ABSTRACT

The Medicines & Products Supplying System (MPSS) is a comprehensive platform designed to streamline the procurement, distribution, and management of pharmaceuticals and related products. Leveraging advanced technology and efficient logistics, MPSS ensures seamless connectivity between healthcare providers, suppliers, and manufacturers, optimizing the supply chain process. Through real-time monitoring and predictive analytics, MPSS enhances inventory management, anticipating demand fluctuations and preventing stockouts or overstock situations. The system incorporates robust security measures to safeguard sensitive patient data and ensure compliance with regulatory standards. Furthermore, MPSS offers customizable features to accommodate diverse healthcare settings, including hospitals, clinics, pharmacies, and community health centers. By fostering collaboration and transparency among stakeholders, MPSS promotes cost-effectiveness, sustainability, and equitable access to essential medications and healthcare products, ultimately improving patient outcomes and advancing public health initiatives.

PROBLEM STATEMENT

The pharmaceutical supply chain confronts a myriad of challenges, including inefficiencies in procurement, distribution bottlenecks, inadequate inventory management, fragmentation among stakeholders, and a lack of transparency exacerbated by outdated infrastructure and manual tracking systems. These issues result in frequent stockouts, overstock situations, and inaccuracies in forecasting, leading to disruptions in healthcare delivery and increased costs. Additionally, the proliferation of counterfeit medications and regulatory compliance issues further compound the complexity of the supply chain. Geopolitical instability, natural disasters, and global health crises underscore the need for supply chain resilience and contingency planning to ensure uninterrupted access to essential medications and healthcare products. A comprehensive solution is urgently required to address these challenges, leveraging technology and collaboration to enhance transparency, efficiency, and resilience throughout the supply chain, ultimately ensuring timely access to quality healthcare resources for patients worldwide.

TABLE OF CONTENTS

PROBLEM UNDERSTANDING

CHAPTER 1

Understanding the complexities of a medicines and products supplying system involves navigating multiple interconnected processes. Firstly, it entails grasping the intricacies of supply chains, which typically involve various stakeholders such as manufacturers, distributors, wholesalers, and retailers. each entity plays a crucial role in ensuring that medicines and products reach their intended destinations efficiently and on time. Supply chain management involves tasks such as procurement, inventory management, transportation, and distribution, all of which must be coordinated seamlessly to prevent delays or shortages.

Secondly, comprehending the regulatory framework governing the supply of medicines and products is essential. Different regions and countries have distinct regulations regarding manufacturing standards, labeling requirements, import/export restrictions, and quality control measures. Understanding and adhering to these regulations are paramount to ensure compliance and avoid legal issues or penalties. Moreover, staying updated on regulatory changes and evolving industry standards is necessary to adapt the supply chain processes accordingly and maintain compliance.

In summary, mastering the medicines and products supplying system requires a deep understanding of supply chain dynamics and regulatory frameworks. Effective management involves optimizing processes, maintaining compliance with regulations, and adapting to changes in the industry landscape. By addressing these aspects comprehensively, stakeholders can enhance the efficiency, reliability, and safety of the supply chain, ultimately benefiting patients and consumers.

OBJECTIVE FOR THE PROJECT :-

To optimize the efficiency and effectiveness of the medicines and products supplying system by:

1. Enhancing supply chain management processes to ensure timely procurement, accurate inventory control, and efficient distribution of medicines and products.

2. Ensuring compliance with regulatory requirements and industry standards to maintain legality, quality, and safety throughout the supply chain.

3. Implementing strategies for continuous improvement, including the adoption of innovative technologies, proactive risk management, and performance monitoring, to drive operational excellence and meet evolving demands.

4. Strengthening collaboration and communication among stakeholders, including manufacturers, distributors, healthcare providers, and regulatory authorities, to foster transparency, accountability, and mutual trust.

5. Enhancing resilience and flexibility within the supply chain to mitigate disruptions, such as supply shortages, regulatory changes, or unforeseen events, and maintain uninterrupted access to essential medicines and products.

By achieving these objectives, the medicines and products supply system can optimize resource utilization, minimize costs, enhance customer satisfaction, and ultimately improve public health outcomes.

IDENTIFICATION OF ENTITY AND RELATIONSHIPS:

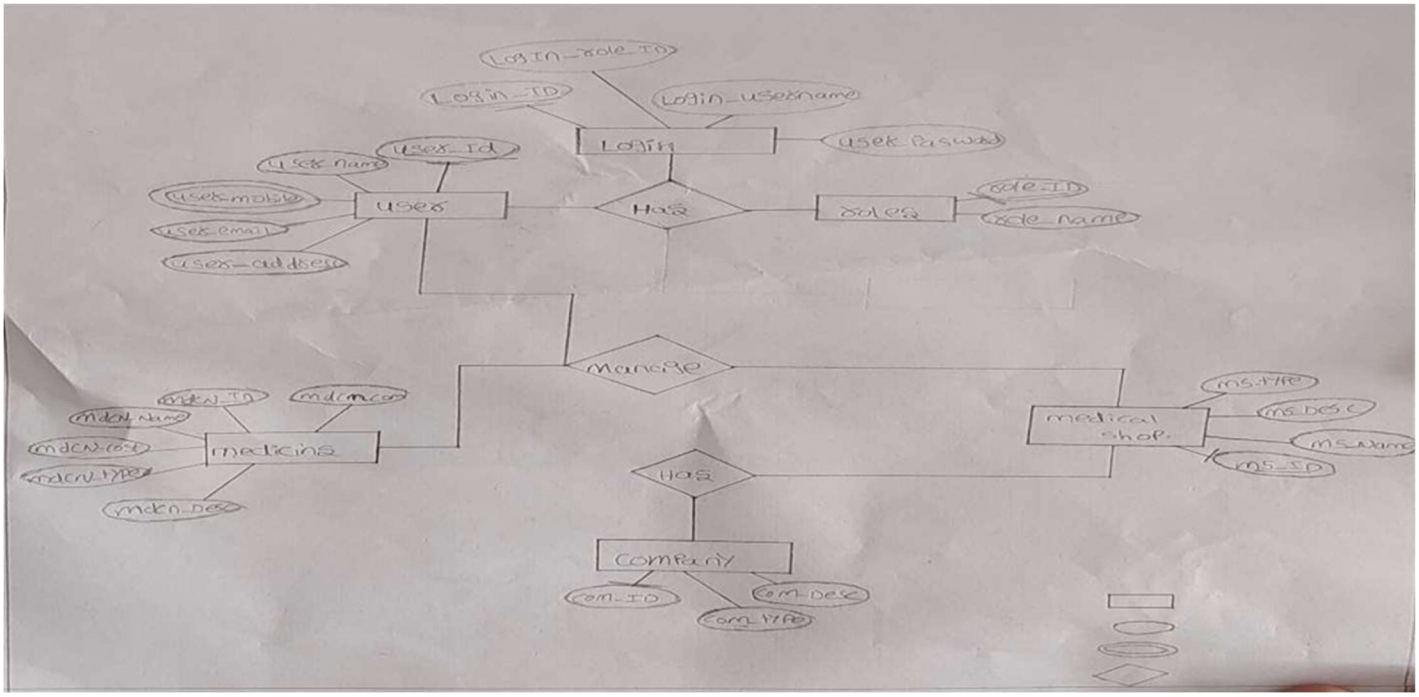
ENTITIES:

- Login
- User
- Role
- MedicalShop
- Medicines
- Stocks

RELATIONSHIPS:

