Class Diagram

Below is a class diagram description capturing the main data models and relationships for the campus-only peer-to-peer ride sharing backend, as specified in the SRS. This diagram reflects a layered Go backend using GORM for MySQL and gRPC for all APIs.

Core Entities and Relationships

User

- **Fields**:
  - ID (string, PK
  - Name (string)
  - Email (string, unique)
  - PhotoURL (string)
  - Geohash (string)
  - LastSeen (timestamp)
- **Relationships**:
  - Can create multiple RideOffers and RideRequests.
  - Can send multiple ChatMessages.
  - Can submit and receive Reviews.

RideOffer

- **Fields**:
  - ID (string, PK
  - DriverID (string, FK → User.ID)
  - FromGeo (string, geohash)
  - ToGeo (string, geohash)
  - Fare (float)
  - Time (timestamp)
  - Seats (int)

- Status (string: active, matched, completed)

- **Relationships**:
  - Associated with one User Driver .
  - Can be matched with RideRequests.
  - Linked to Reviews.

RideRequest

- **Fields**:
  - ID (string, PK
  - UserID (string, FK → User.ID)
  - FromGeo (string, geohash)
  - ToGeo (string, geohash)
  - Fare (float)
  - Time (timestamp)
  - Seats (int)
  - Status (string)
- **Relationships**:
  - Associated with one User Rider .
  - Can be matched to RideOffer.

Match

- **Fields**:
  - ID (string, PK
  - RiderID (string, FK → User.ID)
  - DriverID (string, FK → User.ID)
  - RideID (string, FK → RideOffer.ID)
  - Status (string: requested, accepted, rejected, completed)
  - CreatedAt (timestamp)
- **Relationships**:
  - One Match per pair of RideOffer and RideRequest.

ChatMessage

- **Fields**:
  - ID (string, PK
  - RideID (string, FK → RideOffer.ID)

- SenderID (string, FK → User.ID)
- Content (string)
- Timestamp (datetime)

- **Relationships**:
  - Each message belongs to a RideOffer and Sender User .

