Design Document for Oober-Cy-Lypht

Group 2_HB_2

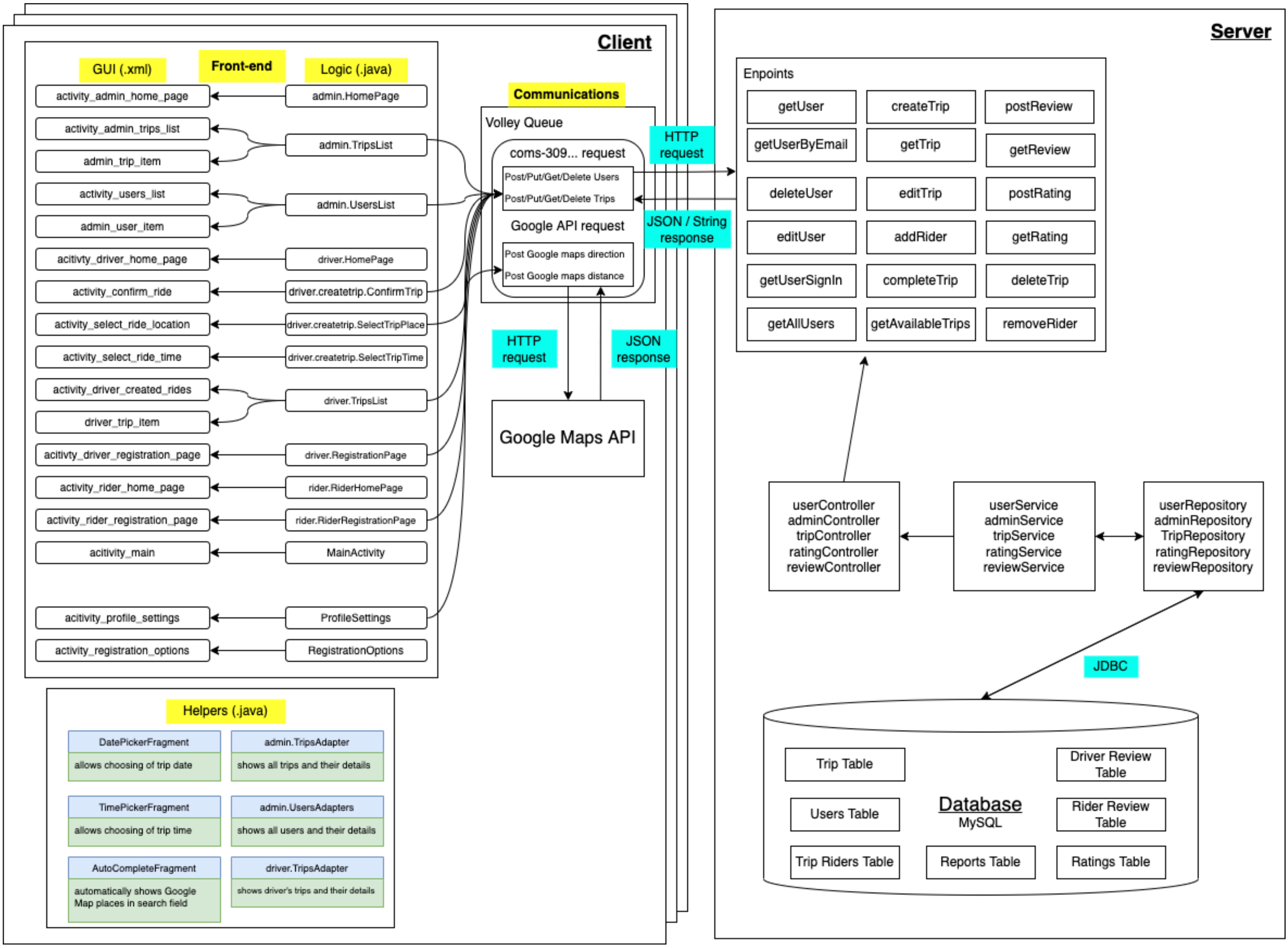
Gautham Ajith : 25% contribution

Yaaseen Basha : 25% contribution

Alex Huynh : 25% contribution

Matt Sinwell : 25% contribution

Block Diagram



Design description:

GUI:

- Each user home page (admin, driver, rider) has buttons that lead to the further activities that are limited to the respective user.
- For admin.TripsList, the corresponding activity is admin_trips_list, but the content is a ListView made of admin_trip_item. The same principle applies to driver.TripsList.
- We separate the classes mostly into three packages: admin, driver, rider. Packages that don't fall into these three categories are placed in the main src directory.

Helpers:

- Adapters: The adapter takes a JSON array and for each element, it designs an .xml file. These .xml files are injected into a main .xml file. For example, the admin.TripsAdapter receives an array of all trips, and for each trip, it creates a small fragment containing details about the trip, along with a button to edit or delete that specific trip. These fragments are defined under admin_trip_item. Finally, all of these fragments are placed into activity_admin_trips_list.
- We use Date- and Time- PickerFragments to allow the user to choose the time and date of their trip.
- We make use of Google's "Place Autocomplete" widget. This allows the user to search for predefined places when creating a trip.
- A non-functional helper includes verification of registration input

