A Minor Project Report

On

ENHANCING PHARMACEUTICAL SUPPLY CHAINS WITH PREDICTIVE MACHINE LEARNING MODELS

Submitted in partial fulfillment of requirements for the award of the degree of

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**

Under the guidance of

**Mr.** **SHANMUGASUNDARAM P**

**Assistant Professor/CSE**

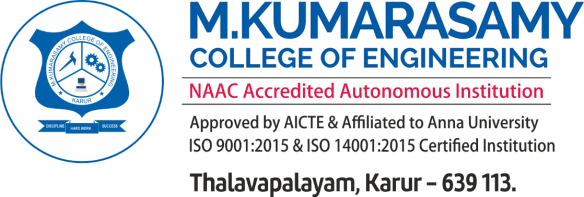
Submitted By

|  |  |
| --- | --- |
| **GIRITHARAN S** | **(927622BCS028)** |
| **GOKUL K** | **(927622BCS029)** |
| **JAYAPRAKASH J** | **(927622BCS042)** |
| **MANOJ KUMAR S** | **(927622BCS061)** |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous) **KARUR – 639 113** APRIL 2023.

# M. KUMARASAMY COLLEGE OF ENGINEERING

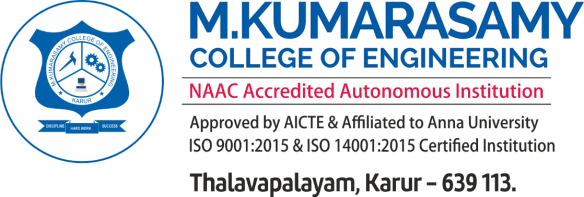
**(Autonomous Institution affiliated to Anna University, Chennai)**

# KARUR – 639113

**BONAFIDE CERTIFICATE**

Certified that this minor project report **“ENHANCING**  **PHARMACEUTICAL SUPPLY CHAINS WITH PREDICTIVE MACHINE LEARNING MODELS”** is the bonafide work of **“GIRITHARAN S (927622BCS028), GOKUL K (927622BCS029), JAYAPRAKASH J (927621BCS042), MANOJ KUMAR S(927622BCS061)”** who carried out the project work during the academic year 2024-2025 under my supervision.

|  |  |
| --- | --- |
| Signature | Signature |
| **Mrs. P. SHANMUGASUNDARAM B.E.,** | **Dr. D. PRADEEP M.E., Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science | Department of Computer Science |
| and Engineering, | and Engineering, |
| M. Kumarasamy College of Engineering, | M. Kumarasamy College of Engineering |
| Thalavapalayam, Karur -639113 | Thalavapalayam, Karur -639113. |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VISION OF THE INSTITUTION**

To emerge as a leader among the top institutions in the field of technical education

**MISSION OF THE INSTITUTION**

 Produce smart technocrats with empirical knowledge who can surmount the global challenges

 Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students

 Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

**VISION OF THE DEPARTMENT**

To achieve education and research excellence in Computer Science and Engineering

**MISSION OF THE DEPARTMENT**

 To excel in academic through effective teaching learning techniques

 To promote research in the area of computer science and engineering with the focus on innovation

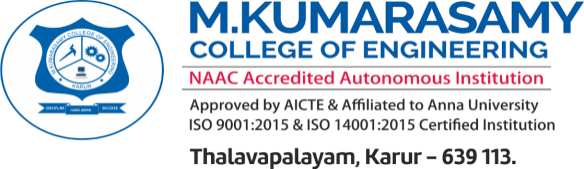
 To transform students into technically competent professionals with societal and ethical responsibilities

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO 1:**Graduates will have successful career in software industries and R&D divisions through continuous learning.

**PEO 2:** Graduates will provide effective solutions for real world problems in the key domain of computer science and engineering and engage in lifelong learning.

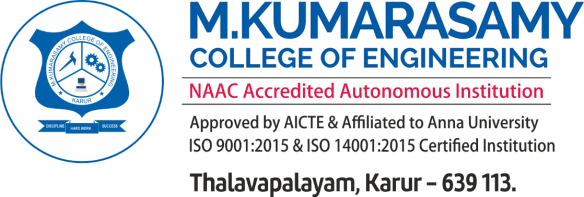
**PEO 3:** Graduates will excel in their profession by being ethically and socially responsible.

**PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **C**and design documentation, make effective presentations, and give and receive clear instructions.

**10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.**

**11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12 .Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

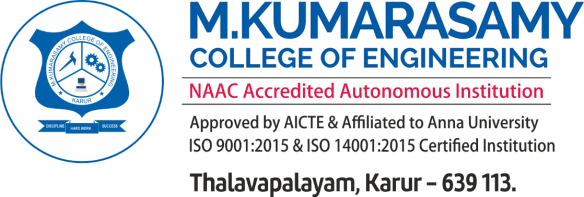
**PROGRAM SPECIFIC OUTCOMES (PSOs)**

 **PSO1: Professional Skills:** Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.

 **PSO2: Successful career:** Ability to utilize the computing skills and ethical values in creating a successful career.

**ABSTRACT**

The pharmaceutical supply chain faces unique challenges due to the sensitive nature of medications, regulatory requirements, and demand variability. Enhancing these supply chains with predictive machine learning (ML) models can provide valuable insights, enabling more accurate forecasting, efficient inventory management, and proactive risk mitigation. By leveraging data from historical sales, patient trends, and environmental factors, ML algorithms can anticipate demand fluctuations, optimize stock levels, and reduce wastage. Additionally, predictive models improve logistics by identifying potential bottlenecks and enabling faster response to disruptions, such as pandemics or natural disasters.

# ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| The pharmaceutical supply chain faces unique challenges due to the sensitive nature of medications, regulatory requirements, and demand variability. Enhancing these supply chains with predictive machine learning (ML) models can provide valuable insights, enabling more accurate forecasting, efficient inventory management, and proactive risk mitigation. By leveraging data from historical sales, patient trends, and environmental factors, ML algorithms can anticipate demand fluctuations, optimize stock levels, and reduce wastage. Additionally, predictive models improve logistics by identifying potential bottlenecks and enabling faster response to disruptions, such as pandemics or natural disasters. | **PO1(3)**  **PO2(3)**  **PO3(2)**  **PO4(2)**  **PO5(2)**  **PO6(1)**  **PO7(3)**  **PO8(2)**  **PO9(3)**  **PO10(3)**  **PO11(2)**  **PO12(2)** | **PSO 1(3)**  **PSO 2(2)** |

Note: 1- Low, 2-Medium, 3- High

**SUPERVISOR HEAD OF THE DEPARTMENT**

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **No.** | **TITLE** | **PAGE No.** |
|  | **Abstract** | **vi** |
|  | **List of Figures** | **Ix** |
|  | **List of Abbreviations** | **X** |
| **1** | **Introduction** | **1** |
|  | 1.1 Overview | 1 |
|  | 1.2 Domain Introduction | 2 |
|  | 1.3 Problem Statement | 3 |
|  | 1.4 Objective | 3 |
| **2** | **Literature Survey** | **4** |
|  | 2.1 Existing System Architecture | 6 |
|  | 2.2 Drawbacks of Existing System **Architecture** | 6 |
|  | 2.3 Proposed System Architecture | 6 |
| **3** | **Feasibility Study** | **10** |
| **4** | **Project Methodology** | **12** |
|  | 4.1 Block diagram of My Companion | 12 |
|  | 4.2 Module Description | 13 |
| **5** | **Results and Discussion** | **15** |
| **6** | **Conclusion** | **19** |
|  | **References** | **20** |

# LIST OF FIGURES

**FIGURE No. TITLE PAGE No.**

|  |  |  |
| --- | --- | --- |
| 1.1 | My Companion | 1 |
| 3.1 | Feasibility study | 9 |
| 4.1  5.1  5.2  5.3  5.4 | Block diagram of My Companion  Screenshot of Home Activity Page  Screenshot of Register Activity Page  Screenshot of Login Activity Page  Screenshot of Map Activity Page | 11  15  16  17  18 |

# LIST OF ABBREVIATIONS

LBS Location Based Service

JDBC Java Data Base

XML Extensible Markup language

# CHAPTER 1

# INTRODUCTION

The pharmaceutical supply chain is complex and faces unique challenges due to strict regulations, demand variability, and sensitivity to global disruptions. These issues often lead to shortages, increased waste, and inefficiencies, impacting patient care and operational costs. Predictive machine learning (ML) offers a solution by analyzing large datasets to uncover trends and predict outcomes across the supply chain. By enhancing demand forecasting, optimizing inventory, and identifying potential disruptions, ML enables more efficient, resilient operations. This paper explores how predictive ML models can transform pharmaceutical supply chains, making them more responsive, efficient, and

reliable.



# Figure: 1.1 Pharmaceutical Supply Chain

**1.1. OVERVIEW**

This project aims to enhance pharmaceutical supply chain operations through the integration of predictive machine learning (ML) models. Given the complexity of the pharmaceutical supply chain, which includes sourcing, production, inventory management, and distribution, disruptions and inefficiencies can lead to shortages, delays, and wastage. By applying predictive ML models, this project seeks to address these challenges by improving demand forecasting, optimizing inventory levels, and proactively identifying risks.

