Introduction

ProfitPath is a one-of-a-kind community focussed application aimed to revolutionise transportation and delivery services in Singapore. Following the “Carousel” model of buyer and seller which had earlier reshaped retail and the second-hand product market in Singapore, ProfitPath allows users to be both passengers and drivers, enhancing user engagement through a versatile platform.

One of the key features of ProfitPath is to allow users to earn money by providing transport and delivery services on daily commutes! This idea harmonises with the Smart Nation initiative of the Singapore Government, by not only enabling a “people-centric” model that employs digital technology to improve people’s lives, it also aligns with the Green Nation goal by reducing carbon emissions and thus, Singapore’s carbon footprint as a whole. ProfitPath serves as a catalyst for socio-economic transformation by encouraging innovative approaches to regular commuting and offering people a chance to earn money by helping the community and the environment at large!

ProfitPath also provides an invaluable solution for individual transport and delivery needs. With immediate or scheduled options, users can access cheap and fast transport services or have their goods delivered in a cost-effective manner. The main part of this is that the customer gets to decide their own reasonable price for the pickup and delivery options. This gives flexibility to the users to set the price which they think is fair for a ride.

Platform

ProfitPath will be available to users as a web-based platform for now. The webapp will be available as a domain name as a web app which can be accessed on any browser on mobile phones, laptops, and tablets. This is designed to increase the versatility of the application and also encourage early onboarding of customers and users.

A study by Facebook in 2018 showed that it costs about $70 per user in marketing to get users to download the app. However, a web-app doesn’t need to be downloaded. It can be made available as a QR code at MRT stations which will open up the web-app on browsers.

Tech Stack

Since this app is a web-app, we have decided to go ahead with the traditional Server-Client Architecture with a REST API based backend service and a frontend service for users to interact with the application.

For the backend we are using the popular Python Framework Django. This framework allows us to build and scale backend apps at lightening speeds and also integrates seamlessly with a SQL database. We have connected our backend to a PostgreSQL database which is hosted on the cloud at https://www.aiven.io.

The frontend is written in React. We chose React because it gives modularity and great flexibility to our application. By splitting our app into separate components, we were able to achieve the Stereotype diagram model architecture.

Key Features

1. User Accounts

Every user will be prompted to make an account as part of the onboarding process. Account creation is a key requirement for the on boarding process. A user cannot access key services such as Requesting a Ride without making an account.
An account provides user with a unique username. Every account will be mapped to a single email address. The account will allow the user to add the details of their car (if they own one and want to become a driver on our platform), will give them access to a digital wallet which will be used for all transactions, and much more.

On the Frontend, we have a Sign-Up Page, a Sign in page, and a profile page which allows the user to see the details of their accounts.

The backend defines several API endpoints such as /accounts/login , /accounts/logout , /accounts/signup , /accounts/profile , etc which provide information to the frontend webpage as well as receive POST requests to update information on the Database.

On the PostgreSQL database, there are 3 tables which store all of this information: Users, Car, WalletHistory.

