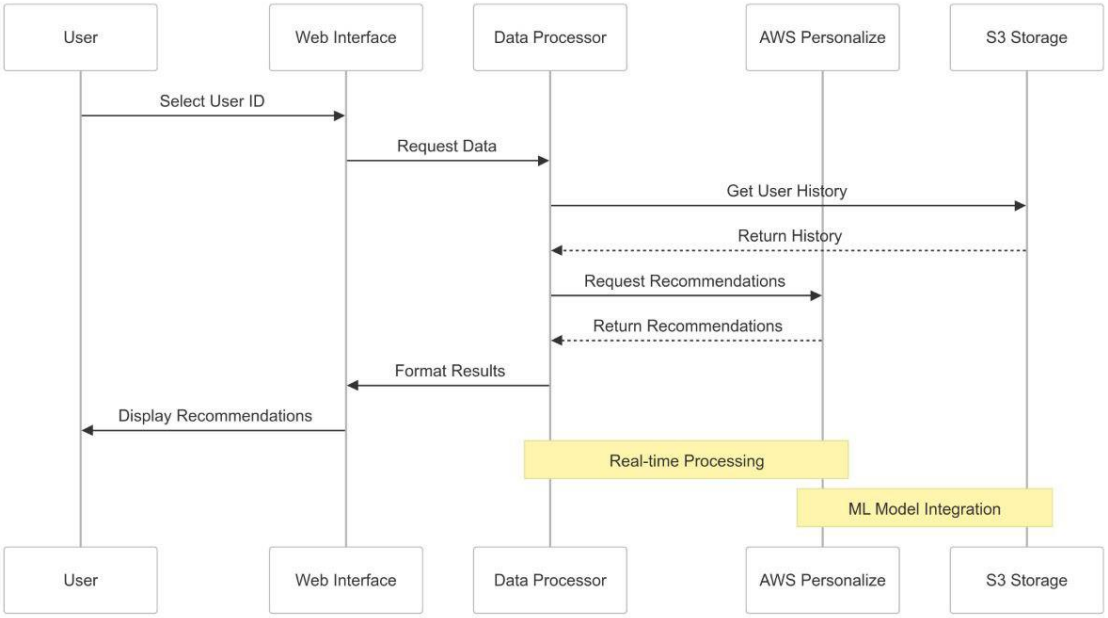
Building a Scalable and Cost-Effective Recommendation System on AWS

Introduction

This report outlines the design and implementation plan for a Service Recommendation System (SRS) built on Amazon Web Services (AWS). The SRS aims to provide personalized recommendations to users based on their preferences and behavior.



