CSM3114 K1 Project 2

Haulier

A Trucking App

Prepared By

GARY LIM KHAI ZHE S62079

Table of Contents

Introduction

It should not take more than 10 seconds to realize how important the trucks that you saw on the highway are. Haulier is one of the most important jobs that has ever existed since the globalization of the world. It is literally how the supply chain functions and creates a bunch of job opportunities for people. Now, with the advancement of technology, we can make the process even more efficient, well, for the administration at least. By using a system to monitor the efficiency of the hauling process, the company will be able to maximize the profit out of the job, preventing waste of manpower, and cut cost as much as possible, which is good for the profit of company, at least, until AI is here to replace all the useless workers.

So yes, as mentioned, a haulier tracking system is good for administration, and management in general, and we could make it even more convenient by porting the system, literally, on one's palm, a mobile app, but not just a mobile app, we can be even more cost efficient by making a cross-platform compatible app, using something like Flutter. As such, this proposal will be about a prototype for a haulier tracking system, written in Flutter.

Executive Summary

Haulier is important to the world supply chain. With the advancement of technology, a haulier tracking app can be created to monitor and improve the efficiency of the company in general. So, our main target will be the hauling companies around the world, with more emphasis on companies that have yet to experience digital transformation.

As our system is focused on user-friendly experience, we provide a lower entrance barrier compared to other systems in the market, thereby being an objectively better choice. Besides that, by having a simple yet functional backend, customers can get the gist of operating and administering the system at ease. Depending on the contract, we will provide post service support for a period of time, preparing customers before they are ready to go on their own.

The project is expected to start on a small scale, converting part of the target companies fleet into using the system, and monitoring the process to measure the effectiveness of the system for better strategy adjustment. Once the experiment is proven to be working, we would be able to convince our customer to go fully digital, thereby completing the transformation, and we would obtain the full payment.

Use Case

For the backend of the system, it consists of 3 major objects / classes:

- Users: recording the entries of drivers as users.
    - Username, password
- Trucks: registering the trucks owned by each user.
    - Plate number
- Schedules: making schedule for hauling.
    - Status, truck, start and end date, start and destination

The backend should be able to handle basic CRUD operations to update the data in database, including:

- Add User
- Edit and Delete current User
- Add Truck, Schedule
- Edit and Delete current User owned Truck and Schedule

Due to the fact that Schedule depends on Truck, and Truck depends on User, cascade delete will be a thing to prevent a dangling pointer… uh I mean data.

App Structure

Of course, nobody is expecting the user to manually input the data into the system, that would be so not user friendly and not secure, which is why we will make a frontend for the haulier tracking system, using Flutter, for obvious reasons such as cross platform compatibility etc. This is how the flow of the app will be:

Start App > Login / Register > Home

Users will be required to log in into the system, or create an account if they don't have it. Then, they will be navigated to the home page. On the home page, there will be a lot of branching, including:

- EditUserButton(?) > View User > Home
- TruckList > View Truck > Home
- Statistic
- ScheduleList > View Schedule > Home
- LogOutButton > Login

Component	Usage
Statistic	The "Overview" page, which is the default landing page after Login, shows statistics of hauling metrics
View User	A form to edit current user data
Truck List	Trucks registered by user
View Truck	A form to add, edit or delete current user truck entry
Schedule List	Schedules created by user
View Schedule	A form to add, edit or delete current user schedule entry
Logout Button	You probably know what it does

