

# CHAPTER-1

## INTRODUCTION

### 1.1 OBJECTIVES:

- 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).
- Less chances of information leakage.
- Provides Security to the data by using login and password method.
- To provide immediate storage and retrieval of data and information.
- Improving arrangements for medicines coordination.
- Reducing paperwork.

### 1.2 LIMITATIONS:

- Time consumption in data entry as the records are to be manually maintained consumes a lot of time.
- Lot of paper work is involved as the records are maintained in the files and registers.
- Storage Requires as files and registers are used the storage space requirement is increased.
- Less Reliable use of papers for storing valuable data information is not at all reliable.
- Aadhar linkage with the official aadhar database has not been done.

## **CHAPTER-2**

### **STUDY OF EXISTING SYSTEM**

#### **2.1 CASE STUDY**

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**

### **3. DATABASE DESIGN**

#### **3.1 SOFTWARE REQUIREMENTS SPECIFICATION**

##### **3.1.2 SOFTWARE REQUIREMENTS:**

Frontend- HTML, CSS, Java Script, Bootstrap

Backend-Python flask (Python 3.7) , SQLAlchemy,

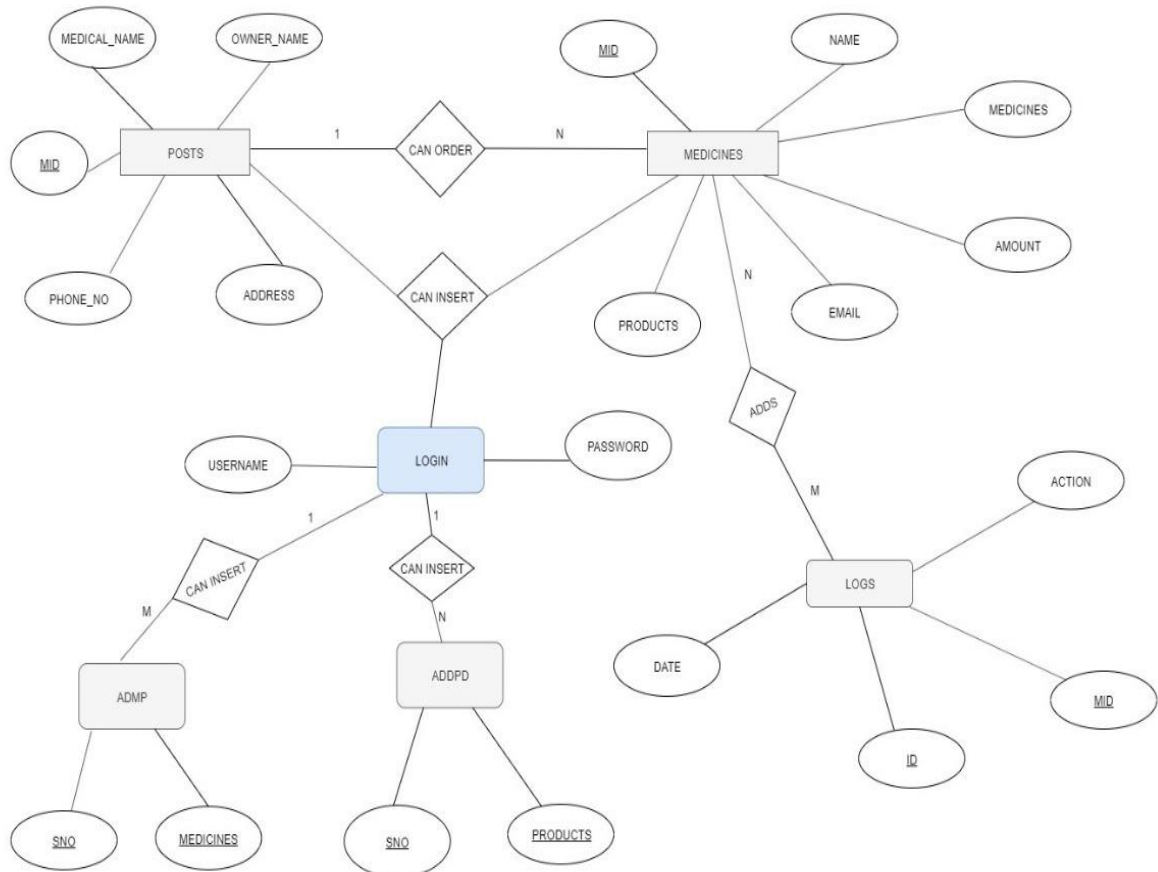
- Operating System: Windows 10
- Google Chrome/Internet Explorer
- AMPPS (Version-3.7)
- Python main editor (user interface): PyCharm Community
- workspace editor: Sublime text 3