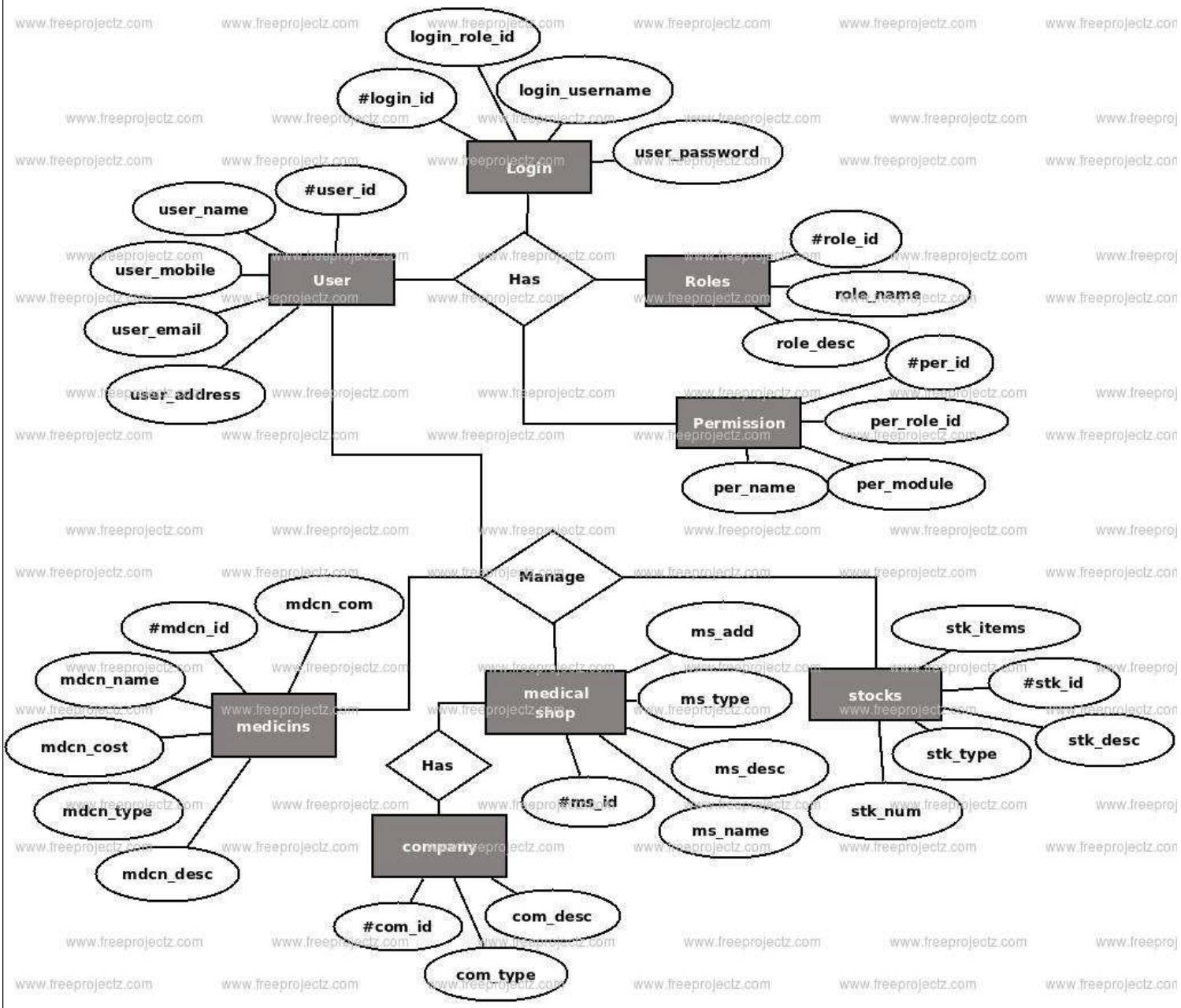
- Manage
- Has

CONSTRUCTION OF DB USING ER DIAGRAM:-



CHAPTER 2

DESIGN OF RELATIONAL SCHEMA:



ER Diagram For Medical Store Management System

CREATION OF DATABASE TABLES FOR THE PROJECT:-

```
mysql> use medicines_products_supplying_system;
Database changed
mysql> show tables;
+-----------------------------------------------+
| Tables_in_medicines_products_supplying_system |
+-----------------------------------------------+
| login                                         |
| medicalshop                                   |
| medicines                                     |
| role                                          |
| stocks                                        |
| user                                          |
+-----------------------------------------------+
6 rows in set (0.00 sec)
```

- **Login_Table**

```
mysql> select * from login;
+----------+----------------+---------------+
| login_id | login_username | login_role_ID |
+----------+----------------+---------------+
|      413 | surya          |        544468 |
|      423 | hitesh         |        858958 |
|      456 | akshay         |        544468 |
+----------+----------------+---------------+
3 rows in set (0.00 sec)
```

- **Medicalshop_Table**

```
mysql> select * from medicalshop;
+-------+---------+----------+--------------+
| ms_id | ms_name | ms_type  | ms_address   |
+-------+---------+----------+--------------+
| 99464 | venkat  | pharmacy | machilipatnam |
+-------+---------+----------+--------------+
1 row in set (0.00 sec)
```

- **Medicines_Table**

```
mysql> select * from medicines;
+---------+-----------+-----------+-----------+
| mdcn_id | mdcn_name | mdcn_type | mdcn_cost |
+---------+-----------+-----------+-----------+
|    5672 | dolo55    | fever     |    112.00 |
+---------+-----------+-----------+-----------+
1 row in set (0.00 sec)
```

- **Roles_Table**

```
mysql> select * from role;
+---------+-----------+
| role_id | role_name |
+---------+-----------+
|   89898 | supplier  |
+---------+-----------+
1 row in set (0.00 sec)
```

- **Stocks_Table**

```
mysql> select * from stocks;
+--------+-----------+-----------+---------+
| stk_id | stk_items | stk_types | stk_num |
+--------+-----------+-----------+---------+
|   6786 | dolo      | fever     |     500 |
| 753159 | injection | injuries  |     500 |
+--------+-----------+-----------+---------+
2 rows in set (0.00 sec)
```

- **User_Table**

```
mysql> select * from user;
+---------+--------------+-------------+--------------------------+--------------+
| user_id | user_name    | user_mobile | user_email               | user_address |
+---------+--------------+-------------+--------------------------+--------------+
|     111 | John Doe     | 1234567890  | john.doe@example.com     | 123 Main St  |
|     222 | Jane Smith   | 0987654321  | jane.smith@example.com   | 456 Elm St   |
|     333 | Alice Johnson | 5551234567 | alice.johnson@example.com | 789 Oak St   |
+---------+--------------+-------------+--------------------------+--------------+
3 rows in set (0.00 sec)
```

CHAPTER 3

Queries:-

```
Create database mmps

use mmps;

CREATE TABLE login (
    login_id INT PRIMARY KEY,
    login_username VARCHAR(255),
    login_role_ID INT
);

CREATE TABLE user (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(255),
    user_mobile VARCHAR(20),
    user_email VARCHAR(255),
    user_address VARCHAR(255)
);

CREATE TABLE role (
    role_id INT PRIMARY KEY,
    role_name VARCHAR(255)
);

CREATE TABLE MedicalShop (
    ms_id INT PRIMARY KEY,
    ms_name VARCHAR(255),
    ms_type VARCHAR(50),
```

```
CREATE TABLE Medicines (
    mdcn_id INT PRIMARY KEY,
    mdcn_name VARCHAR(255),
    mdcn_type VARCHAR(50),
    mdcn_cost DECIMAL(10, 2)
);
CREATE TABLE Stocks (
    stk_id INT PRIMARY KEY,
    stk_items VARCHAR(50),
    stk_types VARCHAR(50),
    stk_num INT
);
insert into login values('413','surya','544468');
insert into login values('456','akshay','544468');
insert into login values('423','hitesh','858958');
INSERT INTO user VALUES ('111','John Doe', '1234567890', 'john.doe@example.com', '123 Main St');
INSERT INTO user VALUES ('222','Jane Smith', '0987654321', 'jane.smith@example.com', '456 Elm St');
INSERT INTO user VALUES ('333','Alice Johnson', '5551234567', 'alice.johnson@example.com', '789 Oak
St');
INSERT INTO medicines values('05672','dolo55','fever','112');
insert into role values ('89898','supplier');
insert into role values ('123456','customer');
insert into medicalshop values('99464','venkat','pharmacy','machilipatnam');
insert into stocks values('6786','dolo','fever','500');
```

VIEWS:-

User_Role_View:

CREATE VIEW user_roles AS
SELECT u.user_id, u.user_name, u.user_mobile, u.user_email, u.user_address, r.role_name
FROM user u
JOIN login l ON u.user_id = l.login_id
JOIN role r ON l.login_role_ID = r.role_id;

```
mysql> select * from user_roles;
+---------+----------------+-------------+-------------------------+--------------+-----------+
| user_id | user_name      | user_mobile | user_email              | user_address | role_name |
+---------+----------------+-------------+-------------------------+--------------+-----------+
|     111 | John Doe       | 1234567890  | john.doe@example.com    | 123 Main St  | supplier  |
|     222 | Jane Smith     | 0987654321  | jane.smith@example.com  | 456 Elm St   | supplier  |
|     333 | Alice Johnson  | 5551234567  | alice.johnson@example.com | 789 Oak St | supplier  |
|     444 | Michael Johnson | 9998887776 | michael@example.com     | 321 Pine St  | supplier  |
+---------+----------------+-------------+-------------------------+--------------+-----------+
4 rows in set (0.00 sec)
```

AVALIBLE _MEDICINES VIEW:-

CREATE VIEW available_medicines AS
SELECT s.stk_id, m.mdcn_name, m.mdcn_type, m.mdcn_cost, s.stk_num
FROM medicines m
JOIN stocks s ON m.mdcn_id = s.stk_id;

```
mysql> select * from available_medicines;
+--------+-----------+-----------+-----------+---------+
| stk_id | mdcn_name | mdcn_type | mdcn_cost | stk_num |
+--------+-----------+-----------+-----------+---------+
|   5672 | dolo55    | fever     |    112.00 |     500 |
+--------+-----------+-----------+-----------+---------+
1 row in set (0.00 sec)
```

JOINS:-

```
    -> JOIN role r ON l.login_role_ID = r.role_id;
+---------+-----------------+-------------+--------------------------+--------------+-----------+
| user_id | user_name       | user_mobile | user_email               | user_address | role_name |
+---------+-----------------+-------------+--------------------------+--------------+-----------+
|     111 | John Doe        | 1234567890  | john.doe@example.com     | 123 Main St  | supplier  |
|     222 | Jane Smith      | 0987654321  | jane.smith@example.com   | 456 Elm St   | supplier  |
|     333 | Alice Johnson   | 5551234567  | alice.johnson@example.com | 789 Oak St   | supplier  |
|     444 | Michael Johnson | 9998887776  | michael@example.com      | 321 Pine St  | supplier  |
+---------+-----------------+-------------+--------------------------+--------------+-----------+
4 rows in set (0.00 sec)
```

