HIMESH RAY

22-06-2024

# Title: AI-Powered Medicine Delivery for Local Pharmacies

## 1.Problem Statement

Local pharmacies often struggle with managing inventory, increasing sales, and ensuring timely delivery of medicines.

Customers demand quick delivery, especially for essential medications, but local pharmacies lack the infrastructure and technology to meet these needs efficiently.

## 2.Market/Customer/Busines Need Assessment

### Market Need:

- Increased demand for quick and reliable medicine delivery services.
- Rising competition from online medicine delivery platforms.
- Need for efficient inventory management to avoid stockouts and overstocking

### Customer Needs:

- Timely delivery of medicines, especially for urgent requirements.
- Personalized recommendations based on medical history and previous purchases.
- Easy access to a wide range of medications.

### Business Needs:

- Improved sales through enhanced customer engagement and loyalty.
- Efficient inventory management to reduce costs and improve profitability.

- Competitive edge through technology adoption.

## 3.Target Specifications and Characterization

### Target Customers:

- Local pharmacies with limited technological infrastructure.
- Customers seeking quick and reliable medicine delivery.
- Elderly and chronically ill patients requiring regular medication.

### Customer Characteristics:

- Small to medium-sized local pharmacies.
- Customers from urban and semi-urban areas.
- Tech-savvy individuals comfortable with using mobile apps for shopping.

## 4.External Search

### Information Sources:

- Online articles and reports on the pharmacy and healthcare delivery market.

- Case studies of successful online medicine delivery platforms.

- Customer reviews and feedback on existing medicine delivery services.

## 5.Benchmarking Alternate Products

### Existing Solutions:

- **PharmEasy**: Offers online medicine ordering and delivery but focuses on larger urban markets.
- **1mg**: Provides a comprehensive platform for medicine delivery, diagnostics, and consultations.
- **Netmeds**: Another popular platform for online medicine delivery.

### Comparison:

- Our solution focuses on local pharmacies, offering quicker delivery within one hour.
- Emphasis on personalized recommendations using AI.
- Direct support for local businesses, helping them compete with larger platforms.

# 6.Applicable Patents

## Relevant Patents:

- Patent on AI-based inventory management systems.
- Patent on real-time delivery tracking algorithms.

# 7.Applicable Regulations

## Government and Environmental Regulations:

- Compliance with local pharmacy regulations and healthcare standards.
- Adherence to data privacy laws  for handling customer information.
- Environmental regulations for sustainable delivery practices.

# 8.Applicable Constraints

## Constraints:

- Need for initial investment in technology and infrastructure.
- Limited technical expertise of local pharmacy staff.
- Budget constraints for small businesses.

# 9.Business Model (Monetization Idea)

## Monetization Strategy:

- **Commission-Based Model**: Charge a commission on each sale made through the app.

- **Subscription Model**: Offer premium features and services for a monthly subscription fee.

- **Delivery Fees**: Charge a small fee for delivery within one hour.

# 10.Concept Generation

## Idea Generation Process:

- Brainstorming sessions with stakeholders (pharmacists, customers, delivery personnel).
- Analysis of pain points and gaps in current medicine delivery systems.
- Conceptualizing an integrated platform that leverages AI for personalized service and efficient logistics.

# Concept Development

## Product/Service Development:

- Develop a mobile app for customers to order medicines and track deliveries.
- Create a web dashboard for pharmacies to manage inventory and view analytics.
- Implement AI algorithms for personalized recommendations and efficient delivery routing.

## Product Prototype:

- **Mobile App**: User-friendly interface for customers to browse, order, and track medicines.
- **Web Dashboard**: Comprehensive tool for pharmacies to manage inventory, view sales data, and optimize operations.
- **AI Algorithms**: Personalized recommendations, demand forecasting, and real-time delivery optimization.

## DIAGRAM

**Customer -> Mobile App -> Order Placement -> Inventory Check -> AI Recommendation -> Delivery Dispatch -> Real-Time**

**Tracking -> Delivery Confirmation-> Web Dashboard -> Inventory Management -> Sales Analytics -> AI Insights**

## 11.Product Details

### How Does It Work?

- Customers browse and order medicines through the mobile app.
- AI algorithms provide personalized recommendations and ensure efficient inventory management.
- Orders are dispatched with real-time tracking, ensuring delivery within one hour.

### Data Sources:

- Customer purchase history and medical records (with consent).
- Inventory data from local pharmacies.
- Real-time delivery and logistics data.

### Algorithms, Frameworks, Software Needed:

- Machine Learning algorithms for recommendations and demand forecasting.
- Real-time tracking and delivery optimization algorithms.
- Software frameworks: Flask/Django for backend, React Native/Flutter for frontend, PostgreSQL for database.

### Team Required to Develop:

- Data Scientists and Machine Learning Engineers.
- Frontend and Backend Developers.
- UX/UI Designers.
- Project Managers and Business Analysts.

