -> FOREIGN KEY (section_name)REFERENCES section(section_name));
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql>
mysql>
mysql> CREATE TABLE srepair_team(
    -> repairteam_ID VARCHAR(30) NOT NULL,
    -> r_teamname VARCHAR(30) NOT NULL,
    -> r_teamleader VARCHAR(30) NOT NULL,
    -> r_teamphone VARCHAR(30) NOT NULL,
    -> PRIMARY KEY (repairteam_ID));
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql>
mysql>
mysql>
mysql> CREATE TABLE sfault(
    -> faultid INT(10) NOT NULL AUTO_INCREMENT,
    -> staff_id INT(30) NOT NULL,
    -> faulttype VARCHAR(10) NOT NULL,
    -> location VARCHAR(100) NOT NULL,
    -> faultdate VARCHAR(30) NOT NULL,
    -> faulttime VARCHAR(20) NOT NULL,
    -> fault_description VARCHAR(200) NOT NULL,
    -> status VARCHAR(50),
    -> repairteam_id VARCHAR(10),
    -> PRIMARY KEY (faultid,staff_id),
    -> FOREIGN KEY (staff_id)REFERENCES sdetails(staff_id),
    -> FOREIGN KEY (repairteam_id)REFERENCES srepair_team(repairteam_ID));
Query OK, 0 rows affected (0.08 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

Figure 4.3: Screen shot showing the creation section table, sdetails, srepair_team and sfault tables.