**1.2 DOMAIN INTRODUCTION**

The pharmaceutical supply chain is a highly complex, tightly regulated domain responsible for delivering medications safely and efficiently from manufacturers to patients. It involves multiple stages, including sourcing raw materials, manufacturing drugs, managing inventories, and distributing products to healthcare providers and pharmacies. Each stage must comply with strict quality and safety standards to maintain drug integrity and ensure patient safety.

Unlike typical supply chains, pharmaceutical supply chains face unique challenges due to the sensitive nature of medications, which often have short shelf lives and require specialized handling. Furthermore, demand for pharmaceuticals can be highly variable, influenced by factors such as seasonal illnesses, global pandemics, and demographic trends. These complexities make efficient supply chain management essential to avoid stockouts, wastage.

The introduction of predictive machine learning models in this domain provides new opportunities to address these challenges. By analyzing data and identifying patterns, machine learning can enhance forecasting, optimize logistics, and detect potential disruptions, ultimately leading to a more efficient, resilient, and responsive supply chain.

**1.3** **PROBLEM STATEMENT**

The pharmaceutical supply chain faces significant challenges due to demand variability, regulatory requirements, and the need for high-quality, time-sensitive products. These inefficiencies not only affect healthcare outcomes but also increase operational costs and risks.

* Traditional methods often struggle to accurately forecast demand, manage inventory efficiently, and respond quickly to disruptions, leading to issues such as medication shortages, wastage, and delays in patient access to essential drugs.
* This approach will help ensure that medications are consistently available when and where they are needed, reducing waste and enhancing overall patient care.

**1.4 OBJECTIVE**

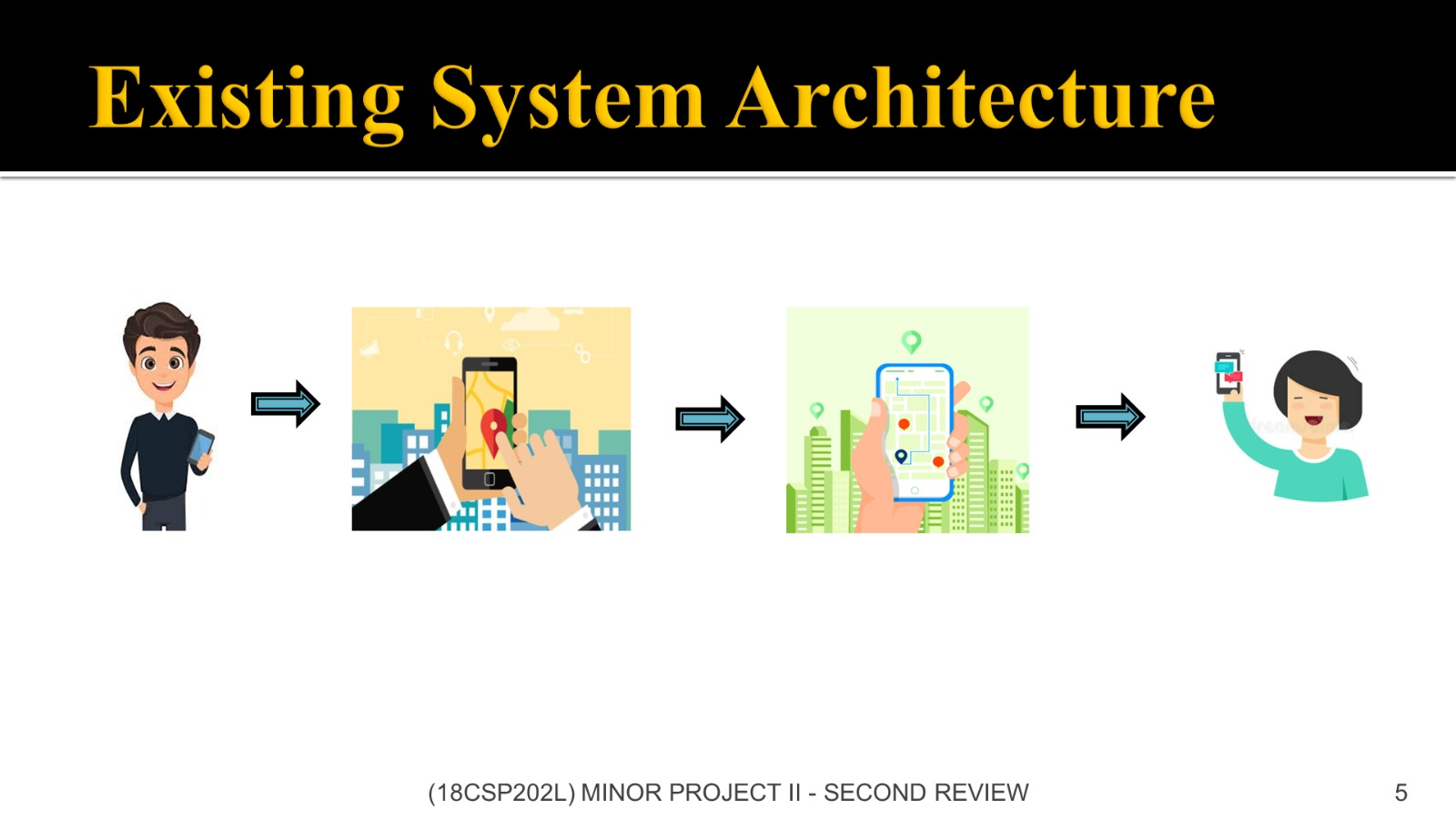
To optimize the efficiency, cost-effectiveness, and reliability of pharmaceutical supply chains by leveraging predictive machine learning models to improve demand forecasting, inventory management, supply chain risk management, route optimization, and production processes. The goal is to reduce stockouts, minimize excess inventory, enhance logistics, improve supplier performance, ensure regulatory compliance, and ultimately improve the quality and availability of pharmaceutical products to meet market demand and customer needs.

# CHAPTER 2

**LITERATURE SURVEY**

1. Location-Based Services have been noted to surge in popularity though (Junglas & Watson, 2008) The newfound popularity of smartphones and localized services have generated specific services around location such as Foursquare and Gowalla (Popescu, 2010), as well as location services provided by platforms that are not internet based such as iOS, Android or Symbian. Foursquare is a location based service which functions both as a social networking website and LBS. While it functions as website, it is also customized for use on smartphones and GPS-enabled mobile devices. For squares primary functions is the “checking”, in which users can register or check in at locations utilizing a web browser, text messaging or a device-specific application. This application detects nearby locations, or venues, and the user can then select the one he is located in, therefore “checking in”. The user can obtain rewards for this, such as points or virtual badges. Gowalla is a similar Location Based Service that also functions as a social network, its main difference being the rewards the user can obtain for “checking in”. A user either checks in through a dedicated website or from an application downloaded to a smartphone. Interestingly, Gowalla allows users to check in via Foursquare, Facebook places, Twitter or Facebook. Facebook’s Places functionality ushered LBS concepts within the Facebook platform, though Facebook’s policy of centralizing its own service hampered its spreading, while consumer LBS gave rise to personalization of services and contextualization of information (Schiller & Voisard, 2004). The Facebook Places functionality is similar to that of Foursquare or Gowalla – it allows users to post status updates in which they “tag” their location. It integrates with Facebook’s other functionalities such as tagging people in posts and status updates.
2. It seems that location-based services are highly connected with mobile phones that can offer mobility while using these services (Curran and Hubrich 2009). Moreover, the content created by these applications is highly correlated by the usage of the typical users (Gummerus and Pihlström 2011). Specific researchers have identified the fact that geospatial information is driven by behavioral characteristics since motivation is a major fact for every single transportation (Girardin et al. 2008; Espeter and Raubal 2009; May, Bayer, and Ross 2007). Others have classified the transportation methods-means into different classifiers in order to better understand how geotagging is performed by end users through social media (Tamas and Toth 2018).
3. The field of LBS lack of researches that focus simultaneously on both the technology stream and the research stream based on stable frameworks and theories (Khruahong et al. 2018). As a result, this field opens significant opportunities for researchers that want to explore Behavior of end users in addition to an innovative solution to approximation location problems such as IPS (Wu, Kao, and Yang 2012).
4. Also, the managerial implications of the current research in LBS seem to be left behind comparing to other relevant disciplines such as information systems-oriented researches (Rinaldi 2009). There seems to be another gap on the management perspectives and consequences of the Location Based User Generated data that can unlock new insights on the managerial enchantments that can be utilized for research purposes (Tiru et al. 2010) (Azlan et al.2017)
5. The impact of Location Analytics into Business intelligence has been confirmed by the fact that more than 60% of the big data on the internet has geo-reference component. Hence Location Analytics is a new area that was merged by Business Intelligence (Rybarczyk et al.2018

