STLDD - Software Top Level Design Document

SG

October 3, 2018

Contents

Document history

Version	Date	Resp	Description
0.1	2018-10-02	SG	Draft version

1 Introduction

This document describes the top-level design of the system. It is a system that helps commuters car pool between cities by matching drivers and commuters based on their departure locations and destinations, as well as the time of the commute. The users will be able to add the routes they will be driving and search for routes they would like to ride with somebody.

The system is developed by group 5 of the course ETSN05 2018.

2 Reference documents

1. SRS - Software Requirements Specification, v. 1.0

2. Documentation for the system. (Documentation.pdf)

3. UML diagram for the system. (UML.png)

3 Overview

This section contains a description of all the classes used for the database and backend in the application. Every subsection is a package in the project.

3.1 Data

class User This is a basic class to represent the data of a user from the Database.

class Location This is a basic class to represent the data of a location from the Database.

class Ride This is a basic class to represent the data of a ride from the Database.

class Role This is a basic class to represent the data of a role from the Database.

class Session This is a basic class to represent the data of a session from the Database.

class Credentials This class handles the credentials of a user. Which is used to authorize the user to do certain actions. Example an user should only be able to have remove access to rides created by themselves.

class Date This class hold the information of a date and time. Then formats in properly and returns the data throught getters.

class UserDataAccess This DataAccess class extends the DataAccess superclass. It contains methods for all the actions that can be done to a user in the database.

class LocationDataAccess This DataAccess class extends the DataAccess superclass. It contains methods for all the actions that can be done to a location in the database.

class RideDataAccess This DataAccess class extends the DataAccess superclass. It contains methods for all the actions that can be done to a ride in the database.

3.2 Database

class CreateSchema This class contains methods for creating the database schema. It is being used at start-up to make sure there exist one. If there isn't a schema, one is created.

class DataAccessException This class is used to wrap in a SQLException and adds an ErrorType to the handled exception.

class DataAccess This class is the superclass for all DataAccess classes. It provides access to the H2 database and contains helper methods for common JDBC use cases. Also contains a method for setting up the database schema.

class ErrorType This is used to communicate to client about usage errors without exposing underlying implementation details.

class Mapper This is a helper class which is used to make working with JDBC's checked exception easier.

class MapMapper This class is used for debugging SQL queries without having to write a fixed Mapper.

3.3 Rest

class UserResources This class is a link between the user using the Front-End trying to communicate with the Database. It's a rest API and this class specifically handles actions related to a user.

class LocationResources This class is a link between the user using the Front-End trying to communicate with the Database. It's a rest API and this class specifically handles actions related to a location.

class RideResources This class is a link between the user using the Front-End trying to communicate with the Database. It's a rest API and this class specifically handles actions related to a ride.

3.4 Provider

class AuthenticationFilter Adds the Session to the current HTTP request to the REST API. This is done by extracting the token in the users cookie and checking the database for the cookie.

class JsonProvider This class is a converter of all the objects in the REST API between Java Object and Json Object. It uses Gson.

class JsonExceptionMapper This converts all Exceptions to HTTP responses for the REST API. It has special handling for WebApplicationException and DataAccessException.

3.5 Ranker

class Ranker This class ranks every ride in a list of rides and return a sorted list based on those rankings. It can rank on several different criteria depending on where it's used.

4 Class diagram

The system's architecture is described by the UML diagram shown in Figure 11.

5 Database

There is one database with four tables;
    Users, Locations, Rides and PassengerRides.

5.1 ER-Diagram

The database is described by the ER-diagram in Figure 1.

5.2 ER-Relations

The following ER-relations describe the databased used by the system. Primary keys are underlined and foreign keys are written in *italics*.

