Ethan Dsouza
Project 3
April 19, 2024
CS 3200 | Prof. Guerra-Gomez

Sustainable Transportation Tracking Database

I. Overview and Proposal

Problem

In many urban areas, transportation contributes significantly to pollution, congestion, and carbon emissions. While there's a growing interest in sustainable transportation options such as cycling, walking, and public transit, tracking the usage and impact of these alternatives can be challenging without proper tools.

Objective

To develop a database that allows the tracking, management, and promotion of sustainable transportation options within communities. The database would help to collect data on usage patterns, emissions reductions, and user feedback, informing transportation planning and policy decisions.

Tables

Transportation_Mode

- mode_id
- mode_name
- description
- carbon_emission_rate_per_mile

User

- user_id
- username
- email
- password
- location
- preferred_transportation_mode

Trip

- trip_id
- user_id
- mode_id
- start_location
- end_location
- distance_traveled
- trip_purpose
- trip_duration
- carbon_emission_saved

Emissions_Data

- emissions_id
- mode_id
- date
- total_emissions_saved

Feedback

- feedback_id
- user_id
- date
- feedback_text
- satisfaction_rating

Community Event

- event_id
- event_name
- location
- date_time
- description
- organizer

II. Business Requirements

Sustainable Transportation Tracking Database Requirements:

Rules

- Users can register with the system using a unique username, email, and password.
- Users provide their location and preferred transportation mode during registration.
- Users can log their trips, including start and end locations, distance traveled, purpose, and duration.
- The system calculates carbon emissions saved for each trip based on the transportation mode chosen.
- Emissions data is aggregated to track total emissions saved over time.
- Users can provide feedback on their transportation experiences, including text feedback and satisfaction ratings.
- Community events promoting sustainable transportation habits are organized, including car-free days, bike-to-work campaigns, and walking tours.
- Users can participate in community events, track their participation, and provide feedback.
- Incentive programs reward users for choosing sustainable transportation options.
- Rewards earned by users are managed within the system.
- Visualizations and reports can be generated from the database to illustrate usage patterns, emissions reductions, user feedback, and community event participation.
- The database system prioritizes security and privacy, encrypting sensitive user information and adhering to data protection regulations.
- The system is designed to handle large volumes of data and support a growing user base.

Possible Nouns

Denoted by yellow highlights in Rules listed above.

- Users
- Transportation Modes
- Trips
- Emissions Data
- Feedback
- Community Events

- Incentive Programs
- Rewards
- Visualizations
- Reports
- Security
- Privacy
- System

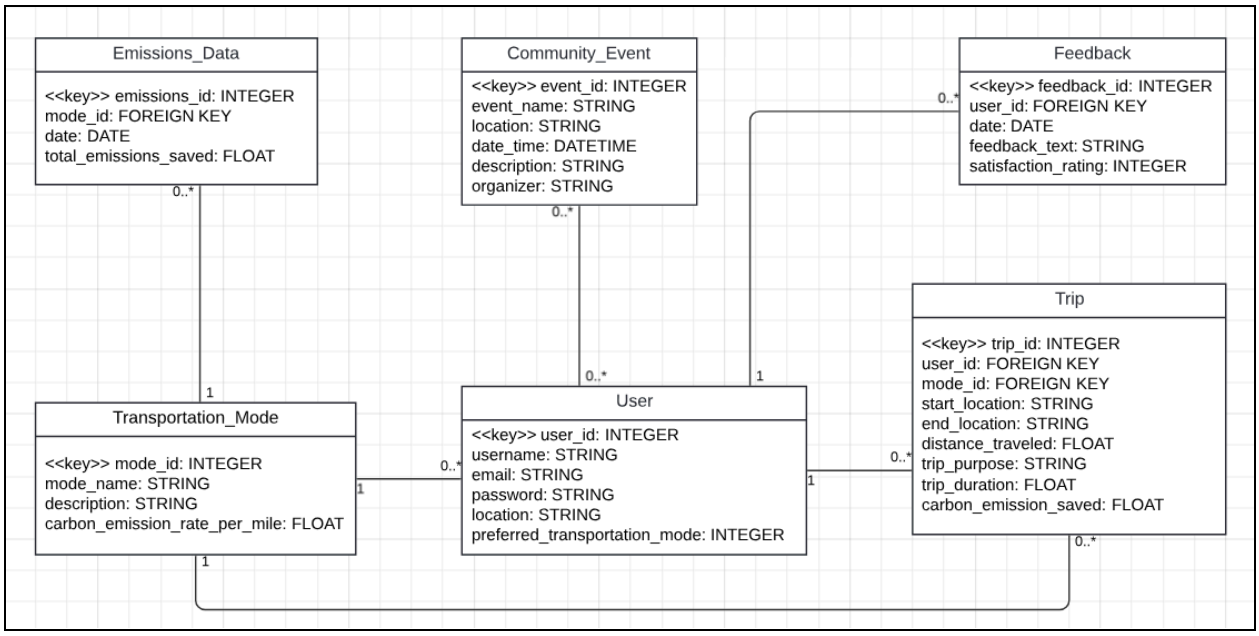
Possible Actions

Denoted by green highlights in Rules listed above.

- Register
- Log
- Calculate
- Aggregate
- Provide
- Organize
- Participate
- Track
- Manage
- Generate
- Illustrate
- Prioritize
- Encrypt
- Support

III. UML Class Diagram

Resued from Projects 1 & 2.