TRIGGERS:-

```
mysql> -- Define a cursor to fetch data from the 'user' table
mysql> DELIMITER //
mysql> CREATE PROCEDURE fetch_users()
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE user_id INT;
    ->     DECLARE user_name VARCHAR(255);
    ->     DECLARE user_mobile VARCHAR(20);
    ->     DECLARE user_email VARCHAR(255);
    ->     DECLARE user_address VARCHAR(255);
    ->
    ->     -- Declare cursor for the 'user' table
    ->     DECLARE user_cursor CURSOR FOR
    ->         SELECT user_id, user_name, user_mobile, user_email, user_address
    ->         FROM user;
    ->
    ->     -- Declare handler for exceptions
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     -- Open the cursor
    ->     OPEN user_cursor;
    ->
    ->     -- Fetch data from the cursor
    ->     user_loop: LOOP
    ->         FETCH user_cursor INTO user_id, user_name, user_mobile, user_email, user_address;
    ->         IF done THEN
    ->             LEAVE user_loop;
    ->         END IF;
    ->
    ->         -- Print or process the fetched data as needed
    ->         SELECT user_id, user_name, user_mobile, user_email, user_address;
    ->     END LOOP;
    ->
    ->     -- Close the cursor
    ->     CLOSE user_cursor;
    -> END//
Query OK, 0 rows affected (0.01 sec)
```

CURSOURS:-

```
mysql> DELIMITER ;
mysql> call fetch_users();
+---------+-----------+-------------+------------+--------------+
| user_id | user_name | user_mobile | user_email | user_address |
+---------+-----------+-------------+------------+--------------+
|    NULL | NULL      | NULL        | NULL       | NULL         |
+---------+-----------+-------------+------------+--------------+
1 row in set (0.01 sec)

+---------+-----------+-------------+------------+--------------+
| user_id | user_name | user_mobile | user_email | user_address |
+---------+-----------+-------------+------------+--------------+
|    NULL | NULL      | NULL        | NULL       | NULL         |
+---------+-----------+-------------+------------+--------------+
1 row in set (0.02 sec)

+---------+-----------+-------------+------------+--------------+
| user_id | user_name | user_mobile | user_email | user_address |
+---------+-----------+-------------+------------+--------------+
|    NULL | NULL      | NULL        | NULL       | NULL         |
+---------+-----------+-------------+------------+--------------+
1 row in set (0.03 sec)

+---------+-----------+-------------+------------+--------------+
| user_id | user_name | user_mobile | user_email | user_address |
+---------+-----------+-------------+------------+--------------+
|    NULL | NULL      | NULL        | NULL       | NULL         |
+---------+-----------+-------------+------------+--------------+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.04 sec)
```

CHAPTER 4

FUNCTIONAL DEPENDENCY :-

To determine the functional dependencies for these tables, we need to analyze how the attributes in each table determine each other. Here's a breakdown of the functional dependencies for each table:

login:
login_id → login_username, login_role_ID
login_role_ID → login_id (assuming one role per login for simplicity)

user:
user_id → user_name, user_mobile, user_email, user_address
user_email → user_id (assuming one email per user for simplicity)

role:
role_id → role_name

MedicalShop:
ms_id → ms_name, ms_type, ms_address

Medicines:
mdcn_id → mdcn_name, mdcn_type, mdcn_cost

Stocks:
stk_id → stk_items, stk_types, stk_num
user_id → user_name, user_mobile, user_email, user_address
user_email → user_id
user_id → medicines, stocks, role, medical_shop
medicines, stocks, role, medical_shop → user_id.

INITIAL TABLE:-

The initial table for medicines & products supplying system

```
mysql> select * from medicine_products;
+---------+-------------+-------------+----------------+-------------+-------------------------+--------+-----------+--------------+
| user_id | user_name   | user_mobile | user_email     | user_address | medicines               | stocks | role      | medical_shop |
+---------+-------------+-------------+----------------+-------------+-------------------------+--------+-----------+--------------+
|     111 | hitesh reddy | 1234567890  | hitesh@gmail.com | 123 Main St  | Ciprofloxacin,paracetamol |    100 | Pharmacist | ABC Pharmacy |
|     222 | akshay varun | 0987654321  | akshay@gmail.com | 456 Elm St   | Diclofenac,Aspirin      |     50 | Pharmacist | XYZ Pharmacy |
|     333 | naga surya  | 5551234567  | surya@gmail.com  | 789 Oak St   | Loratadine,Amoxicillin  |     75 | Pharmacist | PQR Pharmacy |
+---------+-------------+-------------+----------------+-------------+-------------------------+--------+-----------+--------------+
3 rows in set (0.00 sec)
```

NORMAL FORMS:-

- **First Normal Form(1NF):-** In 1NF, all the data in a table is organized into rows and columns, and each column contains atomic values (indivisible). There should be no repeating groups of columns.
- Each column in a table should contain simple, indivisible values.
  Every column should have a unique name.
  The order of rows and columns doesn't matter.

```
mysql> select * from medicine_products_1nf;
+---------+--------------+-------------+-----------------+--------------+-------------+--------+-----------+-------------+
| user_id | user_name    | user_mobile | user_email      | user_address | medicine    | stocks | role      | medical_shop |
+---------+--------------+-------------+-----------------+--------------+-------------+--------+-----------+-------------+
|     111 | hitesh reddy | 1234567890  | hitesh@gmail.com | 123 Main St  | Ciprofloxacin |    100 | Pharmacist | ABC Pharmacy |
|     111 | hitesh reddy | 1234567890  | hitesh@gmail.com | 123 Main St  | Paracetamol |    100 | Pharmacist | ABC Pharmacy |
|     222 | akshay varun | 0987654321  | akshay@gmail.com | 456 Elm St   | Aspirin     |     50 | Pharmacist | XYZ Pharmacy |
|     222 | akshay varun | 0987654321  | akshay@gmail.com | 456 Elm St   | Diclofenac  |     50 | Pharmacist | XYZ Pharmacy |
|     333 | naga surya   | 5551234567  | surya@gmail.com  | 789 Oak St   | Amoxicillin |     75 | Pharmacist | PQR Pharmacy |
|     333 | naga surya   | 5551234567  | surya@gmail.com  | 789 Oak St   | Loratadine  |     75 | Pharmacist | PQR Pharmacy |
+---------+--------------+-------------+-----------------+--------------+-------------+--------+-----------+-------------+
6 rows in set (0.00 sec)
```

Second Normal Form(2NF):- we need to ensure that the table is already in the first normal form (1NF) and that itdoesn't contain any partial dependencies. Partial dependencies occur when part of the primary key determinates other attribute.

```
mysql> select * from medicine_products_users;
+---------+--------------+-------------+------------------+--------------+
| user_id | user_name    | user_mobile | user_email       | user_address |
+---------+--------------+-------------+------------------+--------------+
|     111 | hitesh reddy | 1234567890  | hitesh@gmail.com | 123 Main St  |
|     222 | akshay varun | 0987654321  | akshay@gmail.com | 456 Elm St   |
|     333 | naga surya   | 5551234567  | surya@gmail.com  | 789 Oak St   |
+---------+--------------+-------------+------------------+--------------+
3 rows in set (0.00 sec)

mysql> select * from medicine_products_medicines;
+---------+--------------+--------+
| user_id | medicine     | stocks |
+---------+--------------+--------+
|     111 | Ciprofloxacin |    100 |
|     111 | Paracetamol  |    100 |
|     222 | Aspirin      |     50 |
|     222 | Diclofenac   |     50 |
|     333 | Amoxicillin  |     75 |
|     333 | Loratadine   |     75 |
+---------+--------------+--------+
6 rows in set (0.00 sec)
```

```
mysql> select * from medicine_products_roles;
+---------+------------+
| user_id | role       |
+---------+------------+
|     111 | Pharmacist |
|     222 | Pharmacist |
|     333 | Pharmacist |
+---------+------------+
3 rows in set (0.00 sec)

mysql> select * from medicine_products_shops;
+---------+--------------+
| user_id | medical_shop |
+---------+--------------+
|     111 | ABC Pharmacy |
|     222 | XYZ Pharmacy |
|     333 | PQR Pharmacy |
+---------+--------------+
3 rows in set (0.00 sec)
```

Third Normal Form(3NF):-For 3NF, the table must first be in 2NF. Furthermore, all attributes must be functionally dependent on the primary key, and there shouldbe no transitive dependencies.

```
mysql> select * from medicines;
+-------------+---------------+--------+
| medicine_id | medicine_name | stocks |
+-------------+---------------+--------+
|           1 | Ciprofloxacin |    100 |
|           2 | Paracetamol   |    100 |
|           3 | Diclofenac    |     50 |
|           4 | Aspirin       |     50 |
|           5 | Loratadine    |     75 |
|           6 | Amoxicillin   |     75 |
+-------------+---------------+--------+
6 rows in set (0.00 sec)
```

```
mysql> select * from user_medicines;
+---------+-------------+
| user_id | medicine_id |
+---------+-------------+
|     111 |           1 |
|     111 |           2 |
|     222 |           3 |
|     222 |           4 |
|     333 |           5 |
|     333 |           6 |
+---------+-------------+
6 rows in set (0.00 sec)
```

Boyce-Codd Normal Form(BCNF):- To apply Boyce-Codd Normal Form (BCNF) to the tables, we need to ensurethat for every functional dependency X->Y in the table, where X and Y are sets of attributes, either:

1. X is a superkey, or
   Every attribute in Y is a part of a candidate key

```
mysql> select * from user;
+---------+--------------+------------+-----------------------------------------+--------------+
| user_id | user_name    | user_mobile | user_email                              | user_address |
+---------+--------------+------------+-----------------------------------------+--------------+
|     111 | John Doe     | 1234567890 | john.doe@example.com,hitesh@gmail.com    | 123 Main St  |
|     222 | Jane Smith   | 0987654321 | jane.smith@example.com,akshay@gmail.com  | 456 Elm St   |
|     333 | Alice Johnson | 5551234567 | alice.johnson@example.com,surya@gmail.com | 789 Oak St   |
+---------+--------------+------------+-----------------------------------------+--------------+
3 rows in set (0.00 sec)
```