### Cost Estimates:

- Initial Development: INR 20,00,000
- Monthly Maintenance and Operations: INR 87,500

## 12.Code Implementation/Validation on Small Scale

### Basic Visualizations:

- Sales trends and inventory levels over time.
- Customer demographics and purchase patterns.

### Simple EDA and ML Modelling:

- Exploratory Data Analysis on customer and sales data.
- Machine Learning model to predict demand and optimize inventory.

## Code Implementation

### Data Collection and Preparation

### Datasets Needed:

**Medicine Inventory Data:** This includes information about the medicines available, quantities, and prices.

**Customer Purchase Data**: Historical purchase data, including customer IDs, medicine IDs, quantities purchased, and dates.

**Delivery Data**: Historical delivery times and logistics information.

**Possible Sources:**

- Public healthcare datasets (e.g., Kaggle Healthcare Datasets)
- Synthetic data generation for simulation purposes
- Data from partnering pharmacies

## AI Model Development

## Technologies: Python, scikit-learn, pandas

### CODING IN PYTHON

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Load datasets
customer_data = pd.read_csv('customer_data.csv')
drug_data = pd.read_csv('drug_data.csv')
delivery_data = pd.read_csv('delivery_data.csv')
```

```
# Data preprocessing
# Merge datasets on a common column if available
data = pd.merge(customer_data, drug_data, on='CustomerID')
data = pd.merge(data, delivery_data, on='OrderID')

# Basic Analysis
print(data.head())
print(data.describe())

# Train a simple predictive model
X = data[['Feature1', 'Feature2', 'Feature3']]
y = data['DeliveryTime']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
```

```
print(f'Mean Absolute Error: {mae}'
```

## Backend Development

## Technologies: Flask/Django, PostgreSQL/MySQL

## Flask API Example:

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql://username:password@localhost/medicinedb'
db = SQLAlchemy(app)

class Pharmacy(db.Model):
     id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    inventory = db.relationship('Inventory', backref='pharmacy',
lazy=True)

class Inventory(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    medicine_name = db.Column(db.String(100))
    quantity = db.Column(db.Integer)
    price = db.Column(db.Float)
    pharmacy_id = db.Column(db.Integer,
db.ForeignKey('pharmacy.id'), nullable=False)
```

```
@app.route('/register', methods=['POST'])
def register_pharmacy():
    data = request.get_json()
    new_pharmacy = Pharmacy(name=data['name'])
    db.session.add(new_pharmacy)
    db.session.commit()
    return jsonify({"message": "Pharmacy registered successfully"}), 201


@app.route('/inventory', methods=['POST'])
def add_inventory():
    data = request.get_json()
    new_inventory = Inventory(
        medicine_name=data['medicine_name'],
        quantity=data['quantity'],
        price=data['price'],
        pharmacy_id=data['pharmacy_id']
    )
    db.session.add(new_inventory)
    db.session.commit()
    return jsonify({"message": "Inventory added successfully"}), 201


if __name__ == '__main__':
    app.run(debug=True)
```

## Frontend Development

## Technologies: React Native/Flutter for mobile app, React.js for web dashboard

## Example React Native Component for Pharmacy Registration:

```
import React, { useState } from 'react';
import { View, TextInput, Button, Alert } from 'react-native';
import axios from 'axios';

const RegisterPharmacy = () => {
    const [name, setName] = useState('');

    const handleSubmit = async () => {
        try {
            const response = await axios.post('http://localhost:5000/register', { name });
            Alert.alert('Success', response.data.message);
        } catch (error) {
```

```
      Alert.alert('Error', 'Failed to register pharmacy');
    }
  };

  return (
    <View>
      <TextInput
        placeholder="Pharmacy Name"
        value={name}
        onChangeText={setName}
      />
      <Button title="Register Pharmacy" onPress={handleSubmit} />
    </View>
  );
};

export default RegisterPharmacy;
```

## Integration and Deployment

## Technologies: Docker, AWS/GCP for deployment

## Dockerfile Example:

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
```

```
ENV NAME World


# Run app.py when the container launches

CMD ["python", "app.py"]
```

## Deployment Steps

### 1. Build Docker Image:

```
docker build -t medicine-delivery-app .
```

### 2.Run Docker Container:

```
docker run -p 5000:80 medicine-delivery-app
```

### 3.Deploy on AWS/GCP:

Use services like AWS ECS or Google Cloud Run to deploy the Docker container.

## 14.Conclusion

- This AI-powered medicine delivery app for local pharmacies addresses the need for efficient inventory management, personalized customer service, and timely delivery.

- By leveraging AI and machine learning, local pharmacies can enhance their operational efficiency, increase sales, and provide superior customer experiences. The proposed business model ensures sustainable revenue generation, making it a viable and profitable solution for local businesses.