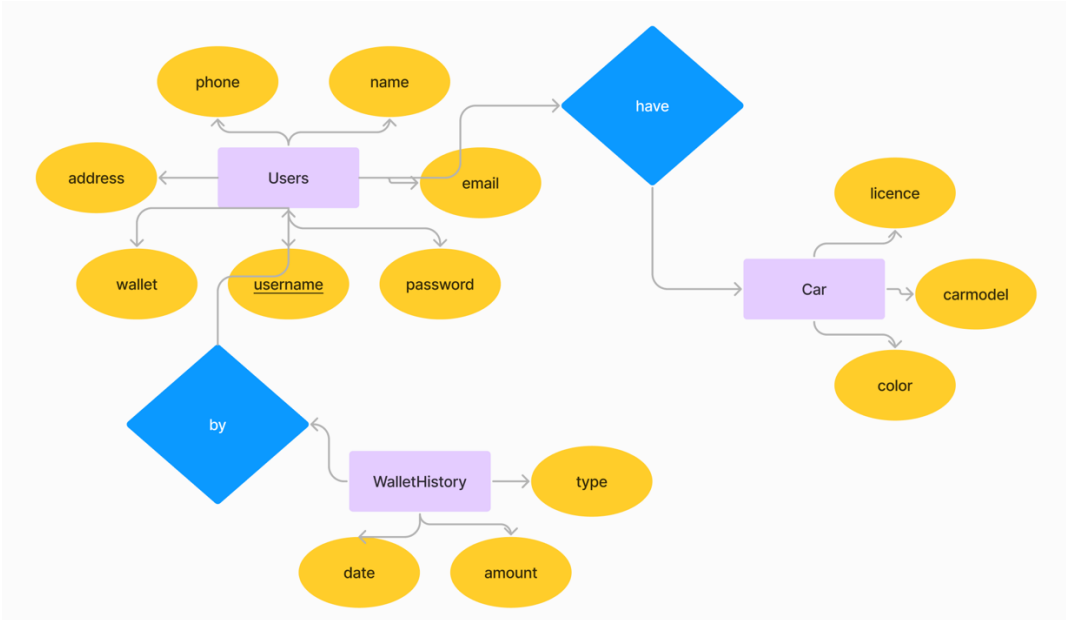

Fig 1: Entity Relationship Diagram (Database)

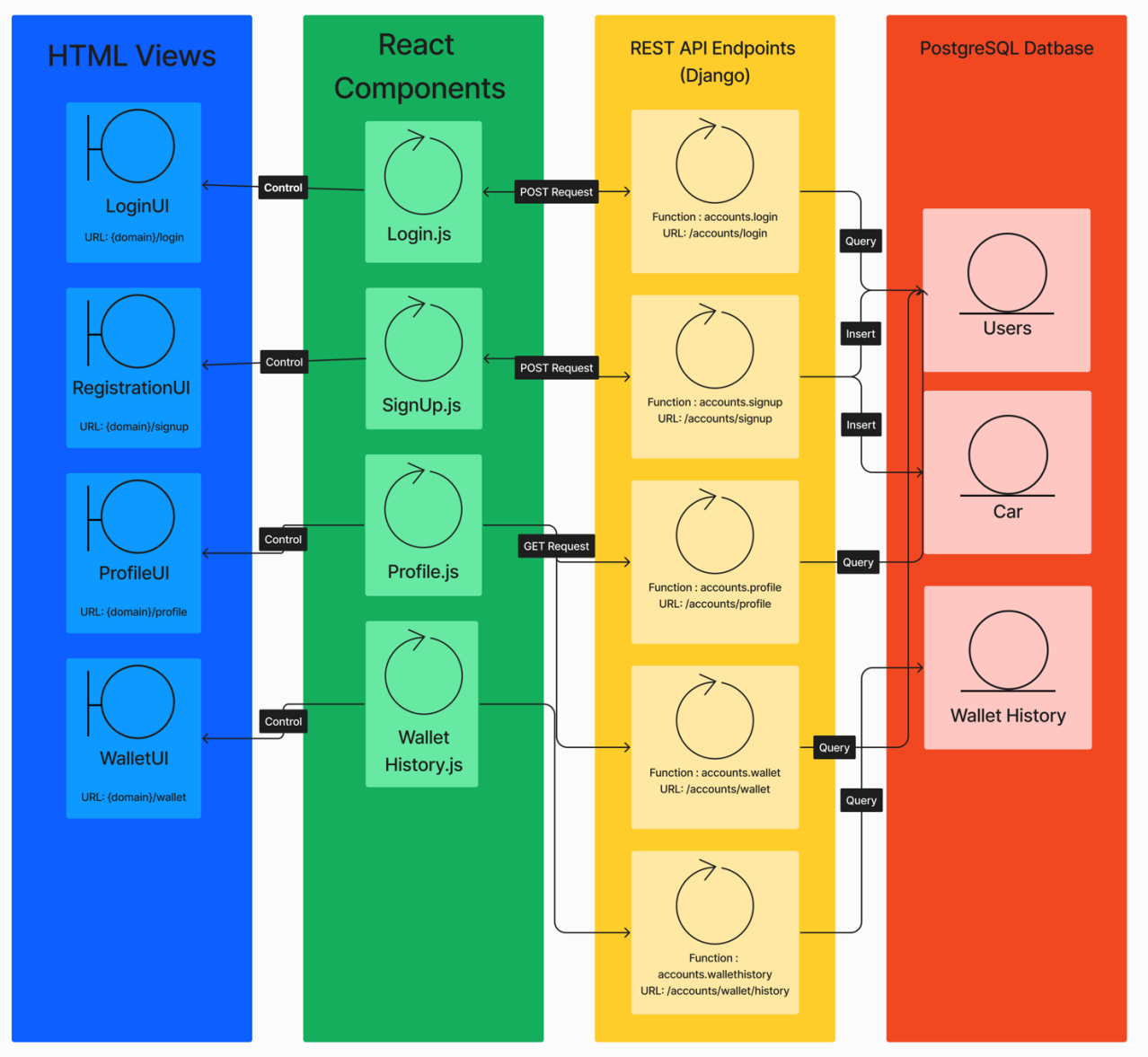

Fig 2: Architecture for the Users Feature

2. Authentication

Authentication is required to enforce security. Every time that a user logs in, a random string token is generated in the backend and sent to the frontend. The frontend caches the token in the LocalStorage of the browser. This token is sent with every subsequent request made in the header of the HTTP Request. In the backend, the token is stored along with the user associated with it on a table called AuthTokens.

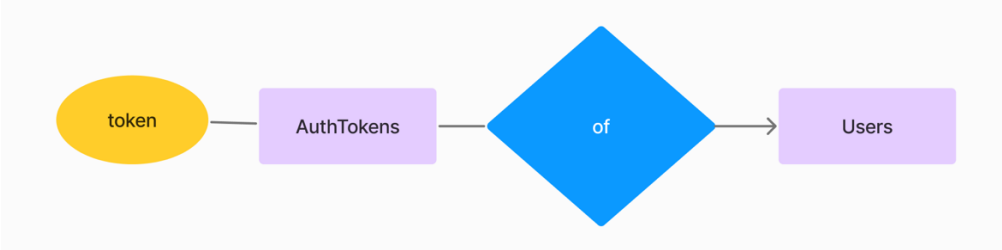

Figure 3: Entity Relationship Diagram

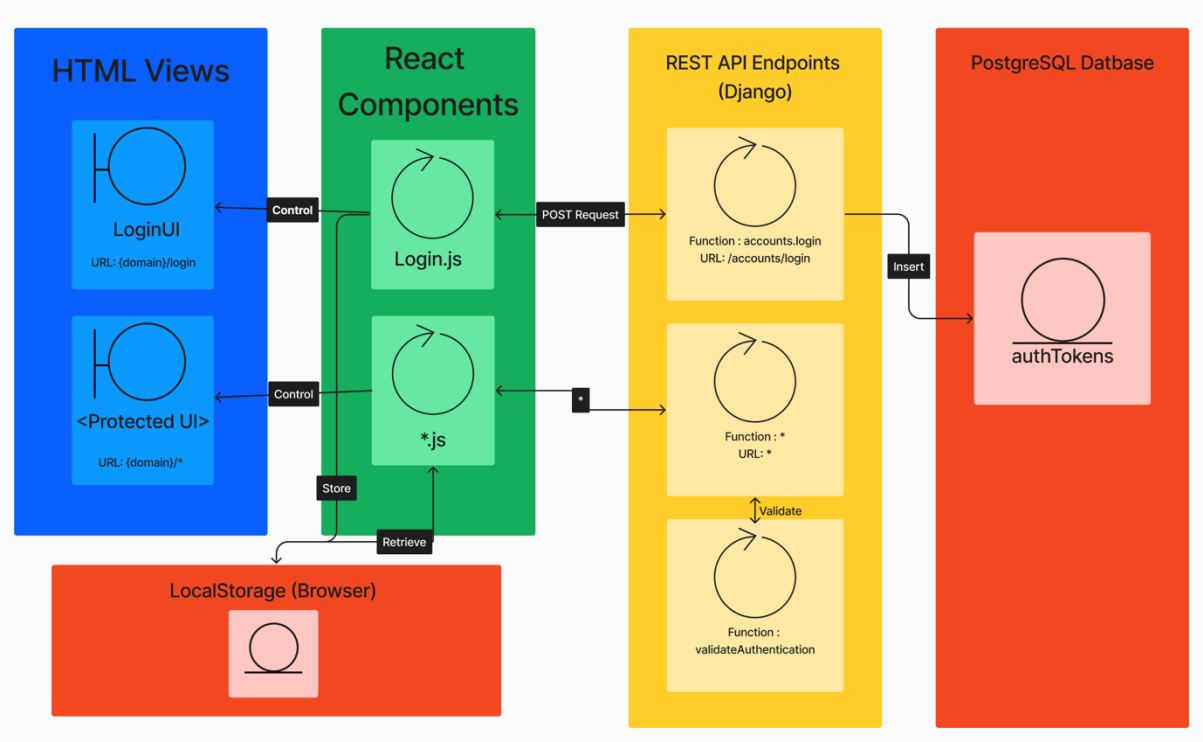

Figure 4: Architecture for Authentication

Note: This is a generic feature which extends to most of the routes and functions within the application. Thus, * , *.js , etc all denote routes/functions which are protected. Protected means routes that require a user to be logged in to the system.

3. Delivery and Pickup Services

The Delivery and Pickup services is the main functionality behind this webapp. There are multiple views and control functions associated with the entire functionality. The exact details of implementation are given in the Sequence Diagrams.

Figure 5: Entity Relationship Diagram

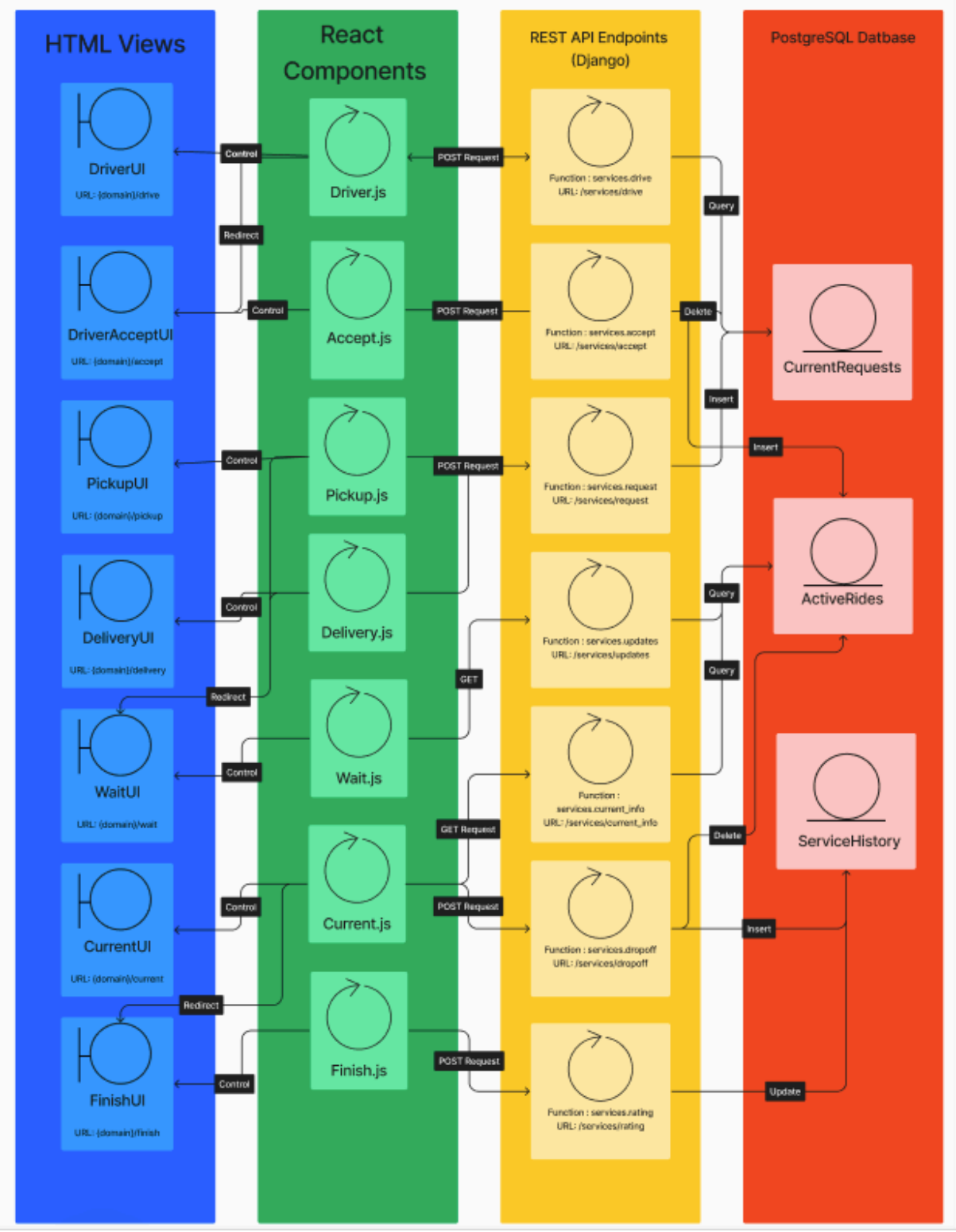

Figure 6: Architecture for Delivery and Passenger Services

4. Real Time Chat

In the CurrentUI page a delivery requester/passenger can chat with the driver in real time. For this, we implement a web-socket approach which allows them to communicate freely.