##### **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

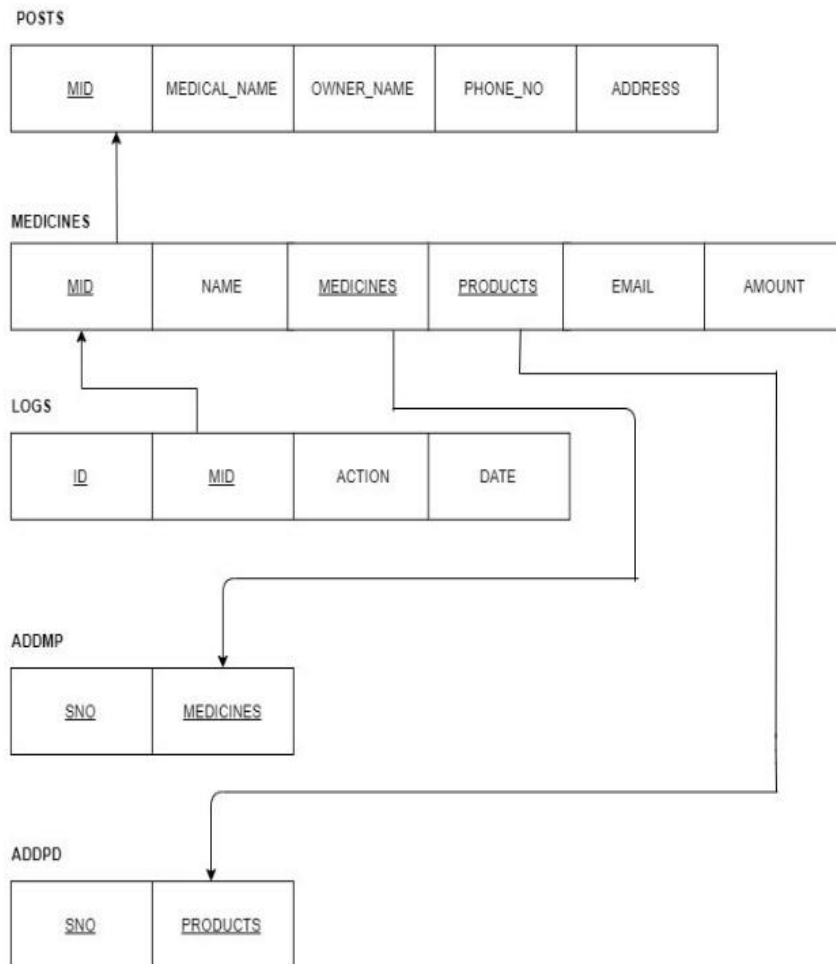
## 3.2 CONCEPTUAL DESIGN:

### 3.2.1 E-R DIAGRAM:



### 3.2.2 SCHEMA DIAGRAM:

SCHEMA DIAGRAM



### 3.3 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 (which implies relatively slow access compared to volatile main memory).A database query language and report

writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.

- 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. to the 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).

#### **4.2: Stored Procedure**

Routine name: proc 1, post proc

Type: procedure

Definition: Select \* from posts;

Select \* from medicines;

#### **4.3: 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 PYHTON 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)
```

## FRONT END 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

<i class="fa fa-bars"></i>

</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" h ref="/list">view ordered list</a>

</li>

<li class="nav-item">

<a class="nav-link" h ref="/medicines ">ADD medicines/products</a>

</li>

<li class="nav-item">

<a class="nav-link" h ref="/details ">Details</a>

</li>

<li class="nav-item">

<a class="nav-link" h ref="/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-itemparams['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>

</footer>

<!-- Bootstrap core JavaScript -->

<script src="{ { url_for('static',filename='vendor/jquery/jquery.min.js') }}"></script>

<script
src="{ { url_for('static',filename='vendor/bootstrap/js/bootstrap.bundle.min.js') }}"></script>

<!-- Custom scripts for this template -->

<script src="{ { url_for('static',filename='js/clean-blog.min.js') }}"></script>
```

</body>

</html>

2.

```
{% extends "layout.html" %}
```

```
{% block body %}
```

```
<!-- Page Header -->
```

```
<header class="masthead mb-0" style="background-image:
url('{{ url_for('static',filename='img/gmail.jpg') }}')">
```

```
<div class="overlay"></div>
```

```
<div class="container">
```

```
<div class="row">
```

```
<div class="col-lg-8 col-md-10 mx-auto">
```

```
<div class="page-heading">
```

```
<h1> MEDICINES AND PRODUCTS </h1>
```

```
<span class="subheading">Do you Have questions? I have answers</span>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</header>
```

```
{% with messages=get_flashed_messages(with_categories=true) %}
```

```
{% if messages %}
```

```
{% for category, message in messages %}
```

```
<div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
```

```
  {{ message }}
```

```
<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>
```

