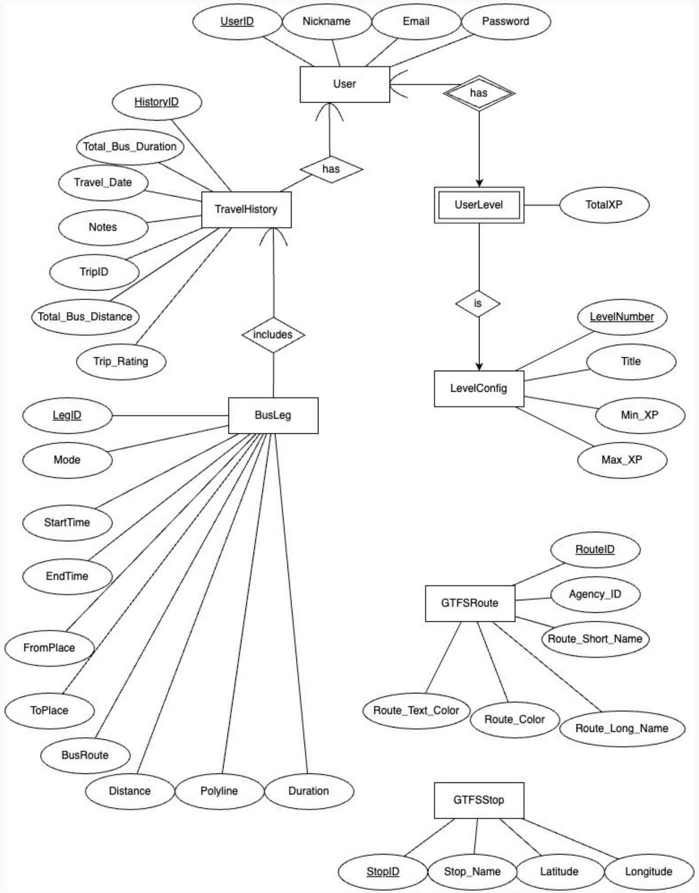
Stage 2: Conceptual and Logical Database Design

Due Mar 4, 2025

Team 099- BigBallers
Rahul Reddy
Jay Malavia
Allen Kaile Yuan
Yu Fu

ER Diagram (Revised Version)



Assumptions for each entity and relationship

User:
Each user has a unique ID with basic account info (nickname, email, password hash) and timestamps for creation/update.

TravelHistory:
Logs each trip taken by a user, capturing the trip's ID, date, and aggregated bus metrics (duration and distance), plus optional notes and ratings.

BusLeg:
Represents individual bus segments within a trip, detailing start/end times, duration, distance, departure/arrival locations, bus route, and an encoded polyline for mapping.

UserLevel:
Tracks a user's cumulative XP, their current level and title; each user has one level record that can be recalculated by joining with level configuration.

Level_Config:
Defines each level's number and title based on their XP thresholds, serving as a lookup table for the leveling system.

GTFS_Route & GTFS_Stop:
Static reference data from the GTFS feed used to display transit route and stop information.

Relationships & Cardinality:

User (1) ↔ (0..*) TravelHistory: One user can have many travel records.

TravelHistory (1) ↔ (0..*) BusLeg: Each trip may consist of several bus legs.

User (1) ↔ (1) UserLevel: Each user has exactly one level record.

UserLevel (0..*) ↔ (1) Level_Config: A user's current level corresponds to a level configuration entry (many users can share the same level).

GTFS_Route & GTFS_Stop: Serve as static lookup tables for transit information.

Normalization (3NF) - Revised

- We noticed that the current database schema violated **3NF**, due to transitive dependencies in the **UserLevel** table.
- Calculating information for relation **UserLevel** having attributes: **User_ID**, **TotalXP**, **Current_Level**, **Title**, **Updated_At.**
- Given input functional dependencies:
  - **User_ID** → **TotalXP**, **Current_Level**, **Current_Level**, **Updated_At**;
  - **TotalXP** → **Current_Level**, **Title**;
- We can see that **TotalXP** is not the primary key for the table (**PK** is **User_ID**), but is still able to uniquely identify **Current_Level** and **Title**.
- To normalize this, we split the relation **UserLevel** into these new relations.
  - **User_Level** (**TotalXP**, **User_ID**, **Updated_At**) having FD(s): **User_ID** → **TotalXP**; **User_ID** → **Updated_At**.
  - **Level_Tracking** (**TotalXP**, **Current_Level**, **Title**) having FD(s): **TotalXP** → **Current_Level**; **TotalXP** → **Title**.
- Now, the relations **User_Level** and **Level_Tracking** are normalized.
- However, since **Level_Tracking** is very similar in purpose to **Level_Config**, we decided to drop it from our schema and instead add the table **User_Level_Progress** (to link **User_Level** with **Level_Config**).
- Normalization (3NF) proof for all tables is shown in the next slide.