**2.1 EXISTING SYSTEM ARCHITECTURE**



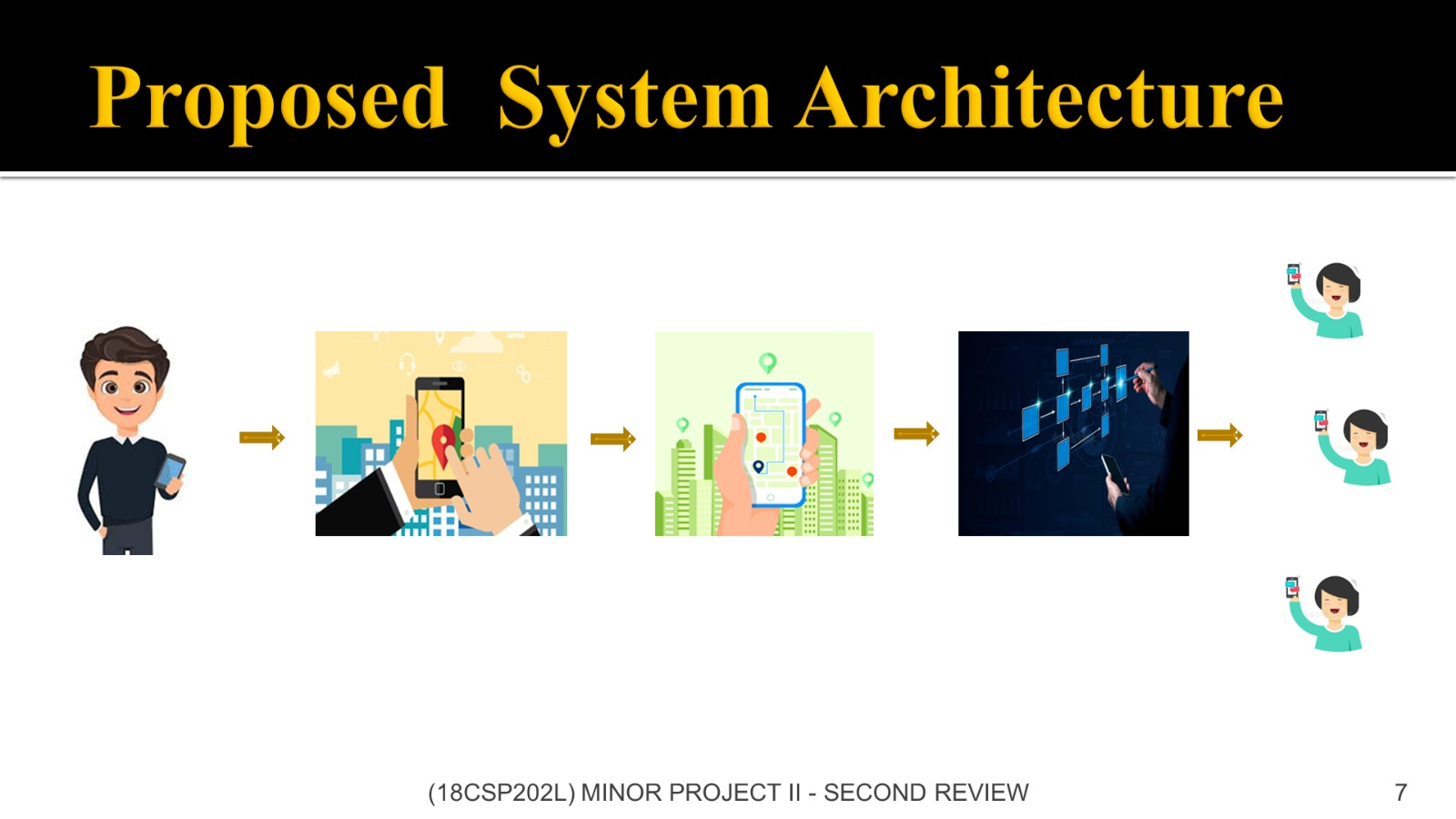
**2.2 DRAWBACKS OF EXISTING SYSTEM ARCHITECTURE**

The location can be shared to particularly one contact.

It takes time to send the location when we are in emergency.

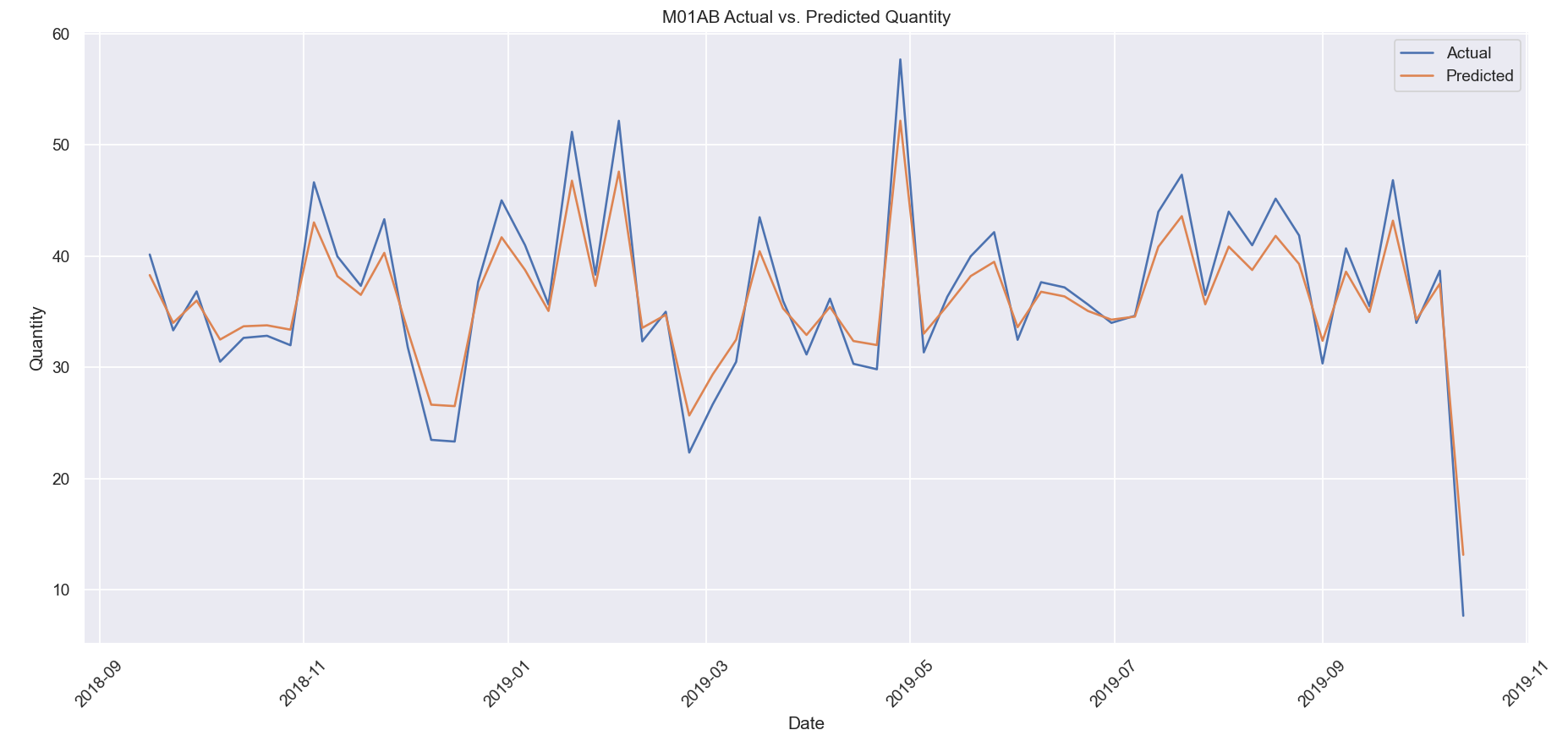
Existing system needed.

**2.3 PROPOSED SYSTEM ARCHITECTURE**



**CHAPTER 3**

**FEASIBILITY STUDY**

****

* 1. **Actual**
* There are noticeable peaks and troughs, suggesting that demand is not constant but varies periodically
* Significant peaks appear around late December 2018, March 2019, and May 2019, indicating increased demand during these periods.
* There are sharp declines in demand at specific points, such as late December 2018 and October 2019, with the latter showing a steep drop-off. This might reflect lower demand or end-of-cycle inventory depletion.
* The overall trend shows consistent variability, but the range remains relatively stable, typically between 20 and 50 units, with occasional spikes.
* The actual quantity data reveals significant demand fluctuations, likely influenced by seasonal factors or external events, which could be crucial for planning in the pharmaceutical supply chain.
  1. **Predict**
* The predicted quantities mirror the actual demand, showing peaks and troughs at nearly the same points, indicating that the predictive model ca overall pattern well.
* The predicted values generally range between 20 and 50 units, aligning with the actual quantities.
* There are occasional minor differences, where the predicted values slightly underestimate or overestimate the actual demand, particularly during sharp peaks or drops.
* The largest deviations are visible during periods of sudden demand spikes, such as in late December 2018 and May 2019.
* The sharp decline at the end of the graph (October 2019) is accurately predicted, suggesting that the model effectively captures the downward trend.
* The predicted quantities show a strong alignment with the actual demand, demonstrating the reliability of the forecasting model in predicting pharmaceutical supply chain needs. Minor discrepancies during high variability periods suggest opportunities for fine-tuning the model.

