Ethan Dsouza
Project 1
February 26, 2024
CS 3200

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.

Potential Tables

Transportation Modes

- mode_id (PK)
- mode_name
- description
- carbon_emission_rate_per_mile

Users

- user_id (PK)
- username
- email
- password
- location
- preferred_transportation_mode

Trips

- trip_id (PK)
- user_id (FK)
- mode_id (FK)
- start_location
- end_location
- distance_traveled
- trip_purpose
- trip_duration
- carbon_emission_saved

Emissions_Data

- emissions_id (PK)
- mode_id (FK)
- date
- total_emissions_saved

Feedback

- feedback_id (PK)
- user_id (FK)
- date
- feedback_text
- satisfaction_rating

Community Events

- event_id (PK)
- 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

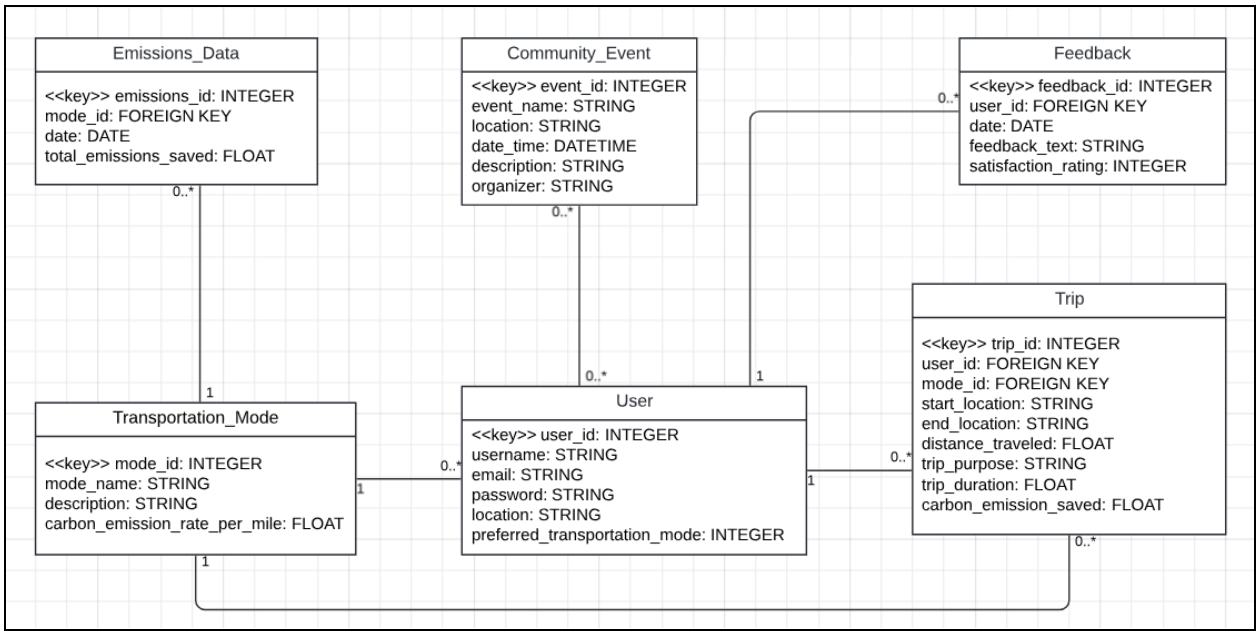
- Incentive Programs
- Rewards
- Visualizations
- Reports
- Security
- Privacy
- System

