

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Pharmacy supply management systems play a crucial role in the efficient operation of healthcare facilities, particularly pharmacies. These systems are designed to streamline the management of pharmaceutical inventory, procurement, distribution, and supplier relationships. The implementation of robust supply management systems addresses various challenges faced by pharmacies, including medication errors, stockouts, expired medications, and compliance with regulatory requirements.

Importance in Healthcare

In healthcare settings, pharmacies are responsible for managing a wide range of medications and supplies essential for patient care. Effective supply management ensures that medications are available when needed, reduces waste, and enhances patient safety through accurate dispensing practices. Without dedicated systems, pharmacies may struggle with manual tracking methods that are prone to errors and inefficiencies.

1.2 OBJECTIVES AND GOALS

- The main objective of the project is to design and develop a user friendly-system
- Easy to use and an efficient computerized system.
- To develop an accurate and flexible system, it will eliminate data redundancy.
- To study the functioning of pharmacy supply management System.
- To make a software fast in processing, with good user interface.
- To make software with good user interface so that user can change it and it should be used for a long time without error and maintenance.
- To provide synchronized and centralized farmer and seller database.
- Computerization can be helpful as a means of saving time and money.
- To provide better Graphical User Interface (GUI).

CHAPTER 2

PROBLEM STATEMENT

2.1 EXISTING SYSTEM

Rising debt, cost-cutting, and layoffs in health care-delivery facilities, alluded to earlier. Models for the design and operation of supply chain networks may be steady state or dynamic and may be deterministic or deal with uncertainties (particularly in product demands). Research in this field started very early on, with location-allocation problems forming part of the earlier set of “classical” operations research problems. The gap between the growing demand and available supply of high-quality, cost effective, and timely health care continues to be a daunting challenge not only in developing and underdeveloped countries, but also in developed countries. Further, the issues involved with the supply chain design in developing countries are prevalent in developed countries, especially with the rising number of uninsured and jobless among the patient populations and with the budget deficits. Thus the project is a sincere effort in simplifying the tasks of administrators in an easily usable format.

2.2 PROPOSED SYSTEM

While there has been no consensus on the definition of Pharmacy Supply Management in the literature, they have proposed that researchers adopt the below definition to allow for the coherent development of theory in the area. In order to have a successful supply management, we need to make many decisions related to the flow of information, product, and funds. Each decision should be made in a way to increase the whole supply chain profitability. Supply management is more complex in healthcare and other industries because of the impact on people’s health requiring adequate and accurate medical supply according to the patient’s need.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 FUNCTIONAL REQUIREMENTS

- 2 Real-time Inventory Tracking: Ensure accurate and up-to-date monitoring of medication and supply levels.
- 3 Batch and Expiry Management: Track batch numbers and expiry dates to prevent dispensing expired medications.
- 4 Supplier Management: Maintain supplier information, track performance, and manage contracts efficiently.
- 5 Regulatory Compliance: Generate reports and maintain documentation to comply with regulatory standards (e.g., FDA, DEA).
- 6 Reporting and Analytics: Provide dashboards and analytics tools for monitoring inventory levels, trends, and compliance metrics.

3.2 SOFTWARE REQUIREMENTS:

Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.7) , SQLAlchemy,

- Operating System: Windows 10
- Google Chrome/Internet Explorer
- AMPPS (Version-3.7)
- Python main editor (user interface): PyCharm Community
- workspace editor: Sublime text 3

HARDWARE REQUIREMENTS:

- Computer with a 1.1 GHz or faster processor
- Minimum 2GB of RAM or more
- 2.5 GB of available hard-disk space
- 5400 RPM hard drive
- 1366 × 768 or higher-resolution display
- DVD-ROM drive

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 SYSTEM ARCHITECTURE

A system architecture serves as the blueprint that defines the arrangement and interaction of system components, guiding the implementation and operation of the system towards fulfilling its intended purpose. It encompasses the organization and integration of software, hardware, data, and networking components, designed to meet specific functional requirements while adhering to performance, security, and scalability standards.

1. Database Layer :Explain the role of the inventory and patient databases in storing critical information such as product details, stock levels, patient prescriptions, and medication histories.

Application Layer :

2.Inventory Management Software: Discuss how it manages stock levels, orders, and generates reports.

3.Order Management System: Explain its role in facilitating seamless ordering processes with suppliers.

4.Integration Layer

Supplier Integration: Describe how the system connects with suppliers for ordering and receiving pharmaceutical products.

5.Security Layer

Discuss the security measures implemented to protect sensitive data, ensuring compliance with healthcare regulations

6.User Interface Layer :Describe user interfaces designed for pharmacy staff and patients to interact with the system effectively.