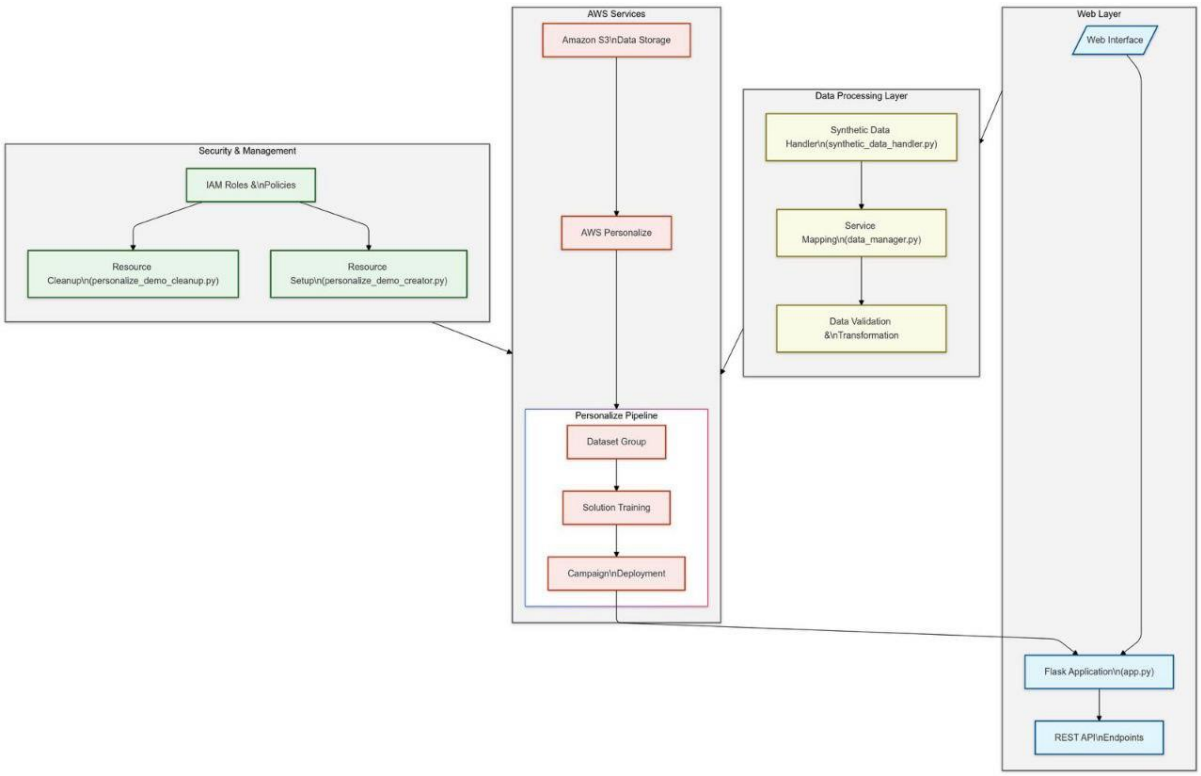
The Recommendation system's workflow

Participants:

- **User:** The end-user who interacts with the system.
- **Web Interface:** The web application that the user interacts with.
- **Data Processor:** A component that processes user data before sending it to AWS Personalize.
- **AWS Personalize:** The Amazon Web Services service responsible for generating recommendations.
- **S3 Storage:** An Amazon S3 bucket used to store user data.

Sequence of Events:

1. **Select User ID:** The user selects a user ID through the web interface.
2. **Request Data:** The web interface sends a request to the Data Processor to retrieve the user's history.
3. **Get User History:** The Data Processor retrieves the user's history from S3 Storage.
4. **Return History:** The Data Processor returns the user's history to the web interface.
5. **Request Recommendations:** The web interface sends the user's history to AWS Personalize to request recommendations.
6. **Return Recommendations:** AWS Personalize returns the recommendations to the web interface.
7. **Format Results:** The web interface formats the recommendations for display.
8. **Display Recommendations:** The web interface displays the recommendations to the user.



The architecture of a recommendation system powered by AWS Personalize

High-Level Architecture

```
Client Layer
├── Web Interface
└── Flask Application Server

Processing Layer
├── Data Transformer
└── Service Mapping Engine

AWS Infrastructure
├── Amazon S3
├── AWS Personalize
└── IAM Security
```

Components:

- **Web Interface Flask application:** The web application that the user interacts with.

```
# Flask Application Structure
app/
├── templates/
│   └── index.html      # User interface
├── static/
│   └── style.css       # Styling
└── app.py             # Application logic
```

Key Components for Web Interface:

1. **Flask Application:**
    - The Flask framework is used to create a web application.
    - Routes are defined to handle user requests and display relevant content.
    - Templates are used to render HTML pages.
2. **User Interaction:**
        - View personalized recommendations.
        - Provide feedback on recommendations.
        - View their interaction history.
3. **Personalize Interaction:**
    - The web application interacts with the AWS Personalize service to:
        - Retrieve personalized recommendations based on user data.
        - Send updated user interactions to Personalize for model retraining

- **Data Processing Layer:**

```
class RecommendationEngine:
    """
    Manages recommendation generation

    Features:
    - Real-time processing
    - Personalized rankings
    - Confidence scoring
    """
    def get_recommendations(self, user_id):
        return personalize_runtime.get_recommendations(
            campaignArn=campaign_arn,
            userId=user_id
        )
```

