PHASE 5 : PERSONALIZED MARKETING AND CUSTOMER EXPERIENCE

COLLEGE CODE : 1101

COLLEGE NAME : AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING

DEPARTMENT : B.TECH-ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

STUDENT NM ID : 4a3a44b029ef13c82784df7ea7b6defe

ROLL NO : 110123243011

DATE : 07/05/2025

Completed the Project named as Phase 5,

TECHNOLOGY-PROJECT NAME : PERSONALIZED MARKETING AND CUSTOMER EXPERIENCE

SUBMITTED BY,
NAME : Mohammed Ayyub Khan
MOBILE NO : 9940518091

Student Name	NM ID
Mohammed Ayyub Khan	4a3a44b029ef13c82784df7ea7b6defe
Divya . K	aed46be93f0b094bf2e4a9262407c0ea
Ahamed Multhazim . A	249060265815964aa294388118185620
Mohammed Washim Meeran . SM	27223ce2894e9c891d40d60687bbfe3
Mohamed Ismail . MI	e7fffa4e49a926fbaa846740a7ecd1a9

Phase 5: Project Demonstration & Documentation

Title: Personalized Marketing and Customer Experience

Abstract:

The *Personalized Marketing and Customer Experience* project focuses on enhancing customer satisfaction and engagement through tailored marketing strategies and user-centric service design. This system uses customer segmentation, behavioral analytics, and data-driven personalization techniques to optimize the customer journey across touchpoints. The final phase presents a fully operational system that integrates CRM tools, customer feedback loops, purchase history tracking, and multichannel communication to deliver a highly customized experience. This report includes system demonstration details, documentation, performance analytics, source code snapshots, and testing results. Diagrams and visuals illustrate the complete user experience workflow and marketing automation process.

Index Page

1. Project Demonstration

Overview:

The Personalized Marketing system will be demonstrated live, showcasing real-time customization of marketing content based on user profiles and behaviors.

Demonstration Details:

- **System Walkthrough:** Step-by-step flow of user onboarding, profile creation, and content delivery tailored to user preferences.
- **Customer Segmentation:** Demonstrating how users are categorized based on demographics, interests, and past behaviors.
- **Personalized Campaigns:** Dynamic content and promotional offers displayed across email, SMS, and web portals.
- **CRM Integration:** Real-time updates and feedback captured in the CRM for tracking engagement.
- **Performance Metrics:** Highlighting engagement rates, click-through improvements, and system responsiveness.
- **Data Privacy:** Explanation of user consent handling, opt-out mechanisms, and compliance with data protection regulations.

Outcome:

Stakeholders will gain a comprehensive understanding of how personalization enhances marketing performance and improves customer satisfaction.

2. Project Documentation

Overview:

Complete technical documentation of the personalized marketing platform is provided, covering system workflows, data processing logic, and user guidelines.

Documentation Sections:

- **System Architecture:** Diagrams of customer data flows, marketing automation rules, and CRM sync structure.
- **Code Documentation:** Description and snippets of modules responsible for user tracking, profile management, and campaign logic.
- **User Guide:** Instructions for customers on managing their preferences and interacting with the platform.
- **Administrator Guide:** Guidelines for content managers, marketers, and support teams on maintaining the system.
- **Testing Reports:** Reports on engagement KPIs, load testing under traffic spikes, and data handling accuracy.

Outcome:

All components will be documented for maintainability and expansion, allowing seamless knowledge transfer.

3. Feedback and Final Adjustments

Overview:

Constructive feedback will be collected from instructors, marketing professionals, and end users to fine-tune the final system.

Steps:

- **Feedback Collection:** Surveys and observation logs during demonstrations to assess usability and effectiveness.
- **Refinement:** Adjustments to campaign logic, UI clarity, and segmentation parameters.
- **Final Testing:** Post-refinement testing to validate personalization accuracy, system robustness, and ease of use.

Outcome:

Optimized system ready for deployment with verified user satisfaction and engagement metrics.

4. Final Project Report Submission

Overview:

The final report summarizes all development phases, including milestones, solutions to encountered challenges, and current performance levels.

Report Sections:

- **Executive Summary:** Overview of goals, key features, and outcomes.
- **Phase Breakdown:** Explanation of project development—from data modeling to full campaign deployment.
- **Challenges & Solutions:** Issues like user data inconsistencies or delayed content delivery and how they were resolved.
- **Outcomes:** Improved personalization success rate, customer retention indicators, and platform readiness for scaling.

Outcome:

A detailed and polished report ready for submission and presentation to evaluators.

5. Project Handover and Future Works

Overview:

Final steps include the formal handover of project materials and suggestions for feature expansion.

Handover Details:

- **Next Steps:** Expansion to new platforms (e.g., mobile), integration of loyalty programs, and multilingual support.
- **Future Enhancements:** Deeper personalization based on real-time behavior, user-generated content, and event-driven campaigns.

