User Story:

1. As a user, I wish to reserve a car for future use on Uber.

When the user opens the Uber app, they can type or select the pickup location and time to view the reservation results, which will be sent via email to the user.

2. As a user, I would like to share information about Uber with others.

Upon opening the page, the user can press the "share" button to share the information with those they choose.

3. As a user, I want to see my current location on a map.

Upon opening the Uber app, a map with a marker for the user's GPS location will be displayed.

4. As a user, I want to search for an address or landmark and see it on the map.

Upon opening the Uber app, the user can click the search bar and enter an address or landmark name to view a list of search results for relevant addresses. Upon clicking on a search result, the address will be marked on the map, and the map view will move to the marker.

5. As a user, I wish to provide feedback on the car/driver.

After the user completes the trip, a window will appear for them to write a comment. When the user clicks on a car, a list of comments collected from other users for that driver will be displayed.

6. As a user, I want to choose the type of car that suits me when reserving an Uber.

When the user opens the Uber app, different car models and service rates will be displayed for them to choose from.

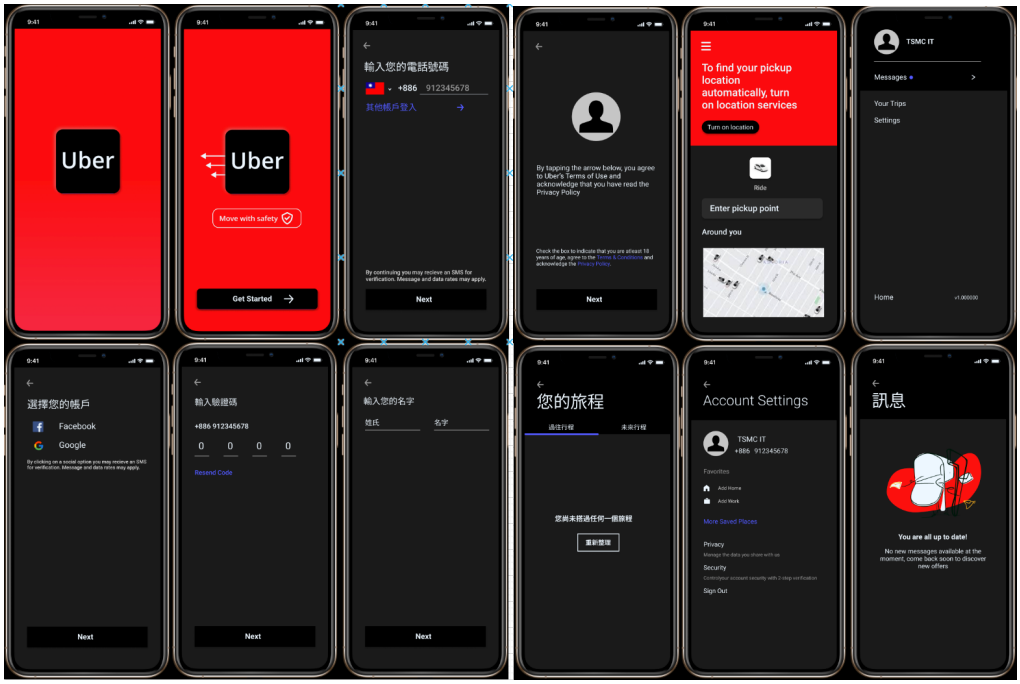
7. As a user, I want to know the prices for different locations and see them on the map.

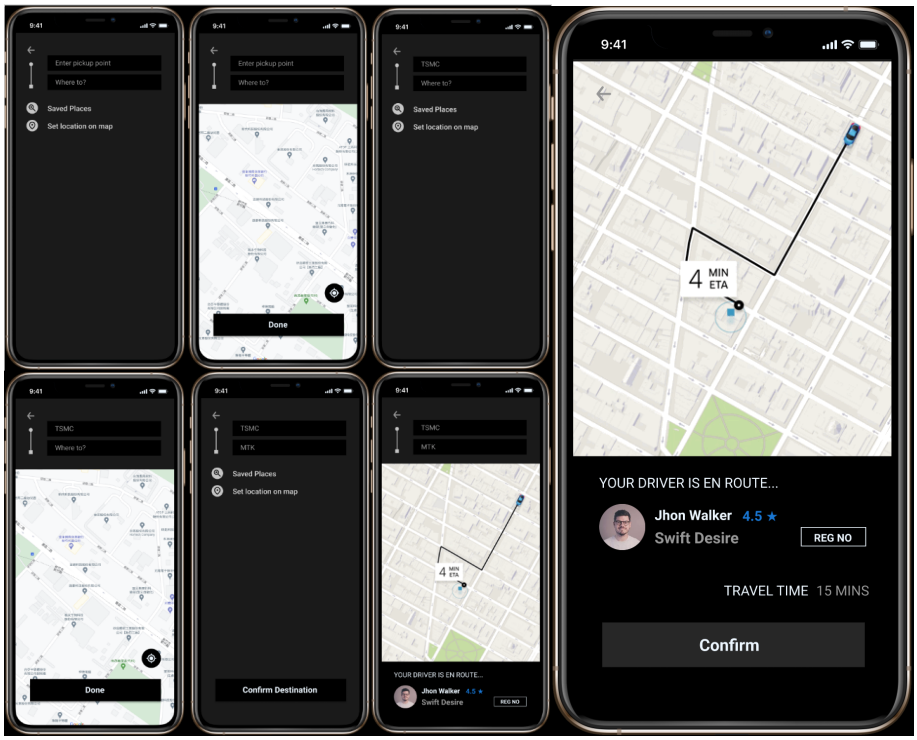
When the user opens the Uber app, they can type in an address or click on a landmark to view the prices for that location on the map.