- **Data directory setup:** Creates a directory named dataset to store source data and processed files.
- **Source data validation:**
  - Checks for the existence and presence of data in the source CSV file (aws_synthetic_service_recommendation_data.csv).
  - Ensures the source data contains required columns (User ID, AWS Service, Interaction Type, and Rating).
  - Validates that there are at least two unique services and one user in the data.
- **Service mapping creation:**
  - Reads unique service names from the source data.
  - Creates a mapping dictionary that assigns a unique ID to each service name (preserving the original service name).
  - Saves the service mapping to a separate CSV file (service_mapping.csv).
- **Data preparation:**
  - Transforms the source data into a format suitable for AWS Personalize:
    - Converts User ID and AWS Service columns to strings.
    - Assigns timestamps to each interaction based on current time.
    - Selects and reorders columns to USER_ID, ITEM_ID (AWS Service mapped to ID), EVENT_TYPE, and TIMESTAMP.
  - Validates the prepared data:
    - Checks for the presence of required columns.
    - Ensures there are no null values in required columns.
    - Confirms there are at least two unique users and items in the data.
- **Data writing:**
  - Writes the prepared dataset to a CSV file (interactions.csv) within the dataset directory.
  - Validates the written file size to ensure successful data storage.

- **<u>AWS Personalize:</u>** The Amazon Web Services service responsible for generating recommendations.

```
def setup_personalize():
    """
    Configures AWS Personalize

    Steps:
    1. Create dataset group
    2. Define schema
    3. Create solution
    4. Deploy campaign
    """
    dataset_group = create_dataset_group()
    solution = train_solution(dataset_group)
    return deploy_campaign(solution)
```

Purpose: Creates and configures a Personalize solution for use with an AWS account.

Functionalities:

- Deletes existing schemas with a specified name if they exist to avoid conflicts.
- Initializes a PersonalizeManager object to handle Personalize resources.
- Configures various components of a Personalize solution, including dataset groups, schemas, data import, and solution creation.
- Evaluates the performance of the created solution.
- Creates a campaign using the Personalized solution.
- Logs informational and error messages throughout the process.

Error Handling:

- Catches exceptions during setup and logs error messages.
- Raises exceptions to terminate the script in case of errors

- **Amazon S3 Storage:** An Amazon S3 bucket used to store user data.

```
class S3Manager:
    """
    Manages S3 operations

    Responsibilities:
    - Data storage
    - File management
    - Access control
    """
    def store_data(self, data):
        return self.s3_client.put_object(
            Bucket=self.bucket_name,
            Key=self.file_key,
            Body=data
        )
```

The S3Manager class manages S3 bucket and IAM role setup for Amazon Personalize.

- **create_bucket_s3**: Creates an S3 bucket with a unique name based on account ID and region.
- **upload_file_to_s3**: Uploads specified data files to the S3 bucket for Amazon Personalize.
- **configure_bucket_policy**: Grants personalize.amazonaws.com read and list access to the bucket.
- **configure_iam_roles_personalize**: Sets up an IAM role for Personalize with necessary permissions and attaches S3 access policies.
- **cleanup**: Removes IAM policies, deletes the role, and clears/deletes the S3 bucket to clean up resources.

- **Security & Management:**
    - **IAM Roles & Policies:** This component manages the security and access control of the system using AWS Identity and Access Management (IAM).
    - **Setup Script (personalize_demo_creator.py)**: Handles the initial setup, including dataset and campaign creation in AWS Personalize. Requires permissions for AWS Personalize and S3 access.
    - **Cleanup Script (personalize_demo_cleanup.py)**: Manages resource deletion, including datasets and campaigns. Needs permissions to remove AWS Personalize resources and delete S3 files.

- **Resource Cleanup:** This component cleans up resources after they are no longer needed.

  Cleanup process:

    - Deletes the Personalize campaign (with retries and waiting for the campaign to be in a deletable state).
    - Deletes the Personalize solution (with retries and waiting for the solution to be in a deletable state).
    - Deletes the Personalize dataset group, schema, and dataset (if they exist).
    - Deletes S3 bucket and contents associated with Personalize (e.g., the bucket name might be it-service-bucket).
    - Deletes IAM roles and policies associated with Personalize (e.g., the role name might be PersonalizeITServiceRole)

Workflow:

1. **User Interaction:** The user interacts with the web interface, providing input or selecting items.
2. **Data Processing:** The input data is processed through the Data Processing Layer:
    - **Synthetic Data:** If synthetic data is used, it is generated by the Synthetic Data Handler.
    - **Data Validation & Transformation:** The data is validated for correctness and transformed into the required format.
    - **Mapping Service:** The data is mapped to the schema required by AWS Personalize.
3. **AWS Personalize:** The processed data is fed into AWS Personalize, which uses machine learning models to generate recommendations based on the user's preferences and past behavior.
4. **Recommendation Generation:** AWS Personalize generates recommendations and returns them to the web interface.
5. **Display Recommendations:** The web interface displays the recommendations to the user.