Fourth Normal Form(4NF):- To achieve 4th Normal Form (4NF), we need to ensure that the table is free from multi-valued dependencies. A table is in 4NF if it is in BCNFand has no non-trivial multi-valued dependencies.
Therefore, since the table is already in BCNF and does not have any non-trivial multi-valued dependencies, itautomatically satisfies the requirements for 4th Normal Form.

```
mysql> select * from user;
+---------+--------------+------------+-----------------------------------------+--------------+
| user_id | user_name    | user_mobile | user_email                              | user_address |
+---------+--------------+------------+-----------------------------------------+--------------+
|     111 | John Doe     | 1234567890 | john.doe@example.com,hitesh@gmail.com    | 123 Main St  |
|     222 | Jane Smith   | 0987654321 | jane.smith@example.com,akshay@gmail.com  | 456 Elm St   |
|     333 | Alice Johnson | 5551234567 | alice.johnson@example.com,surya@gmail.com | 789 Oak St   |
+---------+--------------+------------+-----------------------------------------+--------------+
3 rows in set (0.00 sec)
```

Fifth Normal Form (5NF):-Fifth Normal Form (5NF),also known as Project-Join Normal Form (PJNF), deals withdecomposing tables to eliminate redundancy that arises fromjoins. To achieve 5NF, we need to ensure that every join dependency in the table is a consequence of the candidate keys of the table.

In the table, there are no evident join dependencies apart from those implied by the primary key. Each attribute seems to be functionally determined by the primary key and does not rely on any other non-key attributes. Therefore, the table already seems to satisfy 5NF

```
mysql> select * from user;
+---------+--------------+-------------+-----------------------------------------+--------------+
| user_id | user_name    | user_mobile | user_email                              | user_address |
+---------+--------------+-------------+-----------------------------------------+--------------+
|     111 | John Doe     | 1234567890  | john.doe@example.com,hitesh@gmail.com   | 123 Main St  |
|     222 | Jane Smith   | 0987654321  | jane.smith@example.com,akshay@gmail.com | 456 Elm St   |
|     333 | Alice Johnson| 5551234567  | alice.johnson@example.com,surya@gmail.com| 789 Oak St  |
+---------+--------------+-------------+-----------------------------------------+--------------+
3 rows in set (0.00 sec)
```

CHAPTER 5

Implementation Of Concurrency Control And Recovery Mechanisms:

```
START TRANSACTION;

-- Inserting data into the `user` table
INSERT INTO `user` (`user_name`, `user_mobile`, `user_email`, `user_address`)
VALUES ('John Doe', '1234567890', 'john.doe@example.com', '123 Main St');
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

```
-- Inserting data into the `login` table
INSERT INTO `login` (`login_id`, `login_username`, `login_role_ID`)
VALUES (111, 'JohnDoe', 123456);

-- Inserting data into the `role` table
INSERT INTO `role` (`role_id`, `role_name`)
VALUES (123456, 'customer');

-- Inserting data into the `MedicalShop` table
INSERT INTO `MedicalShop` (`ms_id`, `ms_name`, `ms_type`, `ms_address`)
VALUES (99464, 'venkat', 'pharmacy', 'machilipatnam');

-- Inserting data into the `Medicines` table
INSERT INTO `Medicines` (`mdcn_id`, `mdcn_name`, `mdcn_type`, `mdcn_cost`)
VALUES (5672, 'dolo55', 'fever', 112);

-- Inserting data into the `Stocks` table
INSERT INTO `Stocks` (`stk_id`, `stk_items`, `stk_types`, `stk_num`)
VALUES (6786, 'dolo', 'fever', 500);

-- Check if all inserts are successful
-- If not, ROLLBACK the transaction
COMMIT;
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

-- If there's an error or something unexpected, rollback the transaction
ROLLBACK;

CHAPTER 6

BACKEND PYTHON WITH MYSQL CODE:

```
-- phpMyAdmin SQL Dump
-- version 5.0.2
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jan 24, 2023 at 08:31 AM
-- Server version: 10.4.11-MariaDB
-- PHP Version: 7.2.29

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `medical`
--

--- ---------------------------------------------------------

--
-- Table structure for table `addmp`
--

CREATE TABLE `addmp` (
  `sno` int(11) NOT NULL,
  `medicine` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `addmp`
--

INSERT INTO `addmp` (`sno`, `medicine`) VALUES
(1, 'Dolo 650'),
(2, 'Carpel 250 mg'),
(3, 'Azythromycin 500'),
(4, 'Azythromycin 250'),
```

```
(5, 'Rantac 300'),
(6, 'Omez'),
(7, 'Okacet'),
(8, 'Paracetomol');

--- ---------------------------------------------------------

--
-- Table structure for table `addpd`
--

CREATE TABLE `addpd` (
  `sno` int(11) NOT NULL,
  `product` varchar(200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `addpd`
--

INSERT INTO `addpd` (`sno`, `product`) VALUES
(1, 'colgate'),
(2, 'perfume'),
(3, 'garnier face wash'),
(4, 'garnier face wash');

--- ---------------------------------------------------------

--
-- Table structure for table `logs`
--

CREATE TABLE `logs` (
  `id` int(11) NOT NULL,
  `mid` int(11) NOT NULL,
  `action` varchar(500) NOT NULL,
  `date` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `logs`
--

INSERT INTO `logs` (`id`, `mid`, `action`, `date`) VALUES
(1, 1001, ' INSERTED', '2023-01-24 12:54:41'),
(2, 1001, ' DELETED', '2023-01-24 12:57:48');

--- ---------------------------------------------------------

--
-- Table structure for table `medicines`
--
```

```
CREATE TABLE `medicines` (
  `id` int(11) NOT NULL,
  `amount` int(11) NOT NULL,
  `name` varchar(100) NOT NULL,
  `medicines` varchar(500) NOT NULL,
  `products` varchar(500) NOT NULL,
  `email` varchar(50) NOT NULL,
  `mid` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Triggers `medicines`
--
DELIMITER $$
CREATE TRIGGER `Delete` BEFORE DELETE ON `medicines` FOR EACH ROW INSERT INTO Logs
VALUES(null,OLD.mid,' DELETED',NOW())
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER `Insert` AFTER INSERT ON `medicines` FOR EACH ROW INSERT INTO Logs
VALUES(null,NEW.mid,' INSERTED',NOW())
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER `Update` AFTER UPDATE ON `medicines` FOR EACH ROW INSERT INTO Logs
VALUES(null,NEW.mid,' UPDATED',NOW())
$$
DELIMITER ;

-- --------------------------------------------------------

--
-- Table structure for table `posts`
--

CREATE TABLE `posts` (
  `mid` int(11) NOT NULL,
  `medical_name` varchar(100) NOT NULL,
  `owner_name` varchar(100) NOT NULL,
  `phone_no` varchar(20) NOT NULL,
  `address` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `posts`
--

INSERT INTO `posts` (`mid`, `medical_name`, `owner_name`, `phone_no`, `address`) VALUES
(1001, 'ARK PROCODER MEDICAL', 'ANEES', '7896541230', 'Bangalore');

--
```

```
-- Indexes for dumped tables
--

--
-- Indexes for table `addmp`
--
ALTER TABLE `addmp`
  ADD PRIMARY KEY (`sno`);

--
-- Indexes for table `addpd`
--
ALTER TABLE `addpd`
  ADD PRIMARY KEY (`sno`);

--
-- Indexes for table `logs`
--
ALTER TABLE `logs`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `medicines`
--
ALTER TABLE `medicines`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `posts`
--
ALTER TABLE `posts`
  ADD PRIMARY KEY (`mid`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `addmp`
--
ALTER TABLE `addmp`
  MODIFY `sno` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

--
-- AUTO_INCREMENT for table `addpd`
--
ALTER TABLE `addpd`
  MODIFY `sno` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT for table `logs`
--
```

```
ALTER TABLE `logs`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `medicines`
--
ALTER TABLE `medicines`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `posts`
--
ALTER TABLE `posts`
  MODIFY `mid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1002;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

FRONT END CODE:-

```
from flask import Flask, render_template, request, session, redirect, flash
from flask_sqlalchemy import SQLAlchemy
import json
with open('config.json','r') as c:
    params = json.load(c)["params"]
local_server  = True
app = Flask(_name_)
app.secret_key = 'super-secret-key'
if(local_server):
    app.config['SQLALCHEMY_DATABASE_URI'] = params['local_uri']
else:
    app.config['SQLALCHEMY_DATABASE_URI'] = params['proud_uri']
db = SQLAlchemy(app)
class Medicines(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    amount = db.Column(db.Integer, nullable=False)
    name = db.Column(db.String(500), nullable=False)
    medicines= db.Column(db.String(500), nullable=False)
    products = db.Column(db.String(500), nullable=False)
    email = db.Column(db.String(120), nullable=False)
    mid = db.Column(db.String(120), nullable=False)
class Posts(db.Model):
```

```
    mid = db.Column(db.Integer, primary_key=True)
    medical_name = db.Column(db.String(80), nullable=False)
    owner_name = db.Column(db.String(200), nullable=False)
    phone_no = db.Column(db.String(200), nullable=False)
    address = db.Column(db.String(120), nullable=False)
class Addmp(db.Model):
    sno = db.Column(db.Integer,  primary_key=True)
    medicine = db.Column(db.String, nullable=False)
class Addpd(db.Model):
    sno = db.Column(db.Integer,  primary_key=True)
    product = db.Column(db.String, nullable=False)
class Logs(db.Model)
    id = db.Column(db.Integer, primary_key=True)
    mid = db.Column(db.String, nullable=True)
    action = db.Column(db.String(30), nullable=False)
    date = db.Column(db.String(100), nullable=False)