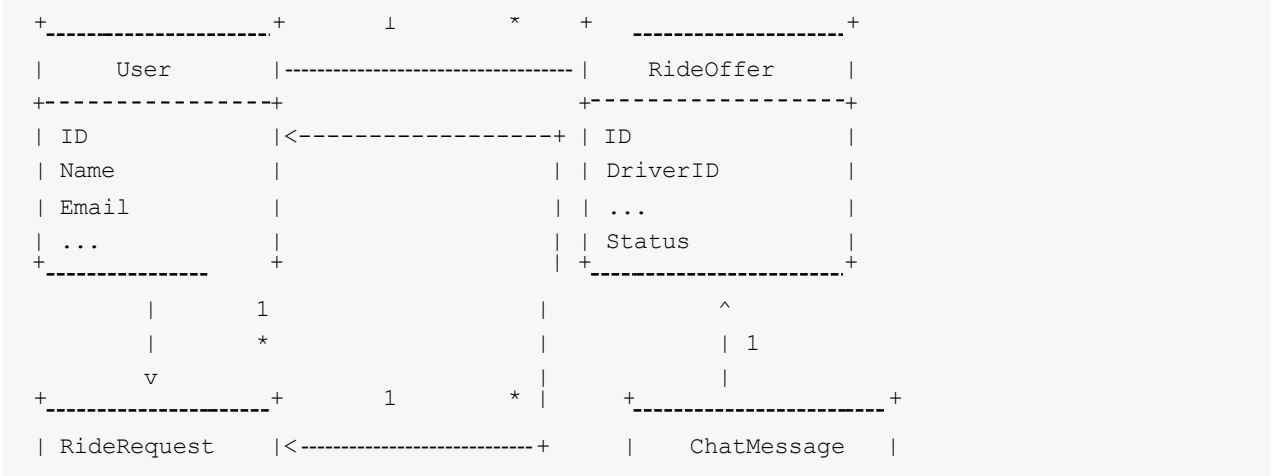
Review

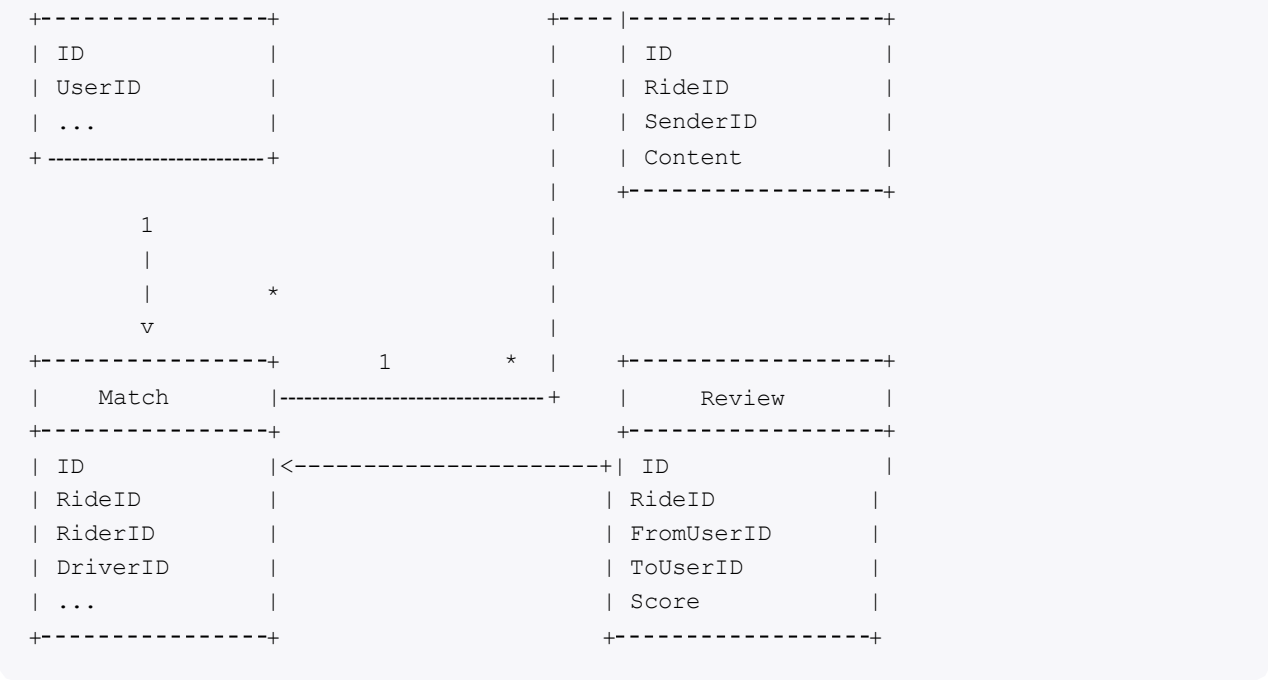
- **Fields**:
  - ID (string, PK
  - RideID (string, FK → RideOffer.ID)
  - FromUserID (string, FK → User.ID)
  - ToUserID (string, FK → User.ID)
  - Score (int)
  - Comment (string)
  - CreatedAt (timestamp)
- **Relationships**:
  - Associated with both the reviewer and reviewee User .

Summary of Services

- **Auth/UserService**: Handles User registration, login, JWT, profile management.
- **RideService**: Handles RideOffer/RideRequest CRUD, geo-search via geohash.
- **MatchService**: Manages ride matching, state transitions.
- **ChatService**: Manages ride-specific chat threads.
- **LocationService**: Streams real-time location (lat/lon, geohash).
- **ReviewService**: Handles rating and feedback after rides.

Class Diagram Structure UML Notation)





Notes

- **User**, **RideOffer**, **RideRequest**, **Match**, **ChatMessage**, and **Review** correspond directly to GORM structs, gRPC messages, and database tables/collections.
- Geohash fields are used for spatial queries (matching by proximity).
- MongoDB/Redis used for non-relational data (chat, live sessions/location), SQL for persistent transactional data.
- The **layered architecture** wraps these models in Handler (gRPC/REST/API , Service (business logic), Repository DB CRUD , and Dependency Injection layers, ensuring modularity and testability.