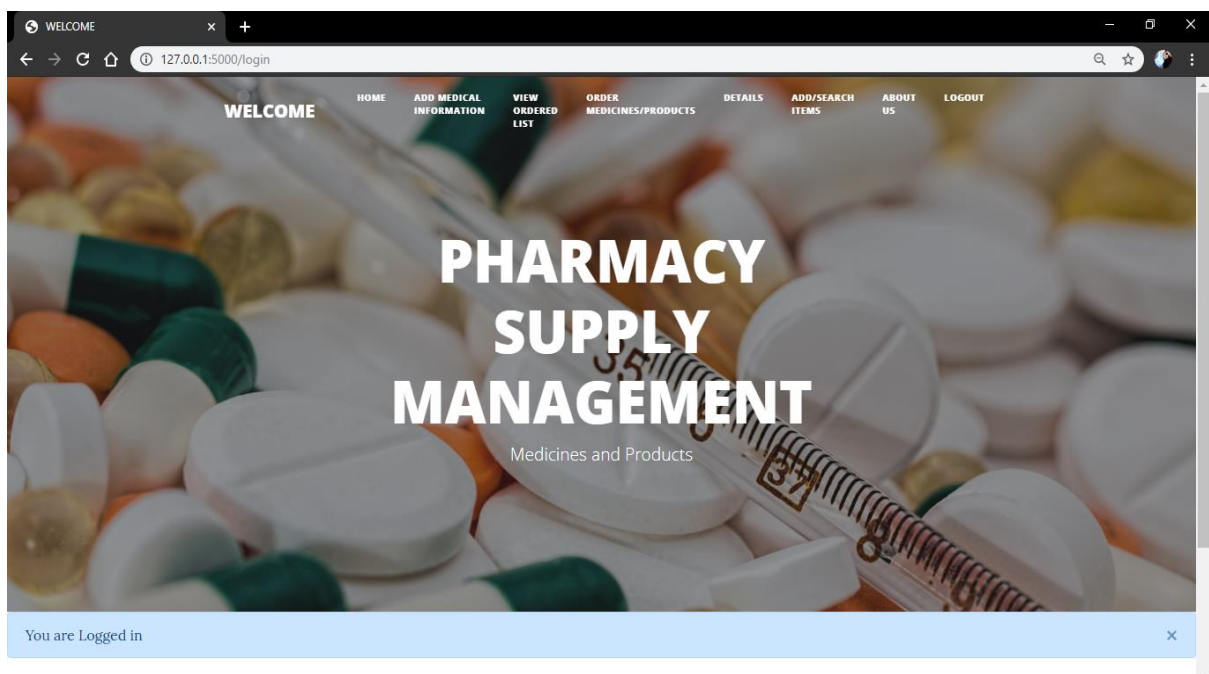
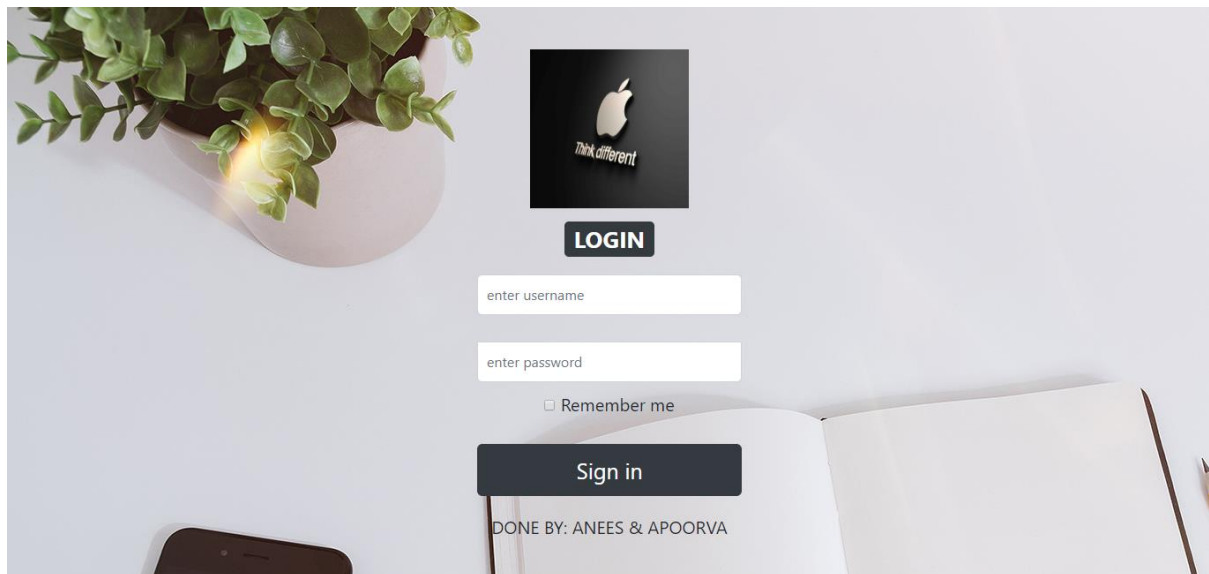
```
</html>
```

```
{% endblock % }
```

