Product Design

|  |  |
| --- | --- |
| **Team** | **Team Number: 58**  **Team Members:**   * **Aaradhya Gupta - 2019114010** * **Abdullah Mujtaba - 2019101093** * **Kartik Garg - 2019101060** * **Kevin Vargis - 2019101092** |

# **Design Overview**

## **Architectural design**

The two major subsystems are:

1. The WebApp
2. The Mobile App

These are linked by a common database/backend.

1. **WebApp**

* Aims to provide support to the supervisor, manager , and higher levels in the company
* Gives customer details of his plan/bills, and notifies before delivery
* Deals with organising and creating profiles
* Creates trips
* Manages stockpoint
* Allows decision making regarding stakeholders

1. **Mobile App**

* To be used by field agents
* Allows viewing of duty/trip details
* Deals with on-field transaction details input

**Common Database/Backend:**

* Holds information regarding each stakeholder
* Contains API’s for functionalities
* Facilitates information sharing between two subsystems

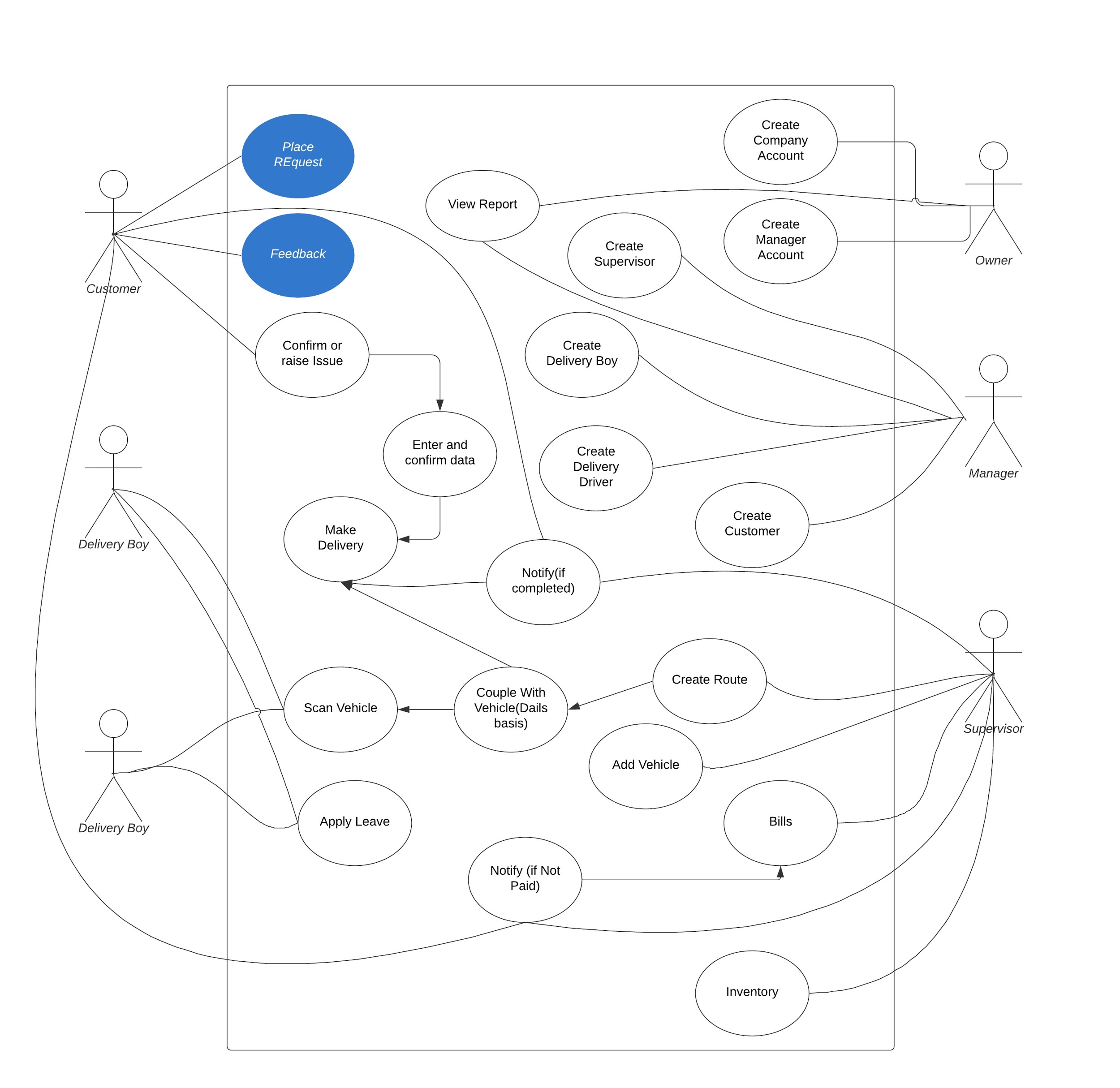
Used by the Management, customers

View all the data to make decisions.

Used by field workers

Captures delivery, trip data

Stores and handles all this data to keep the two systems in sync



## 

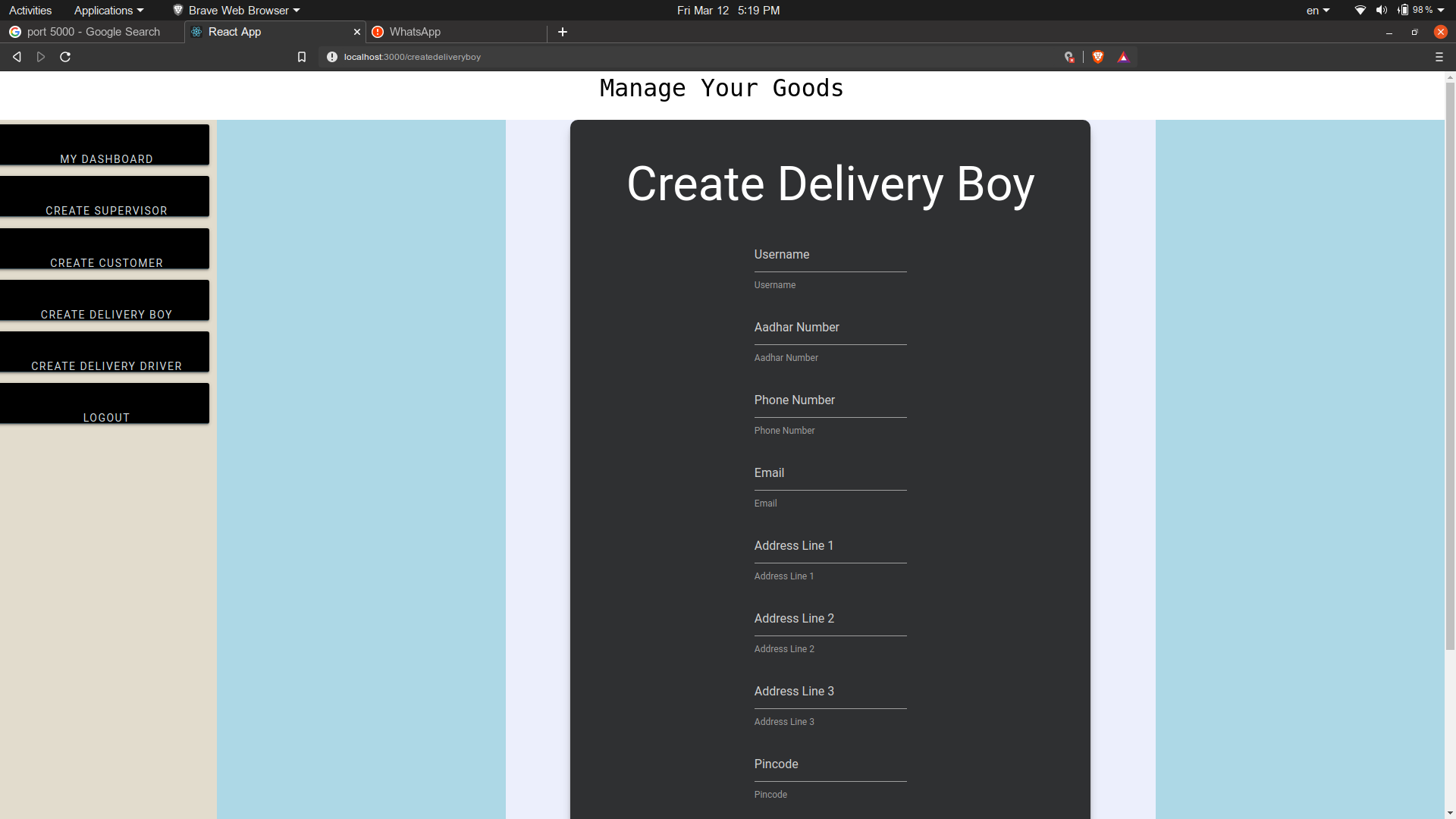