@app.route("/")
def hello():
    return render_template('index.html', params=params)
@app.route("/index")
def home():
    return render_template('dashbord.html', params=params)
@app.route("/search",methods=['GET','POST'])
def search():
    if request.method == 'POST':
```

```
        name = request.form.get('search')
        post = Addmp.query.filter_by(medicine=name).first()
        pro = Addpd.query.filter_by(product=name).first()
        if (post or pro):
            flash("Item Is Available.", "primary")
        else:
            flash("Item is not Available.", "danger")
    return render_template('search.html', params=params)
@app.route("/details", methods=['GET','POST'])
def details():
    if ('user' in session and session['user'] == params['user']):
        posts =Logs.query.all()
        return render_template('details.html', params=params, posts=posts)
@app.route("/aboutus")
def aboutus():
    return render_template('aboutus.html', params=params)
@app.route("/insert", methods = ['GET','POST'])
def insert():
    if (request.method == 'POST'):
        '''ADD ENTRY TO THE DATABASE'''
        mid=request.form.get('mid')
        medical_name = request.form.get('medical_name')
        owner_name = request.form.get('owner_name')
        phone_no = request.form.get('phone_no')
        address = request.form.get('address')
        push = Posts(mid=mid,medical_name=medical_name, owner_name=owner_name,
```

```
phone_no=phone_no, address=address)
        db.session.add(push)
        db.session.commit()
        flash("Thanks for submitting your details","danger")
    return render_template('insert.html',params=params)
@app.route("/addmp", methods = ['GET','POST'])
def addmp():
    if (request.method == 'POST'):
        '''ADD ENTRY TO THE DATABASE'''
        newmedicine = request.form.get('medicine')
        push=Addmp(medicine=newmedicine,)
        db.session.add(push)
        db.session.commit()
        flash("Thanks for adding new items", "primary")
    return render_template('search.html', params=params)
@app.route("/addpd", methods = ['GET','POST'])
def addpd():
    if (request.method == 'POST'):
        '''ADD ENTRY TO THE DATABASE'''
        new product = request.form.get('product')
        push=Addpd(product=newproduct,)
        db.session.add(push)
        db.session.commit()
        flash("Thanks for adding new items", "primary")
    return render_template('search.html', params=params)
@app.route("/list",methods=['GET','POST'])
```

```
def post():
    if ('user' in session and session['user'] == params['user']):


        posts=Medicines.query.all()
        return render_template('post.html', params=params, posts=posts)
@app.route("/items",methods=['GET','POST'])
def items():
    if ('user' in session and session['user'] == params['user']):
        posts=Addmp.query.all()
        return render_template('items.html', params=params,posts=posts)
@app.route("/items2", methods=['GET','POST'])
def items2():
    if ('user' in session and session['user'] == params['user']):
        posts=Addpd.query.all()
        return render_template('items2.html',params=params,posts=posts)
@app.route("/sp",methods=['GET','POST'])
def sp():
    if ('user' in session and session['user'] == params['user']):
        posts=Medicines.query.all()
        return render_template('store.html', params=params,posts=posts)
@app.route("/logout")
def logout():
    session.pop('user')
    flash("You are logout", "primary")
    return redirect('/login')
@app.route("/login",methods=['GET','POST'])
```

```
def login():
    if ('user' in session and session['user'] == params['user']):
        posts = Posts.query.all()
        return render_template('dashbord.html',params=params,posts=posts)
    if request.method=='POST':
        username=request.form.get('uname')
        userpass=request.form.get('password')
        if(username==params['user'] and userpass==params['password']):
            session['user']=username
            posts=Posts.query.all()
            flash("You are Logged in", "primary")
            return render_template('index.html',params=params,posts=posts)
        else:
            flash("wrong password", "danger")
    return render_template('login.html', params=params)


@app.route("/edit/<string:mid>",methods=['GET','POST'])
def edit(mid):
    if('user' in session and session['user']==params['user']):
        if request.method =='POST':
            medical_name=request.form.get('medical_name')
            owner_name=request.form.get('owner_name')
            phone_no=request.form.get('phone_no')
            address=request.form.get('address')
            if mid==0:
```

```
posts=Posts(medical_name=medical_name,owner_name=owner_name,phone_no=phone_no,address=addre
ss)


            db.session.add(posts)
            db.session.commit()
        else:
            post=Posts.query.filter_by(mid=mid).first()
            post.medical_name=medical_name
            post.owner_name=owner_name
            post.phone_no=phone_no
            post.address=address
            db.session.commit()
            flash("data updated ", "success")
            return redirect('/edit/'+mid)
    post = Posts.query.filter_by(mid=mid).first()
    return render_template('edit.html',params=params,post=post)
#       if user is logged in
#delete
@app.route("/delete/<string:mid>", methods=['GET', 'POST'])
def delete(mid):
   if ('user' in session and session['user']==params['user']):
      post=Posts.query.filter_by(mid=mid).first()
      db.session.delete(post)
      db.session.commit()
      flash("Deleted Successfully", "warning")
   return redirect('/login')
```

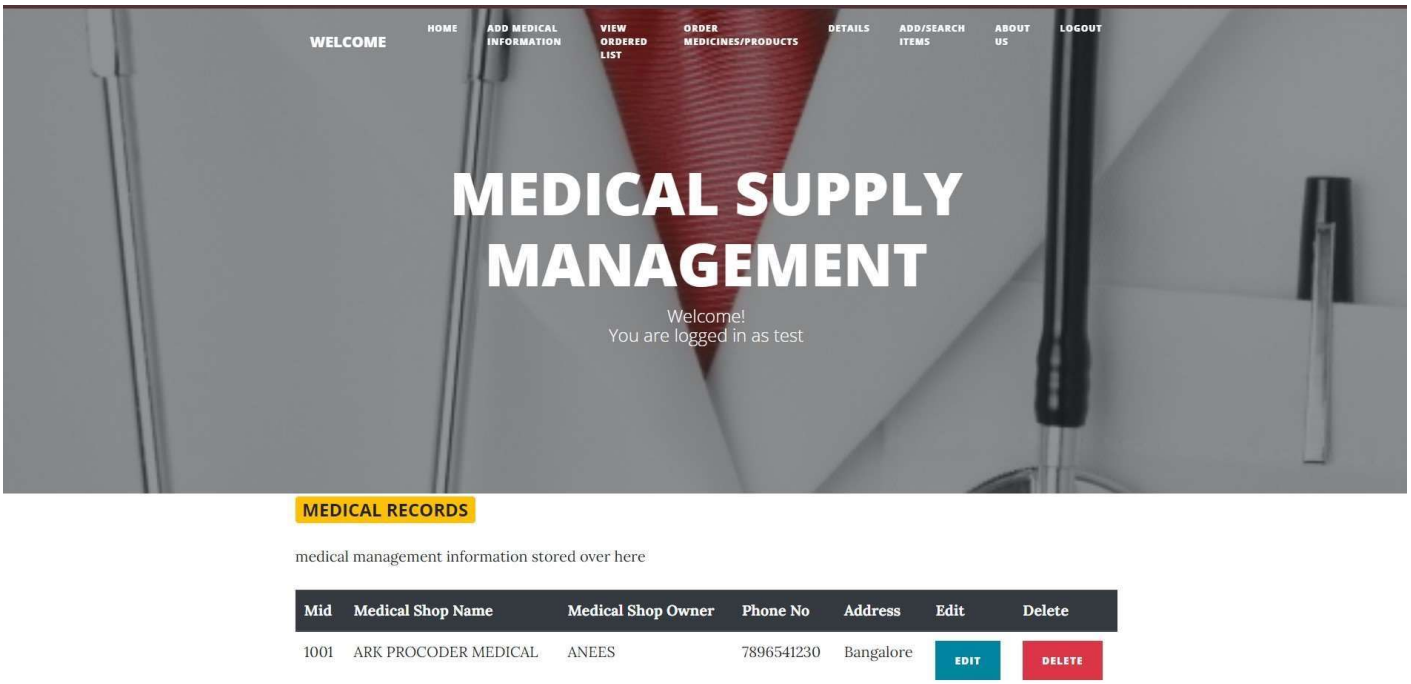
```
@app.route("/deletemp/<string:id>", methods=['GET', 'POST'])
def deletemp(id):
    if ('user' in session and session['user']==params['user']):
        post=Medicines.query.filter_by(id=id).first()
        db.session.delete(post)
        db.session.commit()
        flash("Deleted Successfully", "primary")