4.2 CONCEPTUAL DESIGN:

4.2.1 E-R DIAGRAM:

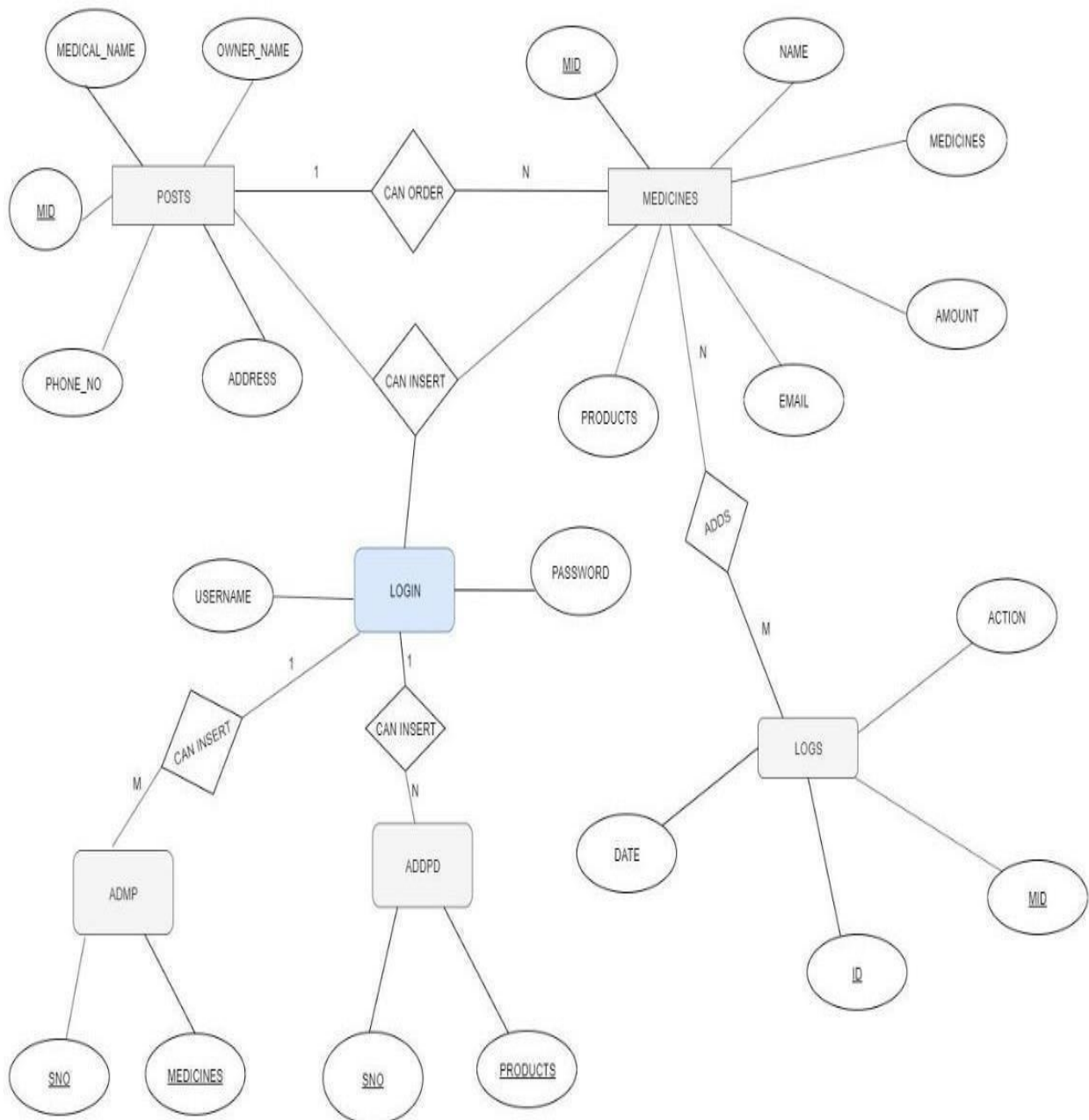


Fig.4.2.1-ER Diagram

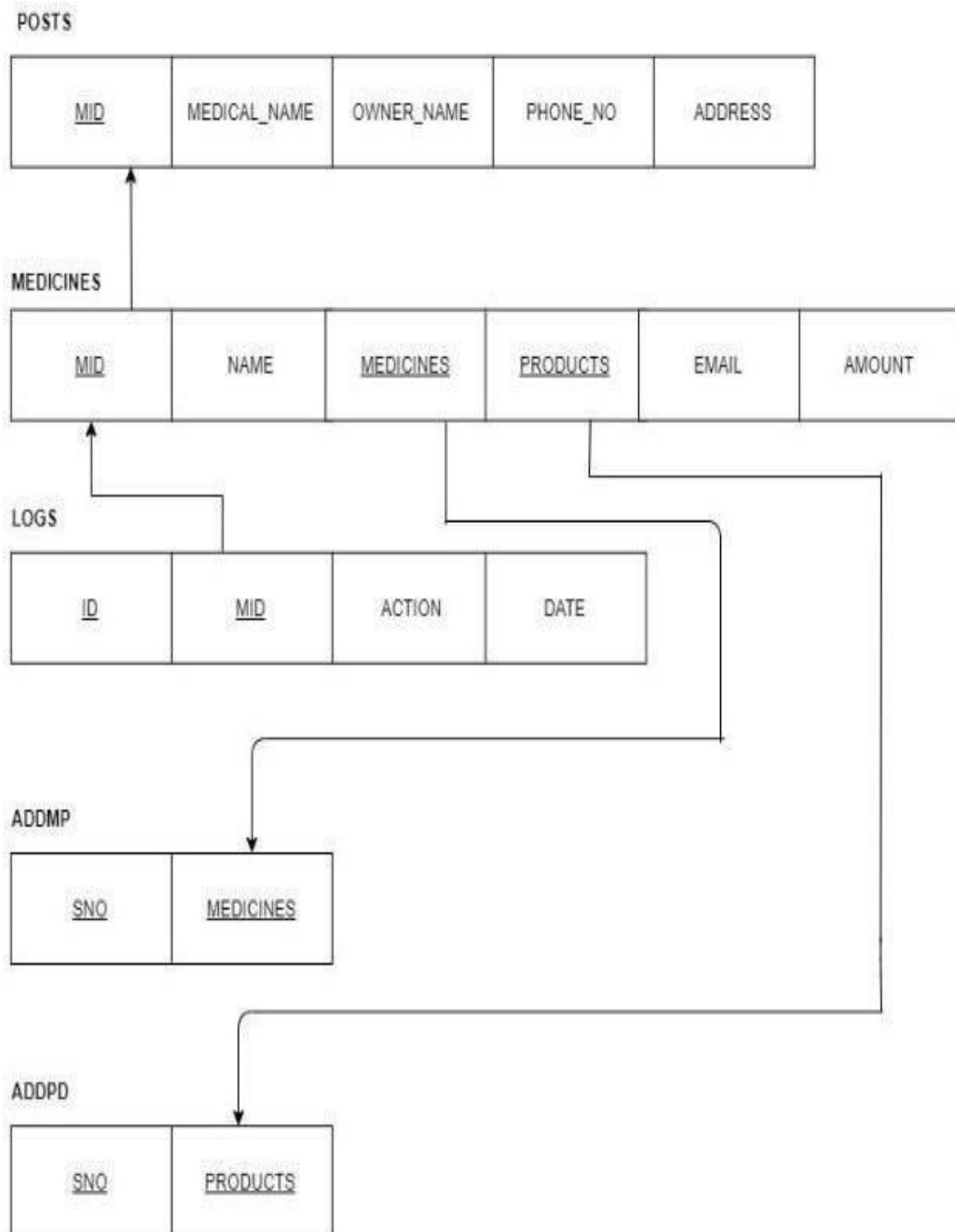
4.2.2 SCHEMA DIAGRAM:

Fig.4.2.2-Schema Diagram

CHAPTER 5

IMPLEMENTATION

An "implementation" of Python should be taken to mean a program or environment which provides support for the execution of programs written in the Python language, as represented by the [CPython](#) reference implementation.

There have been and are several distinct software packages providing of what we all recognize as Python, although some of those are more like distributions or variants of some existing implementation than a completely new implementation of the language.

Back End (MySQL)

Database:

A Database Management System (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems. Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

- A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
- The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.
- Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device.

- Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called sub schemas. For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data.
- If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.
- ✓ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).
- It also maintains the integrity of the data in the database.
- The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

SQL:

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

- In the relational model, data is stored in structures called relations or tables.
- SQL statements are issued for the purpose of:
- Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes).

5.1: Stored Procedure

Routine name: proc1, post proc

Type: procedure

Definition: Select * from posts;

Select * from medicines;

5.2: Triggers

It is the special kind of stored procedure that automatically executes when an event occurs in the database.

Triggers used :

1: Trigger name: on insert Table:

medicines

Time: after Event:

insert

Definition: INSERT INTO logs VALUES(null, new.mid, 'inserted', NOW()); 2:

Trigger name: on delete

Table: medicines

Time: after

Event: delete

Definition: INSERT INTO logs VALUES(null, old.mid, 'deleted', NOW());

BACKEND PYTHON WITH MYSQL CODE

```
from flask import Flask, render_template, request, session, redirect, flash
from flask_sqlalchemy import SQLAlchemy
from flask_mail import Mail
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'
    app.config.update(
        MAIL_SERVER='smtp.gmail.com',
        MAIL_PORT='465',
        MAIL_USE_SSL=True,

        MAIL_USERNAME=params['gmail-user'],
        MAIL_PASSWORD=params['gmail-password'])
    mail = Mail(app)
    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 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("/index")
def home():
    return render_template('dashbord.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")
        @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("/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)
    #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')
            post=Medicines.query.filter_by(id=id).first()
            db.session.delete(post)
            db.session.commit()
            flash("Deleted Successfully", "primary")
    @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,amount=amount)

db.session.add(entry)
db.session.commit()

mail.send_message('new message send from ' + name, sender=email,
recipients=[params['gmail-user']],

body= 'Medicines ordered--->' +medicines + "\n" 'products ordered---> ' + products + "\n"
'customer id --->' + email + "\n" 'total amount=' +amount)

flash("Data sent and Added Successfully","primary")
return render_template('medicine.html',params=params)
app.run(debug=True)
```