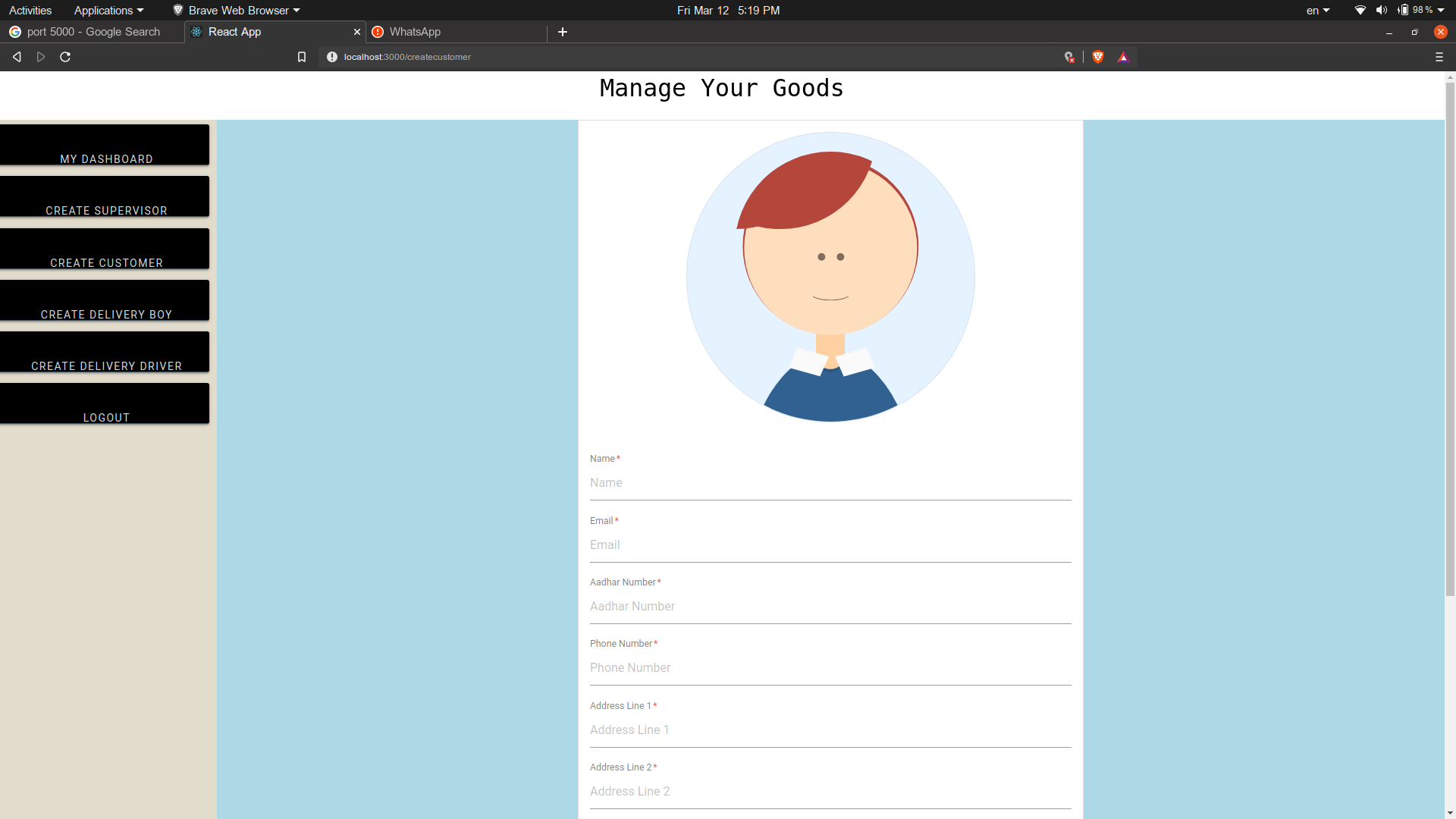
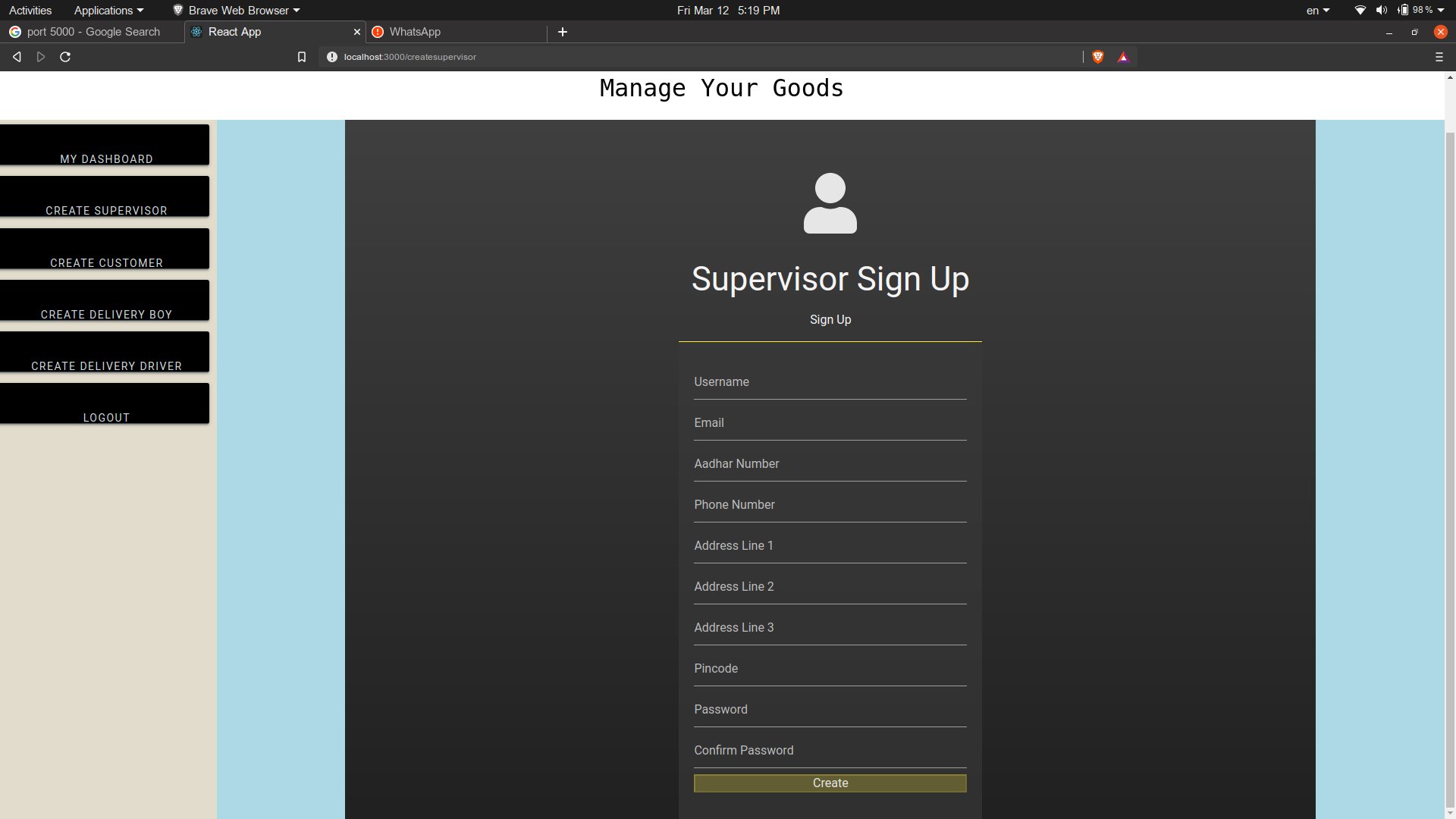
