CS 5200 - Database Management Systems
City Ride
Group Name: MohammadiZWangZh
Mohammadi Zahra, Wang Zhen

1. **README**

Overview
Taxi Booking Application is a Java-based application designed to facilitate the management of taxi orders for both drivers and passengers. It enables drivers to manage their profiles and process rides, while passengers can manage their profiles and book rides.

Features
- **Driver Management**: Drivers can register, log in, view and update their profiles, and manage ride orders.
- **Passenger Management**: Passengers can register, log in, view and update their profiles, and create ride orders.
- **Order Management**: Creation, viewing, updating, and deletion of ride orders.
- **Route Management**: View available routes and distances.
- **Car Management**: Drivers can add, view, and update their cars.
- **Order Matching**: Matches available cars with passenger orders based on location and preferences.

Structure
The project is organized as follows:
- `cli/CLI.java`: Command Line Interface for user interaction.
- `dao/`: Data Access Objects for database interactions.
- `DBConnector.java`: Manages the database connection.
- `model/`: Contains entity classes representing data.
- `Main.java`: The main entry point of the application.

Getting Started
To run the Taxi Booking Application, follow these steps:

1. Ensure Java is installed on your system.
2. Clone the repository to your local machine. github
3. Navigate to the project directory and compile the Java files.
4. Run `cityride_dump.sql` in mysql.
5. Go to `DBConnector.java` and change **DATABASE_PASSWORD** to your own mysql password. (If your mysql username is not “root”, change **DATABASE_USER** to the correct username**.)**
6. Run `Main.java` to start the application.

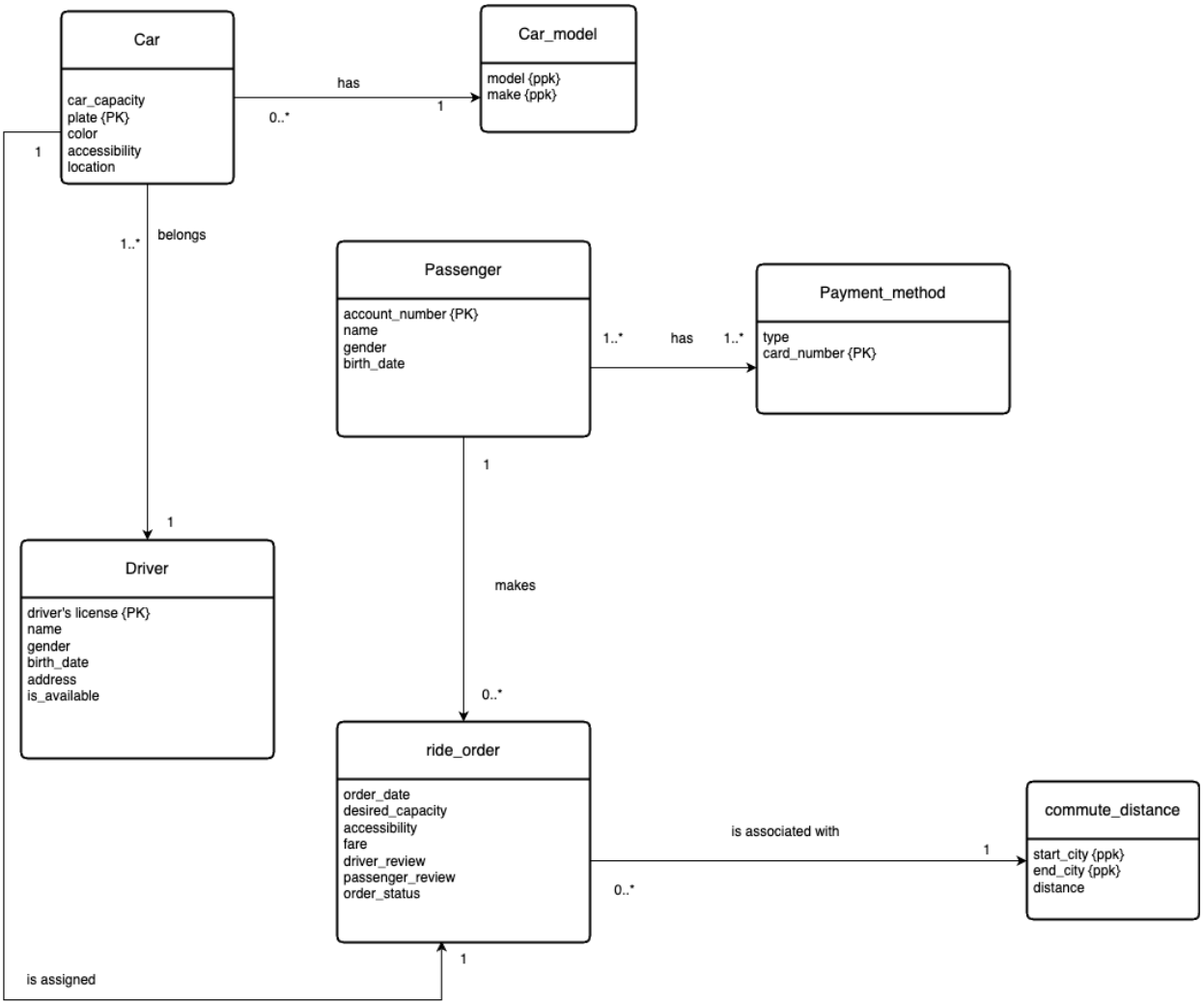
Usage
After starting the application, follow the on-screen prompts to interact with the system. You can choose to operate as a driver or a passenger, and then perform various actions based on your role.