FRONTEND HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<title>{{ params['blog_name'] }}</title>
<!-- Bootstrap core CSS -->
<link href="{{ url_for('static',filename='vendor/bootstrap/css/bootstrap.min.css') }}"
rel="stylesheet">
<!-- Custom fonts for this template -->
<link href="{{ url_for('static',filename='vendor/fontawesome-free/css/all.min.css') }}"
rel="stylesheet" type="text/css">
<link href='https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic'
rel='stylesheet' type='text/css'>
<link
href='https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700ita
lic,800italic,400,300,600,700,800' rel='stylesheet' type='text/css'>
<!-- Custom styles for this template -->
<link href="{{ url_for('static',filename='css/clean-blog.min.css') }}" rel="stylesheet">
</head>
<body>
<!-- Navigation -->
<nav class="navbar navbar-expand-lg navbar-light fixed-top" id="main Nav">
<div class="container">
<a class="navbar-brand" href="#">{{ params['blog_name'] }}</a>
<!-- <button class="navbar-toggler navbar-toggler-right" type="button" data-
toggle="collapse" data-target="#navbar Responsive" aria-controls="navbar Responsive" aria-
expanded="false" aria-label="Toggle navigation">
```

Menu

```
</button> -->
<div class="collapse navbar-collapse" id="navbar Responsive">
<ul class="navbar-nav ml-auto">
<li class="nav-item">
<a class="nav-link" href="/login">home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/insert">add medical information</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/list">view ordered list</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/medicines ">ADD medicines/products</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/details ">Details</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/search ">add/search</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/aboutus">about us</a>
</li>
<!-- <li class="nav-item">
<a class="nav-link" href="/add">search</a>
</li> -->
<li class="nav-item">
<a onclick="return confirm('Are you sure to logout?');" class="nav-link"
href="/logout">logout</a>
</li>
</ul>
</div>
</div>
</nav>
```

```

{% block body %} {% endblock %}

<hr>
<!-- Footer -->
<footer>
<div class="container">
<div class="row">
<div class="col-lg-8 col-md-10 mx-auto">
<ul class="list-inline text-center">
<li class="list-inline-item">
<a href="{ { params['tw_url'] } }" target=_blank>
<span class="fa-stack fa-lg">
<i class="fas fa-circle fa-stack-2x"></i>
<i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
</span>
</a>
</li>
<li class="list-inline-item">
<a href="{ { params['fb_url'] } }" target=_blank>
<span class="fa-stack fa-lg">
<i class="fas fa-circle fa-stack-2x"></i>
<i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i></span>
</a>
</li>
<li class="list-inline-item">
<a href="{ { params['gh_url'] } }" target=_blank>
<span class="fa-stack fa-lg">
<i class="fas fa-circle fa-stack-2x"></i>
<i class="fab fa-github fa-stack-1x fa-inverse"></i>
</span>
</a>
</li>
</ul>
<p class="copyright text-muted">Copyright &copy; Your dbms website 2019</p>
</div>
</div>
</div>

```


2.

Dept. of CSE, CEC

```
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
<span aria-hidden="true">&times;</span>
</button>
</div>
{% endfor %}
{% endif %}
{% endwith %}
<!-- Main Content -->
<div class="container">
<div class="row">
<div class="col-lg-8 col-md-10 mx-auto">
<p>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!</p>
<form name="sendMessage" id="contactForm" novalidate action="/medicines"
method="post">

<div class="control-group">
<div class="form-group floating-label-form-group controls">
<label>mid</label>
<input type="number" name='mid' class="form-control" placeholder="enter id" id="mid"
required data-validation-required-message="Please enter id" >
<p class="help-block text-danger"></p>
</div>
</div>

<div class="control-group">
<div class="form-group floating-label-form-group controls">
<label>name</label>
<input type="text" name="name" class="form-control" placeholder="Enter name" id="name"
required data-validation-required-message="Please enter name">
<p class="help-block text-danger"></p>
</div>
</div>

<div class="control-group">
<div class="form-group floating-label-form-group controls">
<label>Medicines</label>
```

```
<textarea rows="5" class="form-control" placeholder="Enter medicines names"
id="medicines" name="medicines" required data-validation-required-message="Please enter
a medicines/products/syrups" ></textarea>

<p class="help-block text-danger"></p>
</div>
</div>
<div class="control-group">
<div class="form-group floating-label-form-group controls">
<label>Products</label>
<textarea rows="5" class="form-control" placeholder="Enter products names" id="products"
name="products" required data-validation-required-message="Please enter a
medicines/products/syrups" ></textarea>

<p class="help-block text-danger"></p>
</div>
</div>
<div class="control-group">
<div class="form-group floating-label-form-group controls">
<label>Email id</label>
<input type="email" name="email" class="form-control" placeholder="Enter your mail id"
id="email" required data-validation-required-message="Please enter email">

<p class="help-block text-danger"></p>
</div>
</div>
<div class="control-group">
<div class="form-group floating-label-form-group controls">
<label>Amount</label>
<input type="email" name="amount" class="form-control" placeholder="Enter Amount"
id="amount" required data-validation-required-message="Please enter amount">

<p class="help-block text-danger"></p>
</div>
</div>
<br>
<div id="success"></div>
```

```
<div class="form-group">
  <button onclick="return confirm('Are you sure to SaveData?');" type="submit" class="btn
  btn-primary" id="sendMessageButton" >Send</button>

</div>
</form>
</div>
</div>
</div>
<hr>
<!-- Bootstrap core JavaScript -->
<script src="vendor/jquery/jquery.min.js"></script>
<script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<!-- Contact Form JavaScript -->
<script src="js/jqBootstrapValidation.js"></script>
<script src="js/contact_me.js"></script>
<!-- Custom scripts for this template -->
<script src="js/clean-blog.min.js"></script>
</body>
</html>
{% endblock %}
```

CHAPTER 6

RESULTS AND DISCUSSIONS

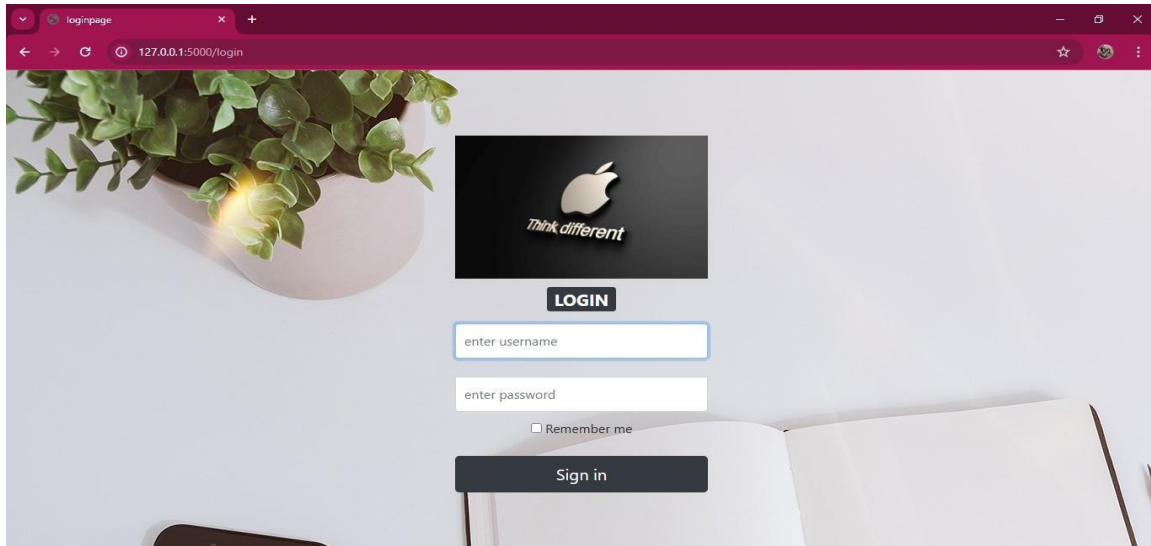


Fig.6.1-Login Page



Fig.6.2-Front Page

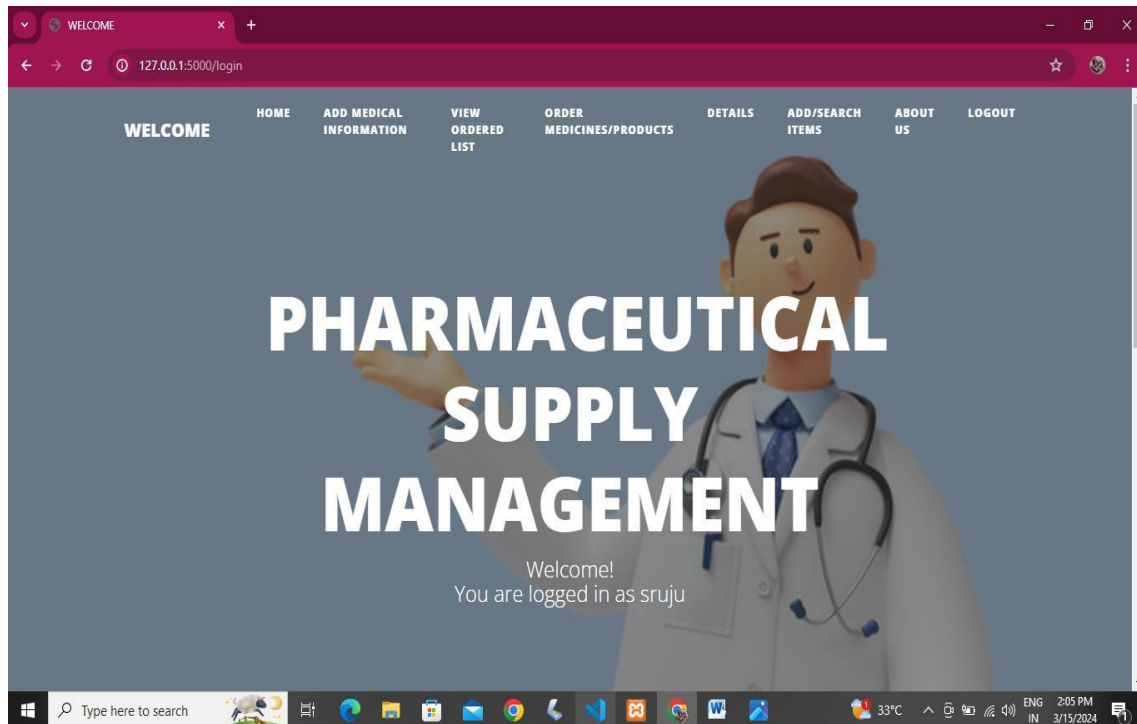


Fig.6.3-After Login

A screenshot of the 'ADD DATA' form in the system. The browser address bar shows '127.0.0.1:5000/insert'. The form contains several input fields with the following labels and values: 'MEDICAL ID' with value '1002', 'MEDICAL SHOP NAME' with value 'WOW medicines', 'OWNER NAME' with value 'srujana', 'PHONE NUMBER' with value '987654321', and 'ADDRESS' with value 'Bangalore'. At the bottom of the form is a blue button labeled 'INSERT DATA'.

Fig.6.4-Add Medicines Information

WELCOME HOME ADD MEDICAL INFORMATION VIEW ORDERED LIST ORDER MEDICINES/PRODUCTS DETAILS ADD/SEARCH ITEMS ABOUT US LOGOUT