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



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

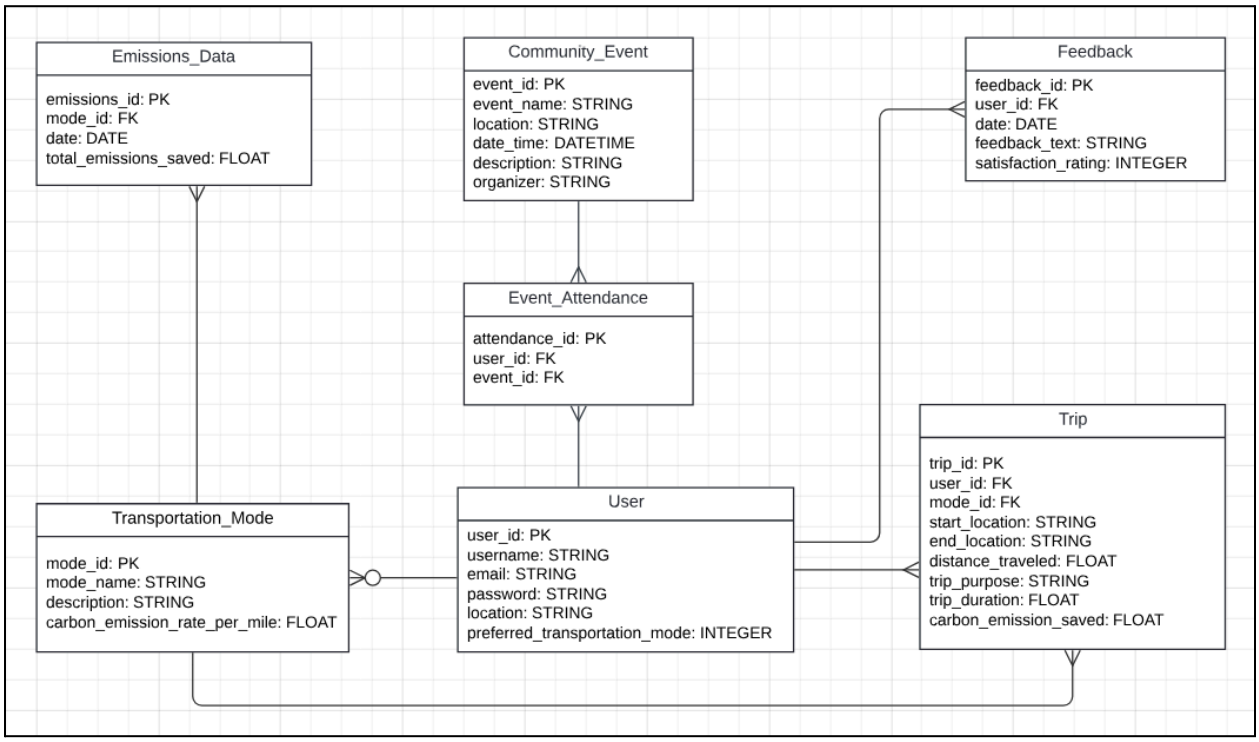
- **Multiplicity**: One User (1) to Many Feedback entries (0..*)
- **Description**: Each user can provide multiple feedback entries, but each feedback entry is provided by exactly one user.

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. ER Diagram (Logical Model)

Crow's Foot notation derived from the UML class diagram depicted above.



URL:
https://lucid.app/lucidchart/b164c01e-8368-4d0f-83b9-ef3cde5ce34f/edit?viewport_loc=-2499%2C-1210%2C4668%2C2159%2C0_0&invitationId=inv_f1d1878c-934a-4844-b132-bd91533cabc5

V. Relational Schema (BCNF)

Relational Schema

Transportation_Mode

- **Attributes:** mode_id (PK), mode_name, description, carbon_emission_rate_per_mile
- **Functional Dependencies:** mode_id → mode_name, description, carbon_emission_rate_per_mile
- **Explanation:** The primary key mode_id is the determinant in all functional dependencies, making it a superkey. There are no partial dependencies or transitive dependencies. Hence, it's in BCNF.

User

- **Attributes:** user_id (PK), username, email, password, location, preferred_transportation_mode (FK)
- **Functional Dependencies:** user_id → username, email, password, location, preferred_transportation_mode
- **Explanation:** The primary key user_id is the determinant for all other attributes, meaning it's a superkey, satisfying the BCNF condition.

Trips

- **Attributes:** trip_id (PK), user_id (FK), mode_id (FK), start_location, end_location, distance_traveled, trip_purpose, trip_duration, carbon_emission_saved
- **Functional Dependencies:** trip_id → user_id, mode_id, start_location, end_location, distance_traveled, trip_purpose, trip_duration, carbon_emission_saved
- **Explanation:** The primary key trip_id is a superkey as it uniquely determines every other attribute in the table. This table is in BCNF.

Emissions_Data

- **Attributes:** emissions_id (PK), mode_id (FK), date, total_emissions_saved
- **Functional Dependencies:** emissions_id → mode_id, date, total_emissions_saved
- **Explanation:** The primary key emissions_id determines all other attributes, and there are no dependencies on non-superkeys, so this table is in BCNF.

