Uber Report

Functionalities and implementation

Driver and Rider Registration

1 Driver and Rider Registration

Implementation:

- **gRPC Methods**: RegisterDriver and RegisterRider
- **Description**:
    - When a driver or rider registers, their details (name, client port, and server port) are stored in the database.
    - A unique driver_id or rider_id is assigned to the new entity, and the system returns a success message along with this ID to the client.
    - Drivers are marked as available upon registration, while riders are simply stored.
    - The server port is used to keep track of which server a driver is connected to, enabling communication across multiple servers.

2 Ride Request and Driver Notification

Implementation:

- **gRPC Methods**: RequestRide, DriverNotificationService, ServerNotificationService
- **Description**:
    - When a rider makes a ride request, the system fetches all available drivers from the MySQL database.
    - A notification is generated for each available driver, and a ride entry is created with the status 'requested'.
    - The system attempts to notify drivers one by one using the send_driver_notification function. If a driver is on the current server, a notification is added to their notification queue. If they are on a different server, the notification is forwarded to the appropriate server using the ServerNotificationService method.
    - A driver has 10 seconds to accept or reject the ride before the system moves on to notify the next available driver.

3 Ride Acceptance/Rejection

Implementation:

- **gRPC Methods**: AcceptRide, RejectRide
- **Description**:
  - When a driver accepts or rejects a ride, the ride's status is updated in the database.
  - If accepted, the ride is assigned to the driver, and the driver is marked as unavailable during the ride.
  - In case of rejection, the system moves on to the next available driver, and the ride remains unassigned until a driver accepts it.

4 Ride Completion

Implementation:

- **gRPC Method**: CompleteRide
- **Description**:
  - Upon ride completion, the driver marks the ride as Completed. The ride status is updated in the database, and the driver is made available for new ride requests.

5 Driver Notifications

Implementation:

- **gRPC Methods**: DriverNotificationService
- **Description**:
  - These are streaming gRPC services that continuously provide notifications to connected drivers.
  - When a notification is generated (e.g., a ride request for a driver), it is queued in the appropriate notification list.
  - The service keeps streaming notifications to the client until the client disconnects or all notifications are processed.

6 Ride Status Check

Implementation:

- **gRPC Method**: GetRideStatus
- **Description**:
  - This method allows a rider or driver to check the status of their most recent ride.
  - The method fetches the ride details from the MySQL database (using a join with the drivers table to retrieve driver information) and returns it to the client.

    - If no rides are found for the rider, an empty response is returned.

7 Multi-server Notification Forwarding

Implementation:

- **gRPC Methods**: ServerNotificationService, send_driver_notification
- **Description**:
    - In a multi-server setup, drivers may be connected to different servers. To ensure that drivers receive ride requests regardless of their server connection, the send_driver_notification function checks whether the driver is on the current server. If not, it forwards the notification to the server where the driver is connected.
    - The ServerNotificationService method is responsible for processing these forwarded notifications and queuing them in the target driver's notification list on the receiving server.

Logging

Implementation:

- **Class**: LoggingInterceptor
- **Description**:
    - A logging interceptor is implemented to intercept all gRPC method calls and log the timestamp, client role, and method being called.
    - This helps in monitoring the system's usage and understanding the flow of operations across the system.