Thanks for submitting your details

ADD DATA

enter medical id

enter medical name

Enter Owner name

Enter phone number

Fig.6.5-Add Medicines Information 2

127.0.0.1:5000/login

WELCOME HOME ADD MEDICAL INFORMATION VIEW ORDERED LIST ORDER MEDICINES/PRODUCTS DETAILS ADD/SEARCH ITEMS ABOUT US LOGOUT

MEDICAL RECORDS

medical management information stored over here

Mid	Medical Shop Name	Medical Shop Owner	Phone No	Address	Edit	Delete
123	wow pharmacy	srujana	987654321	Bangalore	EDIT	DELETE
1001	ARK PROCODER MEDICAL	ANEES	7896541230	Bangalore	EDIT	DELETE
1002	WOW medicines	srujana	987654321	Bangalore	EDIT	DELETE
2334	dfdfc	srujana	987654321	Bangalore	EDIT	DELETE

Fig.6.6-Medicine Information Editing

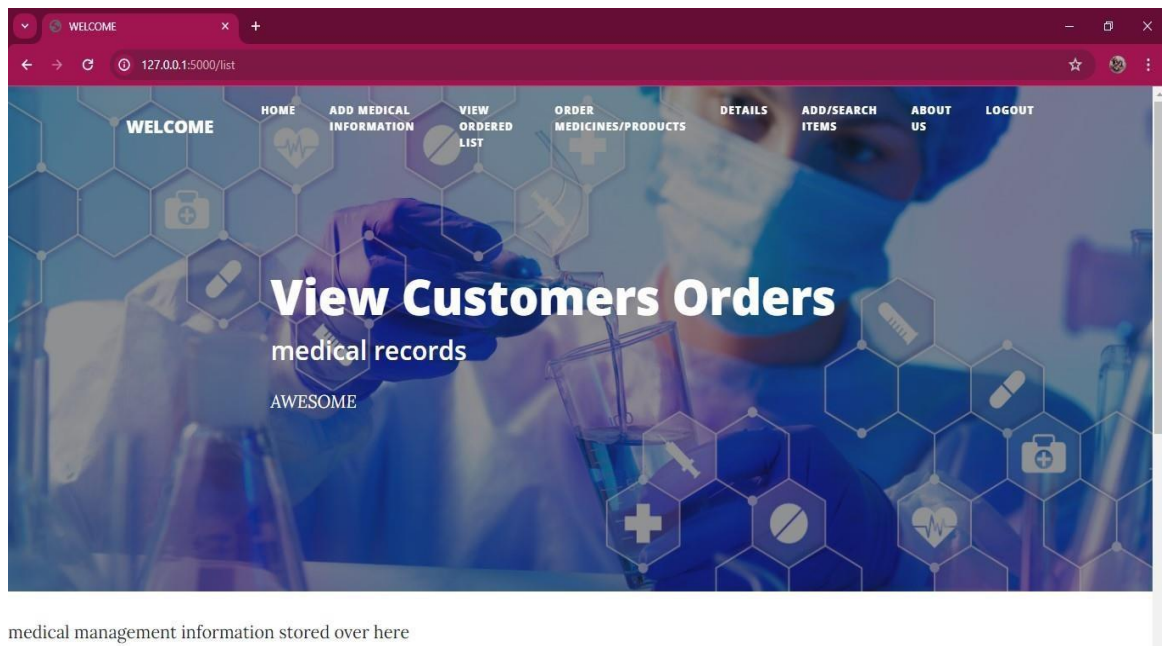


Fig.6.7-Medicine Information Stored

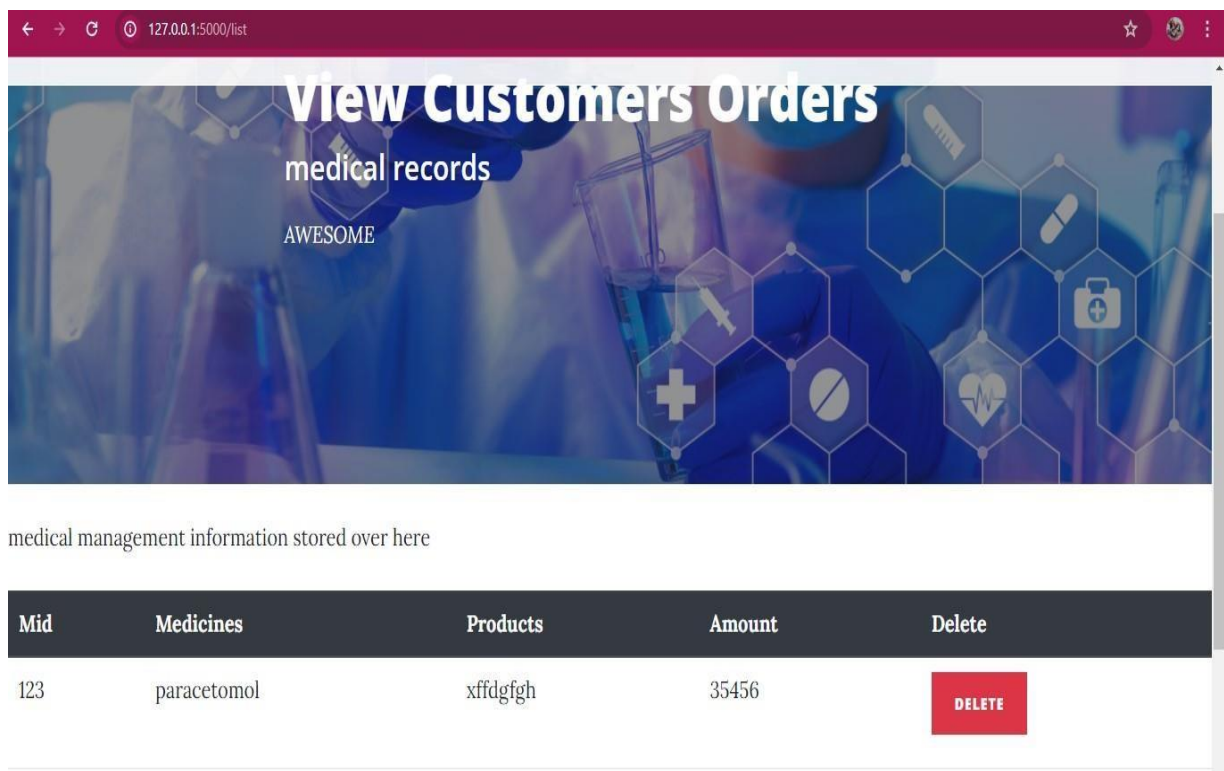
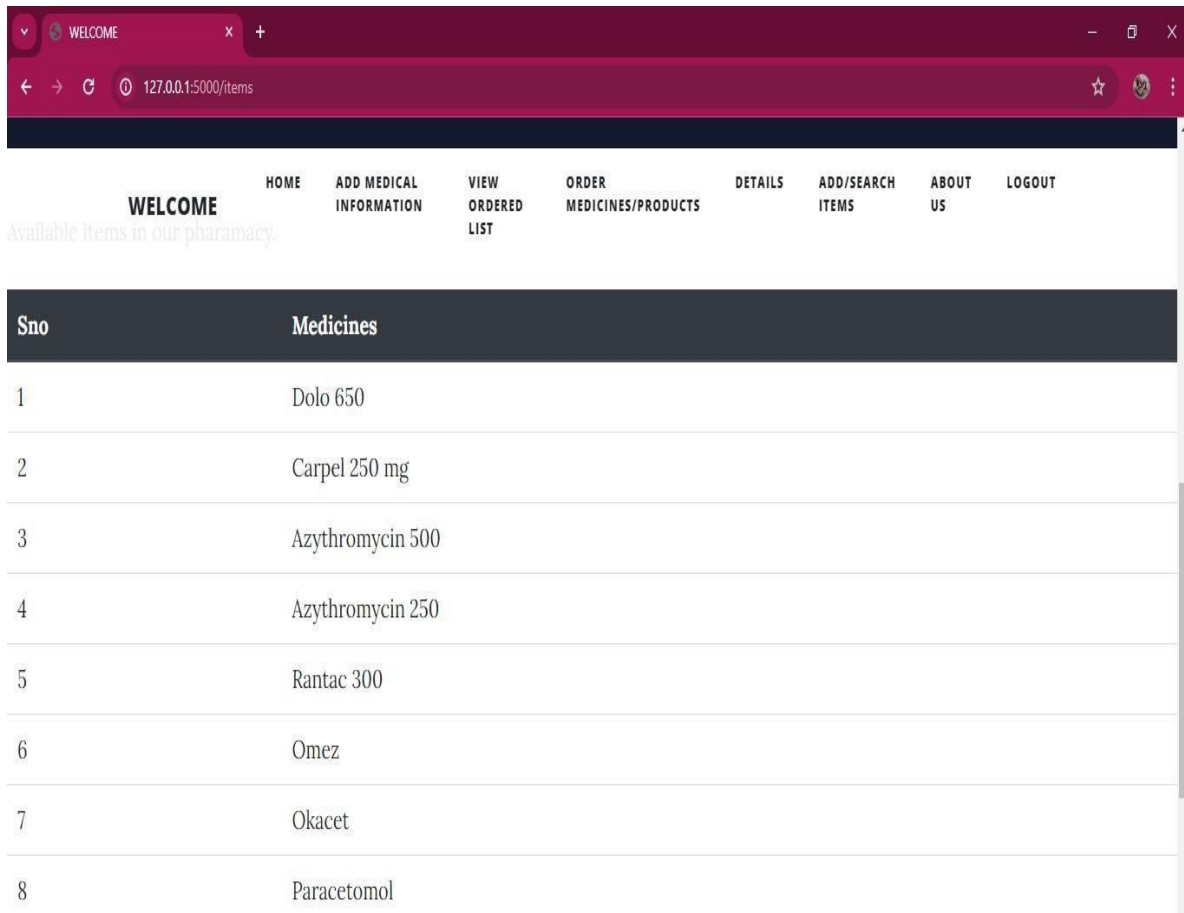


Fig.6.8-Medicine Information Deletion



Sno	Medicines
1	Dolo 650
2	Carpel 250 mg
3	Azythromycin 500
4	Azythromycin 250
5	Rantac 300
6	Omez
7	Okacet
8	Paracetamol

Fig.6.9-Login Database

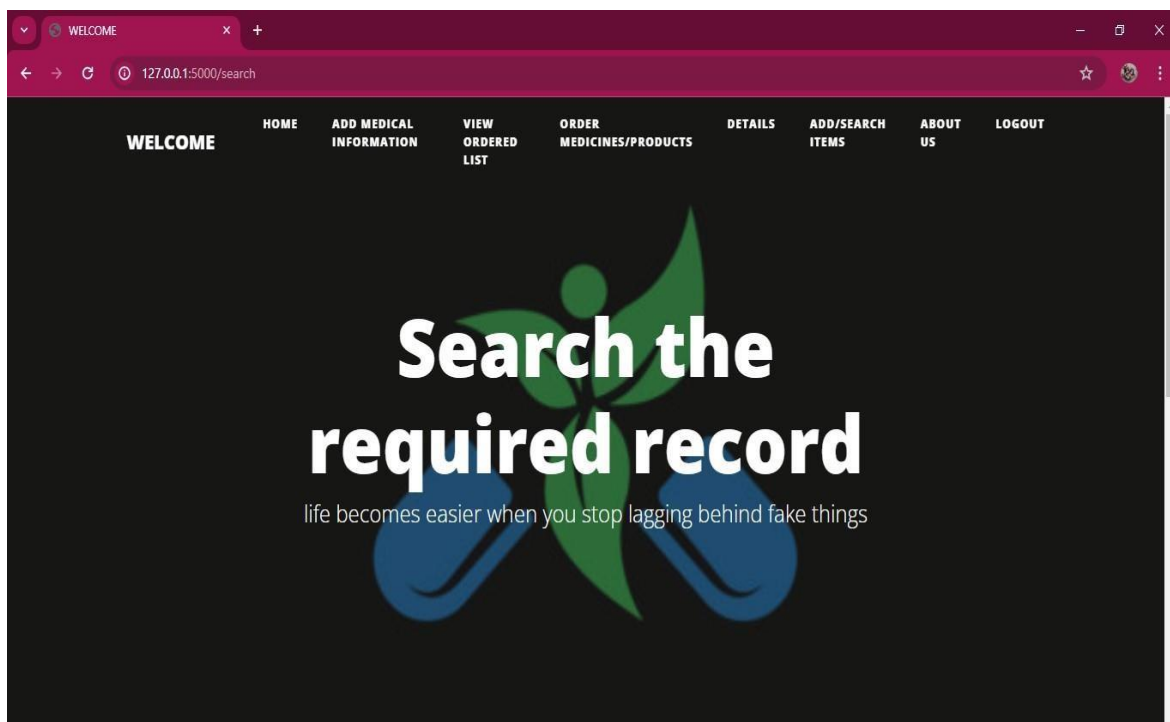


Fig.6.10-Medicine Search