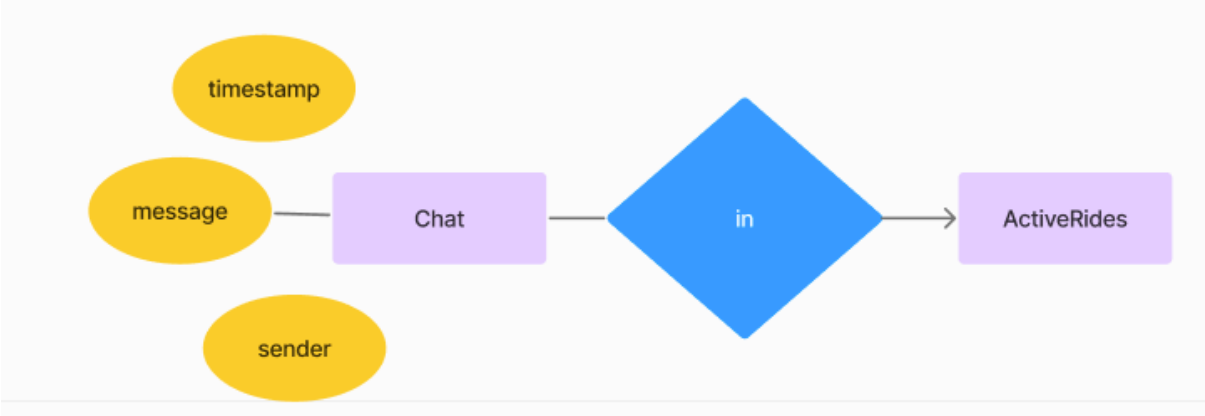

Figure 7: Entity Relationship Diagram

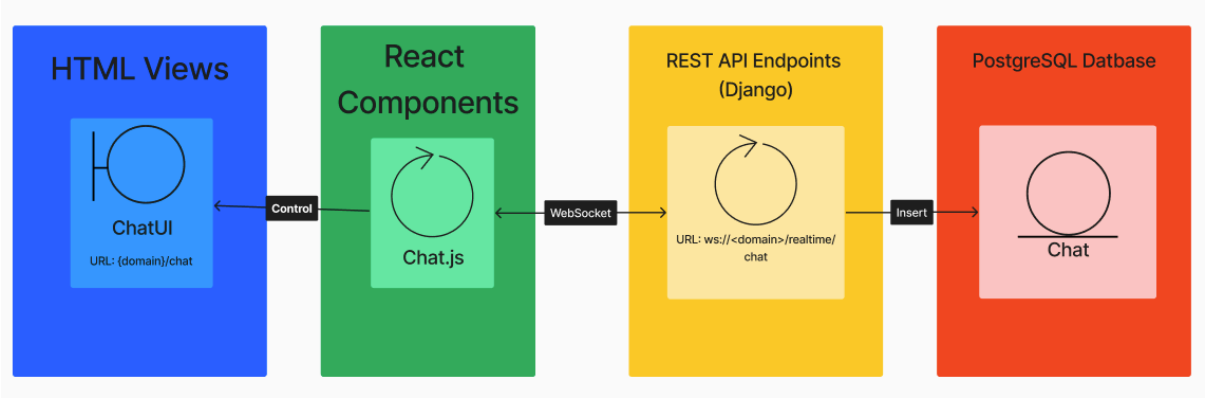

Figure 8: Architecture For Real time chat implementation

Design

For this webapp we have employed the MVC (Model-view-controller) Design pattern. We implemented this pattern because of the microservices based architecture that our backend is based on.
Model: The Django functions and the PostgreSQL database
View: The HTML Views which the user can see and interact with
Controller: The JavaScript code in the React components which connects the HTML view and the Model.