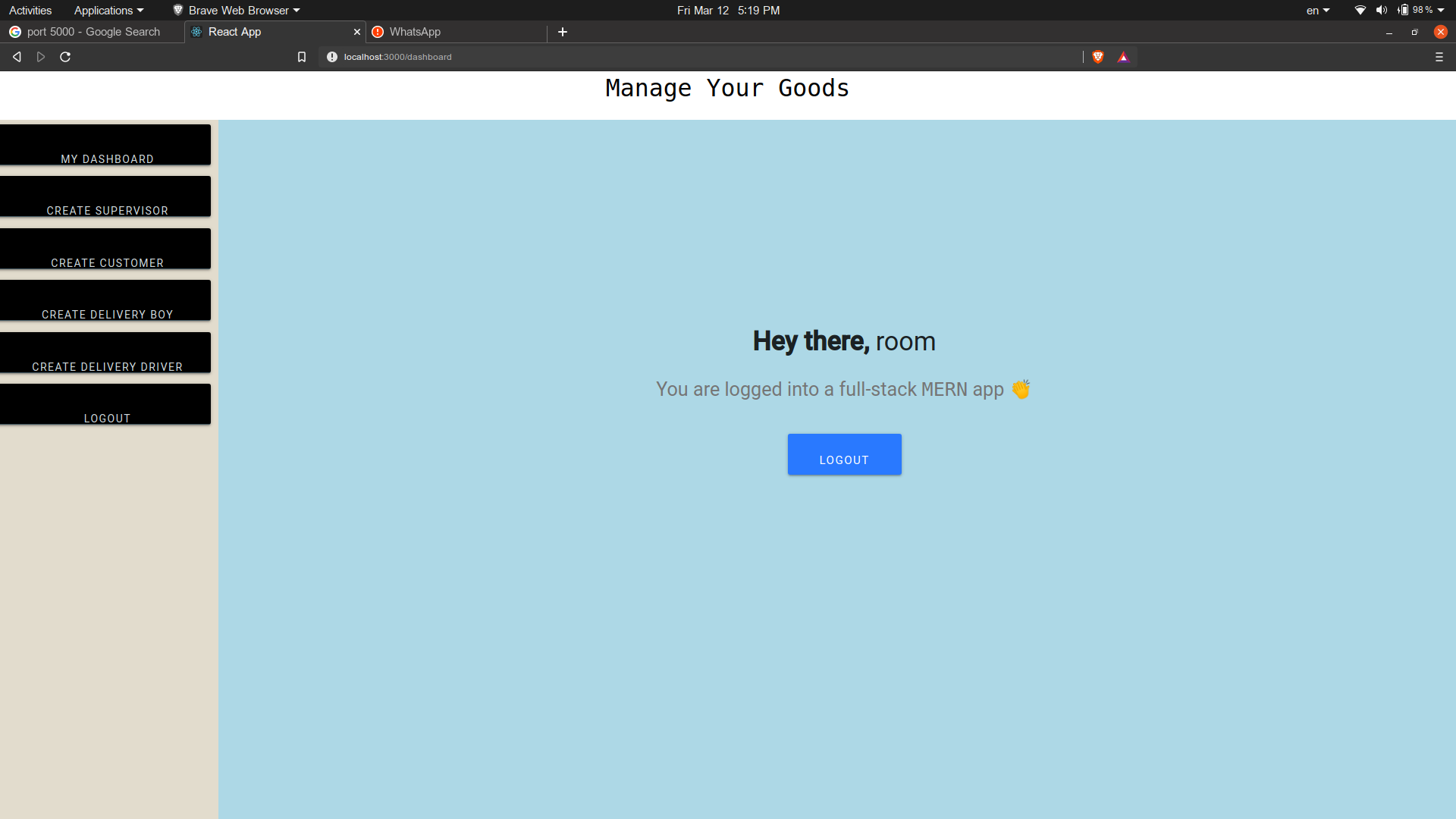
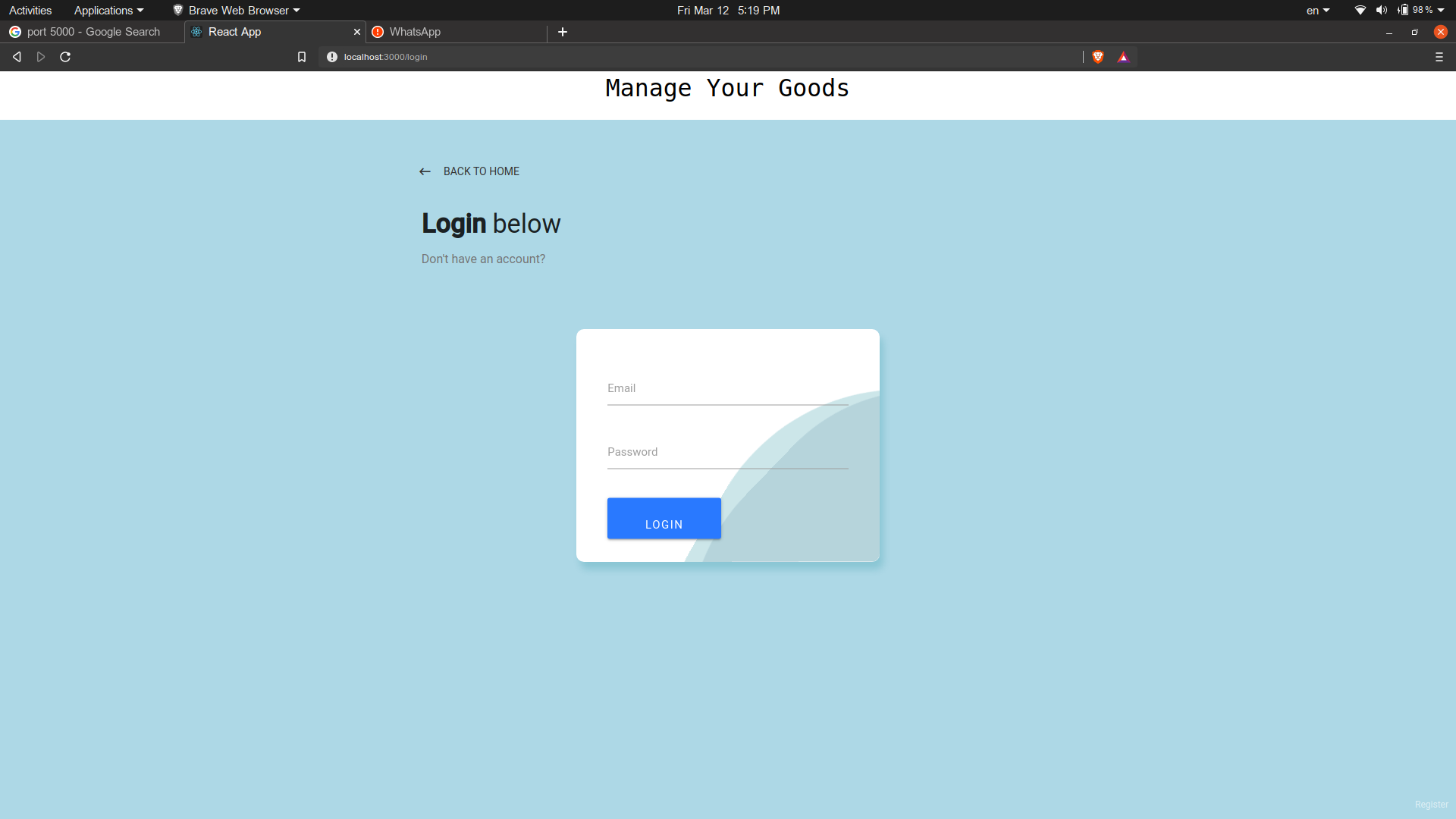