Feedback

- **Attributes:** feedback_id (PK), user_id (FK), date, feedback_text, satisfaction_rating
- **Functional Dependencies:** feedback_id → user_id, date, feedback_text, satisfaction_rating
- **Explanation:** The primary key feedback_id is the determinant for all attributes in this table and is a superkey, making the table BCNF compliant.

Community_Event

- **Attributes:** event_id (PK), event_name, location, date_time, description, organizer
- **Functional Dependencies:** event_id → event_name, location, date_time, description, organizer
- **Explanation:** The primary key event_id determines all other attributes and no non-superkey dependencies, this table is also in BCNF.

Conclusion

A relational schema is in BCNF if, for every one of its non-trivial functional dependencies X → Y, X is a superkey, where a superkey is a set of attributes that uniquely identifies a tuple in a table.

Each table's primary key acts as a superkey and is the only determinant for all functional dependencies within its table, meeting the criteria for BCNF. There are no partial dependencies (where a non-prime attribute is dependent on part of a composite key) or transitive dependencies (where non-prime attributes depend on other non-prime attributes), confirming that the schema is in BCNF.

VI. SQL Data Definitions

SQL Code Executed

Transportation Mode Table

```
CREATE TABLE Transportation_Mode (
    mode_id INTEGER PRIMARY KEY,
    mode_name TEXT  NOT NULL,
```

```
    description TEXT,
    carbon_emission_rate_per_mile NUMERICAL NOT NULL
);
```

User Table

```
CREATE TABLE User (
    user_id INTEGER PRIMARY KEY,
    username TEXT  NOT NULL UNIQUE,
    email TEXT  NOT NULL UNIQUE,
    password TEXT  NOT NULL,
    location TEXT ,
    preferred_transportation_mode INTEGER,
    FOREIGN KEY (preferred_transportation_mode) REFERENCES TransportationModes(mode_id)
);
```

Trip Table

```
CREATE TABLE Trip (
    trip_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    mode_id INTEGER NOT NULL,
    start_location TEXT  NOT NULL,
    end_location TEXT  NOT NULL,
    distance_traveled NUMERICAL NOT NULL,
    trip_purpose TEXT ,
    trip_duration INTEGER NOT NULL,
    carbon_emission_saved NUMERICAL NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (mode_id) REFERENCES TransportationModes(mode_id)
```

);

Emissions Data Table

```
CREATE TABLE Emissions_Data (
    emissions_id INTEGER PRIMARY KEY,
    mode_id INTEGER NOT NULL,
    date TEXT NOT NULL,
    total_emissions_saved NUMERICAL NOT NULL,
    FOREIGN KEY (mode_id) REFERENCES TransportationModes(mode_id)
);
```