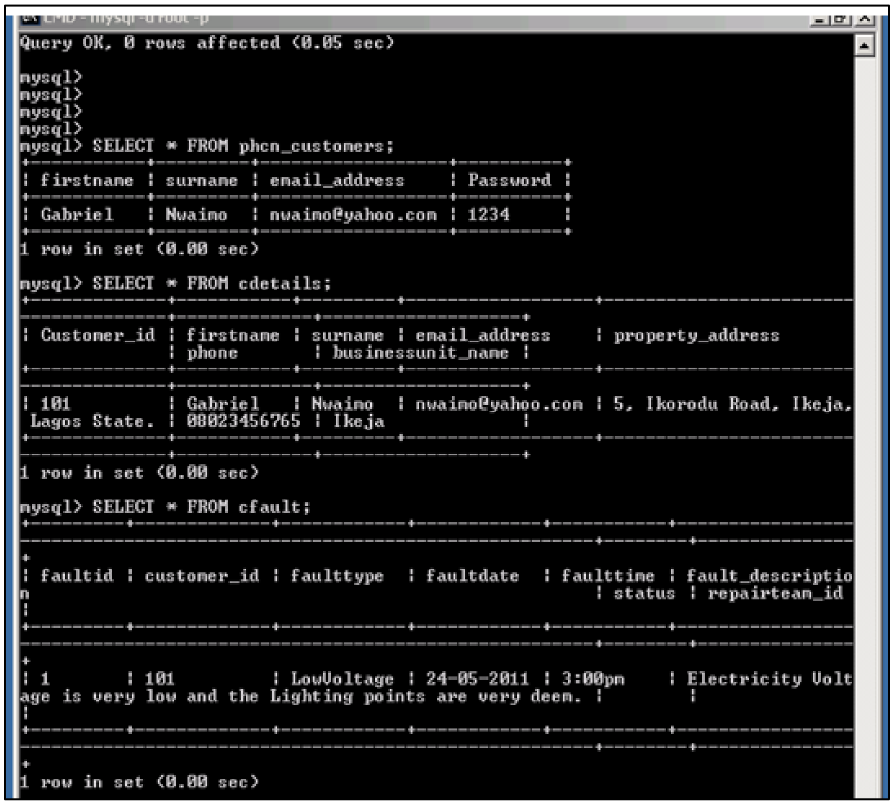
5 SYSTEM TESTING RESULT AND CONCLUSION

To ensure that the developed system is functional as well as possess the ability to support a dynamic fault reporting system for utility companies, some tests were carried out on the developed database. Tests are generally carried out in newly developed systems to ensure the developed system is devoid of any defects that will impact negatively on its performance. Testing is used to confirms if the actual performance of the delivered project or developed system meets expected performance (Gupta, 2016). Customers' details (Registration and Login) were inserted into the tables of the developed database using an online interface. Also, different faults were logged into the system by various users to see if the system will be able to display them.
Figures 5.1 and 5.2 showed that the created database is able to display both faults logged in by users and users' details inputted into the database respectively.



```
mysql> UPDATE TABLE cfault SET status= 'fault fixed'
    -> WHERE faultid='1';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'TABLE
 cfault SET status= 'fault fixed'
WHERE faultid='1'' at line 1
mysql> UPDATE TABLE cfault SET status='fault fixed'
    -> WHERE FAULTID=1;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'TABLE
 cfault SET status='fault fixed'
WHERE FAULTID=1' at line 1
mysql> UPDATE cfault SET status='fault rectified'
    -> WHERE faultid='1';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM cfault;
+---------+-------------+------------+------------+-----------+-----------------
-------------------------------------------------------+-----------------+------
---------+
| faultid | customer_id | faulttype  | faultdate  | faulttime | fault_descriptio
n                                                      | status          | repai
rteam_id |
+---------+-------------+------------+------------+-----------+-----------------
-------------------------------------------------------+-----------------+------
---------+
| 1       | 101         | LowVoltage | 24-05-2011 | 3:00pm    | Electricity Volt
age is very low and the Lighting points are very deem. | fault rectified |
         |
+---------+-------------+------------+------------+-----------+-----------------
-------------------------------------------------------+-----------------+------
---------+
1 row in set (0.00 sec)

mysql>
```

Figure 5.1: Screen shot displaying users' details of fault reported, all duly inserted in their corresponding table.



```
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql>
mysql>
mysql>
mysql> SELECT * FROM phcn_customers;
+-----------+---------+------------------+----------+
| firstname | surname | email_address    | Password |
+-----------+---------+------------------+----------+
| Gabriel   | Nwaimo  | nwaimo@yahoo.com | 1234     |
+-----------+---------+------------------+----------+
1 row in set (0.00 sec)

mysql> SELECT * FROM cdetails;
+-------------+-----------+---------+------------------+-------------------------
-------------+-------------+------------------+
| Customer_id | firstname | surname | email_address    | property_address
             | phone       | businessunit_name |
+-------------+-----------+---------+------------------+-------------------------
-------------+-------------+------------------+
| 101         | Gabriel   | Nwaimo  | nwaimo@yahoo.com | 5, Ikorodu Road, Ikeja,
 Lagos State. | 08023456765 | Ikeja            |
+-------------+-----------+---------+------------------+-------------------------
-------------+-------------+------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM cfault;
+---------+-------------+-----------+------------+----------+--------------------
-----------------------------------------------------------+--------+------------
+
| faultid | customer_id | faulttype | faultdate  | faulttime | fault_descriptio
n                                                          | status | repairteam_id
|
+---------+-------------+-----------+------------+----------+--------------------
-----------------------------------------------------------+--------+------------
+
| 1       | 101         | LowVoltage | 24-05-2011 | 3:00pm   | Electricity Volt
age is very low and the Lighting points are very deem. |        |
|
+---------+-------------+-----------+------------+----------+--------------------
-----------------------------------------------------------+--------+------------
+
1 row in set (0.00 sec)
```

Figure 5.2: Screen shot displaying user login details, details, personal details, all duly inserted in their corresponding table.

From the result of the tests conducted, it can be seen that the developed database for fault reporting systems is functional and also possess the ability to support a dynamic online fault reporting system. It is recommended that future researchers in this area should consider equipping the database with security features that will guarantee the integrity of the data it is holding.

REFERENCES

Blagojche, N., Dimitrovska, G. & Joshevska, E. (2023). The importance of databases in web programming. International Journal of Advanced Natural Sciences and Engineering Researches, 7(4), 319-322.Avalaibleat: https://www.academia.edu/126527972/The_importance_of_databases_in_web_programming

Bower D, (1993) 'From data to database' second edition, Chapman & Hall, London, 1993.

Eriksson H and Penker M (1998) 'UML toolkit' John Wiley & Sons, 1998, USA.

Fowler M (2004) 'UML distilled' 3rd edition, Addison-Wesley, 2004, USA.

Gillenson, M. (2005) ,'Database Modelling', international edition, John wiley & Sons, USA, 2005.

Gupta P, (2016) "Importance of testing in quality assurance and system development life cycle" International Journal of Advanced Science and Research, Available at:

https://www.academia.edu/38403416/Importance_of_testing_in_quality_assurance_and_system_development_life_cycle

Iyidobi J.C, Opata C. and Mba V.C, (2023) "Development of An Online Fault Reporting/Management System for Improving Electric Power Distribution in Nigeria" Journal of Computer Science Review and Engineering (JCSRE) Vol. 4, No. 8.

Kolade C, (2022), "Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples" Free Code Camp website, Available at: https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/