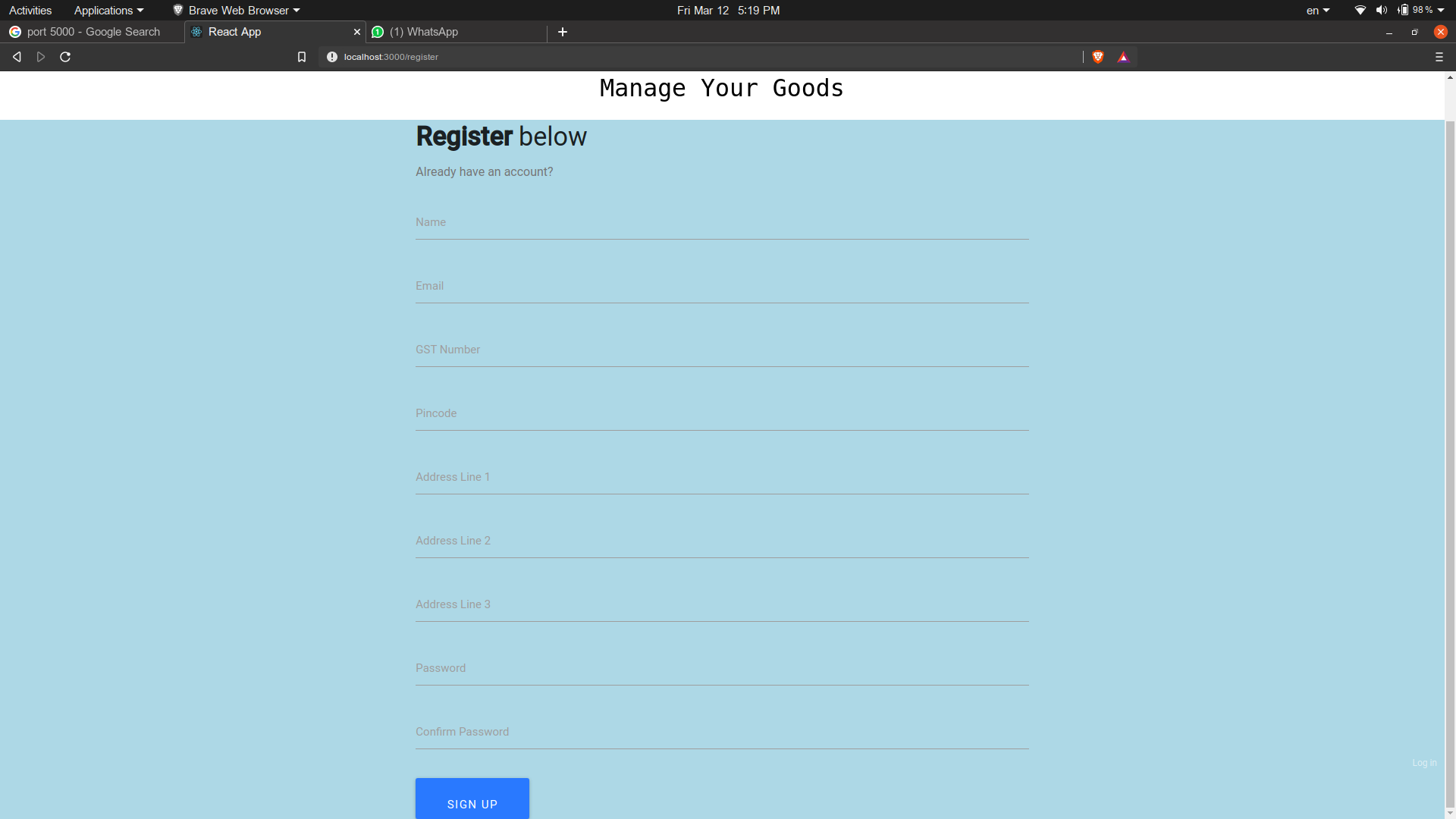
## **System interfaces**