Users (<u>user_id</u>, username, user_full_name, user_email, user_role)
Locations (<u>location_name</u>)
Rides (ride_id, departure_time, arrival_time, nbr_seats, *driver_id, departure_location, destination*)
RidePassengers (*<u>ride_id, passenger_id</u>*)
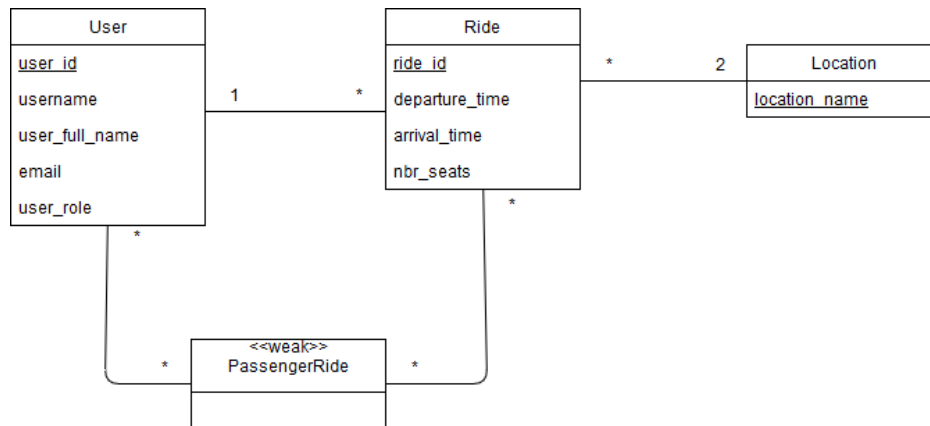
Figure 1: ER-diagram of the database

5.3 SQL-code

The following code should be used to create the tables.

```
CREATE TABLE users (
    user_id INT AUTO_INCREMENT NOT NULL,
    username VARCHAR NOT NULL UNIQUE,
    user_full_name VARCHAR NOT NULL,
    email VARCHAR NOT NULL,
    user_role VARCHAR NOT NULL,
    PRIMARY KEY (user_id));

CREATE TABLE locations (
    location_name VARCHAR NOT NULL,
    PRIMARY KEY (location_name));

CREATE TABLE rides (
    ride_id INT AUTO_INCREMENT NOT NULL,
    departure_time DATE NOT NULL,
    arrival_time DATE NOT NULL,
    nbr_seats INT NOT NULL,
    driver_id INT NOT NULL,
    departure_location VARCHAR NOT NULL,
    destination VARCHAR NOT NULL,
    PRIMARY KEY (ride_id),
    FOREIGN KEY (driver_id) REFERENCES users (user_id),
    FOREIGN KEY (departure_location) REFERENCES locations (location_name)
    FOREIGN KEY (destination) REFERENCES locations (location_name));

CREATE TABLE ridePassengers (
```

```
ride_id NOT NULL,
passenger_id NOT NULL,
PRIMARY KEY (ride_id, passenger_id),
FOREIGN KEY (ride_id) REFERENCES rides (ride_id),
FOREIGN KEY (passenger_id) REFERENCES users (user_id));
```
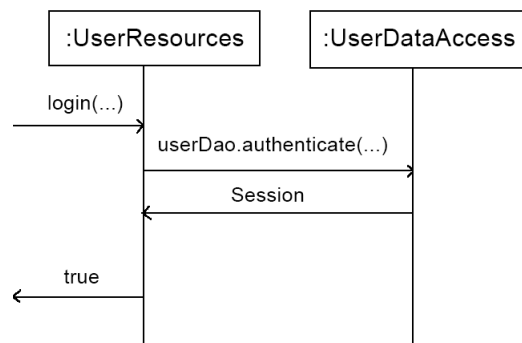
6 Sequence diagrams

Login

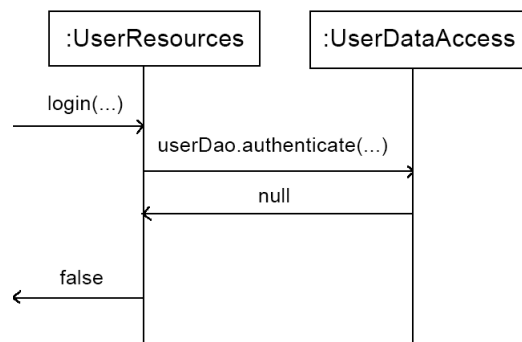Figure 2: Sequence diagram of login procedure.