Feedback Table

```
CREATE TABLE Feedback (
    feedback_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    date TEXT NOT NULL,
    feedback_text TEXT,
    satisfaction_rating INTEGER CHECK (satisfaction_rating BETWEEN 1 AND 5),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

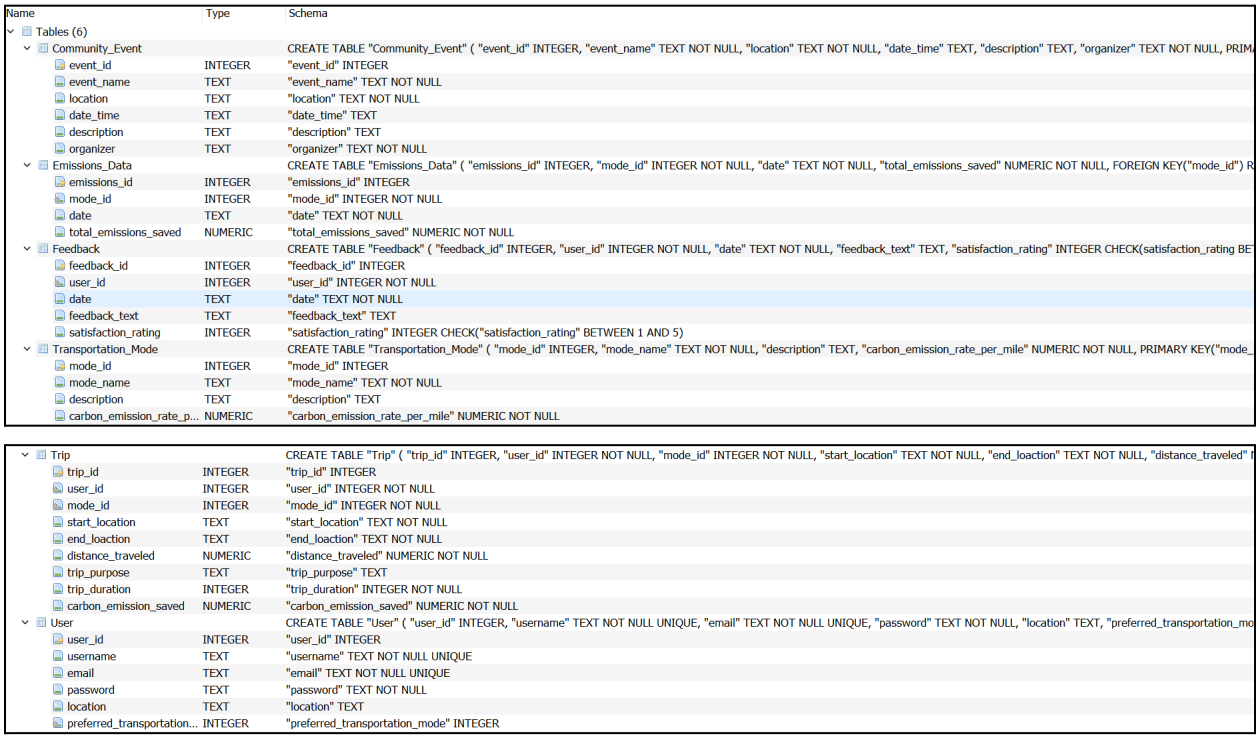
Community Events Table

```
CREATE TABLE Community_Event (
    event_id INTEGER PRIMARY KEY,
    event_name TEXT NOT NULL,
    location TEXT  NOT NULL,
    date_time TEXT NOT NULL,
    description TEXT,
```

organizer TEXT NOT NULL,

);

DB Browser Implementation



Name	Type	Schema
Tables (6)		
Community_Event		CREATE TABLE "Community_Event" ("event_id" INTEGER, "event_name" TEXT NOT NULL, "location" TEXT NOT NULL, "date_time" TEXT, "description" TEXT, "organizer" TEXT NOT NULL, PRIM
event_id	INTEGER	"event_id" INTEGER
event_name	TEXT	"event_name" TEXT NOT NULL
location	TEXT	"location" TEXT NOT NULL
date_time	TEXT	"date_time" TEXT
description	TEXT	"description" TEXT
organizer	TEXT	"organizer" TEXT NOT NULL
Emissions_Data		CREATE TABLE "Emissions_Data" ("emissions_id" INTEGER, "mode_id" INTEGER NOT NULL, "date" TEXT NOT NULL, "total_emissions_saved" NUMERIC NOT NULL, FOREIGN KEY("mode_id") R
emissions_id	INTEGER	"emissions_id" INTEGER
mode_id	INTEGER	"mode_id" INTEGER NOT NULL
date	TEXT	"date" TEXT NOT NULL
total_emissions_saved	NUMERIC	"total_emissions_saved" NUMERIC NOT NULL
Feedback		CREATE TABLE "Feedback" ("feedback_id" INTEGER, "user_id" INTEGER NOT NULL, "date" TEXT NOT NULL, "feedback_text" TEXT, "satisfaction_rating" INTEGER CHECK(satisfaction_rating BE
feedback_id	INTEGER	"feedback_id" INTEGER
user_id	INTEGER	"user_id" INTEGER NOT NULL
date	TEXT	"date" TEXT NOT NULL
feedback_text	TEXT	"feedback_text" TEXT
satisfaction_rating	INTEGER	"satisfaction_rating" INTEGER CHECK("satisfaction_rating" BETWEEN 1 AND 5)
Transportation_Mode		CREATE TABLE "Transportation_Mode" ("mode_id" INTEGER, "mode_name" TEXT NOT NULL, "description" TEXT, "carbon_emission_rate_per_mile" NUMERIC NOT NULL, PRIMARY KEY("mode_
mode_id	INTEGER	"mode_id" INTEGER
mode_name	TEXT	"mode_name" TEXT NOT NULL
description	TEXT	"description" TEXT
carbon_emission_rate_p...	NUMERIC	"carbon_emission_rate_per_mile" NUMERIC NOT NULL

Trip		CREATE TABLE "Trip" ("trip_id" INTEGER, "user_id" INTEGER NOT NULL, "mode_id" INTEGER NOT NULL, "start_location" TEXT NOT NULL, "end_loaction" TEXT NOT NULL, "distance_traveled"
trip_id	INTEGER	"trip_id" INTEGER
user_id	INTEGER	"user_id" INTEGER NOT NULL
mode_id	INTEGER	"mode_id" INTEGER NOT NULL
start_location	TEXT	"start_location" TEXT NOT NULL
end_loaction	TEXT	"end_loaction" TEXT NOT NULL
distance_traveled	NUMERIC	"distance_traveled" NUMERIC NOT NULL
trip_purpose	TEXT	"trip_purpose" TEXT
trip_duration	INTEGER	"trip_duration" INTEGER NOT NULL
carbon_emission_saved	NUMERIC	"carbon_emission_saved" NUMERIC NOT NULL
User		CREATE TABLE "User" ("user_id" INTEGER, "username" TEXT NOT NULL UNIQUE, "email" TEXT NOT NULL UNIQUE, "password" TEXT NOT NULL, "location" TEXT, "preferred_transportation_mo
user_id	INTEGER	"user_id" INTEGER
username	TEXT	"username" TEXT NOT NULL UNIQUE
email	TEXT	"email" TEXT NOT NULL UNIQUE
password	TEXT	"password" TEXT NOT NULL
location	TEXT	"location" TEXT
preferred_transportation...	INTEGER	"preferred_transportation_mode" INTEGER

VII. Generated Data Insertion

Tables populated with test data accessible via database submitted alongside this document.

VIII. SQL Queries

Query 1: Join of at Least Three Tables

This query finds total distance traveled and carbon emissions saved by each user, along with the user's preferred mode of transportation, by joining the User, Trip, and Transportation_Mode tables.

SELECT