**CHAPTER 4**

**PROJECT METHODOLOGY**

**4.1 BLOCK DIAGRAM OF MY COMPANION**

Press “Tap Here” button to share location

**MAP ACTIVITY**

Implementing main() function in android application

Entering details to register

**REGISTER**

**ACTIVITY**

Opens Google map and shows the

**MAIN**

**ACTIVITY**

After registering can login in login page

**HOME ACTIVITY**

Controls over all application

# Figure 4.1 Block diagram of My Companion

**APP ACTIVITY**

**LOGIN ACTIVITY**

This block diagram contains the details about My Companion App. It is a location based service application that helps to share the location to your friends and family.

# 4.2 MODULE DESCRIPTION

**Modules available:**

* Register Activity.
* Login Activity.
* Home Activity
* App Activity
* Main Activity
* Map Activity

**REGISTER ACTIVITY**

Registration Activities means activities that consist of recording of financial instruments and the registration of rights to such financial instruments in the VPS Register.

By entering some required details, we can create an account into the app.

**LOGIN ACTIVITY**

Login Activity is one of the most common things which most of the application have is Login Activity. To have a Login Activity in your android studio project is very simple.

To Implement login Activity you need to create or open an android studio project, give it a name and press Next to configuration panel. After configuration, you just need to choose Login Activity and again press next. In the last choose the name for the name of your activity and press FINISH.

**HOME ACTIVITY**

After clicking this "tap here" button, it will go to the map and it shares the current location to the respective contact number.

**APP ACTIVITY**

App activity means a single activity described in a single sentence in column 2 of the application charge table; “application activity charge” means the charge for the relevant type of application described in columns 3 to 8 of the application charge table, which corresponds to the relevant application activity.

**MAP ACTIVITY**

A Map is an important thing to have in your app as it gives your user a perfect view of the location. The Google map can be implemented easily in android studio project because android studio gives a ready to use activity.

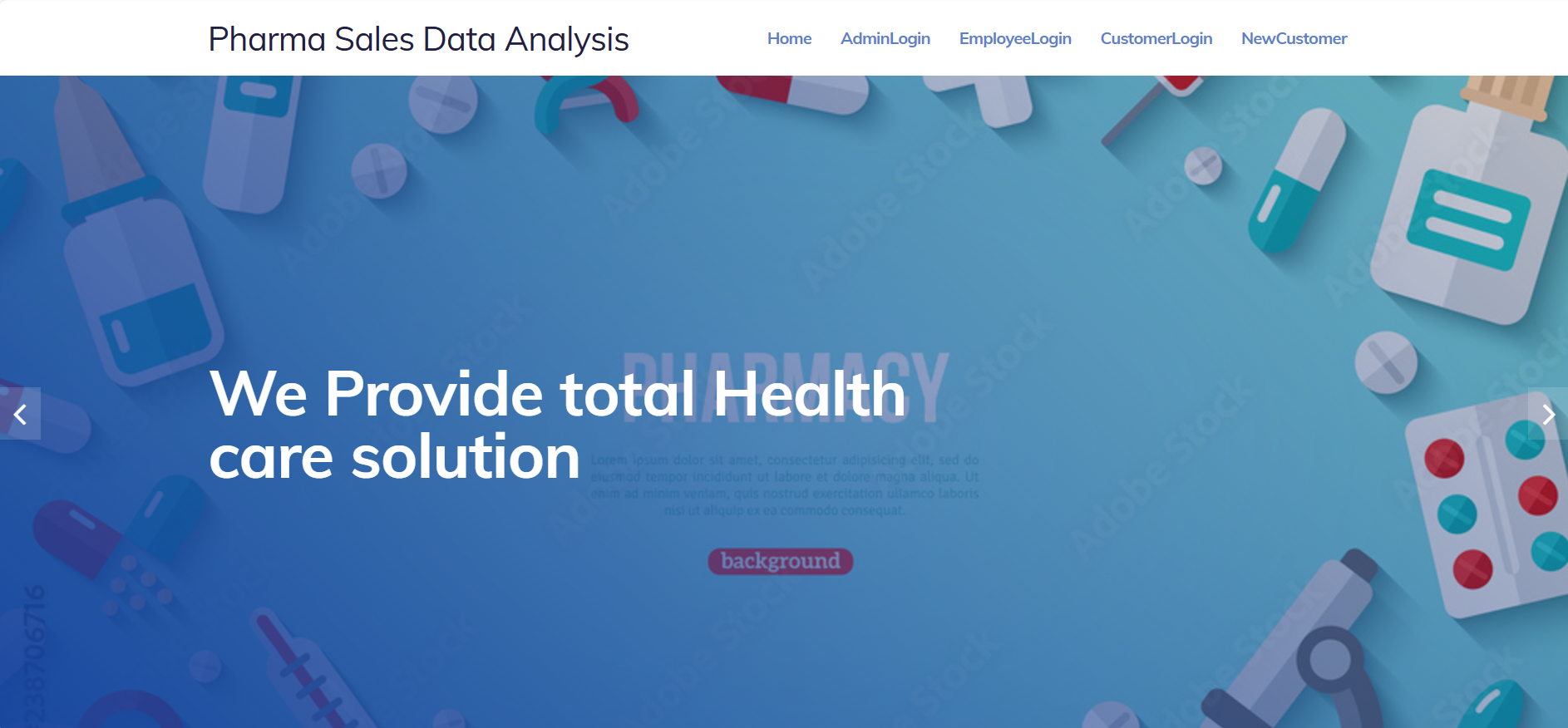
There are few things you have to do to reach this point, first make a new android studio project or open an existing one. In the root directory right-click on the folder and select the new activity and Google Maps Activity. After it gives a name to the activity and you are ready to go.

**MAIN ACTIVITY**

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle.