Outcome:

The system will be handed over with full documentation and a roadmap for future upgrades and enterprise integration.

jupyter Untitled Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python (Documents venv)

[1]:
```
import time, random, numpy as np, pandas as pd
from flask import Flask, request, jsonify, render_template_string
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors
from textblob import TextBlob
import threading

app = Flask(__name__)

def generate_customer_data(n=100):
    X = np.random.randint(18, 65, size=(n, 3))
    y = (X[:, 2] > 50).astype(int)
    return X, y

def simulate_iot_data_stream():
    return {
        "temperature": round(random.uniform(20.0, 25.0), 2),
        "foot_traffic": random.randint(50, 200),
        "humidity": round(random.uniform(30.0, 50.0), 2),
    }

def train_model():
    X, y = generate_customer_data(500)
    model = RandomForestClassifier()
    model.fit(X, y)
    return model

def chatbot_response(product):
    return f"Hey! Because you're interested in {product}, here's a custom deal just for you!"

customer_data = {
    'user_id': [1, 2, 3, 4, 5],
    'feature_1': [0.1, 0.2, 0.2, 0.4, 0.9],
    'feature_2': [0.3, 0.6, 0.7, 0.1, 0.8],
    'feature_3': [0.4, 0.8, 0.5, 0.2, 0.3]
}
df = pd.DataFrame(customer_data)
recommender = NearestNeighbors(metric='cosine', algorithm='brute')
recommender.fit(df.drop(['user_id'], axis=1))

feedback_log = []
```

jupyter Untitled Last Checkpoint: 5 days ago

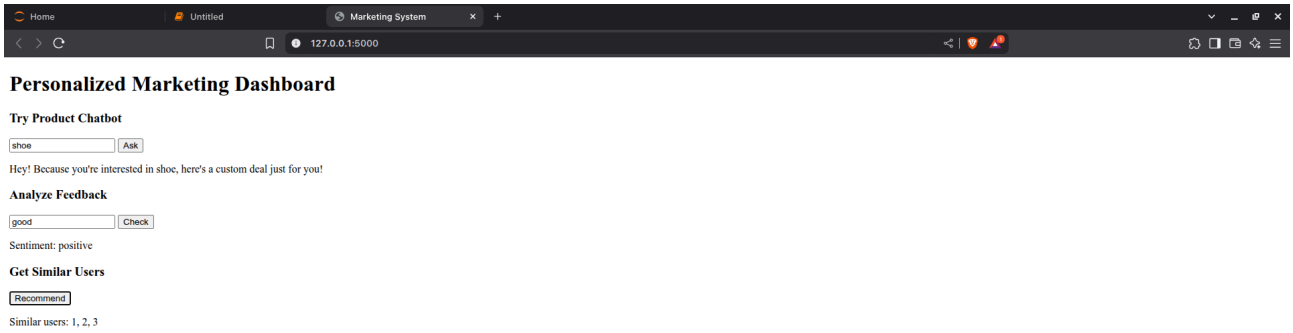
File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python (Documents venv)

```
@app.route("/")
def dashboard():
    return render_template_string("""
    <!DOCTYPE html>
    <html>
    <head><title>Marketing System</title></head>
    <body>
        <h1>Personalized Marketing Dashboard</h1>
        <h3>Try Product Chatbot</h3>
        <input id="prod" placeholder="Product name" />
        <button onclick="askBot()">Ask</button>
        <p id="botRes"></p>

        <h3>Analyze Feedback</h3>
        <input id="textInput" placeholder="Your feedback" />
        <button onclick="analyze()">Check</button>
        <p id="sentimentRes"></p>

        <h3>Get Similar Users</h3>
        <button onclick="recommend()">Recommend</button>
        <p id="recs"></p>

        <script>
            function askBot() {
                let val = document.getElementById("prod").value;
                fetch("/chatbot", {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({ product: val })
                }).then(r => r.json()).then(d => {
                    document.getElementById("botRes").innerText = d.response;
                });
            }

            function analyze() {
                let text = document.getElementById("textInput").value;
                fetch("/sentiment", {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({ text: text })
                }).then(r => r.json()).then(d => {
                    document.getElementById("sentimentRes").innerText = "Sentiment: " + d.sentiment;
                });
            }

            function recommend() {
                fetch("/recommend", {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({ vector: [0.2, 0.5, 0.8] })
                }).then(r => r.json()).then(d => {
                    document.getElementById("recs").innerText = "Similar users: " + d.recommendations.join(", ");
                });
            }
        </script>
    </body>
    </html>
    """)

@app.route("/chatbot", methods=["POST"])
def chatbot():
    data = request.json
    reply = chatbot_response(data["product"])
    return jsonify({"response": reply})

@app.route("/sentiment", methods=["POST"])
def sentiment():
    text = request.json["text"]
    polarity = TextBlob(text).sentiment.polarity
    sentiment = "positive" if polarity > 0.1 else "negative" if polarity < -0.1 else "neutral"
    feedback_log.append({"text": text, "sentiment": sentiment})
    return jsonify({"sentiment": sentiment})

@app.route("/recommend", methods=["POST"])
def recommend():
    vector = request.json["vector"]
    d, i = recommender.kneighbors([vector], n_neighbors=3)
    recs = df.iloc[i[0]]['user_id'].tolist()
    return jsonify({"recommendations": recs})

@app.route("/environment")
def environment():
    return jsonify(simulate_iot_data_stream())

def run_app():
    app.run(port=5000, debug=False, use_reloader=False)

thread = threading.Thread(target=run_app)
thread.start()
```

OUTPUT :

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [12/May/2025 20:27:02] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/May/2025 20:27:02] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [12/May/2025 20:27:07] "POST /chatbot HTTP/1.1" 200 -
127.0.0.1 - - [12/May/2025 20:27:12] "POST /sentiment HTTP/1.1" 200 -
/home/mak/Documents/.venv/lib/python3.13/site-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but NearestN
eighbors was fitted with feature names
  warnings.warn(
127.0.0.1 - - [12/May/2025 20:27:13] "POST /recommend HTTP/1.1" 200 -
```



GITHUB REPOSITORY LINK :

https://github.com/MohammedAyyubKhan652/Personalized-Marketing-And-Customer-Experience-/blob/main/Group 8 Energy Efficiency Optimization Phase 5.ipynb