Widgets & Features

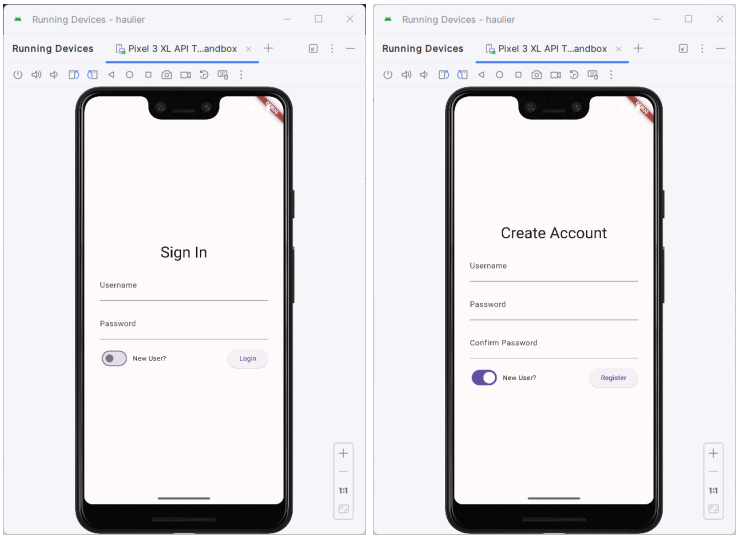
Generally speaking, each “View” is a full page Scaffold widget, while each “List” is generally ListView, unless otherwise specified. Now for the details. Firstly, no, there are no SharedPreferences in this app to save the user’s logged in status, but it is possible to implement it. Secondly, class objects are not used in this system, because the developer is experimenting with Map<K,V> based database oriented data structures, which means the logic is applied on the backend.

For this project, the Key, GlobalKey and TextFormField are being used in an esoteric, inhumane and twisted way. In terms of Form in general, since the backend are the ones that handles the form logic, there is supposedly no need to do a validator before onSaved. But, we still want the error to reach the user, so, instead of the usual validator then save approach, we did it in reverse, by submitting the request after save, we use the error received from response, if any, to populate the validator.

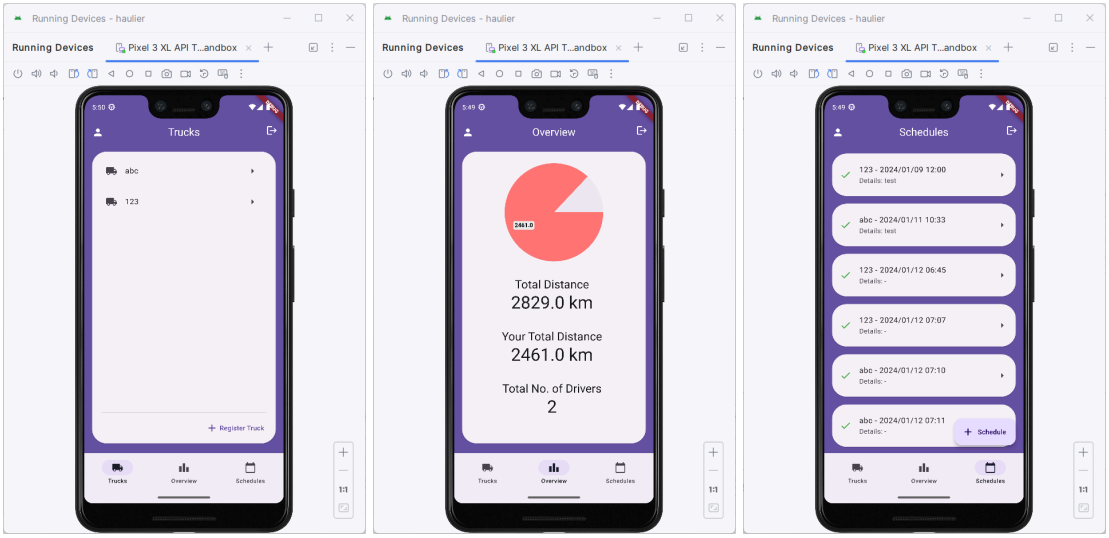
For the Schedule View in particular, as you might have noticed (if not, you can still refer to the next section), not every text form field accepts text input, some are dropdown select, which is understandable, while some others, are text form field that are read only, and only triggers function to edit the data on tap, which is shown on “initialValue”. The issue is that, Flutter is smart, it won’t simply redraw the initialValue even on set sata by default, which means you have to find a creative way to redraw the form field widget, which one of it is by declaring the Key for the field using the value of form field itself, which forces the redraw on set state. I find the person that first found this exploit ingenious.

For fancy date time, intl is used for easy formatting, while Geocoding is used for getting the location name based on coordinates given. For the location tracking, an await Timer is used to spam getting user location every 10 seconds or so, updating the live location of the user, while FlutterMap is used to display the location (or allow user to pick one), using OpenStreetMap. For statistics, a PieChart is used for visualizing the data for truck utilization.

