Design Document

Ghazal Bakhshande

Output:

```
(venv) D:\work\Porsline\Task\recommendation_system>python main.py
Interaction matrix shape: (6, 8)

Recommendations for User 1 (Existing user with rich interaction history (Alice)):
- Gaming Mouse (Product ID: 106)
  Reasons: Because you interacted with Electronics products
- Laptop Stand (Product ID: 105)
  Reasons: Users similar to you purchased this
- Smartphone Case (Product ID: 102)
  Reasons: Popular product

Recommendations for User 2 (Existing user with rich interaction history (Bob)):
- Electric Toothbrush (Product ID: 104)
  Reasons: Popular product
- Wireless Earbuds (Product ID: 101)
  Reasons: Because you interacted with Electronics products, Users similar to you purchased this
- Yoga Mat (Product ID: 103)
  Reasons: Popular product

Recommendations for User 5 (Cold start user (Eve)):
- Winter Jacket (Product ID: 108)
  Reasons: Popular product
- Wireless Earbuds (Product ID: 101)
  Reasons: Popular product
- Laptop Stand (Product ID: 105)
  Reasons: Popular product
```

```
Recommendations for User 6 (User testing contextual recommendations (Frank)):
- Wireless Earbuds (Product ID: 101)
  Reasons: Because you interacted with Electronics products
- Yoga Mat (Product ID: 103)
  Reasons: Popular product
- Laptop Stand (Product ID: 105)
  Reasons: Users similar to you purchased this

--- Testing Recommendations with a New Product Added ---
Interaction matrix shape: (6, 8)

Recommendations for User 2 after adding new product:
- Electric Toothbrush (Product ID: 104)
  Reasons: Popular product
- Wireless Earbuds (Product ID: 101)
  Reasons: Because you interacted with Electronics products, Users similar to you purchased this
- Yoga Mat (Product ID: 103)
  Reasons: Popular product
```

Tests Output:

```
(venv) D:\work\Porsline\Task\recommendation_system>python -m unittest discover tests
Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.Interaction matrix shape: (5, 5)
.
----------------------------------------------------------------------
Ran 11 tests in 1.968s

OK
```

Data Structures

- **df_users:** Contains user profiles with fields like user_id, name, and device.
- **df_products:** Holds product details such as product_id, name, category, tags, and rating.
- **df_interactions:** Records user interactions with products, including user_id, product_id, and weight (e.g., ratings).
- **df_purchase_history:** Logs users' purchase transactions, capturing user_id, product_id, purchase_date, and quantity. This helps in understanding users' buying patterns and preferences.
- **df_browsing_history:** Tracks users' browsing activities, including user_id, product_id, and timestamp. This provides insights into users' interests and potential future purchases.
- **df_context:** Stores contextual information like peak days and seasons for product categories.
- **df_products_encoded:** An encoded version of df_products, where categorical features are transformed into numerical formats suitable for clustering and modeling.
- **User-Item Interaction Matrix (user_item_matrix):** A pivot table of user interactions used for collaborative filtering.
- **TF-IDF Matrix:** Used in content-based filtering to represent product metadata numerically.

Algorithm Choices and Reasoning

1. **Content-Based Filtering**
   - **Algorithm:** Uses TF-IDF vectorization of product metadata (tags, categories) and computes cosine similarity between products.
   - **Reasoning:** Recommends products similar to those a user has previously interacted with, focusing on product attributes.
2. **Collaborative Filtering**

- **Algorithm:** Calculates user-user similarity based on interaction patterns using cosine similarity.
- **Reasoning:** Identifies users with similar preferences to recommend products that similar users have liked.

3. **Matrix Factorization**
   - **Algorithm:** Employs Singular Value Decomposition (SVD) to factorize the user-item interaction matrix.
   - **Reasoning:** Uncovers latent factors representing underlying preferences, improving recommendation accuracy.
4. **Clustering**
   - **Algorithm:** Applies K-Means clustering to group users and products based on interaction data and features.
   - **Reasoning:** Enhances collaborative filtering by considering cluster-based similarities.
5. **Context-Aware Adjustments**
   - **Algorithm:** Modifies recommendations based on contextual factors (e.g., device type, time, season).
   - **Reasoning:** Improves relevance by incorporating situational variables affecting user behavior.
6. **Hybrid Recommendation**
   - **Algorithm:** Combines outputs from all models using a weighted scoring system to generate final recommendations.
   - **Reasoning:** Leverages the strengths of each method to create a more robust and personalized recommendation list.

Optimization Techniques and Trade-Offs

1. **Data Encoding and Preprocessing**
   - **Technique:** One-hot encoding of categorical features (e.g., product tags) for numerical processing.
   - **Trade-Off:** Increases dimensionality, which can impact computational efficiency but is necessary for algorithms requiring numerical input.
2. **Dimensionality Reduction**
   - **Technique:** Uses PCA to reduce the dimensionality of high-dimensional data (e.g., interaction matrix) before clustering.
   - **Trade-Off:** Balances computational efficiency with potential loss of information. A reduced number of components may speed up processing but could omit important variance.
3. **Weighted Hybrid Model**
   - **Technique:** Assigns weights to different recommendation sources to control their influence.
   - **Trade-Off:** Requires careful tuning; inappropriate weights can skew recommendations away from personalization.
4. **Handling Cold Start Problem**
   - **Technique:** For new users with no interaction history, the system recommends popular products or products popular within clusters.

- **Trade-Off:** Recommendations may be less personalized but ensure that new users receive relevant suggestions.

5. **Diversity Enhancement**
    - **Technique:** Incorporates diversity in recommendations by ensuring a mix of product categories.
    - **Trade-Off:** May slightly reduce relevance if less-preferred categories are included but increases the chance of user engagement with diverse options.
6. **Model Performance Optimization**
    - **Technique:** Uses efficient algorithms and limits model complexity (e.g., limiting the number of latent factors in SVD).
    - **Trade-Off:** Improves speed and scalability at the potential cost of reduced model expressiveness.