Normalization - 3NF

Tables:

- **User**
  - **FDs:** User_ID (PK) -> Nickname, Email, Password_Hash, Created_At, Updated_At
  - Since User_ID is the **primary key**, and this is the only **FD**, **User** table is in **3NF.**
- **GTFS_Route**
  - **FDs:** Route_ID (PK) -> Agency_ID, Route_Short_Name, Route_Long_Name, Route_Color, Route_Text_Color
  - Since Route_ID is the **primary key**, and this is the only **FD**, **GTFS_Route** table is in **3NF**.
- **GTFS_Stop**
  - **FDs:** Stop_ID (PK) -> Stop_Name, Latitude, Longitude
  - Since Stop_ID is the **primary key**, and this is the only **FD**, **GTFS_Stop** table is in **3NF**.
- **Travel_History**
  - **FDs:** History_ID (PK) -> User_ID, Trip_ID, Travel_Date, Total_Bus_Duration, Total_Bus_Distance, Notes, Trip_Rating, Created_At, Updated_At
  - Since History_ID is the **primary key**, and this is the only **FD**, **Travel_History** table is in 3NF.
- **Bus_Leg**
  - **FDs:** Leg_ID (PK) -> History_ID, Mode, StartTime, EndTime, Duration, Distance, FromPlace, ToPlace, BusRoute, Polyline, Created_At, Updated_At
  - Since Leg_ID is the **primary key**, and this is the only **FD**, **Bus_Leg** table is in 3NF.
- **User_Level**
  - **FDs:** User_ID (PK) -> Total_XP, Updated_At
  - Since User_ID is the **primary key**, and this is the only **FD**, **User_Level** table is in **3NF.** *(Prior to normalization, TotalXP -> Current_Level, Title)*
- **User_Level_Progress**
  - **FDs:** User_ID (PK) -> Level_Number, Updated_At
  - Since User_ID is the **primary key**, and this is the only **FD**, **User_Level_Progress** table is in **3NF**.
- **Level_Config**
  - **FDs:** Level_Number -> Title, Min_XP, Max_XP
  - Since Level_Number is the **primary key**, and this is the only **FD**, **Level_Config** table is in **3NF**.

Now, all tables are normalized i.e. adhere to 3NF.

FD: Functional Dependency & PK: Primary Key

Relational Schema

```
User(
  User_ID: INT [PK],
  Nickname: VARCHAR(50),
  Email: VARCHAR(100) UNIQUE,
  Password_Hash: VARCHAR(255),
  Created_At: TIMESTAMP,
  Updated_At: TIMESTAMP
)

TravelHistory(
  History_ID: INT [PK],
  User_ID: INT [FK to User.User_ID],
  Trip_ID: VARCHAR(50) UNIQUE,
  Travel_Date: DATE,
  Total_Bus_Duration: INT,
  Total_Bus_Distance: DECIMAL(10,2),
  Notes: TEXT,
  Trip_Rating: DECIMAL(3,1),
  Created_At: TIMESTAMP,
  Updated_At: TIMESTAMP
)

BusLeg(
  Leg_ID: INT [PK],
  History_ID: INT [FK to TravelHistory.History_ID],
  Mode: VARCHAR(20),
  StartTime: TIMESTAMP,
  EndTime: TIMESTAMP,
  Duration: INT,
  Distance: DECIMAL(10,2),
  FromPlace: VARCHAR(100),
  ToPlace: VARCHAR(100),
  BusRoute: VARCHAR(20),
  Polyline: TEXT,
  Created_At: TIMESTAMP,
  Updated_At: TIMESTAMP
)

UserLevel(
  User_ID: INT [PK, FK to User.User_ID],
  Total_XP: INT,
  Updated_At: TIMESTAMP
)

UserLevelProgress(
  User_ID: INT [PK, FK to UserLevel.User_ID],
  Level_Number: INT [FK to Level.Level_Number],
  Updated_At: TIMESTAMP
)

Level_Config(
  Level_Number: INT [PK],
  Title: VARCHAR(50),
  Min_XP: INT,
  Max_XP: INT,
  CHECK(Min_XP < Max_XP)
)

GTFS_Route(
  Route_ID: VARCHAR(20) [PK],
  Agency_ID: VARCHAR(20),
  Route_Short_Name: VARCHAR(20),
  Route_Long_Name: VARCHAR(100),
  Route_Color: VARCHAR(10),
  Route_Text_Color: VARCHAR(10)
)

GTFS_Stop(
  Stop_ID: VARCHAR(20) [PK],
  Stop_Name: VARCHAR(100),
  Latitude: DECIMAL(10,6),
  Longitude: DECIMAL(10,6)
)
```

Relational Schema Explanation

- Designed a relational schema to manage users, travel history, and transit data.
- Implemented foreign keys to link users with their trips and trip segments.
- Integrated GTFS transit data for routes and stops to enhance trip tracking.
- Used timestamps and ratings to analyze travel patterns and user experiences.