**CHAPTER 5**

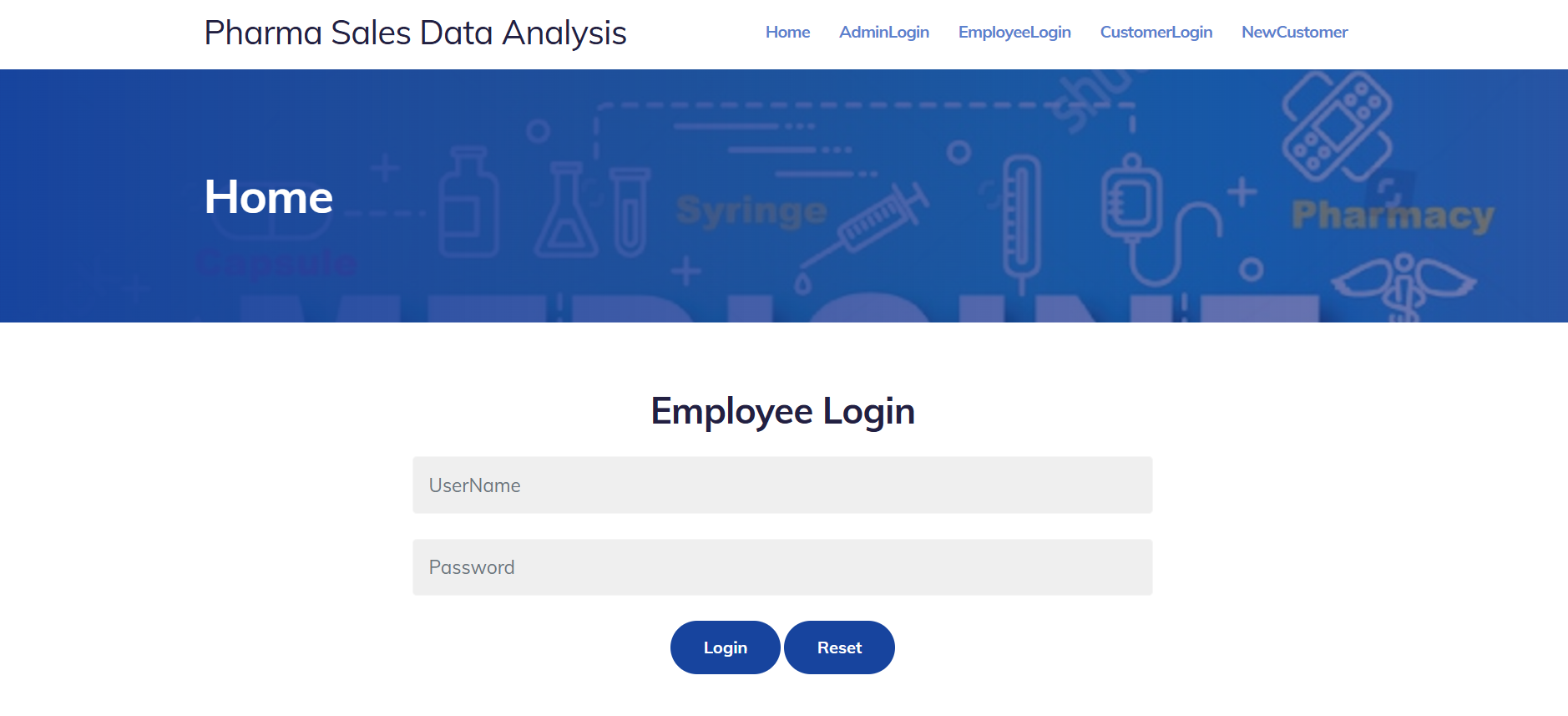
**RESULTS AND DISCUSSION**

****

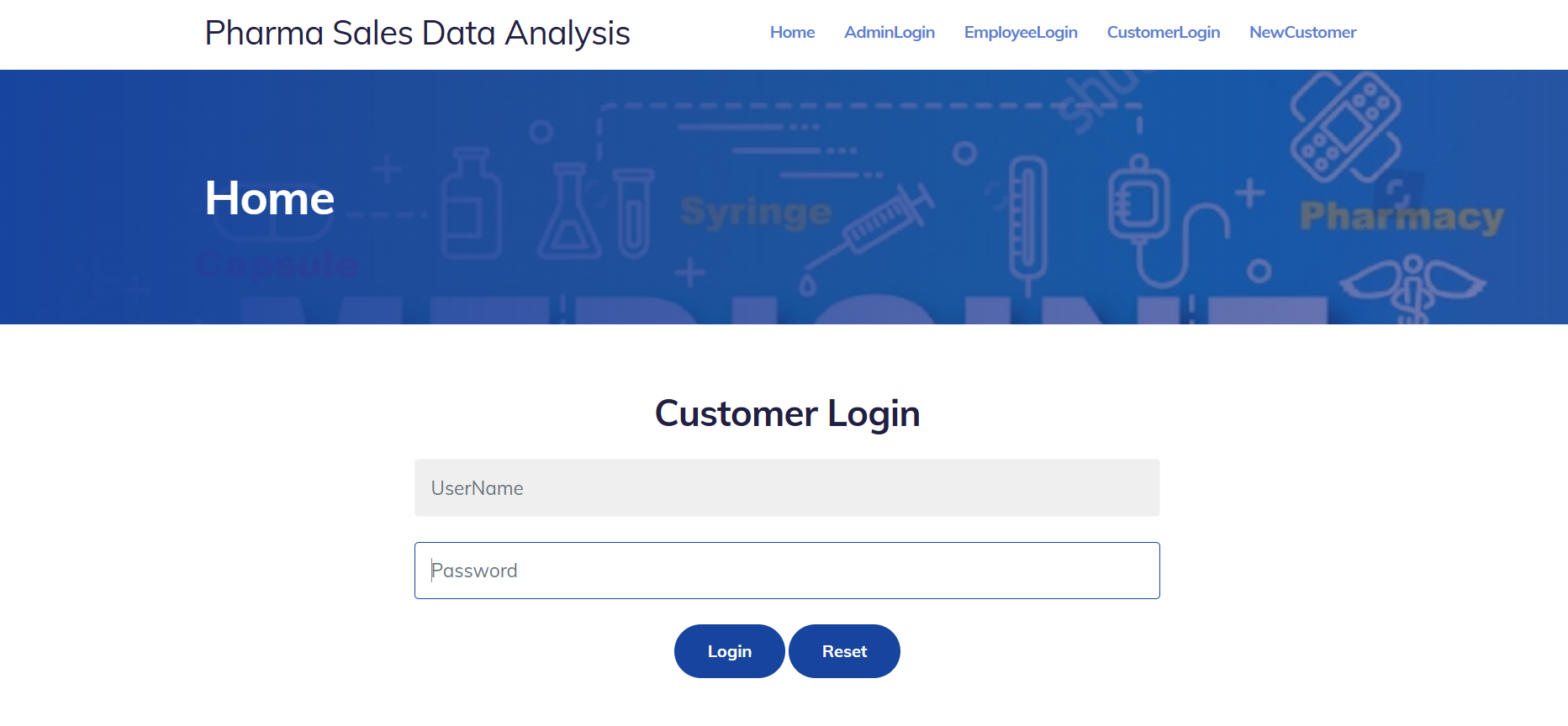
# Figure 5.1 Screenshot of Home Activity Page

# 

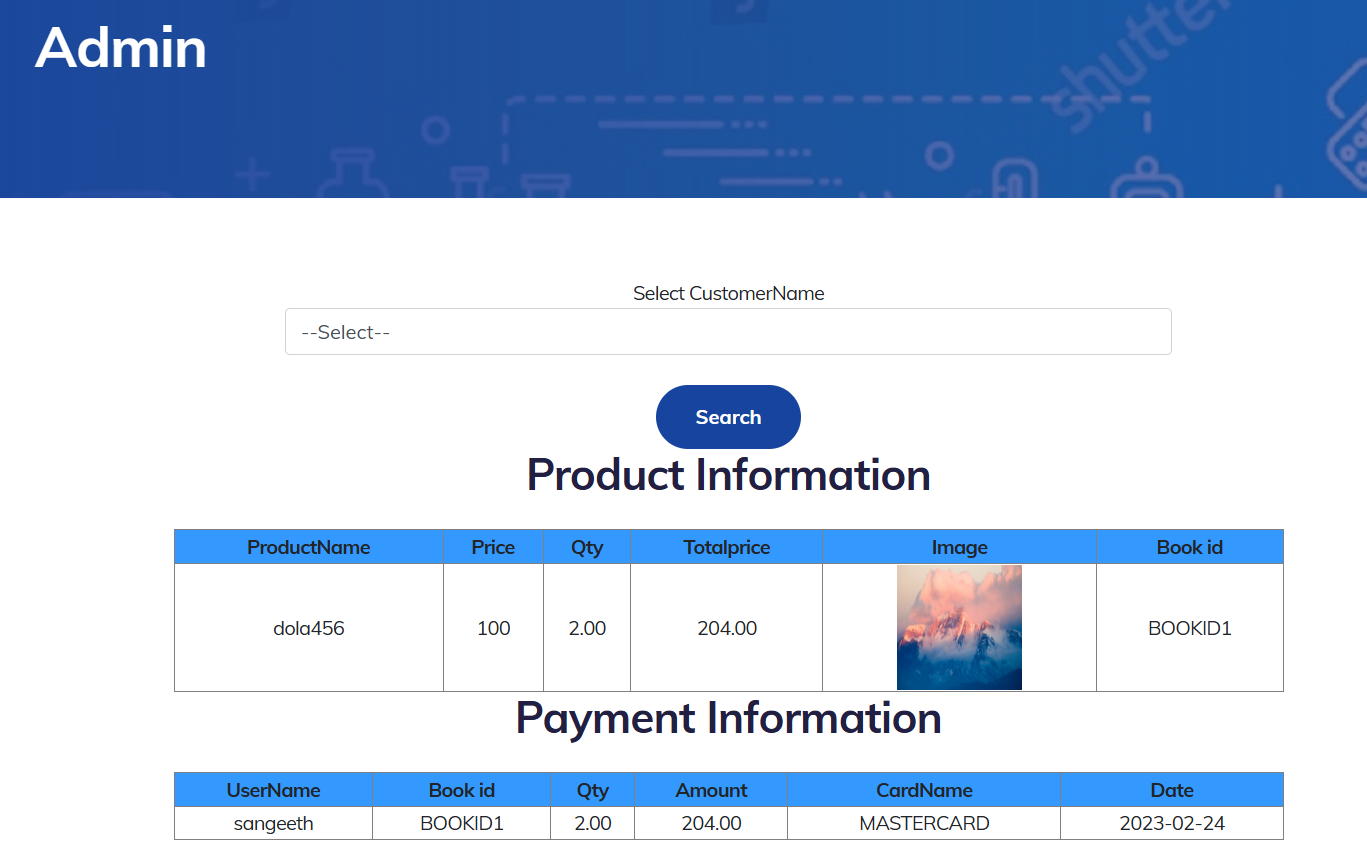
# Figure 5.2 Screenshot of Admin Login Activity Page