8. As a user, I want the option to carpool with others.

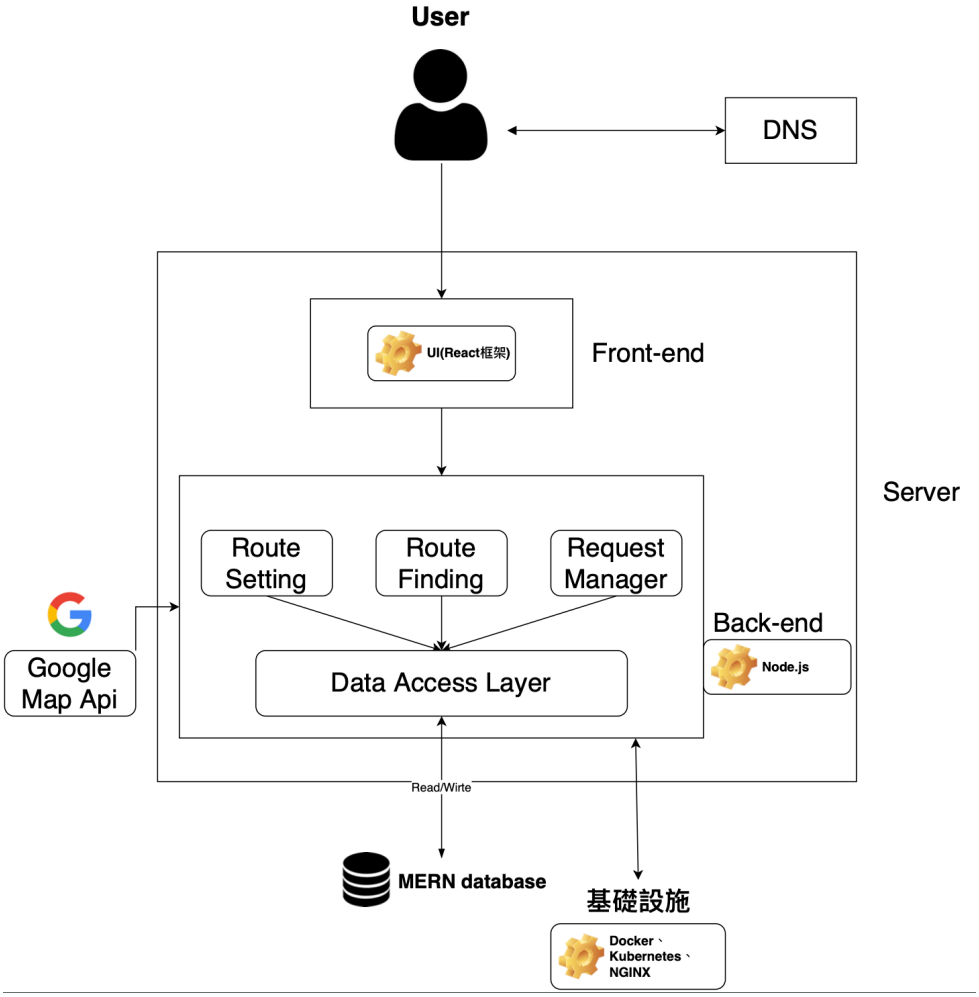
When the user checks the carpool checkbox, multiple possible paths and their fees will be displayed. After the user confirms the carpool path, the app will display the amount of time and money saved through carpooling.

UI Design & User Experience:
https://www.figma.com/file/26kS34uOBubyVtprvY5rzm/UBER-APP-DESIGN-31155185?node-id=0-1&t=rX225HYdz1yptMh4-0
細圖，詳細流程可點上方網址實際操作。





Application Architrcture:



The system architecture of the ride-sharing platform is divided into three parts: front-end, back-end, and infrastructure.

Front-end architecture: The front-end is implemented using the React framework to create a single-page application (SPA). Users can search for available ride-sharing services, select their pickup and drop-off locations, along with any waypoints, and perform booking and management of their rides.

Back-end architecture: The back-end is implemented using Node.js and uses RESTful API for data exchange. The back-end consists of the following services:

- User authentication service: used for validating user login information and access rights.
- Ride search service: based on the user's search criteria, it searches the MERN database for matching rides.
- Booking and management service: used for managing the user's ride-sharing trips, including booking, modification, and cancellation.
- Driver service: drivers can use this service to set up rides and confirm passengers.

- Payment service: used for processing user payment requests.

The MERN database uses MongoDB as the storage medium, Express as the web application framework, React as the front-end JavaScript library, and Node.js as the cross-platform JavaScript runtime environment. This provides better support for the ride-sharing platform's front-end React framework and back-end Node.js services, improving system performance and scalability.

The map and navigation service uses Google Maps API to enable drivers to plan the best route and confirm ride completion. The monitoring and alerting service uses Prometheus and Grafana to monitor system metrics and quickly detect and handle faults.

Infrastructure architecture: The infrastructure is implemented using Docker containerization technology and uses Kubernetes for automated deployment and management. Load balancing and reverse proxy are implemented using NGINX to improve system stability and scalability. Additionally, features such as log management and backup recovery are implemented to ensure system data integrity and reliability.

The advantages of this system architecture include the use of modern technology stacks and containerized deployment methods, allowing for quick response to business needs and high concurrency. The front-end and back-end separation architecture also improves system maintainability and scalability, making the system more flexible and easy to deploy. Additionally, by using monitoring and alerting services, faults can be quickly detected and handled, ensuring system stability and reliability.