Contributing

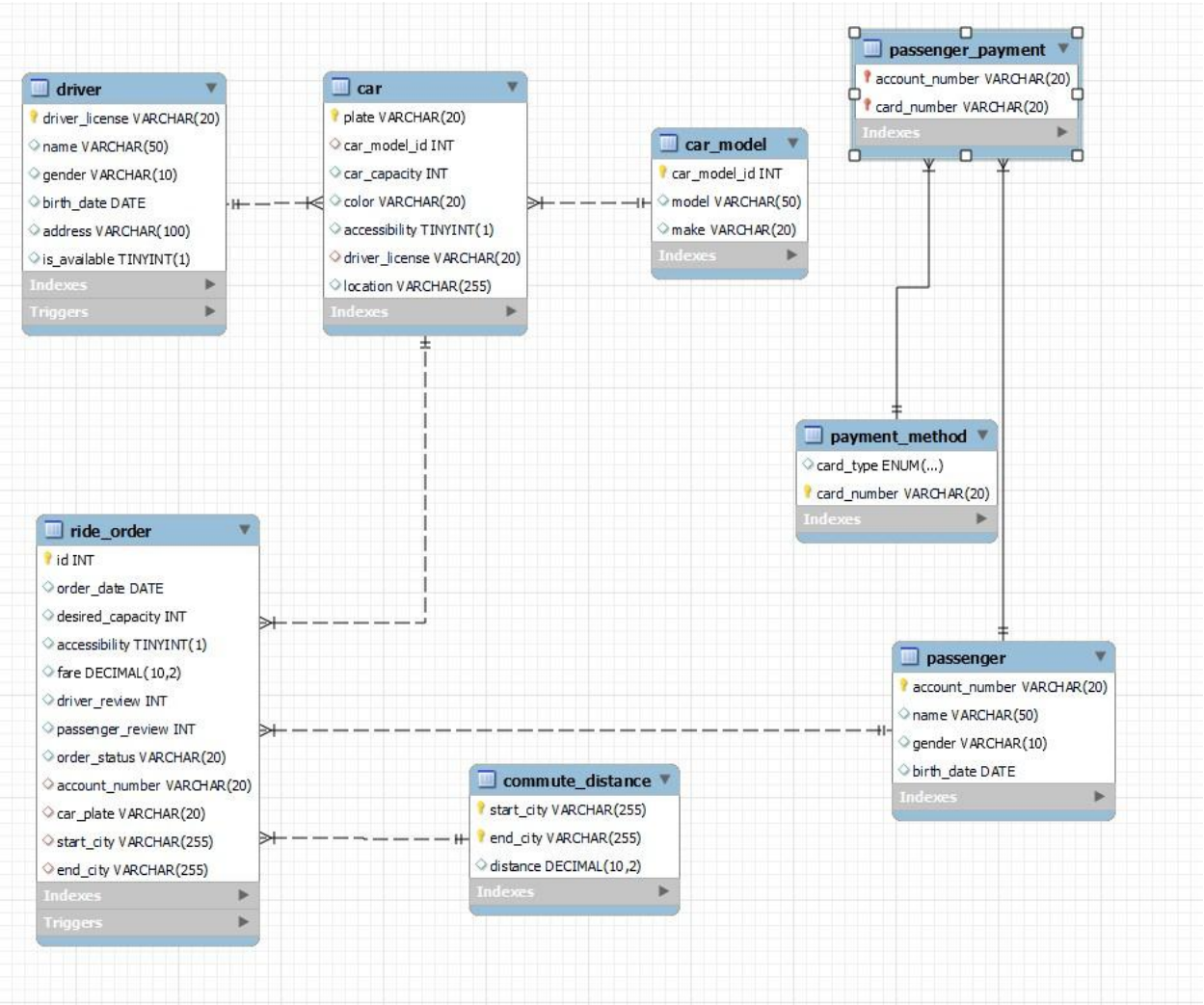
Contributions to the Taxi Booking Application are welcome. Please fork the repository and submit a pull request with your proposed changes.

