Project Documentation: Intelligent Travel Itinerary Generation System

---

Overview

This project aims to build an intelligent travel itinerary generation system that leverages user preferences, past trips, spending habits, and social data such as Instagram photos and reviews. The application generates personalized recommendations for destinations and activities, using both AI/ML techniques and backend engineering.

Tech Stack Used:

- **Languages & Frameworks:** Node.js, Express.js, MySQL
- **Database:** AWS RDS MySQL
- **Version Control:** Git, GitHub
- **Deployment:** AWS Elastic Beanstalk
- **Containerization:** Docker
- **Other Tools:** Nodemon for development, Sequelize ORM, Postman for API testing.

Scope of the Assignment

This project is divided into several key tasks, each focusing on data preprocessing, machine learning-driven segmentation, recommendation engine development, and personalized itinerary generation. Below is a detailed breakdown of each task and its output.

Tasks Breakdown and Implementation

---

1. Data Preprocessing

The goal is to clean and preprocess the input data (users, trips, and Instagram photos). Inconsistent or missing entries are handled, and Instagram metadata is analyzed using text processing techniques.

- **Input:** `users.csv`, `trips.csv`, `instagramPhotos.csv`
- **Output:** Cleaned and validated data for further use.

Preprocessing Steps:

1. **Data Validation:** Checked for missing entries and handled them by either removing or imputing values.
2. **Text Processing:** Extracted meaningful information from Instagram captions and hashtags (e.g., location, activity hashtags).
3. **Data Normalization:** Standardized user trip expenses and Instagram metadata for consistent comparisons.

Code Implementation:

- In the `preprocessing.js` file, a script is written to handle these tasks.
- Libraries like `csv-parser` and `text-analysis` were used.

2. User Segmentation

Using unsupervised machine learning (e.g., clustering), users were segmented based on preferences, past behaviors, and reviews.

- **Input:** Cleaned data from the preprocessing step.
- **Output:** User clusters (e.g., "Adventurers", "Relaxers", "Socialites").

Segmentation Steps:

1. **Feature Engineering:** Extracted features such as "vibe", "preferred activities", "spending patterns", and "frequent destinations".
2. **Clustering Algorithm:** Implemented K-means clustering to segment users into groups based on these features.
3. **User Assignment:** Users were tagged with their cluster IDs.

Code Implementation:

- In the `recommendationController.js`, a function `segmentUsers()` was built using the `KMeans` module from the `ml-kmeans` library.
- The segmentation results are stored in the database, linked to each user.

---

3. Recommendation Engine

The recommendation engine suggests personalized destinations and activities for a new user based on their cluster, preferences, and social context.

- **Input:** User’s cluster ID, past trips, and preferences.
- **Output:** List of personalized recommendations for destination X, activities, and restaurants.

Recommendation Logic:

1. **Matching User Preferences:** Based on the user's cluster, the system matches them to destinations with similar attributes (e.g., "Hawaii" for users in the "Relaxers" group).
2. **Instagram Influence:** Recommendations are further refined by analyzing trending spots and activities in their Instagram photos.
3. **Dynamic Adjustments:** The system adapts the recommendations based on the user's previous travel companions, spending patterns, and reviews.

Code Implementation:

- In `itineraryController.js`, the function `generateRecommendations()` is responsible for generating this output.
- It uses a combination of SQL queries and ML models for prediction.

---

4. Itinerary Generation

The system produces a full itinerary for the selected destination, with day-wise suggestions, approximate costs, and alignment to the user's preferences and past behaviors.

- **Input:** Destination X, user’s preferences, past trip data.
- **Output:** A detailed itinerary for the trip, with costs for each activity.

Itinerary Logic:

1. **Day-Wise Breakdown:** Divides the trip into multiple days, suggesting activities based on the user’s preferences.
2. **Cost Prediction:** Uses past spending habits to estimate the cost of each day’s itinerary (e.g., dining, activities).
3. **Vibe Matching:** Aligns the itinerary to the user's vibe and adjusts the recommendations accordingly.