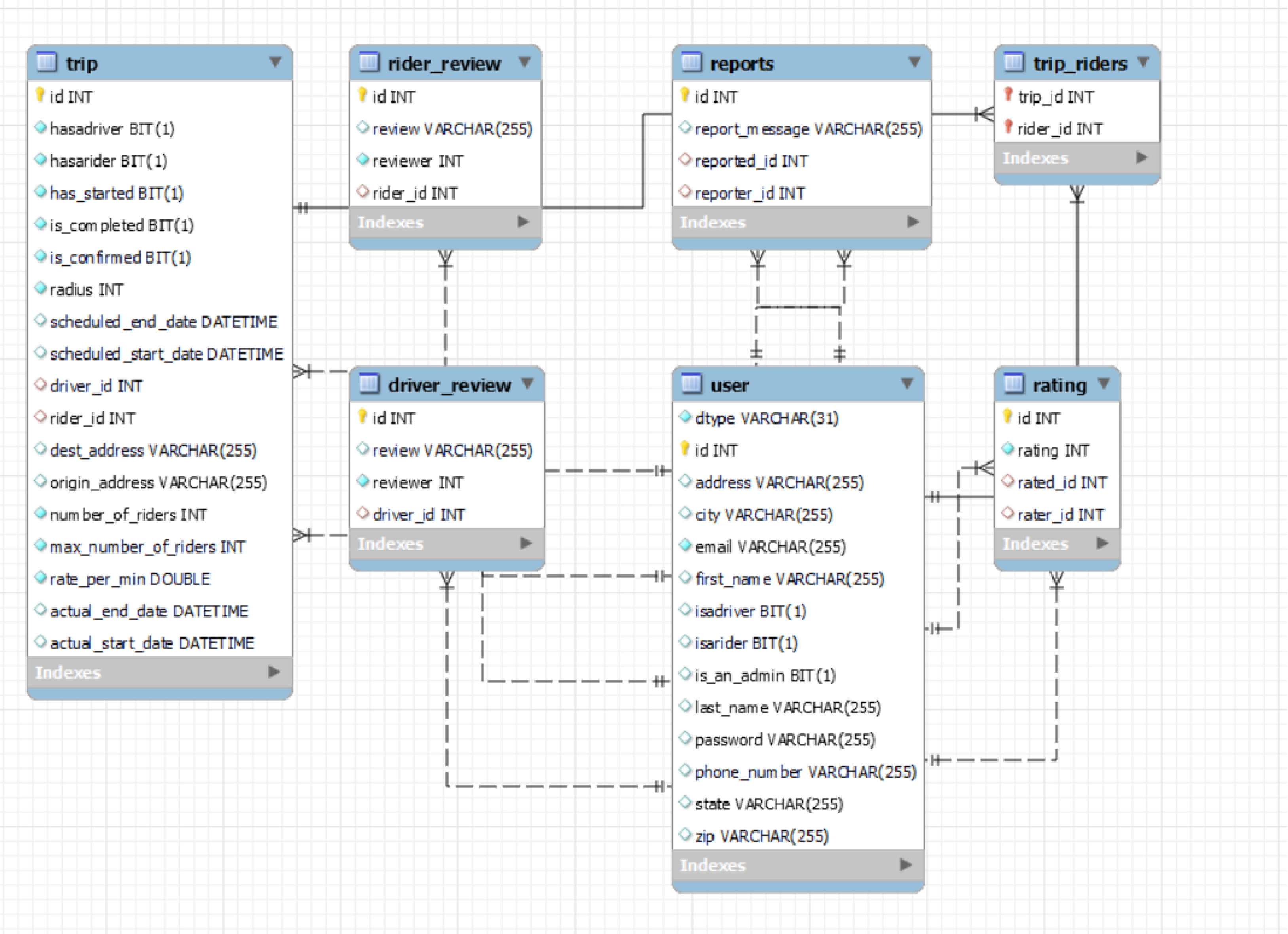
Communication:

- We use the Volley library to send HTTP requests.
- For {registering, updating, deleting} a user, we send the details to the remote backend server using coms-309-[etc].. The same goes for {registering, updating, deleting} a trip.
- For making a trip, once an origin and destination is chosen, we use Google Maps' Direction API to calculate the route. We draw this route locally onto our Google Maps fragment. When the user confirms their origin and destination, we use Google Maps' Distance API to calculate the distance and time of the trip.
- Our endpoints are defined in a class called endpoints.java.

Server side:

- Our MySQL database connects to our SpringBoot Application via JDBC.
- We directly change and retrieve information from the database through our repository classes.
- We then use the repository classes in our service classes which is where most of the logic lies. Our controller classes are where our endpoints are defined that are then used by the frontend side.

DB Schema



trip
id INT
hasadriver BIT(1)
hasarider BIT(1)
has_started BIT(1)
is_completed BIT(1)
is_confirmed BIT(1)
radius INT
scheduled_end_date DATETIME
scheduled_start_date DATETIME
driver_id INT
rider_id INT
dest_address VARCHAR(255)
origin_address VARCHAR(255)
number_of_riders INT
max_number_of_riders INT
rate_per_min DOUBLE
actual_end_date DATETIME
actual_start_date DATETIME
Indexes
rider_review
id INT
review VARCHAR(255)
reviewer INT
rider_id INT
Indexes
reports
id INT
report_message VARCHAR(255)
reported_id INT
reporter_id INT
Indexes
trip_riders
trip_id INT
rider_id INT
Indexes
driver_review
id INT
review VARCHAR(255)
reviewer INT
driver_id INT
Indexes
user
dtype VARCHAR(31)
id INT
address VARCHAR(255)
city VARCHAR(255)
email VARCHAR(255)
first_name VARCHAR(255)
isadriver BIT(1)
isarider BIT(1)
is_an_admin BIT(1)
last_name VARCHAR(255)
password VARCHAR(255)
phone_number VARCHAR(255)
state VARCHAR(255)
zip VARCHAR(255)
Indexes
rating
id INT
rating INT
rated_id INT
rater_id INT
Indexes