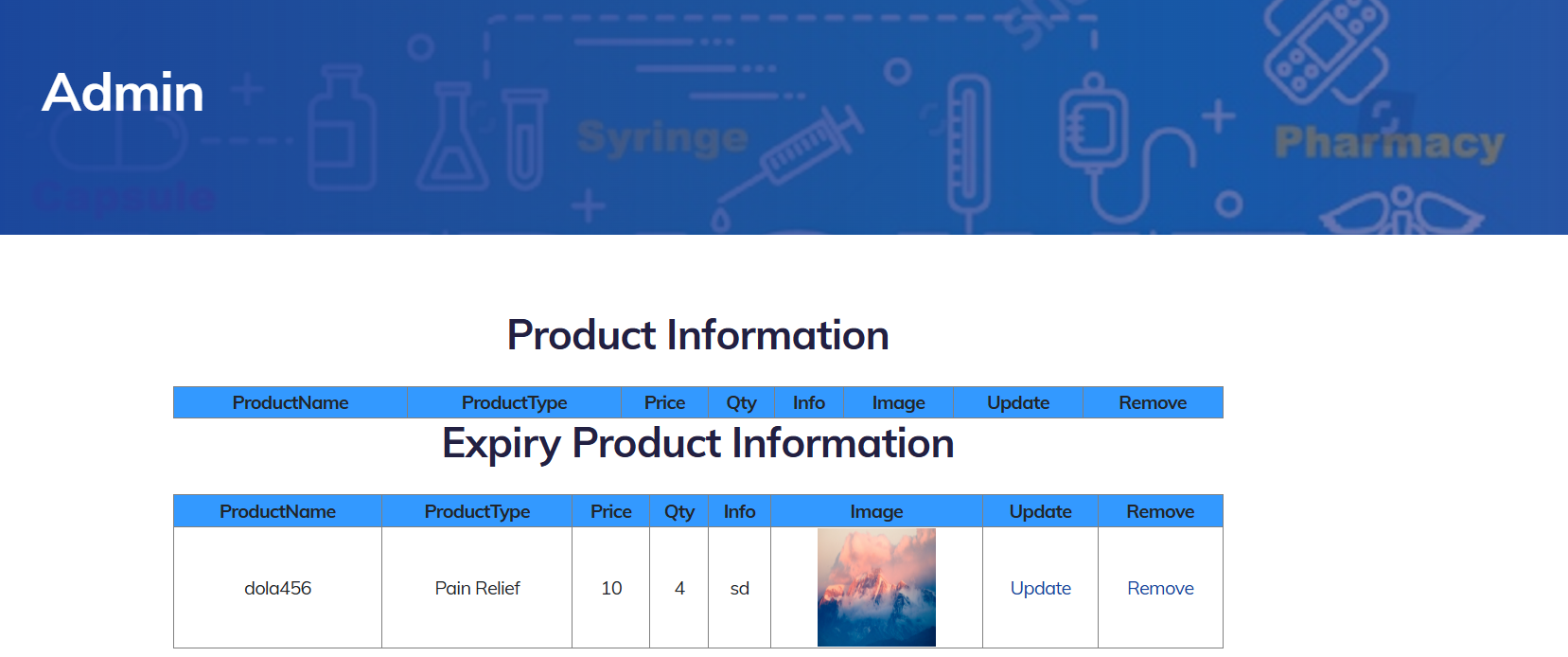
**Figure 5.3 Screenshot of Employee Login Activity Page**

****

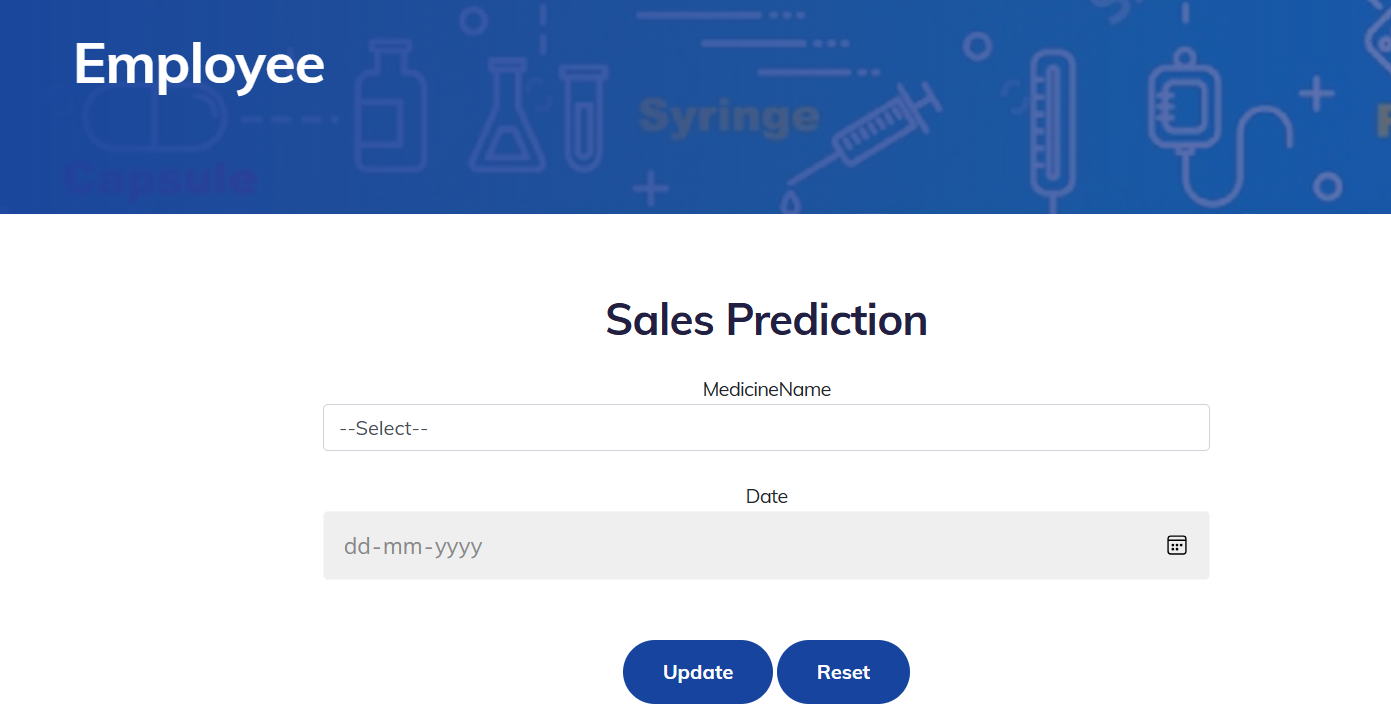
**Figure 5.4 Screenshot of Customer Login Activity Page**



**Figure 5.5 Screenshot of Sales Information Activity Page**



**Figure 5.5 Screenshot of Product Information Activity Page**



**Figure 5.6 Screenshot of Sales Prediction Activity Page**

**CHAPTER 6**

**CONCLUSION AND SCOPE FOR FUTURE WORKS**

From the current research we identify a potential big gap regarding location sharing and the exploitation of research models.

This potential future work can provide with correlations and knowledge that location sharing sector might need to move on and be further established in the near future for both academics and businesses.

**REFERENCES**

1. Deuker, André (2008). "Del 11.2: Mobility and LBS". FIDIS Deliverables. 11 (2).Quercia, Daniele; Lathia, Neal; Calabrese, Francesco; Di Lorenzo, Giusy; Crowcroft, Jon (2010). Recommending Social Events from Mobile Phone Location Data (PDF). 2010 IEEE International Conference on Data Mining. p. 971. doi:10.1109/ICDM.2010.152. ISBN 978-1-4244-9131-5.
2. B. Guo; R. Fujimura; D. Zhang; M. Imai (2011). "Design-in-Play: Improving the Variability of Indoor Pervasive Games". Multimedia Tools and Applications. 59: 259–277. doi: 1007/s11042-010-0711-z. S2CID 9008319.
3. B. Guo, S. Satake, M. Imai. Home-Explorer: Ontology-based Physical Artifact Search and Hidden Object Detection System Archived 2011-09-04 at the Way back Machine. Mobile Information Systems, Vol. 4 No.2 (2008), 81–103, IOS Press, 2008.
4. Schiller, Jochen; Voisard, Agnès (2004-05-21). Location-Based Services. Elsevier. p. 9. ISBN 978-0-08-049172-1.

**APPENDIX**

**SOURCE CODE**

**REGISTER ACTIVITY**

from flask import Flask, render\_template, flash, request, session  
  
import mysql.connector  
import sys  
  
app = Flask(\_\_name\_\_)  
app.config['DEBUG']  
app.config['SECRET\_KEY'] = '7d441f27d441f27567d441f2b6176a'  
  
  
@app.route("/")  
def homepage():  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb ")  
 data = cur.fetchall()  
 return render\_template('index.html', data=data)  
  
  
@app.route("/Home")  
def Home():  
 return render\_template('index.html')  
  
  
@app.route("/AdminLogin")  
def DoctorLogin():  
 return render\_template('AdminLogin.html')  
  
  
@app.route("/NewEmployee")  
def NewEmployee():  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM employeetb ")  
 data = cur.fetchall()  
 return render\_template('NewEmployee.html', data=data)  
  
  
@app.route("/EmployeeLogin")  
def EmployeeLogin():  
 return render\_template('EmployeeLogin.html')  
  
  
@app.route("/UserLogin")  
def UserLogin():  
 return render\_template('UserLogin.html')  
  
  
@app.route("/NewUser")  
def NewUser():  
 return render\_template('NewUser.html')  
  
  
@app.route("/NewProduct")  
def NewProduct():  
 return render\_template('NewProduct.html')  
  
  
@app.route("/AdminHome")  
def AdminHome():  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM regtb ")  
 data = cur.fetchall()  
 return render\_template('AdminHome.html', data=data)  
  