The screenshot displays a web browser window with the URL `127.0.0.1:5000/search`. The page features a dark blue header with a navigation menu: **WELCOME**, **HOME**, **ADD MEDICAL INFORMATION**, **VIEW ORDERED LIST**, **ORDER MEDICINES/PRODUCTS**, **DETAILS**, **ADD/SEARCH ITEMS**, **ABOUT US**, and **LOGOUT**. Below the header, there is a search bar with a **Search** button. The main content area is titled **ADD NEW MEDICINES AND PRODUCTS**. It contains two sections: **Medicines** and **Products**. Each section has a text input field labeled 'Add Medicine' and 'Add Products' respectively, followed by a blue **ADD MEDICINES** button and a blue **ADD PRODUCTS** button.

Fig.6.11-Medicine Adding

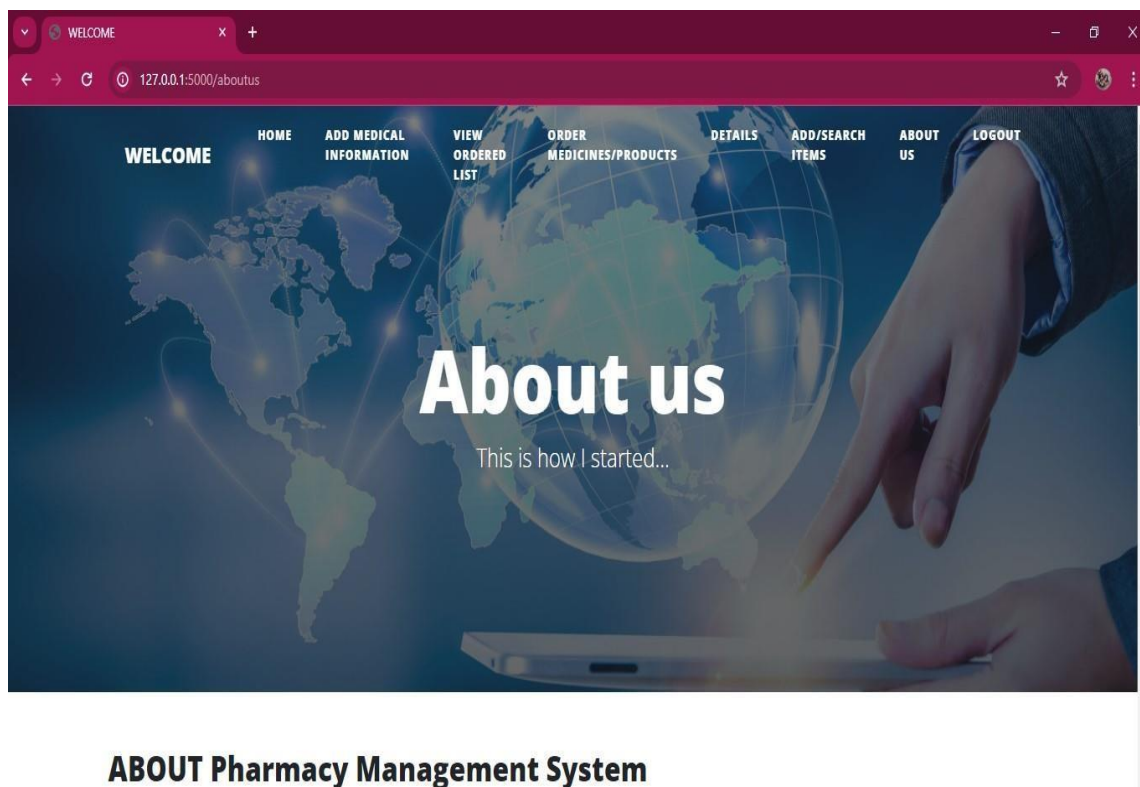


Fig.6.12-About Us

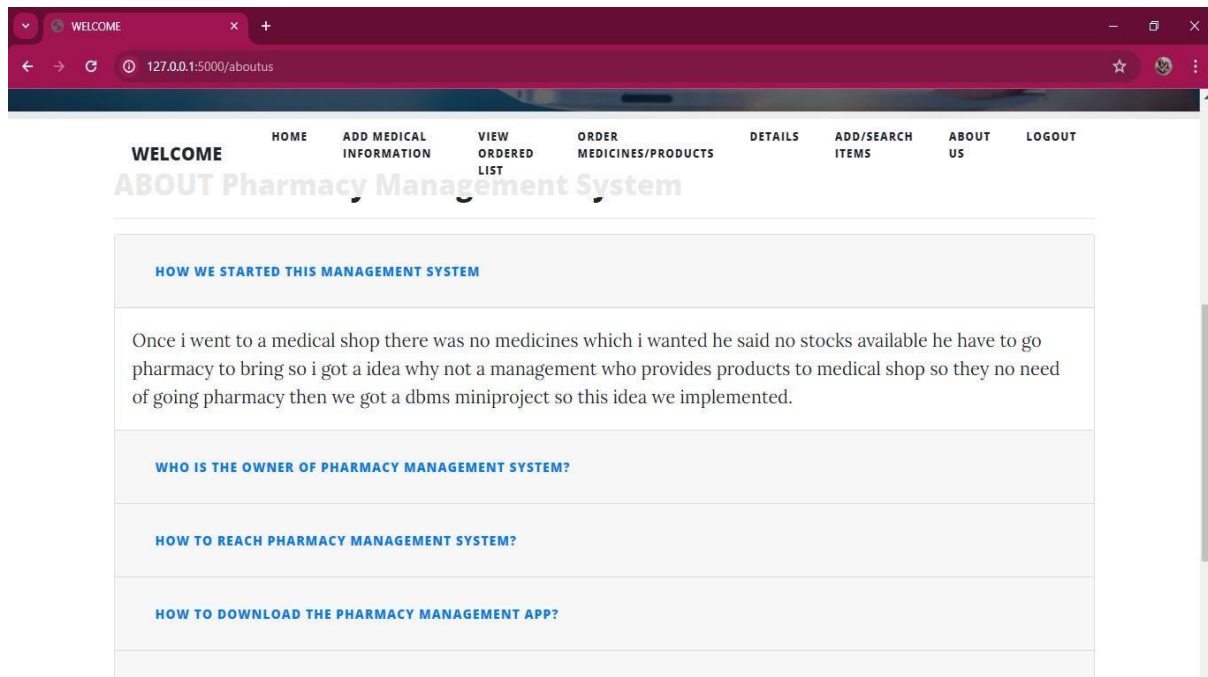


Fig.6.13-About Us Page

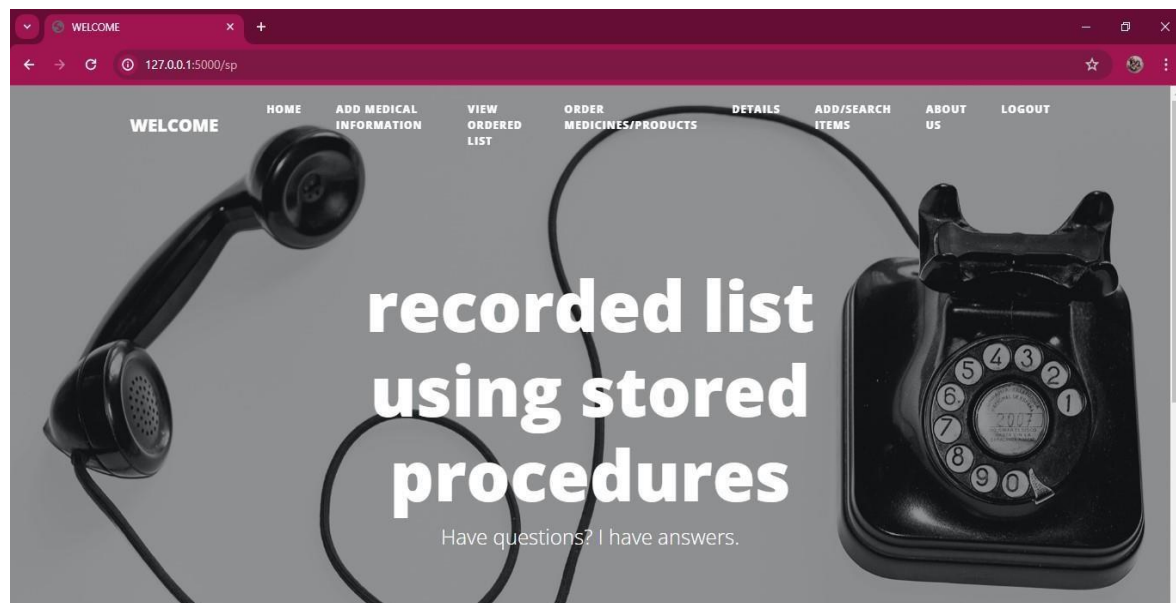


Fig.6.14-Questions Page

The screenshot shows the phpMyAdmin interface for the 'posts' table in the 'coding' database. The table has 4 rows. The columns are: mid, medical_name, owner_name, phone_no, and address. The data is as follows:

mid	medical_name	owner_name	phone_no	address
1	national medical	akhil	9874563214	chickjala
2	params medical	Aadithyaa	9874563215	Bangalore
3	Indian medicals	Anees	7259462891	Bangalore
4	arksa medicals	pritha	9986786453	Bangalore

Fig.6.15-PHP Table

The screenshot shows the phpMyAdmin interface for the 'medicines' table in the 'coding' database. The table has 1 row. The columns are: id, mid, name, medicines, products, email, and amount. The data is as follows:

id	mid	name	medicines	products	email	amount
1	1	akhil	glimepiride (Amaryl)-[5 sheet] glimepiride-piogli...	lux soap-[5 piece] olay cream-[5 item] patanja...	aneesurrehman423@gmail.com	78957

Fig.6.16-Table Details

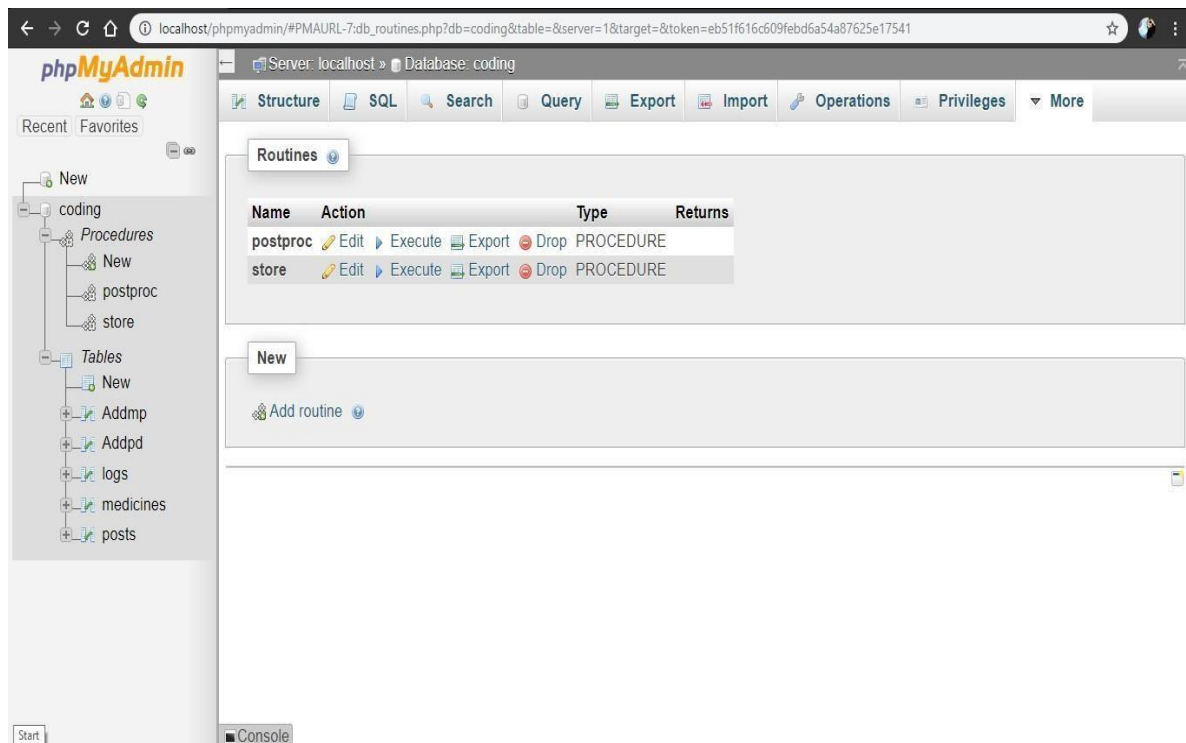


Fig.6.17-Table Input

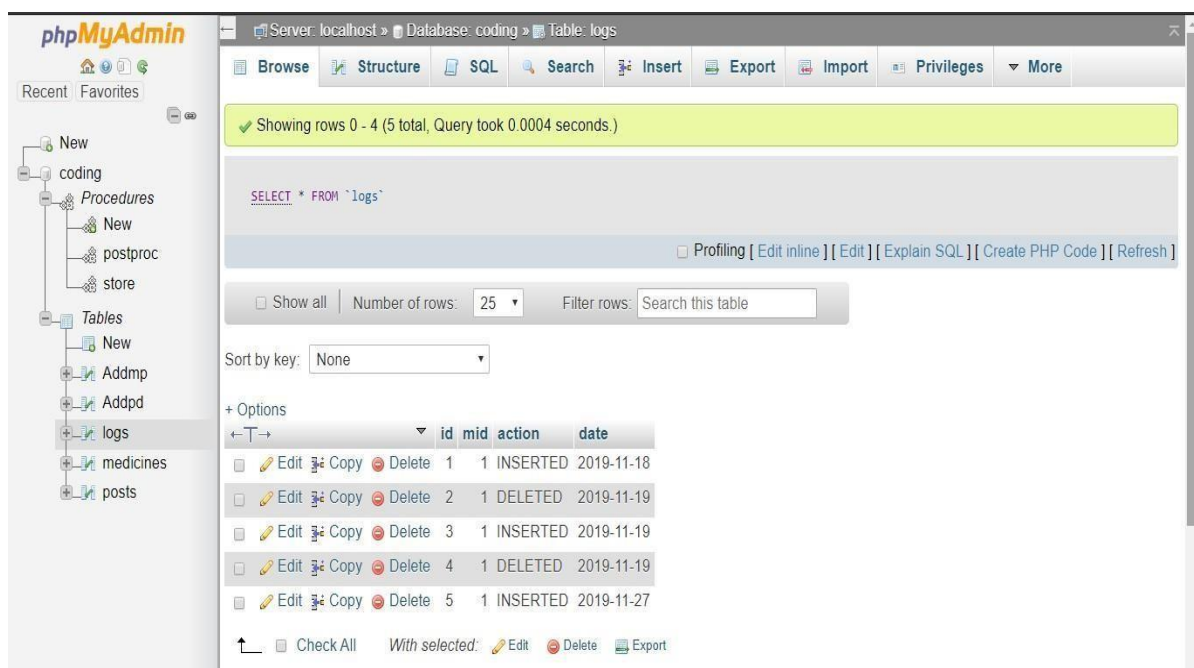


Fig.6.18-Table Input 2

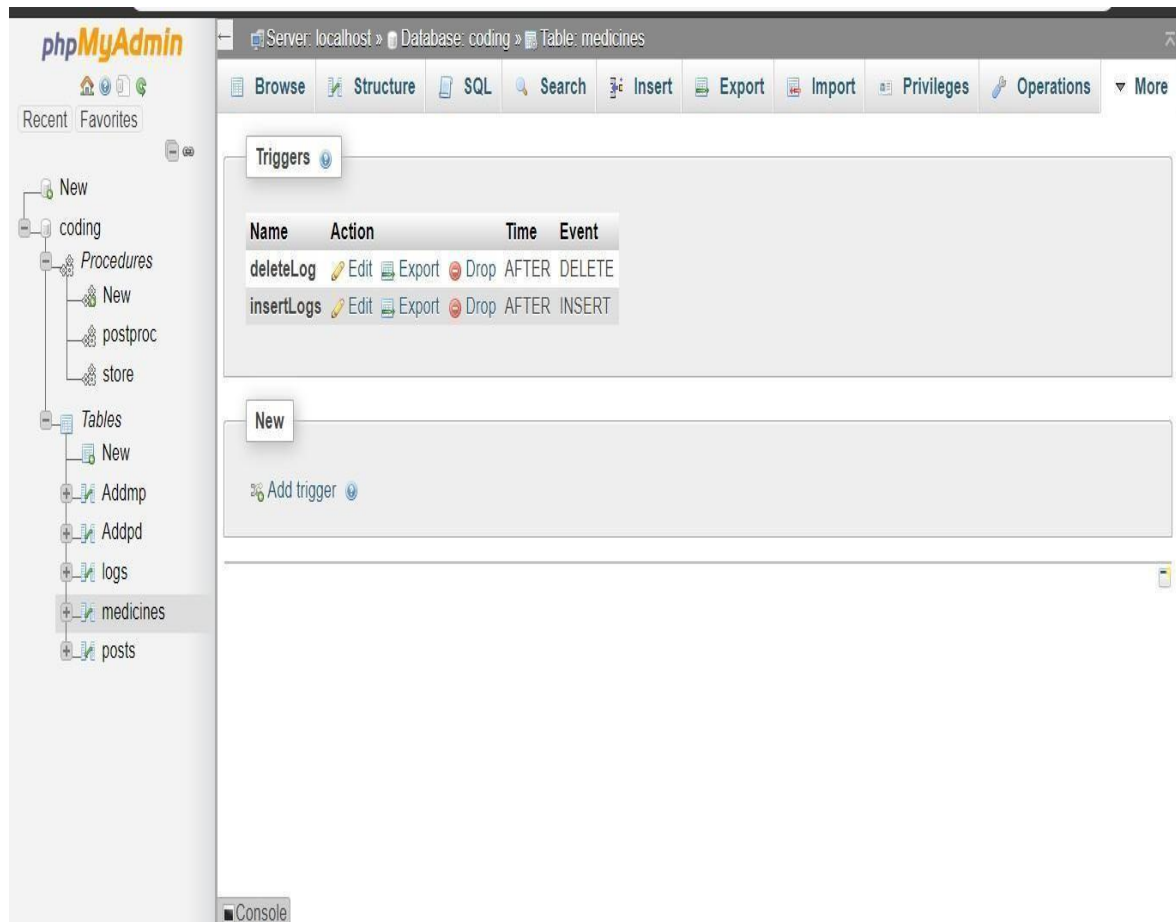


Fig.6.19-Table Deletion and Insertion

CHAPTER 7

CONCLUSION

PHARMACY MANAGEMENT SYSTEM successfully implemented offline medicines supply management database which helps us in administrating the data user for managing the tasks performed in medicines supply. The project successfully used various functionalities of Amppps and python flask and also create the fully functional database management system for offline pharmacy.

Using MySQL as the database is highly beneficial as it is free to download, popular and can be easily customized. The data stored in the MySQL database can easily be retrieved and manipulated according to the requirements with basic knowledge of SQL.

With the theoretical inclination of our syllabus it becomes very essential to take the utmost advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project “Pharmacy Supply Management System” was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

- The planning that goes into implementing a project.
- The importance of proper planning and an organized methodology.
- The key element of team spirit and co-ordination in a successful project.

FUTURE SCOPE

- Enhanced database storage facility
- Enhanced user friendly GUI
- more advanced transportation of medicines
- online Bill payments

BIBILOGRAPHY

1. “Database System Concepts” by Abraham Silberschatz and
2. “Fundamentals of database system” by R Elmasri & Navath
3. “Database Management System” by Raghu Ramakrishnan
4. <https://www.youtube.com/>
5. <http://www.getbootstrap.com/>
6. <https://www.google.com/>