## 4. USER INTERFACES

## 4.1 SCREEN SHOTS

### LOGIN PAGE:



### ADD MEDICINES INFO:

A screenshot of a web browser displaying the 'ADD DATA' form of the Pharmacy Supply Management System. The browser's address bar shows '127.0.0.1:5000/insert'. The page has a navigation bar with links: WELCOME, HOME, ADD MEDICAL INFORMATION, VIEW ORDERED LIST, ORDER MEDICINES/PRODUCTS, DETAILS, ADD/SEARCH ITEMS, ABOUT US, and LOGOUT. The form itself is titled 'ADD DATA' and contains the following fields: 'MEDICAL ID' with the value '4', 'MEDICAL SHOP NAME' with 'arksa medicals', 'OWNER NAME' with 'pritha', 'PHONE NUMBER' with '9986786453', and 'ADDRESS' with 'Bangalore'. A teal 'INSERT DATA' button is at the bottom of the form.

127.0.0.1:5000/insert

WELCOME you are logged in

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

## ADD DATA

MEDICAL ID  
4

MEDICAL SHOP NAME  
arksa medicals

OWNER NAME  
pritha

PHONE NUMBER  
9986786453

ADDRESS  
Bangalore

INSERT DATA

A screenshot of the same 'ADD DATA' form after the data has been submitted. A pink notification banner at the top reads 'Thanks for submitting your details'. The form fields are now empty and labeled with placeholder text: 'enter medical id', 'enter medical name', 'Enter Owner name', 'Enter phone number', and 'enter Address'. The teal 'INSERT DATA' button remains at the bottom.

127.0.0.1:5000/insert

Thanks for submitting your details

## ADD DATA

enter medical id

enter medical name

Enter Owner name

Enter phone number

enter Address

INSERT DATA

127.0.0.1:5000/login

MEDICAL RECORDS

medical management information stored over here

Mid	Medical Shop Name	Medical Shop Owner	Phone No	Address	Edit	Delete
1	national medical	akhil	9874563214	chickjala	EDIT	DELETE
2	params medical	Aadithyaa	9874563215	Bangalore	EDIT	DELETE
3	Indian medicals	Anees	7259462891	Bangalore	EDIT	DELETE
4	arksa medicals	pritha	9986786453	Bangalore	EDIT	DELETE

127.0.0.1:5000/medicines

WELCOME

HOME

ADD MEDICAL INFORMATION

VIEW ORDERED LIST

ORDER MEDICINES/PRODUCTS

DETAILS

ADD/SEARCH ITEMS

ABOUT US

LOGOUT

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!

mail

name

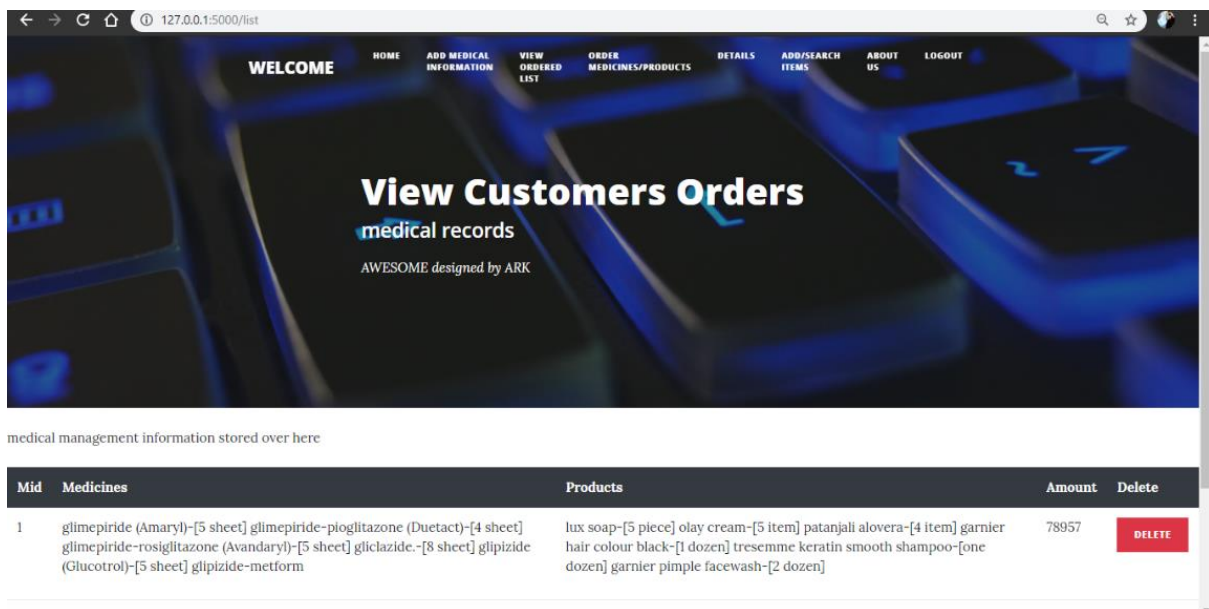
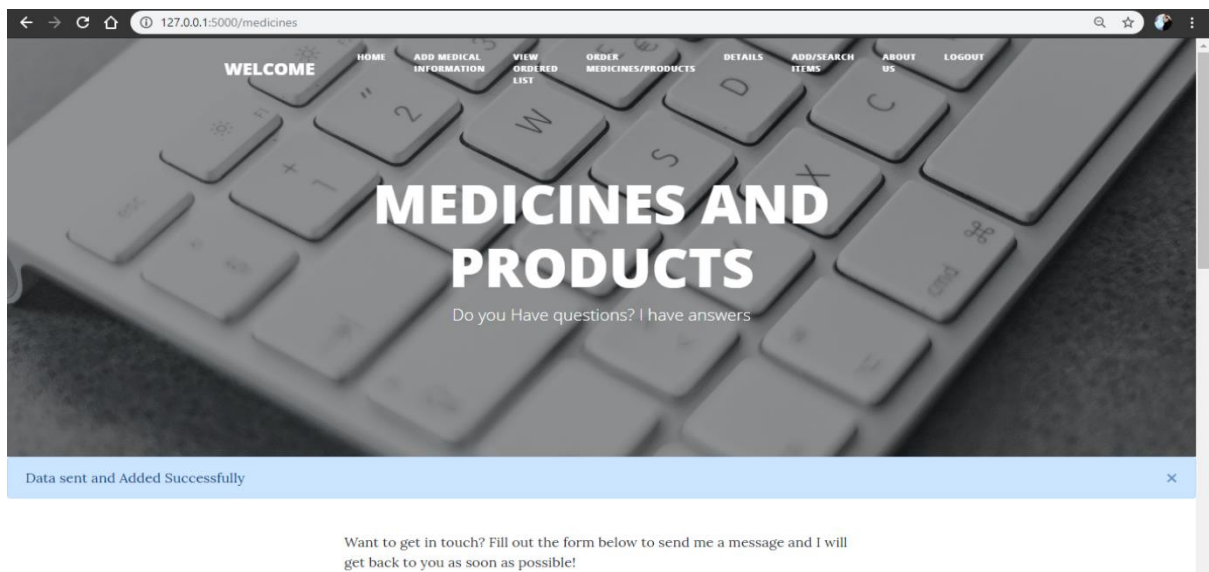
Medicines

Products

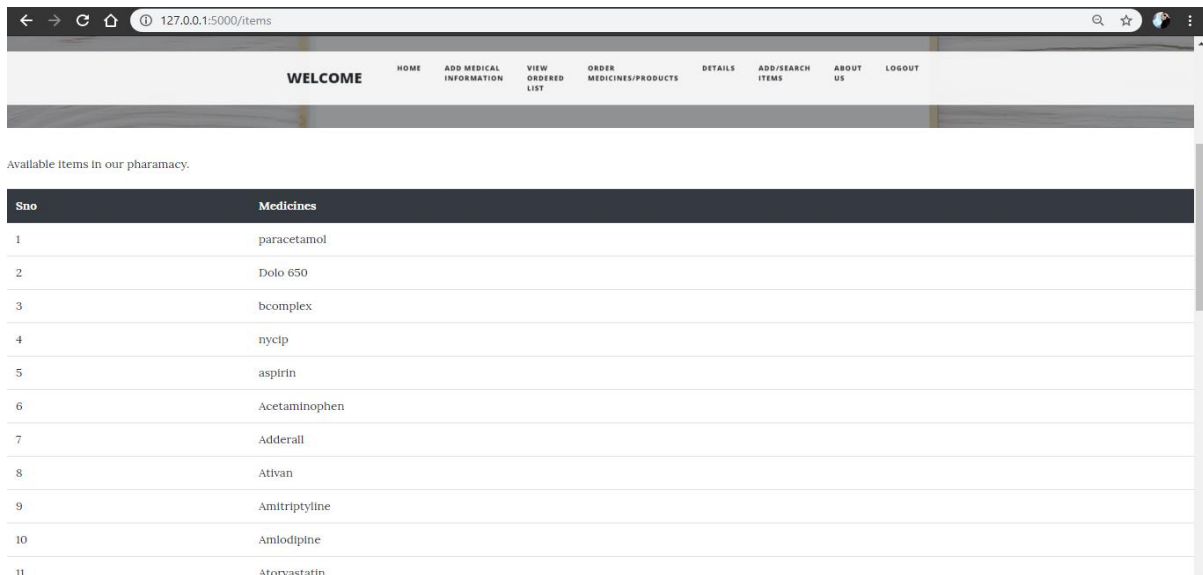
Email id

Amount

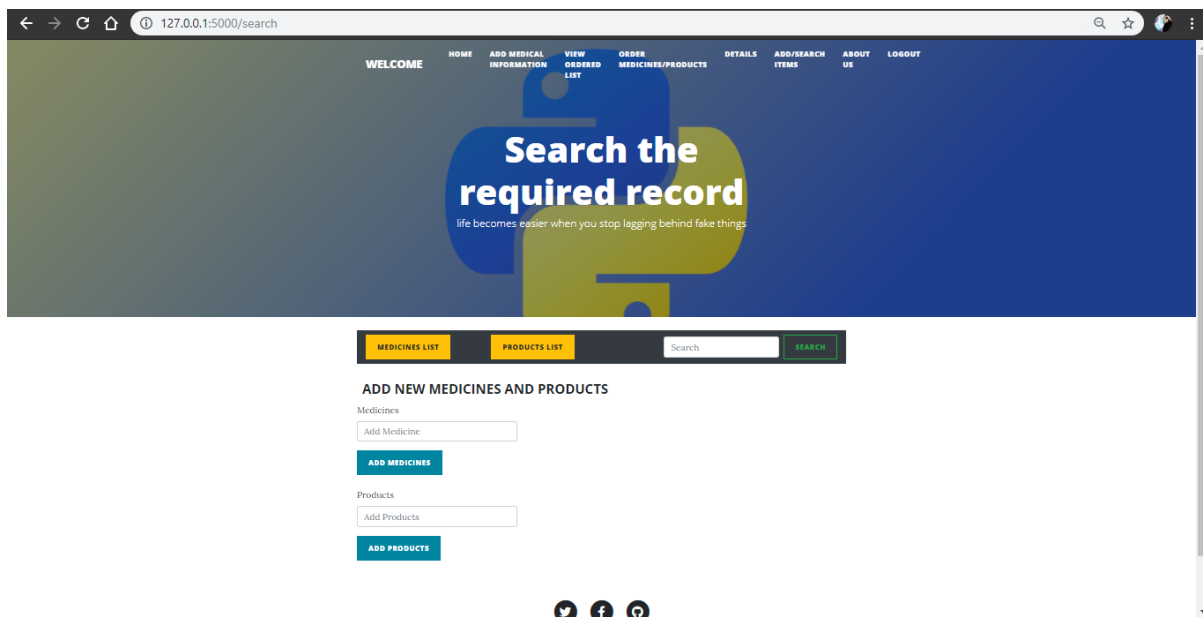
SEND



# Pharmacy Supply Management System



Sno	Medicines
1	paracetamol
2	Dolo 650
3	bcomplex
4	nycip
5	aspirin
6	Acetaminophen
7	Adderall
8	Ativan
9	Amitriptyline
10	Amlodipine
11	Atorvastatin



Search the required record  
life becomes easier when you stop lagging behind fake things

MEDICINES LIST PRODUCTS LIST Search SEARCH

ADD NEW MEDICINES AND PRODUCTS

Medicines  
Add Medicine  
ADD MEDICINES

Products  
Add Products  
ADD PRODUCTS

Twitter Facebook LinkedIn

## Pharmacy Supply Management System

Available items in our pharmacy

Sno	Products
1	santoor soap
2	ponds cream
3	nivea skin care
4	evion
5	ponds cold cream
6	olay
7	lakme
8	maybelin newyork
9	lacto calamine
10	patanjali alovera
11	ayush
12	lotus whiteglow
13	biotique

127.0.0.1:5000/search

WELCOME

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

# Search the required record

life becomes easier when you stop lagging behind fake things

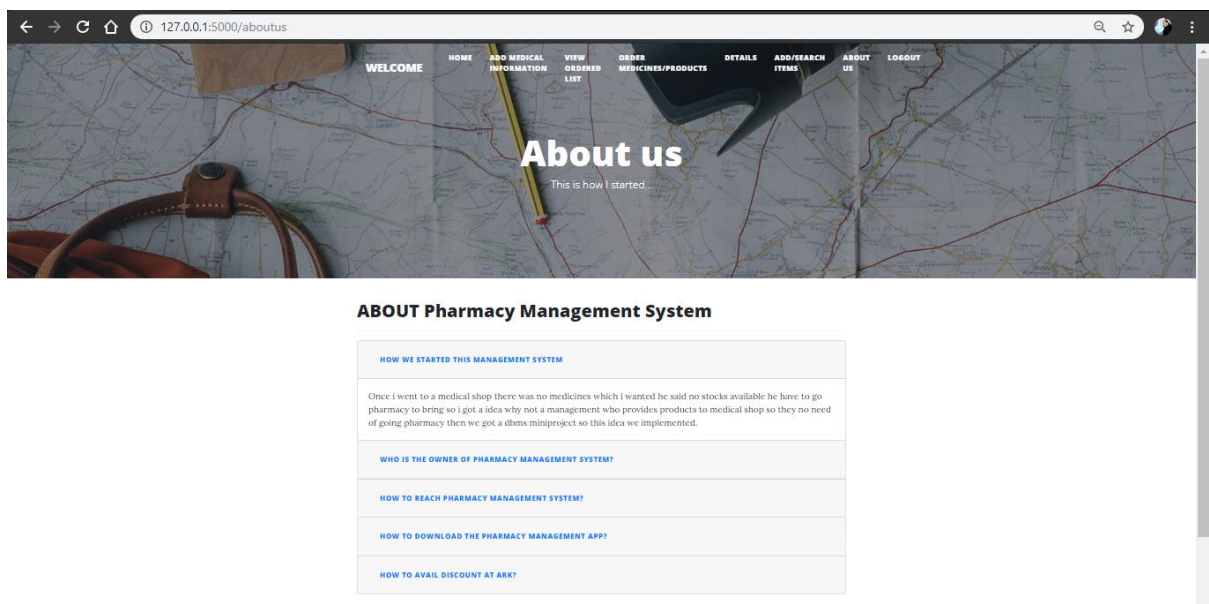
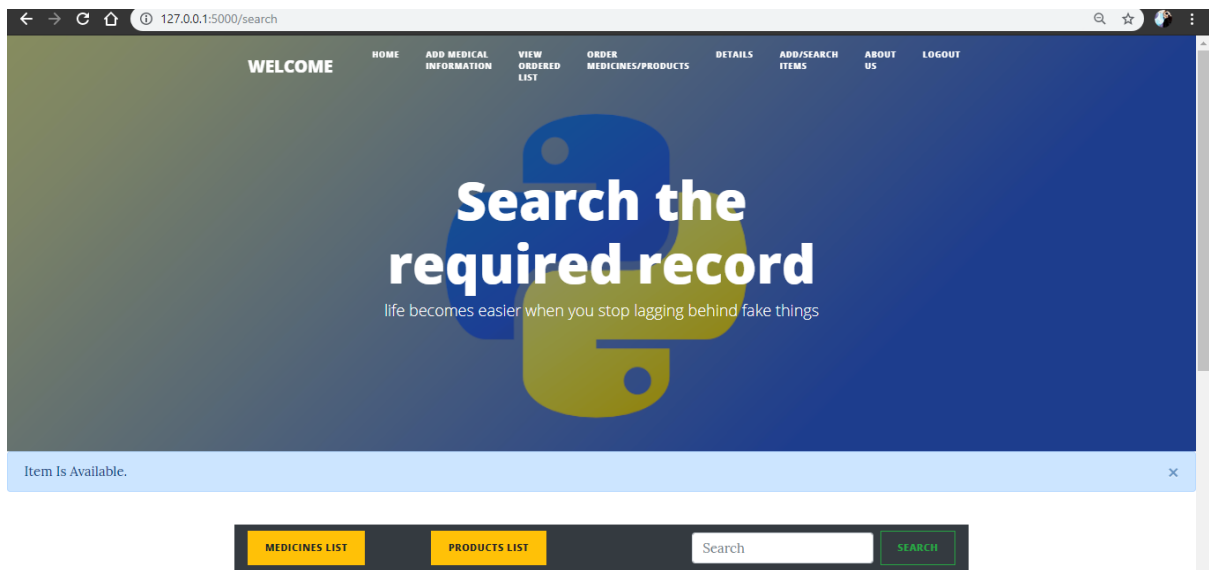
MEDICINES LIST PRODUCTS LIST

Acetaminophen SEARCH

### ADD NEW MEDICINES AND PRODUCTS

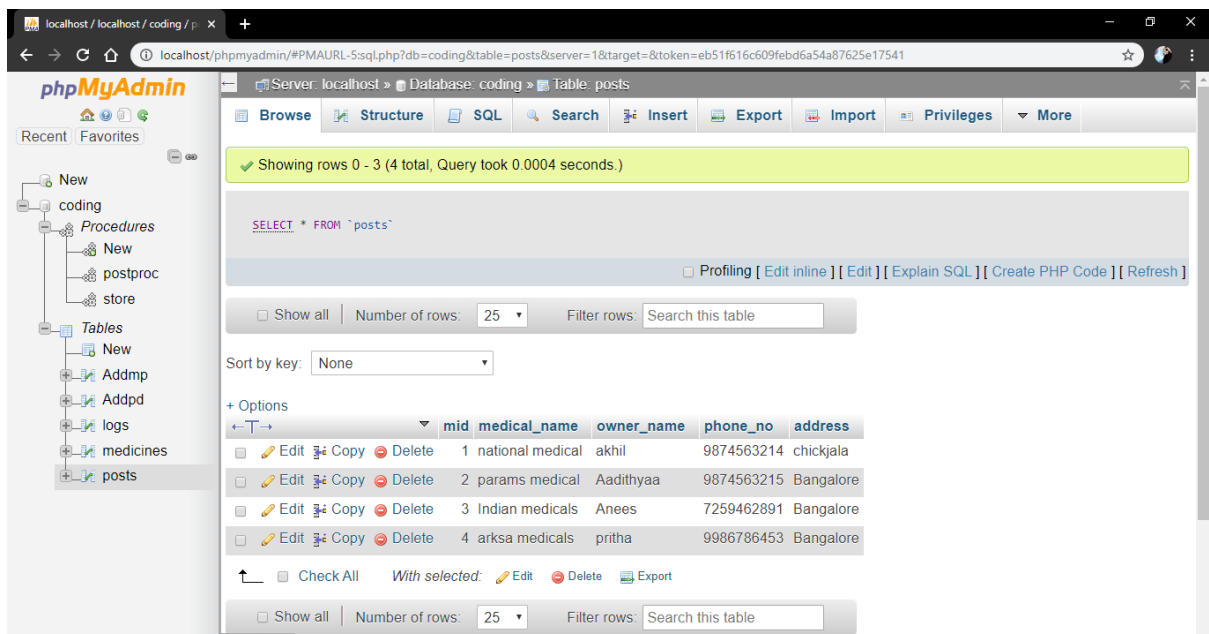
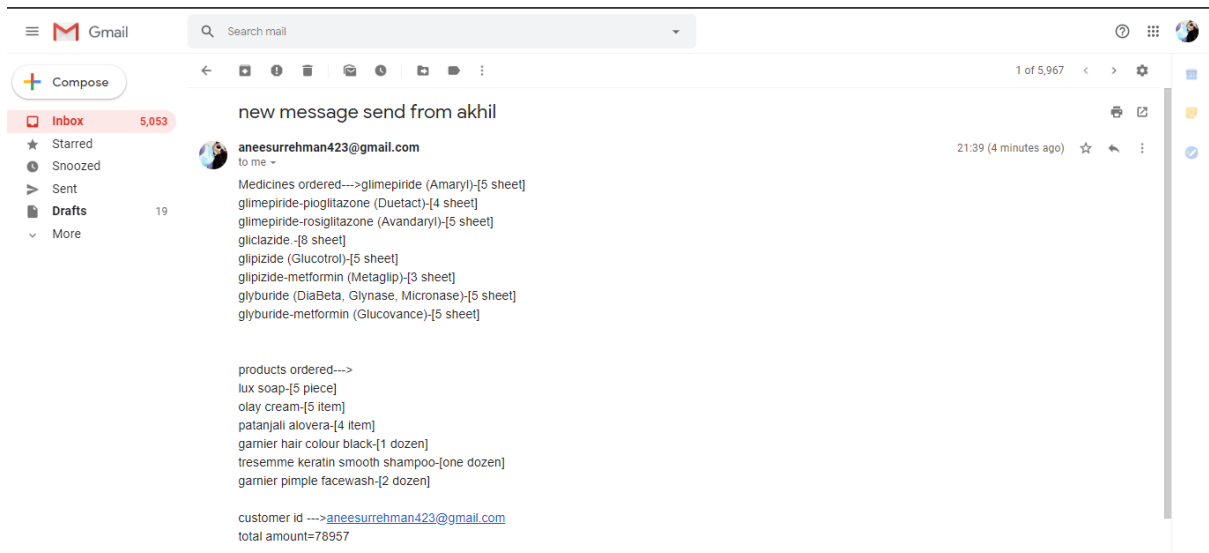
Medicines

Add Medicine





# Pharmacy Supply Management System



The screenshot shows the phpMyAdmin interface for a database named 'coding'. The 'Table: medicines' is selected. The left sidebar shows the database structure with 'coding' as the selected database. The main panel displays the 'Structure' tab for the 'medicines' table. The table has columns: id, mid, name, medicines, products, email, and amount. The data row shows: id=1, mid=1, name=akhil, medicines=glimepiride (Amaryl)-[5 sheet], products=lux soap-[5 piece], email=aneesurrehman423@gmail.com, amount=78957. The 'Query results operations' section is visible at the bottom.

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

The screenshot shows the phpMyAdmin interface for the 'coding' database. The 'Routines' tab is selected. The left sidebar shows the database structure with 'coding' as the selected database. The main panel displays the 'Routines' tab for the 'coding' database. The table shows the following routines:

Name	Action	Type	Returns
postproc	Edit Execute Export Drop	PROCEDURE	
store	Edit Execute Export Drop	PROCEDURE	

Below the table, there is a 'New' section with an 'Add routine' button.

The screenshot shows the phpMyAdmin interface for the 'coding' database. The 'logs' table is selected, and its structure and data are displayed. The table has columns: id, mid, action, and date. The data shows five rows of log entries.

id	mid	action	date
1	1	INSERTED	2019-11-18
2	1	DELETED	2019-11-19
3	1	INSERTED	2019-11-19
4	1	DELETED	2019-11-19
5	1	INSERTED	2019-11-27

The screenshot shows the phpMyAdmin interface for the 'coding' database. The 'medicines' table is selected, and its triggers are displayed. There are two triggers: 'deleteLog' and 'insertLogs', both set to fire AFTER their respective events (DELETE and INSERT).

Name	Action	Time	Event
deleteLog	Drop	AFTER	DELETE
insertLogs	Drop	AFTER	INSERT

## 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 Ampps 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 ENHANCEMENT**

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

## REFERENCES

- <https://www.youtube.com/>
- <https://www.google.com/>
- <http://www.getbootstrap.com/>