URL:
https://lucid.app/lucidchart/f6c4b4c6-b47b-4421-a71b-3dc1d8d982e2/edit?viewport_loc=-1055%2C53%2C2687%2C1243%2CHWEp-vi-RSFO&invitationId=inv_c76ad321-bfca-4417-9cad-678299fe2a7c

Explanation of Multiplicities

1. User to Transportation_Mode
   - **Type**: Many-to-One
   - **Multiplicity**: Many Users (0..*) to One Transportation_Mode (1)
   - **Description**: Each user may have one preferred transportation mode, but each transportation mode can be preferred by many users.

2. User to Trip
   - **Type**: One-to-Many
   - **Multiplicity**: One User (1) to Many Trips (0..*)
   - **Description**: Each user can log multiple trips, but each trip is logged by exactly one user.

3. Trip to Transportation_Mode
   - **Type**: Many-to-One
   - **Multiplicity**: Many Trips (0..*) to One Transportation_Mode (1)
   - **Description**: Each trip is associated with one transportation mode, while each transportation mode can be used for many trips.

4. User to Feedback
   - **Type**: One-to-Many

- **Multiplicity**: One User (1) to Many Feedback entries (0..*)
- **Description**: Each user can provide multiple feedback entries, but each feedback entry is provided by exactly one user.

5. Transportation_Mode to Emissions_Data
    - **Type**: One-to-Many
    - **Multiplicity**: One Transportation_Mode (1) to Many Emissions_Data entries (0..*)
    - **Description**: Each transportation mode can have multiple emissions data entries, but each entry is associated with exactly one transportation mode.

6. User to Community_Event
    - **Type**: Many-to-Many
    - **Multiplicity**: Many Users (0..*) to Many Community_Events (0..*)
    - **Description**: Users can attend multiple community events, and each community event can have multiple attendees.

IV. Functionalities

Functionalities selected to be used as an in-memory key-value storage.

Selected Functionalities

1) User Session Management
2) Live Trip Updates
3) Real-Time Transportation Mode Analytics

V. Structures

1) User Session Management
   Redis Structure used: ***Hashes***

Usage: Store session-related data (e.g., authentication tokens, user preferences, last accessed times) using Redis hashes. Hashes allow storing multiple fields within a single key, making them ideal for encapsulating all attributes of a session.

2) Live Trip Updates

Redis Structure: ***Pub/Sub***

Usage: Implements real-time notifications and updates for trips using Redis' Pub/Sub system. This facilitates a publish/subscribe model where updates are sent to all subscribed clients instantly, making it perfect for dynamic, real-time data like trip status or location updates.

3) Real-Time Transportation Mode Analytics
Redis Structure: ***Sorted Sets***

Usage: Track the number of trips per transportation mode using sorted sets. Each mode can be a member of a sorted set with the number of trips as the score, allowing for quick, ordered access to the most popular modes.

VI. Commands

1) User Session Management

Example commands:

Create or Update Session: HSET session:userID1234 token "abcd1234" lastAccessed "2023-10-05T14:25:00Z" preferences "{...}"

Retrieve session: HGETALL session:userID1234

Delete session: DEL session:userID1234

Set expiration date: EXPIRE session:userID1234 3600 # Expires after 3600 seconds (1 hour)

2) Live Trip Updates

Example commands:

Publish updates: PUBLISH tripUpdates:tripID5678 "User1234 has moved to new location x,y"

Subscribe to updates: SUBSCRIBE tripUpdates:tripID5678

Unsubscribe from updates: UNSUBSCRIBE tripUpdates:tripID5678

3) Real-Time Transportation Mode Analytics

Example commands:

Increment count for mode: ZINCRBY transportationModeCounts 1 "Bicycle"

Retrieve modes by popularity: ZREVRANGE transportationModeCounts 0 -1 WITHSCORES

Get count for specific mode: ZSCORE transportationModeCounts "Bicycle"

VII. Fulfillment of CRUD Requirements

1) Create **(C)**
   - **User Session Management**: The HSET command is used to create a new session or update an existing session with new data. This effectively handles the "Create" part of CRUD.
   - **Live Trip Updates**: The PUBLISH command, while primarily for sending messages, indirectly serves a creation role by initiating new real-time messages about trip updates.
   - **Real-Time Transportation Mode Analytics**: The ZINCRBY command, used for incrementing the count of trips for a particular transportation mode, also implicitly creates a new member in the sorted set if it does not already exist.

2) Read **(R)**
   - **User Session Management**: The HGETALL command retrieves all the data associated with a particular session, fulfilling the "Read" operation.
   - **Live Trip Updates**: The SUBSCRIBE command allows a client to listen to updates, which can be considered a form of reading, as it receives data pushed from the server.
   - **Real-Time Transportation Mode Analytics**: The ZREVRANGE and ZSCORE commands are used to read data from sorted sets, either to get a list of all transportation modes sorted by their usage or to get the count for a specific mode.

3) Update **(U)**
    - **User Session Management**: The HSET command also serves to update existing fields within a session hash, clearly covering the "Update" aspect.
    - **Live Trip Updates**: Each PUBLISH operation could conceptually be seen as updating subscribers with new state information.
    - **Real-Time Transportation Mode Analytics**: The ZINCRBY command updates the score (count) of an existing mode, directly mapping to the "Update" operation.

4) Delete **(D)**
    - **User Session Management**: The DEL command is used to completely remove a session from the store, which is a straightforward implementation of the "Delete" operation.
    - **Live Trip Updates**: Unsubscribing (UNSUBSCRIBE) doesn't delete data per se but does end the reception of new data. Redis doesn't typically store the data sent via Pub/Sub beyond the point of it being delivered to subscribed clients.
    - **Real-Time Transportation Mode Analytics**: Deleting specific data or resetting counters could be achieved with commands like ZREM to remove specific modes from the sorted set.