Login failure

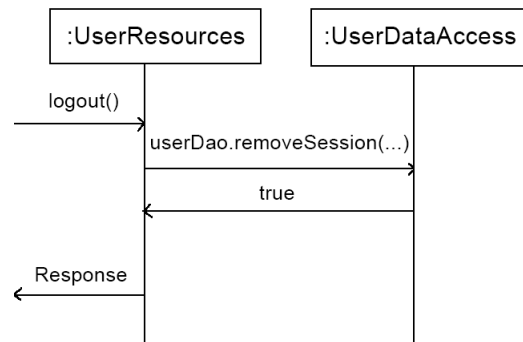Figure 3: Sequence diagram of failure of login procedure.

Logout

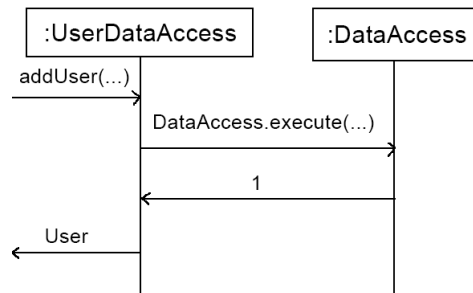Figure 4: Sequence diagram of logout procedure.

Create account

Figure 5: Sequence diagram of the procedure of creating an account.

Create account failure
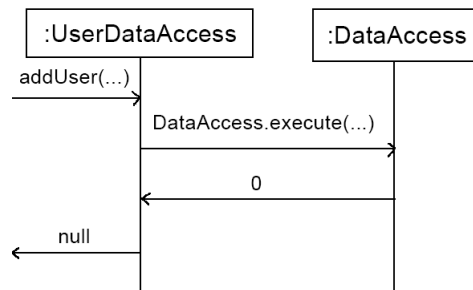
Figure 6: Sequence diagram of failing to create an account.
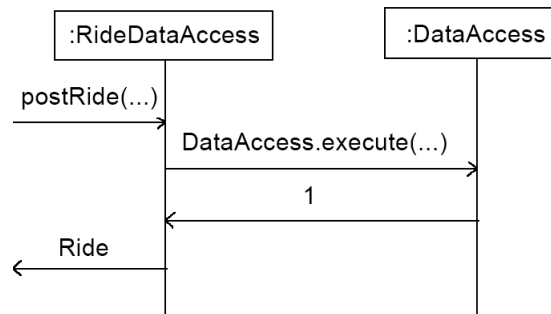
Create ride

Figure 7: Sequence diagram of creating a ride.
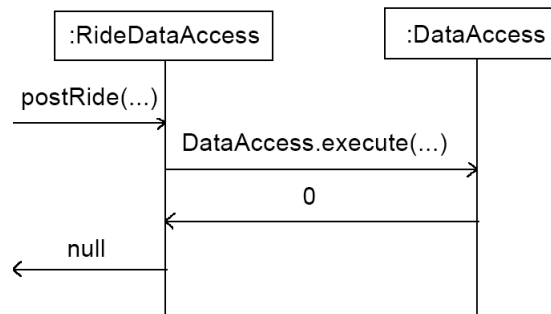
Create ride failure

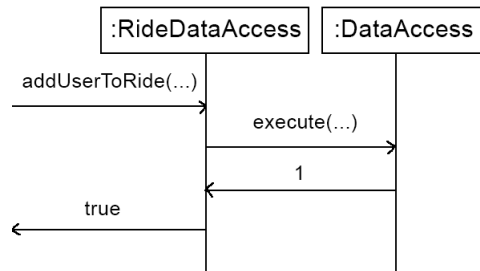Figure 8: Sequence diagram of failing to create a ride.

Join ride

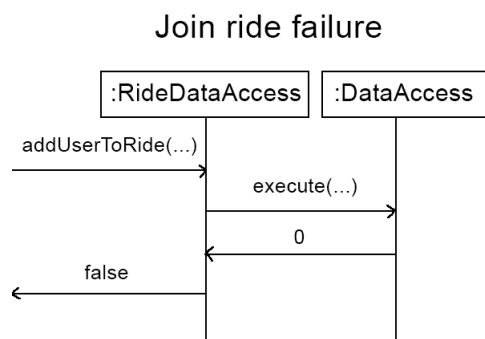Figure 9: Sequence diagram of joining a ride.

Join ride failure

Figure 10: Sequence diagram of failing to join a ride.

Figure 11: UML describing the system(source: