**User Requirements Document (URD)**

**Project Title: Product Comparison and Recommendation Platform**

**Document Version: 1.0**

**Prepared by: [jiang hwang administrators]**

**Date: [16/11/2024]**

**1. Introduction**

**1.1 Purpose**

This document outlines the user requirements for a **Product Comparison and Recommendation Platform** that allows users to:

* Browse and select products from different stores.
* Compare product prices across stores.
* Receive personalized product recommendations based on their preferences and browsing history.

The platform integrates **web scraping**, **RESTful APIs**, **Azure cloud services**, and **machine learning algorithms** to collect and process product data, manage user interactions, and deliver recommendations.

**1.2 Scope**

The platform will:

1. Scrape product and pricing data from multiple online stores.
2. Expose RESTful APIs to fetch product categories, store information, product comparisons, and recommendations.
3. Provide users with personalized product recommendations using machine learning.
4. Implement logging, monitoring, and performance tracking using **Azure services** such as **Prometheus** and **Application Insights**.

**2. System Overview**

**2.1 Functional Overview**

The **Product Comparison and Recommendation Platform** will consist of the following key components:

1. **Web Scraping Module**: Collects product data from external websites (e.g., pricing, availability).
2. **Backend (API Layer)**: Exposes RESTful APIs to handle requests from the frontend and serve data such as product listings, price comparisons, and recommendations.
3. **Frontend (UI Layer)**: The user interface where users can browse categories, view product prices across stores, and receive recommendations.
4. **Machine Learning Pipeline**: Analyzes user behavior and product attributes to generate personalized recommendations.
5. **Cloud Integration**: Use **Azure services** for hosting the system, including databases, APIs, and monitoring.

**3. System Architecture**

**3.1 High-Level Architecture Diagram**

+---------------------+ +---------------------+

| Frontend | | Machine Learning |

| (React/React Native)|----> REST API ---->| Pipeline |

+---------------------+ +---------------------+

| |

v v

+---------------------------+ +---------------------------+

| Backend (Node.js) |<------------------>| Web Scraping Module |

| (Express / Flask API) | +---------------------------+

+---------------------------+ |

| v

v +---------------------------+

+--------------------------+ | Database (Azure SQL |

| Azure API Management |<---------------->| / Cosmos DB) |

+--------------------------+ +---------------------------+

|

v

+---------------------------+

| Azure Monitor & Prometheus|

+---------------------------+

**3.2 Key Components**

**1. Web Scraping Module**

* **Function**: Scrapes product data (name, price, availability) from multiple online stores.
* **Technology**: **Scrapy** or **Puppeteer** for scraping. **Azure Functions** for scheduling and running scraping tasks.
* **Data Storage**: Scraped data is stored in **Azure SQL Database** or **Cosmos DB**.
* **Frequency**: Scraping is done periodically (e.g., daily or weekly) to keep product and price data updated.

**2. Backend (API Layer)**

* **Function**: Exposes RESTful APIs to interact with the frontend and provide data such as product listings, price comparisons, and recommendations.
* **Technology**: **Node.js (Express)** or **Python (Flask/Django)** for API implementation.
* **Endpoints**:
  + **GET /categories**: Fetch all available product categories.
  + **GET /products**: Fetch products within a selected category, with optional filters (e.g., price range, availability).
  + **GET /product/**

: Fetch detailed information about a selected product.

* + **POST /compare-prices**: Compare prices across different stores for a selected product.
  + **GET /recommendations**: Fetch personalized recommendations for the user.
* **Security**: The API will use **OAuth 2.0** or **JWT** for user authentication.

**3. Frontend (UI Layer)**

* **Function**: The user interface where users browse product categories, view price comparisons, and get recommendations.
* **Technology**: **React** (for web) or **React Native** (for mobile).
* **Components**:
  + Product category browsing and filtering.
  + Price comparison display.
  + Personalized recommendation display.
  + User authentication and profile management.
* **API Integration**: Communicates with the backend via RESTful APIs for fetching data.

**4. Machine Learning Pipeline**

* **Function**: Provides personalized product recommendations based on user interactions (e.g., viewed products, previous purchases).
* **Technology**: **Azure Machine Learning Service** or **Databricks** for model training and deployment. **Python** for building recommendation models (e.g., collaborative filtering, content-based filtering).
* **Recommendation Logic**:
  + **Collaborative Filtering**: Based on user behaviour and product ratings (if available).
  + **Content-Based Filtering**: Based on product attributes like brand, category, price, etc.

**5. Cloud Integration (Azure Services)**

* **Function**: Host the platform on **Azure**, including databases, APIs, and monitoring tools.
* **Services**:
  + **Azure SQL Database** or **Cosmos DB**: Store product, store, and user data.
  + **Azure API Management**: Secure and manage the RESTful API layer.
  + **Azure Functions**: Schedule and run scraping jobs.
  + **Azure Monitor & Prometheus**: Monitor application performance, backend services, and infrastructure health.
  + **Azure Application Insights**: Track errors, API response times, and user behaviour on the frontend.

**4. Functional Requirements**

**4.1 Web Scraping**

* The system must be able to scrape product data from external websites on a regular basis (daily, weekly, etc.).
* The scraping process must extract product details such as:
  + Product name
  + Price
  + Availability
  + Store information (name, URL)
  + Product specifications (e.g., size, color, brand)
* The scraped data must be stored in a structured format in the database (Azure SQL Database or Cosmos DB).

**4.2 RESTful API**

The system must expose the following RESTful APIs:

1. **GET /categories**: Fetch a list of product categories.
2. **GET /products**: Fetch products from a selected category, with optional price range filters.
3. **GET /product/**

: Fetch detailed information about a specific product.

1. **POST /compare-prices**: Compare prices for a selected product across different stores.
2. **GET /recommendations**: Fetch personalized product recommendations for the user.

**4.3 Frontend (UI)**

* The UI must allow users to:
  + Browse and filter products by category.
  + View product details, including pricing, availability, and store information.
  + Compare prices of a selected product across multiple stores.
  + View personalized recommendations based on their browsing history or preferences.

**4.4 Machine Learning Pipeline**

* The system must provide personalized recommendations using either:
  + **Collaborative filtering**: Based on user behaviour (e.g., previously viewed or purchased products).
  + **Content-based filtering**: Based on product attributes like category, price, or brand.
* The machine learning model must be deployed on **Azure Machine Learning** and must be able to provide real-time recommendations through the API.

**4.5 Logging, Monitoring, and Alerts**

* **Prometheus** should be integrated to monitor backend services, APIs, and containers, and track metrics like response times, request counts, and error rates.
* **Azure Monitor** and **Application Insights** should be used to monitor the application, log errors, and provide real-time performance metrics.
* Alerts should be set up for:
  + High response times or API failures.
  + Scraping job failures.
  + Unusual user activity patterns.

**5. Non-Functional Requirements**

**5.1 Scalability**

* The system must be scalable to handle an increasing number of products, users, and requests.
* The web scraping module must be able to scale and handle scraping from multiple websites concurrently.

**5.2 Availability**

* The platform should be highly available with minimal downtime. Azure’s services like **App Service** or **AKS** should ensure high availability and redundancy.

**5.3 Security**

* All APIs must be secured using **OAuth 2.0** or **JWT** to protect user data.
* User credentials should be securely stored, using best practices for password hashing and storage.

**5.4 Performance**

* The system should be responsive with low latency, especially for product lookups, price comparisons, and recommendations.
* Scraping jobs should not overload the system or cause delays in response times.

**5.5 Usability**

* The frontend should be user-friendly, intuitive, and responsive across devices (mobile, tablet, and desktop).

**6. Conclusion**

This **User Requirements Document (URD)** provides a comprehensive overview of the requirements for the **Product Comparison and Recommendation Platform**. By combining web scraping, RESTful APIs, machine learning, and Azure cloud services, the platform aims to provide users with an efficient and personalized product browsing and purchasing experience.