    return redirect('/list')
@app.route("/medicines", methods = ['GET','POST'])
def medicine():
    if(request.method=='POST'):
        '''ADD ENTRY TO THE DATABASE'''
        mid=request.form.get('mid')
        name=request.form.get('name')
        medicines=request.form.get('medicines')
        products=request.form.get('products')
        email=request.form.get('email')
        amount=request.form.get('amount')
            entry=Medicines(mid=mid,name=name,medicines=medicines,products=products,email=email,amou
            nt=amount)
        db.session.add(entry)
        db.session.commit()
        flash("Data Added Successfully","primary")
    return render_template('medicine.html',params=params)
app.run(debug=True).
```

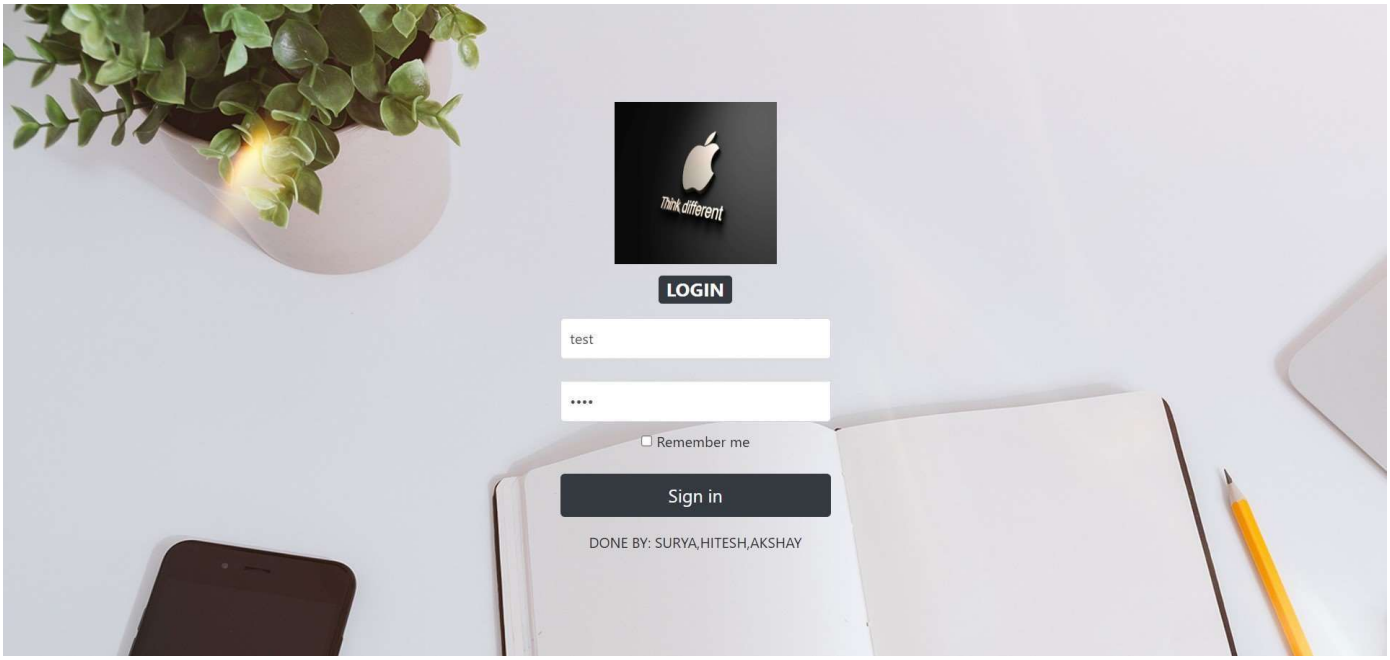
CHAPTER 7

SCREENSHOTS OF THE IMPLEMENTATION WITH FRONT END:-

HOME PAGE:-



LOGIN PAGE:-



ADD MEDICAL INFORMATION ABOUT THE SHOP:-

ADD DATA

enter medical id

enter medical name

Enter Owner name

Enter phone number

enter Address

TO VIEW CUSTOMERS ORDERS:-

medical management information stored over here

Mid	Medicines	Products	Amount	Delete
666	dolo 650	huggies	500	DELETE

TO ORDER MEDICINES & PRODUCTS:-

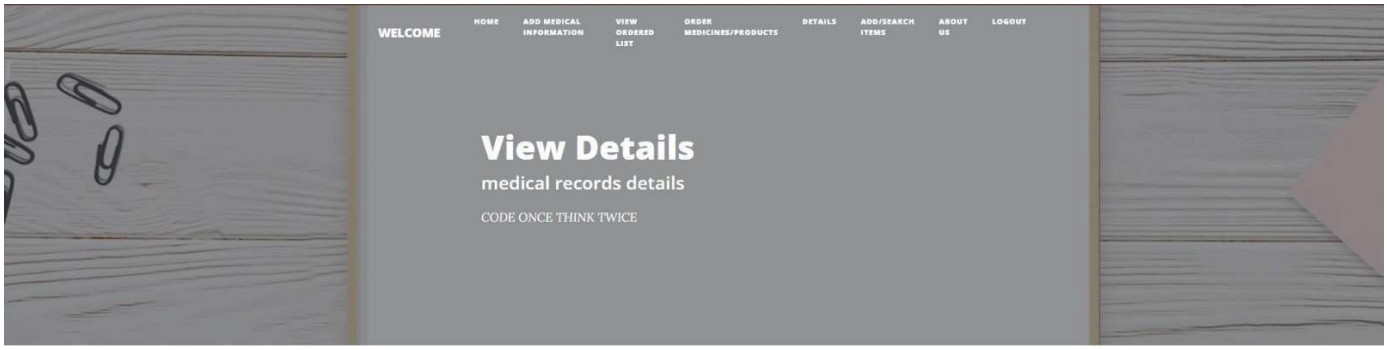
Want to get in touch? Fill out the form below to send me a message and I will get back to you as soon as possible!

enter id

Enter name

Enter medicines names

TO VIEW DETAILS FOR INSERTION OR DELETION:-



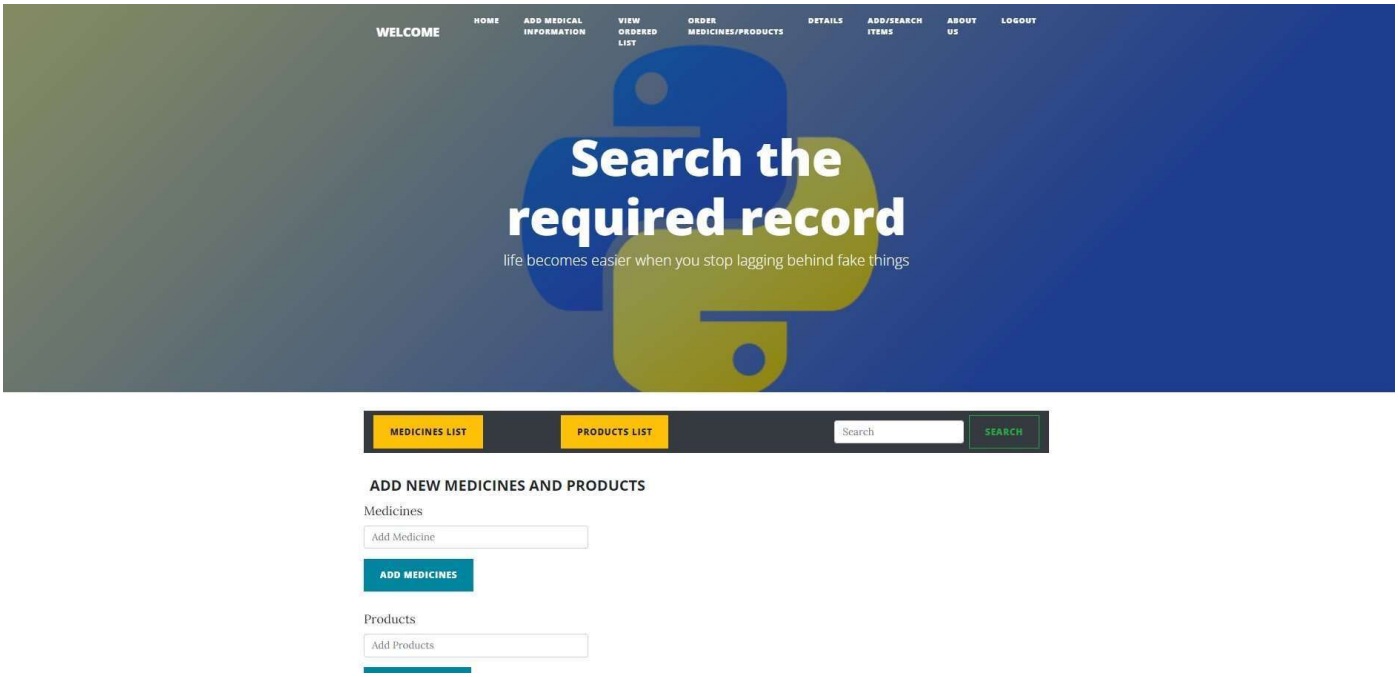