Code Implementation:

- In the `itineraryController.js`, the function `createItinerary()` builds the day-wise schedule.
- The `trips` and `InstagramPhotos` data are used to fine-tune the recommendations.

System Architecture

1.

Frontend: (Not included in this scope)

2.

- Could be built using React.js or any other frontend framework to interact with the API.

3.

Backend:

4.

- **Node.js/Express.js:** Handles API routes and business logic.
- **Sequelize ORM:** Manages database interactions with AWS RDS MySQL.
- **Machine Learning Algorithms:** Used in user segmentation and recommendation generation.

5.

Database:

6.

- **AWS RDS (MySQL):** Stores user data, trip history, and Instagram photo metadata.
- Schema design is based on the provided CSV data:
- `users`, `trips`, and `instagram_photos` tables.

7.

Containerization:

8.

- **Docker:** The application is containerized using Docker. Docker Compose is used to set up both the application and the MySQL database.

Languages and AWS RDS Instance

1.

Programming Languages:

2.

- **Node.js** for backend logic and API.
- **JavaScript** for core logic.

3.

AWS RDS MySQL:

4.

- Hosted the MySQL instance on AWS RDS.
- **Instance Type:** db.t2.micro
- **Connection String:** Connected using the Sequelize ORM in the `db.js` configuration file.
- Credentials and endpoint stored in the `.env` file.

Project Directory Structure

```
travel-itinerary-api/
├── backend/
│   ├── config/
│   │   └── db.js          # Database configuration (MySQL)
│   ├── controllers/
│   │   ├── itineraryController.js   # Itinerary generation logic
│   │   ├── recommendationController.js   # Logic for recommendations
│   │   └── preprocessing.js  # Handles data preprocessing
│   ├── models/
│   │   └── userModel.js   # Models that interact with the database (optional)
│   ├── routes/
│   │   └── itineraryRoutes.js  # Route definitions
│   ├── data/
│   │   ├── users.csv  # Original dataset
│   │   ├── trips.csv  # Trip data
│   │   └── instagramPhotos.csv  # Instagram data
│   ├── app.js            # Express app configuration
│   └── server.js         # Server entry point
├── node_modules/         # External libraries
├── .env                  # Environment variables
├── package.json          # NPM dependencies
└── README.md             # Documentation
```

Deployment on GitHub

The repository for this project can be found on GitHub at:

Repository URL: https://github.com/pranjalsinha1965/Backend-App-Saathi.Aap

To clone and run the project:

```
git clone https://github.com/pranjalsinha1965/Backend-App-Saathi.Aap.git

cd backend

npm install

npm run dev
```

Evaluation Metrics

The performance of the recommendation engine can be evaluated using the following metrics:

- **User Satisfaction:** Feedback loops from users rating their itineraries.
- **Relevance:** How well the recommendations match user preferences.
- **Novelty:** Introducing new but relevant places in the itineraries based on user behavior.

Standard Based Output:

Simple API Call:

Input:

```
curl http://localhost:3000/api/users/2/destination/Hawaii
```

Output:

```
{
  "user": "Bob",
  "destination": "Hawaii",
  "itinerary": [
    {
      "day": 1,
      "activities": [
        "Relax at the spa",
        "Dinner at Sunset View Lounge"
      ]
    },
    {
      "day": 2,
      "activities": [
        "Reading at Beachside Library",
        "Visit Tranquil Beach"
      ]
    }
  ]
}
```

Output

The following URLs return the desired outputs when the system is deployed locally or on AWS:

1. **Personalized Itinerary:**
   - **Endpoint:** `http://localhost:3000/api/users/1/destination/Hawaii`
   - **Output:** Returns the itinerary in JSON format for user ID 1 for the destination "Hawaii".
2. **Default Route:**
   - **Endpoint:** `http://localhost:3000`
   - **Output:** "Welcome to the Travel Itinerary API!"