Implementation Plan Summary for SRS on AWS

Objective: Develop and deploy a scalable, high-performance, and cost-effective recommendation system on AWS.

Key Considerations

- **Scalability**: Ensuring the system can handle growth in data and user requests.
- **Performance**: Using AWS services optimized for recommendation engines.
- **Cost Optimization**: Minimizing expenses through efficient AWS service usage.

AWS Services and Cost Analysis

1. **Amazon S3** (Storage for Dataset)
    - **Cost**: ₹1.85 per GB/month (for 1GB data usage) = ₹2/month
    - **Data Transfer**: Free within the same region (ap-south-1).
    - **Total S3 Cost**: ₹2/month
2. **AWS Personalize** (Core Recommendation Engine)
    - **Data Processing**: ₹1.42 per GB, for ~100MB data ≈ ₹0.14
    - **Training**: ₹19.77/hour, initial training (2-3 hours) ≈ ₹59.31
    - **Inference**: ₹0.40 per 1,000 recommendations, estimated 1,000 recommendations/day ≈ ₹12/month
    - **Total Personalize Cost**: ~₹71.45/month
3. **Verified Monthly AWS Cost**: ~₹75

Development and Maintenance Costs

1. **Development Cost** (One-Time)
    - **Junior Developer** (2 weeks at ₹40,000-50,000/month): ~₹25,000
    - Tasks: Setup & Configuration (2-3 days), Development (7-8 days), Testing (2-3 days), Documentation (1-2 days)

- **Total Development Cost**: ~₹25,000

2. **Monthly Maintenance Cost**
    - **Tasks**: Minimal monitoring and updates (~2-3 hours/month)
    - **Cost**: ~₹1,000-2,000/month for basic maintenance

Total Project Cost Estimate (First Year)

- **One-Time Costs**:
    - Development: ₹25,000
    - Initial AWS Setup: ₹100
    - **Total Initial Cost**: ₹25,100
- **Ongoing Monthly Costs**:
    - AWS Services and Maintenance: ~₹1,575/month
- **First-Year Total**:
    - Initial Costs + (Monthly Costs × 12) = ₹44,000

Summary

This plan leverages AWS services, particularly S3 and Personalize, to build a scalable recommendation system while keeping costs low. Development is done efficiently within two weeks, with minimal ongoing maintenance. The estimated first-year cost is ₹44,000, making this an affordable and robust solution for a recommendation system on AWS.

Key AWS Services Used in the Service Recommendation System

1. Amazon S3 (Simple Storage Service)

- **Purpose**: Amazon S3 is a highly scalable storage service used to store data in the cloud.
- **Key Uses for SRS**:
    - **Store Interaction Data**: S3 can store user interaction data, such as preferences, usage patterns, or clickstream data. This data can then be analyzed or used to train recommendation models.
    - **Store Service Mappings**: S3 can also hold mappings between different services or components used within the SRS.
    - **Data Management**: S3 is effective for general data management tasks, including organizing, accessing, and securing datasets used within the recommendation system.

2. AWS Personalize

- **Purpose**: AWS Personalize is a machine learning service designed for creating personalized recommendations for users in real-time.
- **Key Uses for SRS**:
    - **Real-Time Recommendations**: Personalize enables real-time recommendations based on user behavior, helping to keep recommendations relevant and up-to-date.
    - **User Personalization Recipe**: Personalize offers a range of pre-built machine learning models ("recipes") tailored for different personalization tasks. Choosing an appropriate recipe can improve the recommendation accuracy and relevance.
    - **Campaign Deployment**: Campaigns in AWS Personalize allow you to continuously deliver personalized recommendations to users. The campaign can be updated with new data as it becomes available, ensuring recommendations are dynamically updated.

3. IAM (Identity and Access Management)

- **Purpose**: IAM is an AWS service that manages access to resources securely by defining permissions and roles.

- **Key Uses for SRS**:
    - **Security Roles**: IAM allows you to create roles for different components or services in the SRS, ensuring they have access only to necessary resources.
    - **Access Permissions**: With IAM, you can set precise permissions for users, applications, or services, controlling who can access the SRS and its data.
    - **Service Policies**: IAM policies help define the actions and resources each service can access, protecting sensitive data and operations within the SRS.