```
    User.username,
    Transportation_Mode.mode_name,
    SUM(Trip.distance_traveled) AS total_distance,
    SUM(Trip.carbon_emission_saved) AS total_emissions_saved
FROM
    User
JOIN
    Trip ON User.user_id = Trip.user_id
JOIN
    Transportation_Mode ON User.preferred_transportation_mode = Transportation_Mode.mode_id
GROUP BY
    User.username, Transportation_Mode.mode_name;
```

Query 2: Subquery

This query selects all users who have saved more carbon emissions than the average across all trips.

```
SELECT
    username,
    location
FROM
    User
WHERE
    user_id IN (
        SELECT
            user_id
        FROM
            Trip
        GROUP BY
            user_id
        HAVING
            SUM(carbon_emission_saved) > (
                SELECT AVG(carbon_emission_saved)
                FROM Trip
            )
    );
```

Query 3: GROUP BY with HAVING Clause

This query identifies transportation modes that have saved more than an average of 1000 units of carbon emissions per trip.

```
SELECT
    mode_name,
    AVG(carbon_emission_saved) AS average_emissions_saved
FROM
    Trip
```

```
JOIN
   Transportation_Mode ON Trip.mode_id = Transportation_Mode.mode_id
GROUP BY
   mode_name
HAVING
   AVG(carbon_emission_saved) > 1000;
```

Query 4: Complex Search Criterion

This query finds trips longer than 5 miles that either saved more than 500 units of carbon emissions or took longer than 30 minutes.

```
SELECT
   trip_id,
   distance_traveled,
   carbon_emission_saved,
   trip_duration
FROM
   Trip
WHERE
   distance_traveled > 5
   AND (carbon_emission_saved > 500 OR trip_duration > 30);
```

Query 5: Advanced Query Mechanism - SELECT CASE/WHEN

This query categorizes users based on their preferred transportation mode into 'Eco-Friendly', 'Moderate', or 'Other'.

```
SELECT
   username,
   CASE
      WHEN preferred_transportation_mode IN (SELECT mode_id FROM Transportation_Mode WHERE
mode_name IN ('Cycling', 'Walking')) THEN 'Eco-Friendly'
      WHEN preferred_transportation_mode IN (SELECT mode_id FROM Transportation_Mode WHERE
mode_name = 'Public Transit') THEN 'Moderate'
      ELSE 'Other'
   END AS eco_category
FROM
   User;
```