STORED DETAILS

Click here to view Recorded Information

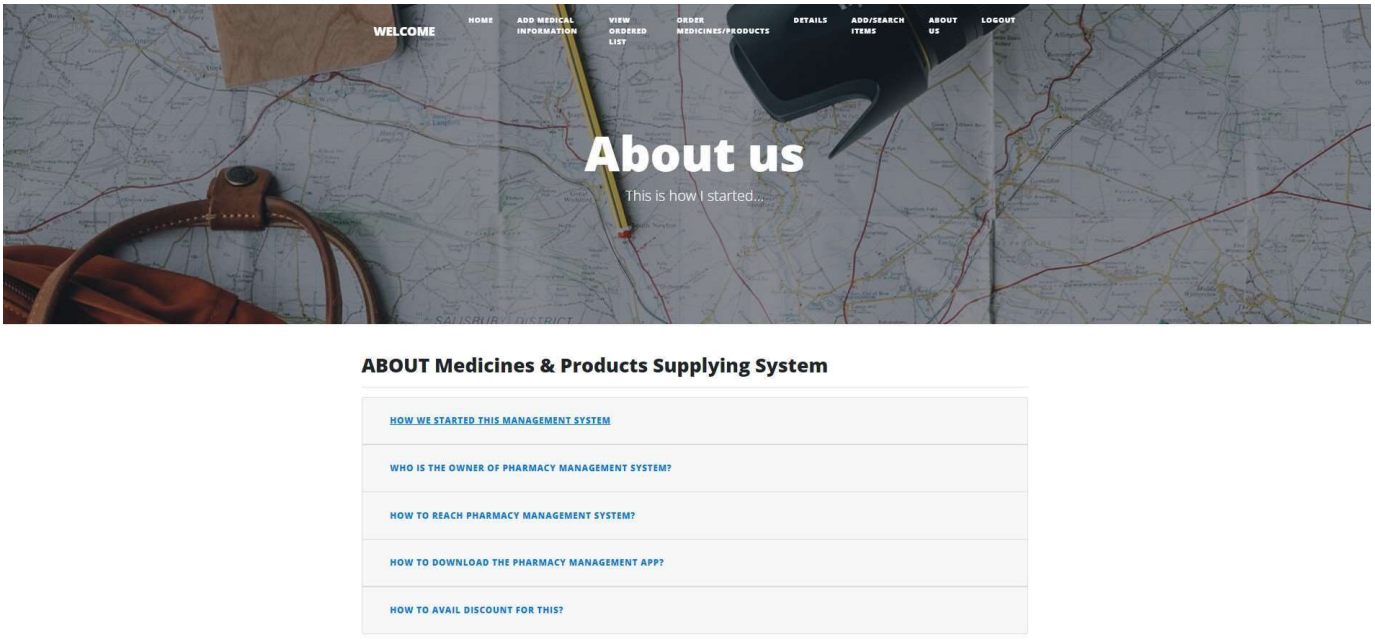
medical management information date and timings stored over here

mid	actions	Date
1001	INSERTED	2023-01-24 12:54:41
1001	DELETED	2023-01-24 12:57:48
666	INSERTED	2024-05-02 22:22:27

SEARCH FOR REQUIRED MEDICINE OR PRODUCTS:-

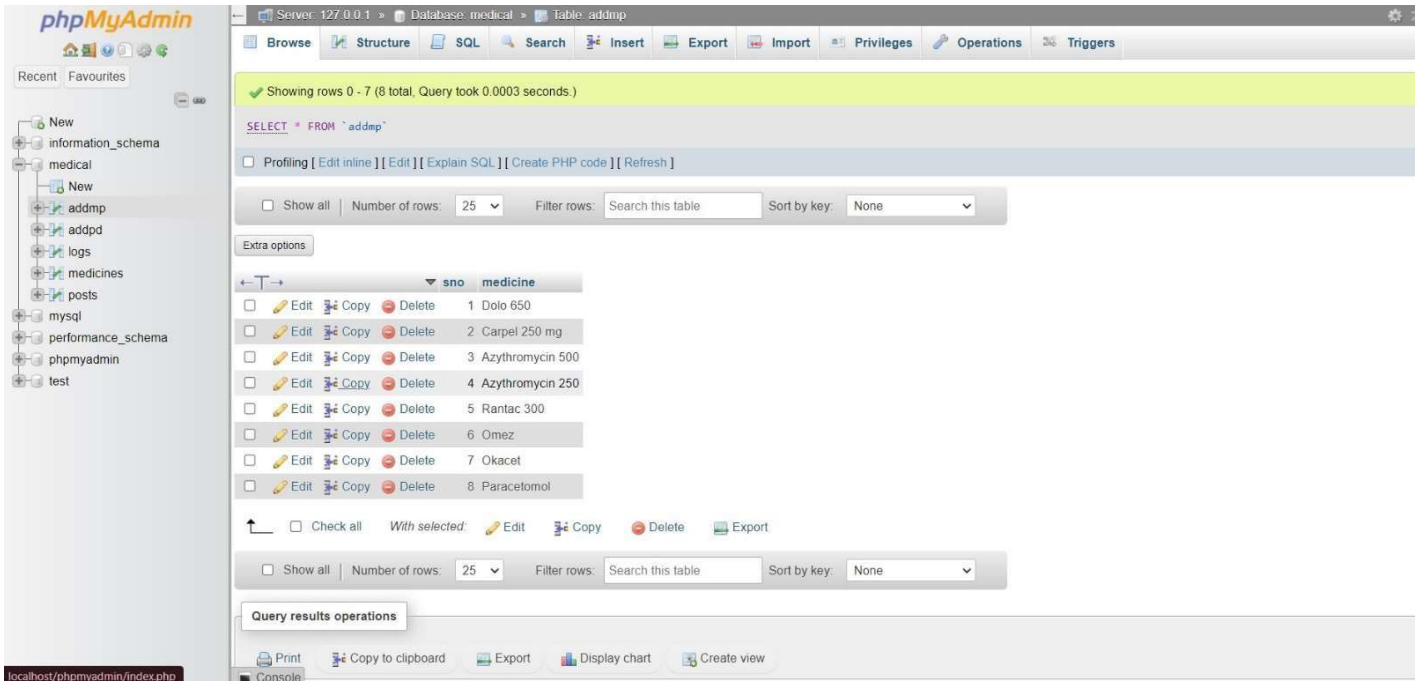


ABOUT THE MEDICAL SHOP:-

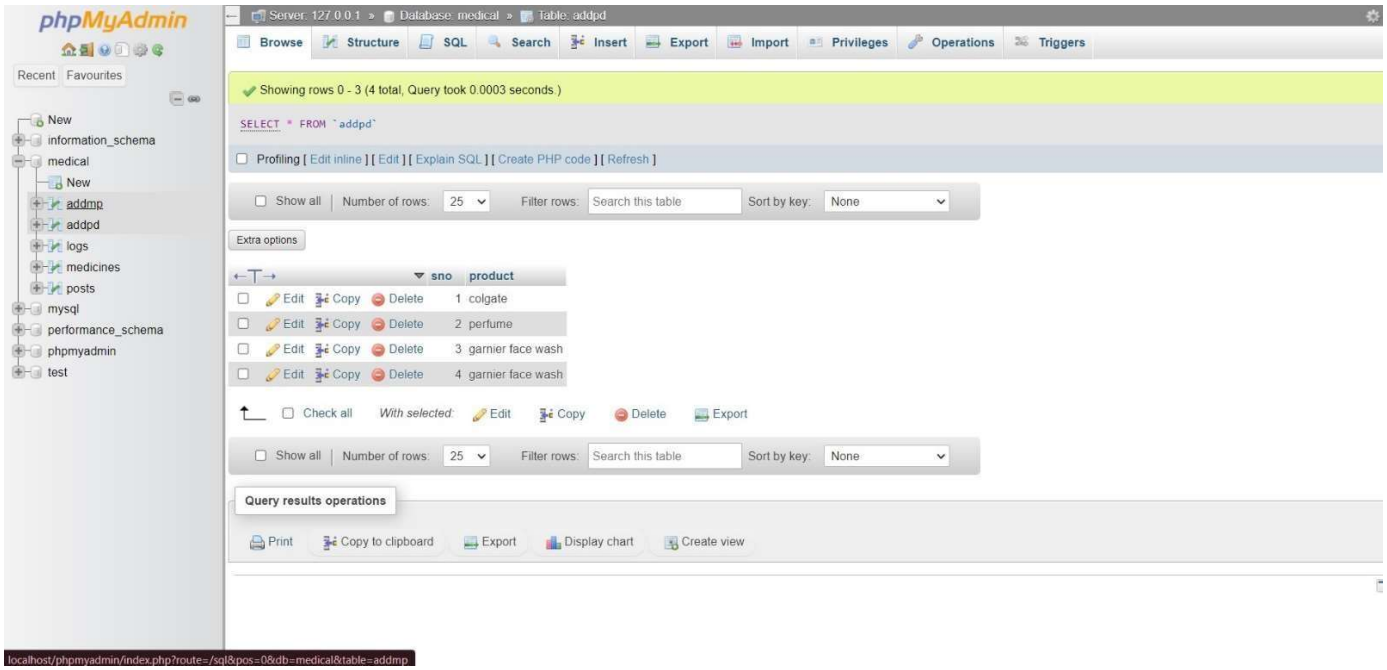


DATABASES:-

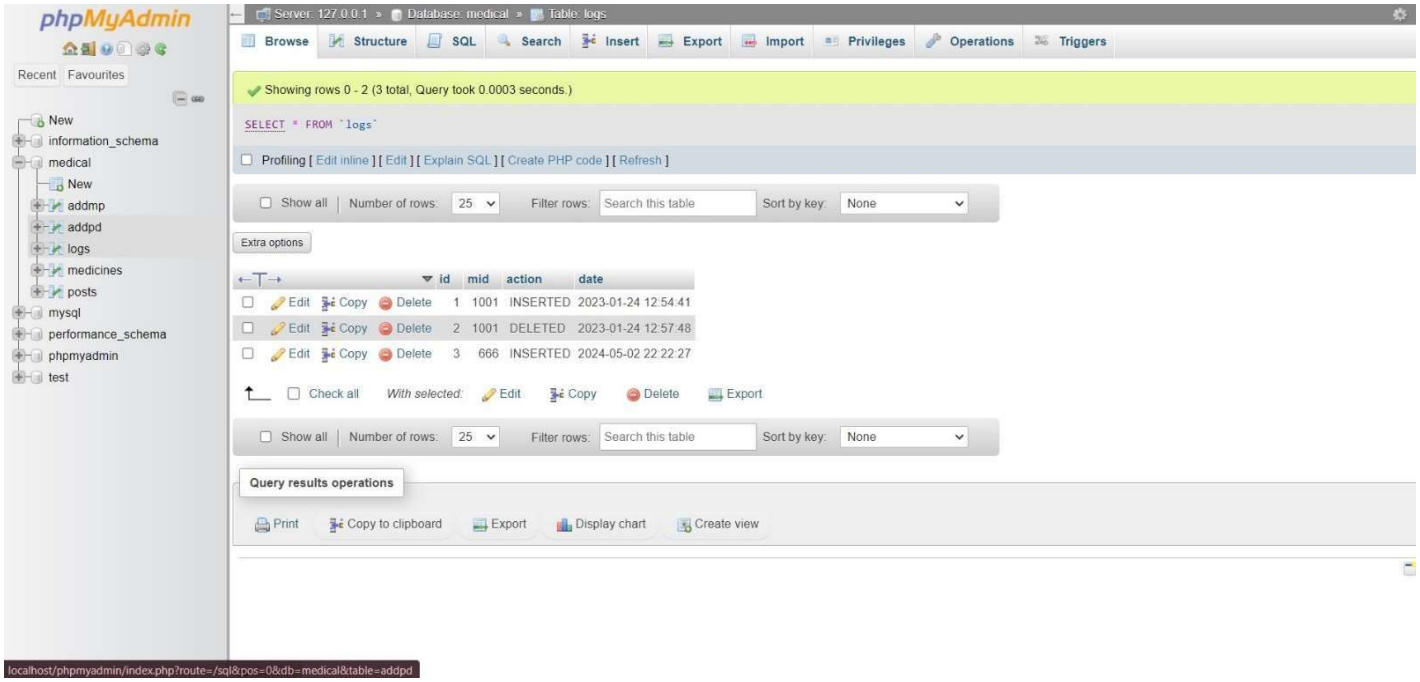
DATABASE OF MEDICINES:-



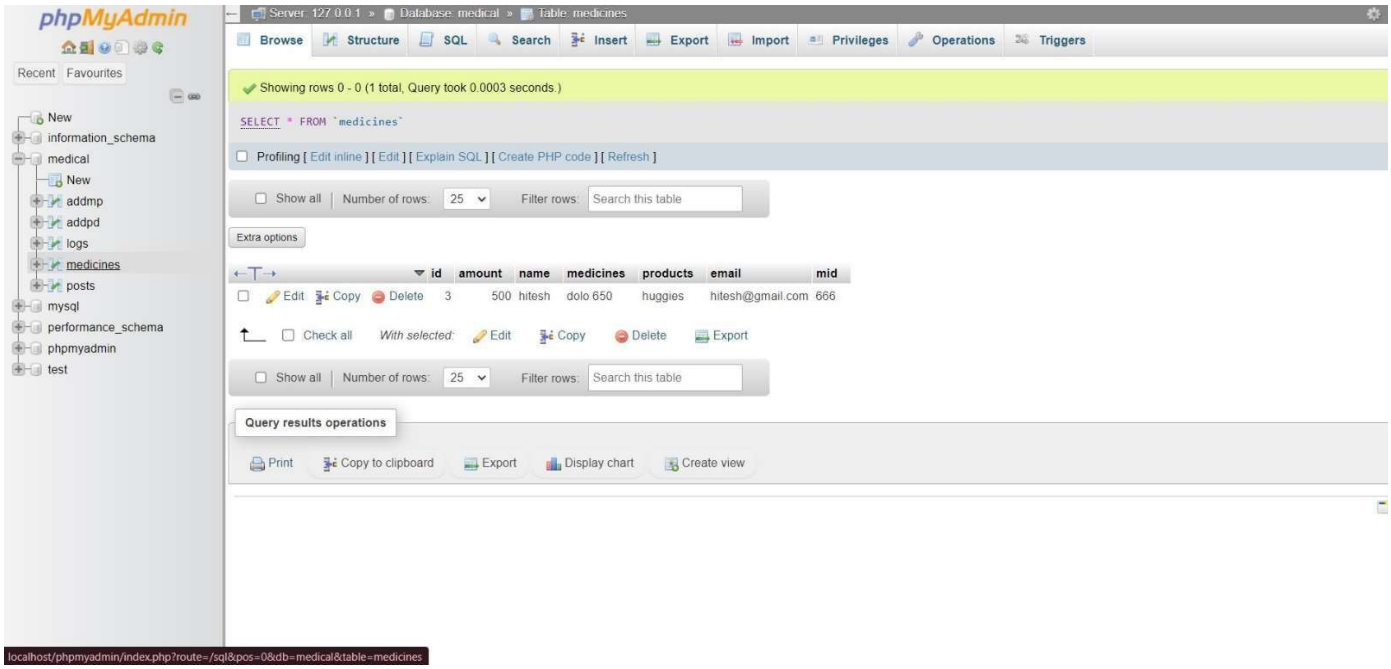
DATABASE OF PRODUCTS:-



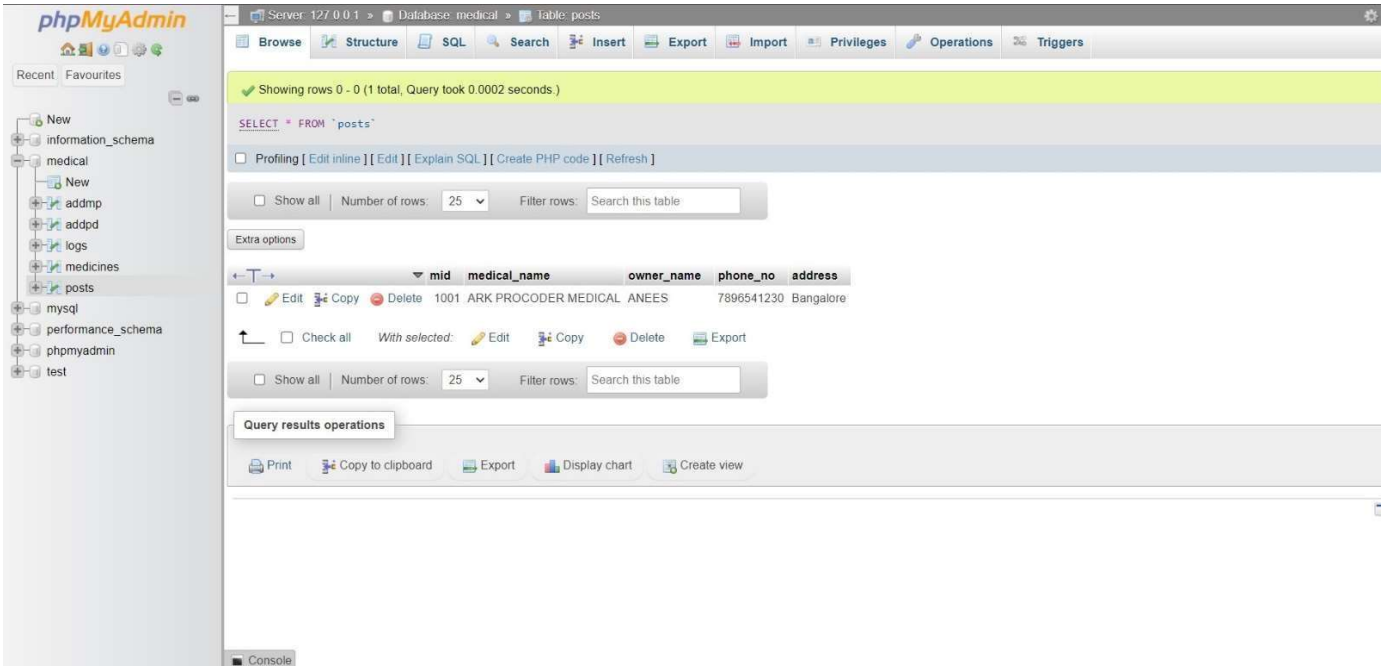
LOGS(Time of Insertion & Deletion):-



DATABASE OF CUSTOMER ORDERS:-



DATABASE FOR MEDICAL SHOPS:-



CONCLUSION:-

The efficient functioning of a medicines and products supplying system is crucial for ensuring the availability, affordability, and quality of essential healthcare commodities. Through a comprehensive analysis, several key considerations have emerged, pointing towards the need for a multifaceted approach to address existing challenges and optimize supply chain operations. Firstly, the integration of digital solutions such as blockchain and IoT offers promising opportunities to enhance supply chain transparency, traceability, and efficiency. By leveraging these technologies, stakeholders can mitigate issues related to counterfeit drugs, stockouts, and expiration of medicines, thereby safeguarding patient safety and public health. Moreover, collaborative partnerships among pharmaceutical manufacturers, distributors, healthcare facilities, and regulatory bodies are indispensable for optimizing procurement, inventory management, and distribution channels. Through shared data and resources, stakeholders can streamline supply chain processes, improve forecasting accuracy, and respond promptly to evolving healthcare needs. Proactive monitoring and forecasting of demand patterns are also critical for maintaining optimal stock levels and preventing shortages. By leveraging data analytics tools and predictive models, decision-makers can anticipate demand fluctuations, optimize resource allocation, and minimize supply chain disruptions. Additionally, ensuring the quality and safety of medicines throughout the supply chain remains paramount. Stringent adherence to regulatory standards, rigorous quality control measures, and robust cold chain management are essential for preserving the efficacy of pharmaceutical products and safeguarding patient health. By embracing a holistic approach that integrates technology, collaboration, and data-driven insights, stakeholders can address existing gaps, optimize supply chain operations, and improve access to essential healthcare commodities. Ultimately, these efforts contribute to improving patient outcomes, advancing public health objectives, and fostering a more resilient and sustainable healthcare system.

REFERENCES:-

- **Youtube**
- **Github**
- **Bootstraps**