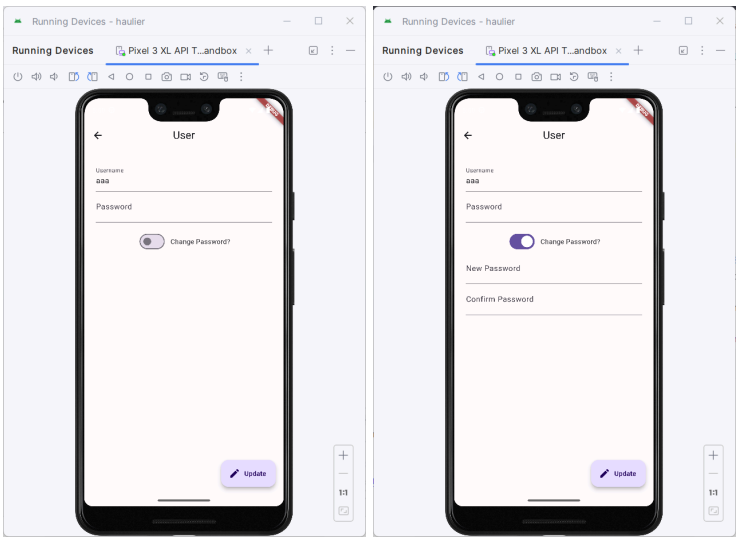
User Interface



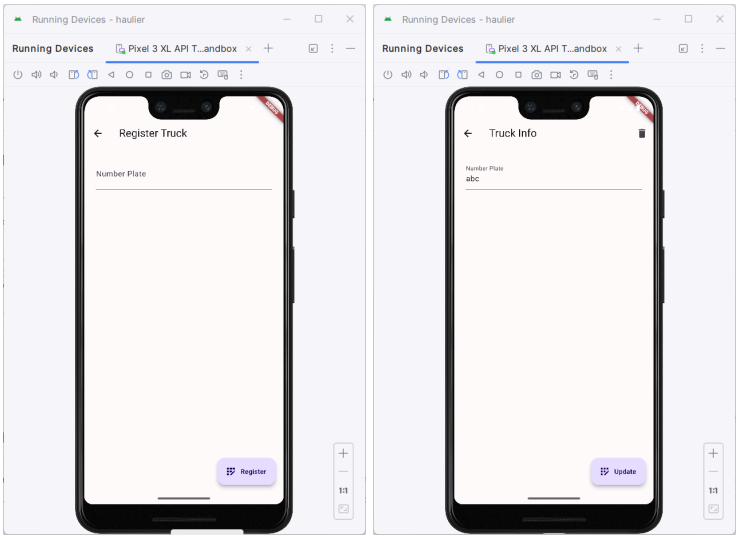
The login page, with built-in switch that allows the user to register an account instead.



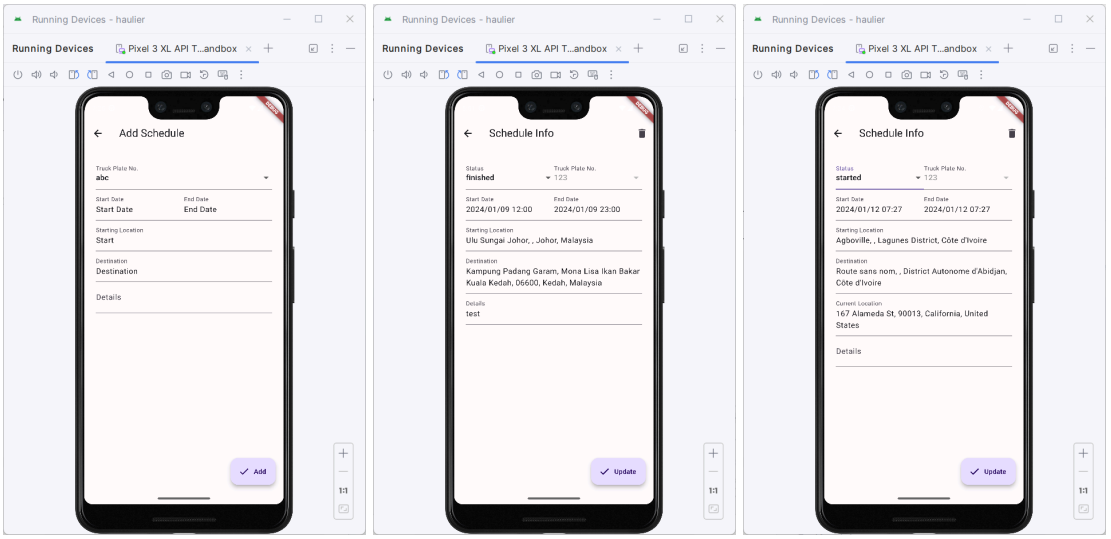
From left to right, Truck List, Statistic and Schedule List, on the left and right side of the app bar, there's a User View and Logout Button, each with features outlined in the App Structure section. Note: Statistics are scrollable.



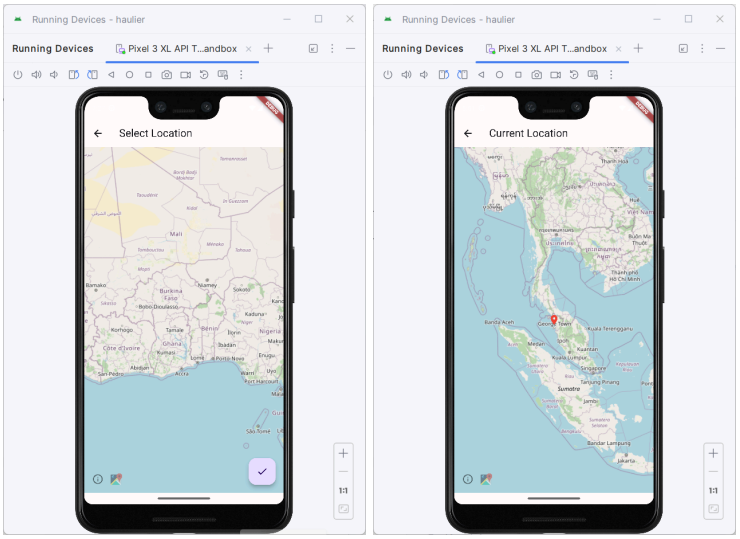
User View, a page where users can edit their own username, or password if they want, there's a switch for that, it is neat.



Truck View allows users to register a truck under their user account or edit/delete truck data that they have registered, it looks a little bit empty due to the number plate being the only important data that actually matters, but of course, it can be modified as per customer requirements. The system does not allow duplicates of truck's number plate, which means no two users can share the same truck.



Schedule View works mostly similar to Truck View, add, edit & delete data, yada yada, **except**, when editing an existing schedule, the user is allowed to change the status of the schedule. When the schedule status is started, the app will automatically try to get and update the location of the current device, which acts as a tracker to track the progress of the hauling. If location permission is not given, the current location will default to the starting location (in theory). Oh, and, most (pseudo) text fields are only enabled on “scheduling” status, except details, which is literally notes for the driver.



When selecting a “location” relevant field, you will be sent into a Location View that shows your current location / allows you to choose a location if the schedule is “scheduling”.

Conclusion

In conclusion, a Hauling Tracking App is planned and implemented in prototype phase, and is proven functional. It is written in Flutter, able to perform basic CRUD operations, and has a dedicated backend database that is based in SQLite. But most importantly, it does what it was named as, a hauling tracking app that tracks driver's live location, and calculates overall truck utilization.

Personally, this prototype is kind of lacking to be honest. I am not impressed with the outcome for this one, but I have since then learned to accept the fact that sometimes there is just not enough budget (time, money and resources) to achieve what you want, compromises are needed. That said, I have learned new stuff, and sometimes that's all that matters.

Reference

1. https://www.linkedin.com/pulse/vital-importance-freight-transport-economy-trancasa-corp
2. https://www.ddcfpo.com/freight-process-insights/12-metrics-you-should-be-tracking-in-fleet-utilization
3. https://stackoverflow.com/questions/61276701 (unused)
4. https://pub.dev/documentation/flutter_map/latest/flutter_map/InteractionOptions-class.html
5. https://stackoverflow.com/questions/67464084
6. https://stackoverflow.com/questions/29628989
7. https://stackoverflow.com/questions/45900387