  
@app.route("/AProductInfo")  
def AProductInfo():  
 ii = 0  
 iii = 0  
 import datetime  
 date = datetime.datetime.now().strftime('%Y-%m-%d')  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where Qty >= '6' ")  
 data = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where Qty < 6 ")  
 data1 = cur.fetchall()  
 msg = ''  
  
 for item1 in data1:  
 pname = "ProductName : " + item1[1] + " "  
 qty = "Quantity : " + str(item1[4]) + "\n"  
  
 msg += pname + qty  
 iii = 1  
 # print(msg)  
  
 if iii == 1:  
 sendmsg("manojkumarsakthi061@gmail.com", msg)  
  
 return render\_template('AProductInfo.html', data=data, data1=data1)  
  
  
def sendmsg(Mailid, message):  
 import smtplib  
 from email.mime.multipart import MIMEMultipart  
 from email.mime.text import MIMEText  
 from email.mime.base import MIMEBase  
 from email import encoders  
  
 fromaddr = "projectmailm@gmail.com"  
 toaddr = Mailid  
  
 # instance of MIMEMultipart  
 msg = MIMEMultipart()  
  
 # storing the senders email address  
 msg['From'] = fromaddr  
  
 # storing the receivers email address  
 msg['To'] = toaddr  
  
 # storing the subject  
 msg['Subject'] = "Alert"  
  
 # string to store the body of the mail  
 body = message  
  
 # attach the body with the msg instance  
 msg.attach(MIMEText(body, 'plain'))  
  
 # creates SMTP session  
 s = smtplib.SMTP('smtp.gmail.com', 587)  
  
 # start TLS for security  
 s.starttls()  
  
 # Authentication  
 s.login(fromaddr, "qmgn xecl bkqv musr")  
  
 # Converts the Multipart msg into a string  
 text = msg.as\_string()  
  
 # sending the mail  
 s.sendmail(fromaddr, toaddr, text)  
  
 # terminating the session  
 s.quit()  
  
  
@app.route("/adminlogin", methods=['GET', 'POST'])  
def adminlogin():  
 if request.method == 'POST':  
 if request.form['uname'] == 'admin' or request.form['password'] == 'admin':  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM regtb ")  
 data = cur.fetchall()  
 flash("Login successfully")  
 return render\_template('AdminHome.html', data=data)  
  
 else:  
 flash("UserName Or Password Incorrect!")  
 return render\_template('AdminLogin.html')  
  
  
@app.route("/newemp", methods=['GET', 'POST'])  
def newemp():  
 if request.method == 'POST':  
 name = request.form['name']  
 gender = request.form['gender']  
 mobile = request.form['mobile']  
  
 email = request.form['email']  
  
 address = request.form['address']  
  
 uname = request.form['uname']  
 password = request.form['password']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "INSERT INTO employeetb VALUES ('" + name + "','" + email + "','" + mobile + "','" + address + "','" + uname + "','" + password + "','" + gender + "')")  
 conn.commit()  
 conn.close()  
 flash('Employee Info Register Successfully')  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 # cursor = conn.cursor()  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM employeetb ")  
 data = cur.fetchall()  
 return render\_template('NewEmployee.html', data=data)  
  
  
@app.route("/ARemove")  
def ARemove():  
 id = request.args.get('id')  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "delete from protb where id='" + id + "'")  
 conn.commit()  
 conn.close()  
  
 flash('Product info Remove Successfully!')  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb ")  
 data = cur.fetchall()  
 return render\_template('AProductInfo.html', data=data)  
  
  
@app.route("/emplogin", methods=['GET', 'POST'])  
def emplogin():  
 if request.method == 'POST':  
 username = request.form['uname']  
 password = request.form['password']  
 session['ename'] = request.form['uname']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* from employeetb where username='" + username + "' and Password='" + password + "'")  
 data = cursor.fetchone()  
 if data is None:  
  
 flash('Username or Password is wrong')  
 return render\_template('EmployeeLogin.html', data=data)  
 else:  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM employeetb where username='" + username + "' and Password='" + password + "'")  
 data = cur.fetchall()  
 flash("Login successfully")  
 return render\_template('EmployeeHome.html', data=data)  
  
  
@app.route("/EmployeeHome")  
def EmployeeHome():  
 username = session['ename']  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM employeetb where username='" + username + "' ")  
 data = cur.fetchall()  
 return render\_template('EmployeeHome.html', data=data)  
  
  
@app.route("/newproduct", methods=['GET', 'POST'])  
def newproduct():  
 if request.method == 'POST':  
 pname = request.form['pname']  
 ptype = request.form['ptype']  
 price = request.form['price']  
 qty = request.form['qty']  
 info = request.form['info']  
 date = request.form['date']  
 gst = request.form['gst']  
 mdate = request.form['mdate']  
  
 import random  
 file = request.files['file']  
 fnew = random.randint(1111, 9999)  
 savename = str(fnew) + ".png"  
 file.save("static/upload/" + savename)  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "INSERT INTO protb VALUES ('','" + pname + "','" + ptype + "','" + price + "','" + qty + "','" + info + "','" + date + "','" + savename + "','" + gst + "','" + mdate + "')")  
 conn.commit()  
 conn.close()  
  
 flash('Product Register successfully')  
 return render\_template('NewProduct.html')  
  
  
@app.route("/EProductInfo")  
def EProductInfo():  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb ")  
 data = cur.fetchall()  
 return render\_template('EProductInfo.html', data=data)  
  
  
@app.route("/Remove")  
def Remove():  
 id = request.args.get('id')  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "delete from protb where id='" + id + "'")  
 conn.commit()  
 conn.close()  
  
 flash('Product info Remove Successfully!')  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb ")  
 data = cur.fetchall()  
 return render\_template('EProductInfo.html', data=data)  
  
  
@app.route("/newuser", methods=['GET', 'POST'])  
def newuser():  
 if request.method == 'POST':  
 name = request.form['name']  
 gender = request.form['gender']  
 mobile = request.form['mobile']  
  
 email = request.form['email']  
  
 address = request.form['address']  
  
 uname = request.form['uname']  
 password = request.form['password']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "INSERT INTO regtb VALUES ('" + name + "','" + email + "','" + mobile + "','" + address + "','" + uname + "','" + password + "','" + gender + "')")  
 conn.commit()  
 conn.close()  
 flash('User Register successfully')  
  
 return render\_template('UserLogin.html')  
  
  
@app.route("/userlogin", methods=['GET', 'POST'])  
def userlogin():  
 if request.method == 'POST':  
 username = request.form['uname']  
 password = request.form['password']  
 session['uname'] = request.form['uname']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* from regtb where username='" + username + "' and Password='" + password + "'")  
 data = cursor.fetchone()  
 if data is None:  
  
 flash('Username or Password is wrong')  
 return render\_template('UserLogin.html')  
 else:  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM regtb where username='" + username + "' and Password='" + password + "'")  
 data = cur.fetchall()  
 flash("Login successfully")  
  
 return render\_template('UserHome.html', data=data)  
  
  
@app.route("/Search")  
def Search():  
 import datetime  
 date = datetime.datetime.now().strftime('%Y-%m-%d')  
 print(date)  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where Exdate >='" + date + "'")  
 data = cur.fetchall()  
  
 return render\_template('Search.html', data=data)  
  
  
@app.route("/search", methods=['GET', 'POST'])  
def search():  
 if request.method == 'POST':  
 ptype = request.form['ptype']  
 import datetime  
  
 date = datetime.datetime.now().strftime('%Y-%m-%d')  
 print(date)  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where Exdate >='" + date + "' and ProductType ='" + ptype + "'")  
 data = cur.fetchall()  
  
 return render\_template('Search.html', data=data)  
  
  
@app.route("/search1", methods=['GET', 'POST'])  
def search1():  
 if request.method == 'POST':  
 ptype = request.form['ptype']  
 import datetime  
  
 date = datetime.datetime.now().strftime('%Y-%m-%d')  
 print(date)  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where Exdate >='" + date + "' and ProductType ='" + ptype + "'")  
 data = cur.fetchall()  
  
 return render\_template('index.html', data=data)  
  
  
@app.route("/UserHome")  
def UserHome():  
 uname = session['uname']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 # cursor = conn.cursor()  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM regtb where username='" + uname + "' ")  
 data = cur.fetchall()  
  
 return render\_template('UserHome.html', data=data)  
  
  
@app.route("/Add")  
def Add():  
 id = request.args.get('id')  
 session['pid'] = id  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where id='" + id + "' ")  
 data = cur.fetchall()  
 return render\_template('AddCart.html', data=data)  
  
  
@app.route("/add")  
def add():  
 flash('Please Login!')  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb ")  
 data = cur.fetchall()  
 return render\_template('index.html', data=data)  
  
  
@app.route("/addcart", methods=['GET', 'POST'])  
def addcart():  
 if request.method == 'POST':  
 import datetime  
 date = datetime.datetime.now().strftime('%Y-%m-%d')  
  
 pid = session['pid']  
 uname = session['uname']  
 qty = request.form['qty']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM protb where id='" + pid + "'")  
 data = cursor.fetchone()  
  
 if data:  
 ProductName = data[1]  
 Producttype = data[2]  
 price = data[3]  
 gst = data[8]  
 cQty = data[4]  
  
 Image = data[7]  
  
 else:  
 return 'No Record Found!'  
  
 tprice = float(price) \* float(qty)  
  
 clqty = float(cQty) - float(qty)  
  
 if clqty < 0:  
  
 flash('Low Product ')  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where id='" + pid + "' ")  
 data = cur.fetchall()  
 return render\_template('AddCart.html', data=data)  
  
 else:  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "update protb set Qty='" + str(clqty) + "' where id='" + pid + "' ")  
 conn.commit()  
 conn.close()  
  
 gstamount = (float(gst) / 100) \* float(tprice)  
  
 payamt = gstamount + tprice  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "INSERT INTO carttb VALUES ('','" + uname + "','" + ProductName + "','" + Producttype + "','" + str(  
 price) + "','" + str(qty) + "','" + str(payamt) + "','" +  
 Image + "','" + date + "','0','','" + gst + "')")  
 conn.commit()  
 conn.close()  
  
 flash('Add To Cart Successfully')  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM protb where id='" + pid + "' ")  
 data = cur.fetchall()  
 return render\_template('AddCart.html', data=data)  
  