2. **Technical specs:**
   Language and Frameworks: Java for application logic.
   MySQL for database.
   Libraries: MySQL Connector/J for JDBC connectivity. You can download it from MySQL Connector/J Download.

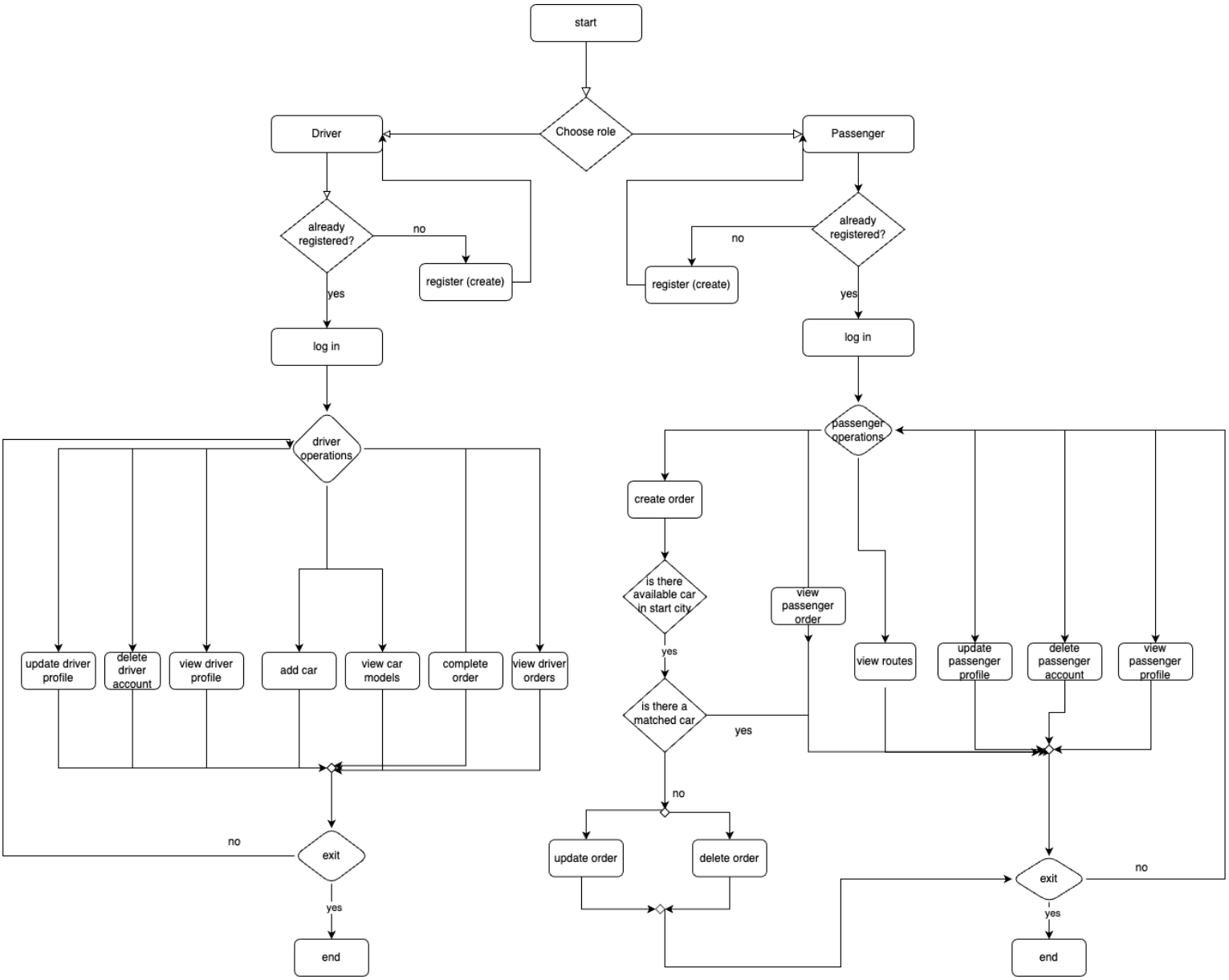
3. **Conceptual Design (UML)**



4. Logical Design



5. User Flow of the system (Activity Diagram)



a. User runs the application.
b. The application prompts the user with options using the CLI.
c. User performs actions like viewing data, adding records, or exiting.
d. The application interacts with the database based on CRUD operations provided and user input

7. Lessons Learned

Developing a database application from the ground up is a highly rewarding and enriching experience. After reviewing and discussing the feedback for our project proposal, we kept improving our UML and logical design, including changing table/attributes names easier to understand, making invoice (later changed to ride-order) and payment one table and redefining the ride-order table as a weak entity, leading to a more efficient database schema. These modifications emphasize the importance of iterative design and refinement in database development.

Throughout this project, we've picked up some useful lessons that have really helped us grow professionally.

We've got a much better grip on JDBC, which is basically how Java talks to databases. Understanding this has made connecting Java applications to databases a whole lot clearer.

Besides that, working on this project has been a hands-on way to get better at developing Java applications.
We learned about Importance of thorough planning before starting development. Time management is crucial, especially in balancing feature development and bug fixing. Also, we explored different CLI libraries for a more user-friendly interface.

Code Not Working:
Our code works as expected so we do not have any code that's not working.

8. Future Work:

Some of the functionalities that we originally planned to implement are not implemented due to time constraints but we will keep working on it to make it more user friendly and useful. We plan to extend the database schema for additional features. We will implement user authentication for personalized experiences as well as integration with an external API for real-time data. We will implement data validation for user inputs, also we will develop a nice frontend for our application.

Bonus features that we implemented

- Interesting queries:
  1.create order (This application is able to process the order created by the passenger and then sends corresponding prompts to the passenger if the order is in progress or needs to be updated or deleted (when there is no matched car for the order)
  2.complete order (driver update the status of the order) car, order, driver status will be updated as well. You will notice that after the completion of an order, the order status will become "completed", and the fare will be updated; location in the car table will be updated to the end city of the order; driver will become available in the driver table.
  3.login/register (Both drivers and passengers can only manage their profiles and orders when they have registered and logged in. We provided a simple authentication and authorization in this application)
- User friendly interface: We provided simple and clear prompts to guide the user to operate this application.

- Overly complicated translations from user operations to database operations. This application is more than a simple profile management. By introducing order functionalities in this project with reasonable database design and employing procedures and triggers, we make sure that our database stays consistent and available.
- Application supports multiple user roles: Users can choose to be either a passenger or a driver.