High-Level Design Document

Project Title: **Book Recommendation System Using K-Means Clustering**

1. Overview

The Book Recommendation System aims to suggest books to users based on similarity in genres, authors, and publishers. The core of the system leverages **K-Means Clustering** to group books into clusters of similar characteristics. Users can input a book title and receive recommendations of other books within the same cluster.

The system is deployed as a web application, accessible via a front-end interface, and backed by a machine learning model (K-Means) for clustering and recommendation logic.

---

2. Objectives

- **Primary Goal**: Provide book recommendations based on the clustering of similar book attributes.
- **Secondary Goals**:
    - Use machine learning (K-Means) to identify clusters of books with similar attributes.
    - Deploy the application using **Azure Web App** for accessibility.
    - Implement continuous integration and deployment (CI/CD) using **GitHub Actions**.

---

3. System Architecture

3.1 Logical Architecture

The system consists of the following primary components:

1. **User Interface (Front-End)**:
    - A simple HTML/CSS/JavaScript front-end allowing users to input a book title and retrieve recommendations.
    - Users interact with the system via a browser.
2. **Web Server (Back-End)**:
    - A Flask web server that processes user requests and returns recommendations.
    - The back-end also handles routes for:
        - Retrieving book recommendations (/recommend).
        - Fetching all book titles for the autocomplete feature (/get_book_titles).
3. **Machine Learning Model**:
    - **K-Means Clustering** model to classify books based on attributes (Title, Author, Publisher, Genre, Subgenre).

- **TF-IDF** (Term Frequency-Inverse Document Frequency) vectorization used to convert text data into numerical format for clustering.

4. **Database (Data Storage)**:
    - A CSV file (or database) containing book metadata such as Title, Author, Publisher, Genre, SubGenre.
5. **Deployment**:
    - Application hosted on **Azure Web App**.
    - Integration with **GitHub** for source control and deployment pipeline.

3.2 Technical Architecture

Technology Stack:

- **Front-End**: HTML, CSS, JavaScript, Jinja2 (templating with Flask).
- **Back-End**: Python (Flask), Pandas, Scikit-learn, TF-IDF for feature extraction, K-Means for clustering.
- **Database**: CSV file for book metadata.
- **Deployment**: Azure Web App, GitHub Actions for CI/CD.
- **ML Model Management**: MLflow for model logging and versioning.

---

4. Detailed Design

4.1 Data Flow

1. **User Input**: The user enters a book title into the front-end form.
2. **API Request**: The form submission triggers an API request to the Flask back-end with the input book title.
3. **Recommendation Logic**:
    - The back-end uses the K-Means model to determine the cluster for the input book.
    - Books from the same cluster are identified and returned as recommendations.
4. **API Response**: The recommended book titles are returned as a JSON response.
5. **Display Results**: The front-end displays the list of recommended books to the user.

4.2 Back-End Design

Routes:

- GET /: Loads the homepage where the user can input a book title.
- GET /recommend: Accepts a book title as input, retrieves similar books from the same cluster, and returns the recommendations in JSON format.
- GET /get_book_titles: Provides a list of book titles for the autocomplete feature.

Functions:

- **get_all_book_titles()**: Retrieves all book titles from the dataset for the autocomplete feature.
- **get_recommendations(title)**: Takes a book title as input, determines its cluster, and returns books from the same cluster as recommendations.

---

5. Deployment Workflow

The deployment process is automated using GitHub Actions and Azure Web App.

1. **Code Push**: Developers push code to the GitHub repository.
2. **CI/CD Pipeline**:
    - GitHub Actions detects the push event and triggers the CI pipeline.
    - The code is built, and tests are run.
    - The pipeline deploys the latest version of the web application to **Azure Web App** using the stored publish profile.

5.1 Deployment Steps:

- Configure the **Azure Web App** in the Azure Portal.
- Connect the GitHub repository to **Azure** via the Deployment Center in the Azure Web App.
- Add the Azure Web App publish profile as a secret in the GitHub repository.
- Set up the CI/CD pipeline using GitHub Actions.
- Deploy the Flask application, including the K-Means model, to the Azure Web App.

---

6. System Components

6.1 Key Components:

- **ML Model**: K-Means model to cluster books based on their metadata.
- **Web App**: Flask-based application to serve the recommendation system.
- **Database**: A CSV file containing book metadata such as Title, Author, Publisher, Genre, Subgenre.
- **GitHub Actions**: For CI/CD automation and deployment to Azure.
- **Azure Web App**: Hosting platform for the web application.

---

7. Security Considerations

- **Environment Variables**: Sensitive information such as the Azure publish profile is stored in GitHub secrets.

- **Input Validation**: Ensure that user inputs are sanitized to avoid potential security risks like injection attacks.

---

8. Testing and Validation

- Unit tests are implemented for core functions such as get_recommendations() and get_all_book_titles().
- Test the web application by simulating user interactions to ensure the correct functionality of recommendation and autocomplete features.
- The deployment pipeline will automatically run tests during the CI/CD process.

---

9. Future Enhancements

- **Model Improvement**: Improve the clustering algorithm to account for additional features such as user ratings or reviews.
- **User Profiles**: Implement personalized recommendations by integrating user profiles.
- **Database**: Migrate from CSV to a more scalable database (e.g., Azure SQL, MongoDB).
- **UI/UX**: Improve the user interface for a more interactive and modern experience.

---

10. Conclusion

This document outlines the high-level design of the Book Recommendation System. The system uses K-Means clustering to suggest books based on similar attributes and is deployed on Azure Web App using CI/CD through GitHub Actions. Further enhancements can include model improvements, user personalization, and database scalability.

End of Document