  
@app.route("/Cart")  
def Cart():  
 uname = session['uname']  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where UserName='" + uname + "' and Status='0' ")  
 data = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "SELECT sum(Qty) as qty ,sum(Tprice) as Tprice FROM carttb where UserName='" + uname + "' and Status='0' ")  
 data1 = cursor.fetchone()  
 if data1:  
 tqty = data1[0]  
 tprice = data1[1]  
 else:  
 return 'No Record Found!'  
  
 return render\_template('Cart.html', data=data, tqty=tqty, tprice=tprice)  
  
  
@app.route("/RemoveCart")  
def RemoveCart():  
 id = request.args.get('id')  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM carttb where id='" + id + "'")  
 data1 = cursor.fetchone()  
  
 if data1:  
 ProductName = data1[2]  
 cQty1 = data1[5]  
  
 else:  
 return 'No Record Found!'  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM protb where ProductName='" + ProductName + "'")  
 data2 = cursor.fetchone()  
  
 if data2:  
 cQty = data2[4]  
  
 else:  
 return 'No Record Found!'  
  
 total = float(cQty1) + float(cQty)  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "update protb set Qty='" + str(total) + "' where ProductName='" + ProductName + "' ")  
 conn.commit()  
 conn.close()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "delete from carttb where id='" + id + "'")  
 conn.commit()  
 conn.close()  
  
 flash('Product Remove Successfully!')  
  
 uname = session['uname']  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where UserName='" + uname + "' and Status='0' ")  
 data = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "SELECT sum(Qty) as qty ,sum(Tprice) as Tprice FROM carttb where UserName='" + uname + "' and Status='0' ")  
 data1 = cursor.fetchone()  
 if data1:  
 tqty = data1[0]  
 tprice = data1[1]  
  
 return render\_template('Cart.html', data=data, tqty=tqty, tprice=tprice)  
  
  
@app.route("/payment", methods=['GET', 'POST'])  
def payment():  
 if request.method == 'POST':  
 import datetime  
 date = datetime.datetime.now().strftime('%Y-%m-%d')  
 uname = session['uname']  
 cname = request.form['cname']  
 Cardno = request.form['cno']  
 Cvno = request.form['cvno']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "SELECT sum(Qty) as qty ,sum(Tprice) as Tprice FROM carttb where UserName='" + uname + "' and Status='0' ")  
 data1 = cursor.fetchone()  
 if data1:  
 tqty = data1[0]  
 tprice = data1[1]  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "SELECT count(\*) As count FROM booktb ")  
 data = cursor.fetchone()  
 if data:  
 bookno = data[0]  
 print(bookno)  
  
 if bookno == 'Null' or bookno == 0:  
 bookno = 1  
 else:  
 bookno += 1  
  
 else:  
 return 'Incorrect username / password !'  
  
 bookno = 'BOOKID' + str(bookno)  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "update carttb set status='1',Bookid='" + bookno + "' where UserName='" + uname + "' and Status='0' ")  
 conn.commit()  
 conn.close()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "INSERT INTO booktb VALUES ('','" + uname + "','" + bookno + "','" + str(tqty) + "','" + str(  
 tprice) + "','" + cname + "','" + Cardno + "','" + Cvno + "','" + date + "')")  
 conn.commit()  
 conn.close()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where UserName='" + uname + "' and Status='1' ")  
 data1 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM booktb where username='" + uname + "'")  
 data2 = cur.fetchall()  
  
 return render\_template('UserBook.html', data1=data1, data2=data2)  
  
  
@app.route("/BookInfo")  
def BookInfo():  
 uname = session['uname']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where UserName='" + uname + "' and Status='1' ")  
 data1 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM booktb where username='" + uname + "'")  
 data2 = cur.fetchall()  
  
 return render\_template('UserBook.html', data1=data1, data2=data2)  
  
  
@app.route("/ASalesInfo")  
def ASalesInfo():  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where Status='1' ")  
 data1 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM booktb ")  
 data2 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT distinct username FROM booktb ")  
 data = cur.fetchall()  
  
 return render\_template('ASalesInfo.html', data1=data1, data2=data2, data=data)  
  
  
@app.route("/ESalesInfo")  
def ESalesInfo():  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where Status='1' ")  
 data1 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM booktb")  
 data2 = cur.fetchall()  
  
 return render\_template('ESalesInfo.html', data1=data1, data2=data2)  
  
  
@app.route("/asale", methods=['GET', 'POST'])  
def asale():  
 if request.method == 'POST':  
 uname = request.form['username']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM carttb where UserName='" + uname + "' and Status='1' ")  
 data1 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT \* FROM booktb where username='" + uname + "'")  
 data2 = cur.fetchall()  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cur = conn.cursor()  
 cur.execute("SELECT distinct username FROM booktb ")  
 data = cur.fetchall()  
  
 return render\_template('ASalesInfo.html', data1=data1, data2=data2, data=data)  
  
  
@app.route("/Update")  
def Update():  
 uid = request.args.get('uid')  
 session["uid"] = uid  
  
 return render\_template('Update.html')  
  
  
@app.route("/update", methods=['GET', 'POST'])  
def update():  
 if request.method == 'POST':  
 price = request.form['price']  
 Qty = request.form['qty']  
 date = request.form['date']  
 mdate = request.form['date']  
  
 conn = mysql.connector.connect(user='root', password='', host='localhost', database='1ProductExpirypy')  
 cursor = conn.cursor()  
 cursor.execute(  
 "update protb set price='" + price + "',Qty='" + Qty + "',exdate='" + date + "',Mdate ='" + mdate + "' where id='" +  
 session[  
 'uid'] + "' ")  
 conn.commit()  
 conn.close()  
  
 flash('Product Info Update')  
  
 return render\_template('AProductInfo.html')  
  
  
@app.route("/APredict")  
def APredict():  
 return render\_template('APredict.html')  
  
  
@app.route("/pred", methods=['GET', 'POST'])  
def pred():  
 if request.method == 'POST':  
 ptype = request.form['ptype']  
 mdate = request.form['mdate']  
 from datetime import datetime  
  
 date\_string = mdate  
  
 # Convert to a datetime object  
 date\_object = datetime.strptime(date\_string, "%Y-%m-%d")  
  
 # Extract year, month, and day  
 year = date\_object.year  
 month = date\_object.month  
 day = date\_object.day  
 mres = "Medicine Name:"+ptype + " Date: "+mdate + "Medicine Need Count: "  
  
 if ptype == "M01AB":  
  
 import numpy as np  
 from tensorflow.keras.models import load\_model  
 from sklearn.preprocessing import StandardScaler  
  
 # Load the trained model  
 model = load\_model('M01AB.h5')  
  
 single\_date\_features = [day, month] # Example placeholder for features  
  
 # Convert the input to a 2D array  
 single\_date\_features = np.array(single\_date\_features).reshape(1, -1)  
  
 # Standardize the input features using the previously fitted scaler  
 # scaled\_features = scaler.transform(single\_date\_features)  
  
 # Make the prediction  
 predicted\_scaled\_quantity = model.predict(single\_date\_features)  
  
 print(int(predicted\_scaled\_quantity[0, 0]))  
  
 mres = mres + str( int(predicted\_scaled\_quantity[0, 0]))  
  
  
 elif ptype == "M01AE":  
 import numpy as np  
 from tensorflow.keras.models import load\_model  
 from sklearn.preprocessing import StandardScaler  
  
 # Load the trained model  
 model = load\_model('M01AE.h5')  
  
 single\_date\_features = [day, month] # Example placeholder for features  
  
 # Convert the input to a 2D array  
 single\_date\_features = np.array(single\_date\_features).reshape(1, -1)  
  
 # Standardize the input features using the previously fitted scaler  
 # scaled\_features = scaler.transform(single\_date\_features)  
  
 # Make the prediction  
 predicted\_scaled\_quantity = model.predict(single\_date\_features)  
  
 print(int(predicted\_scaled\_quantity[0, 0]))  
  
 mres = mres + str( int(predicted\_scaled\_quantity[0, 0]))  
  
 else:  
 import numpy as np  
 from tensorflow.keras.models import load\_model  
 from sklearn.preprocessing import StandardScaler  
  
 # Load the trained model  
 model = load\_model('N02BA.h5')  
  
 single\_date\_features = [day, month] # Example placeholder for features  
  
 # Convert the input to a 2D array  
 single\_date\_features = np.array(single\_date\_features).reshape(1, -1)  
  
 # Standardize the input features using the previously fitted scaler  
 # scaled\_features = scaler.transform(single\_date\_features)  
  
 # Make the prediction  
 predicted\_scaled\_quantity = model.predict(single\_date\_features)  
  
 print(int(predicted\_scaled\_quantity[0, 0]))  
  
 mres = mres + str( int(predicted\_scaled\_quantity[0, 0]))  
  
 return render\_template('APredict.html', res=mres)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True, use\_reloader=True)