### **User Interface**

A separate screen takes care of each use case described. Following are some of the screens that have already been created, with additional formatting and design still to be incorporated during subsequent development stages

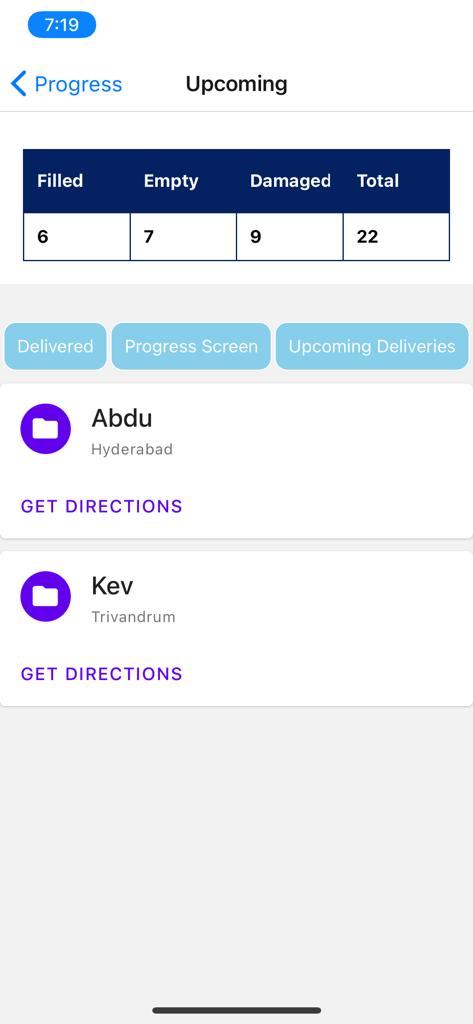
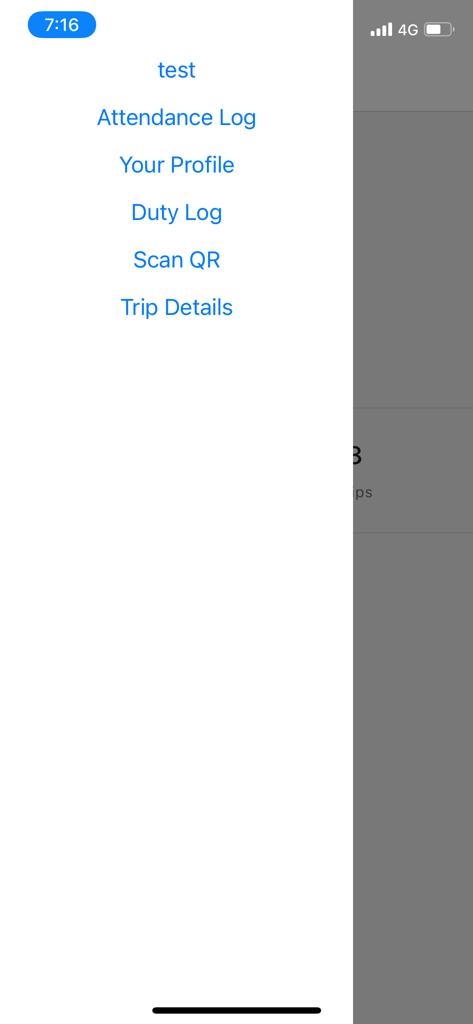
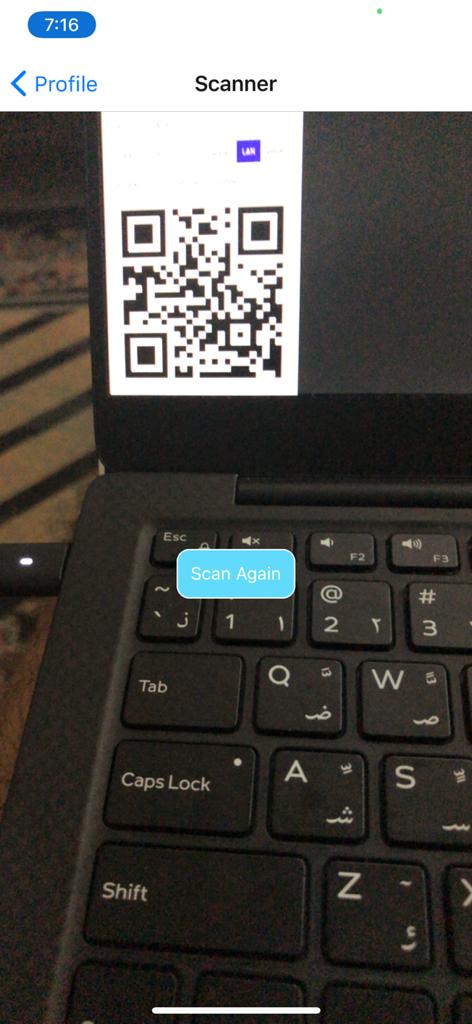
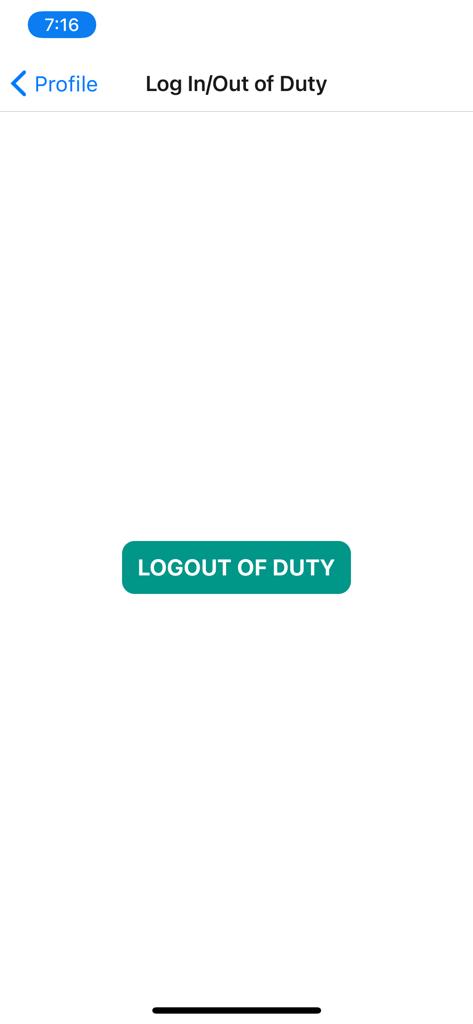
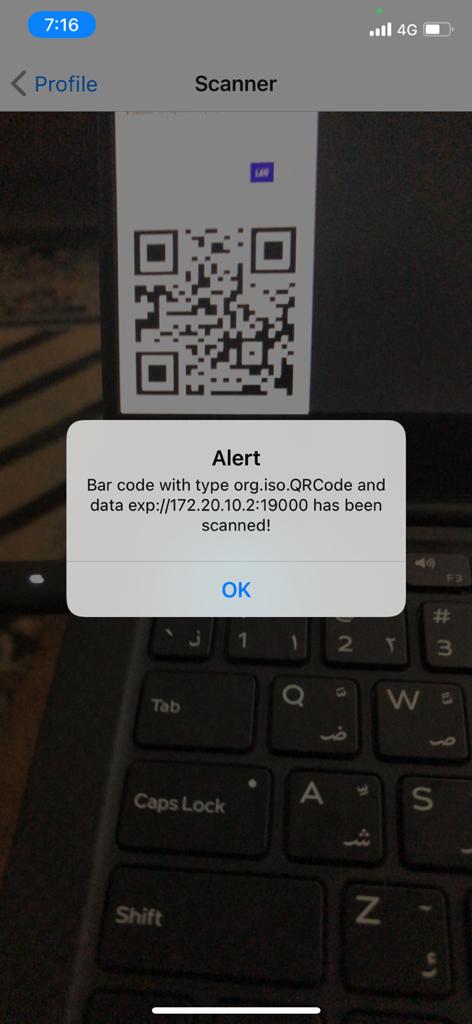
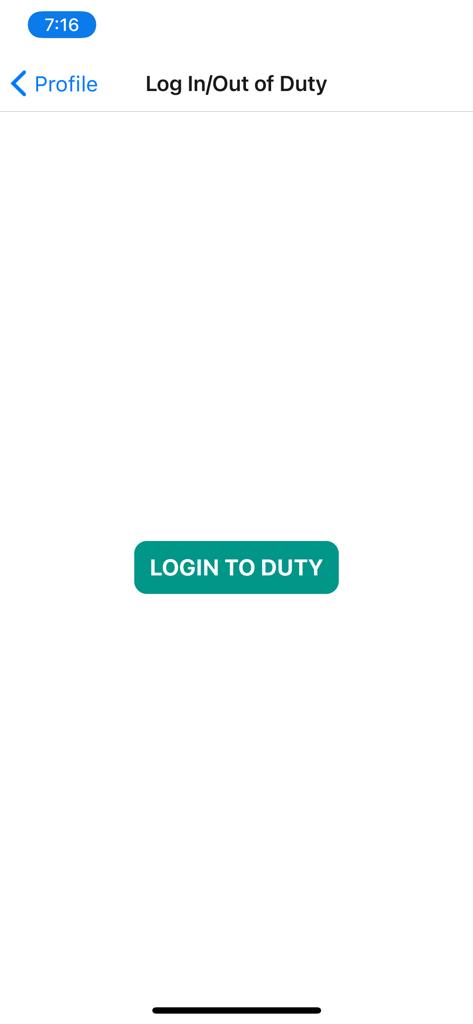
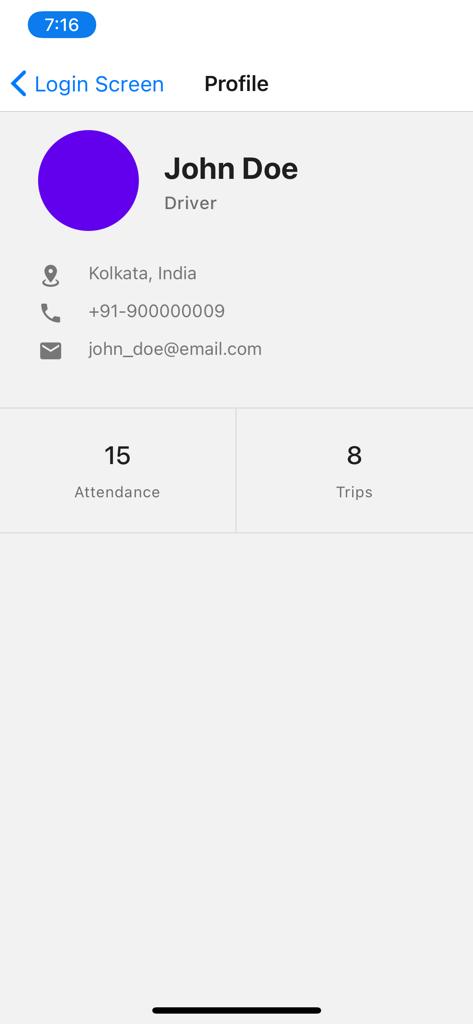
**WebApp**

* Profile Creation for :
  + Company
  + Delivery Boy
  + Driver
  + Manager
  + Supervisor



**MobileApp**

* Login
* ScanQR
* Login and Out of duty
* View Trip Details

# 

### **APIs**

**MobileApp**

## native-camera and QR code scanner: to recognise barcodes

1. Google Maps API’s: for map views
2. Login : Login to the app
3. Profile: Get details for the profile
4. Attendance Details: Access and manipulate attendance details for on-field agents

**WebApp**

Login: Login as a Owner/ Manager / Supervisor

RegisterCompany: Register Your Company

RegisterManager: Register Manager by Company

RegisterSupervisor: Register the Supervisors (Manager)

RegisterDeliveryBoy: Register Delivery Boy (Manager)

RegisterDeliveryDriver: Register Delivery Driver (Manager)

Similarly each one of them will have their deletion API’s

CreateVehicle: Create a new Vehicle with the Company

GetInventory: Get the details of the empty, filled and damaged cans on the Stock Point

CreateRoute: SuperVisor will create the daily routes.

ViewBills: This API will be mainly used by Supervisor and Customer to view their monthly bills.

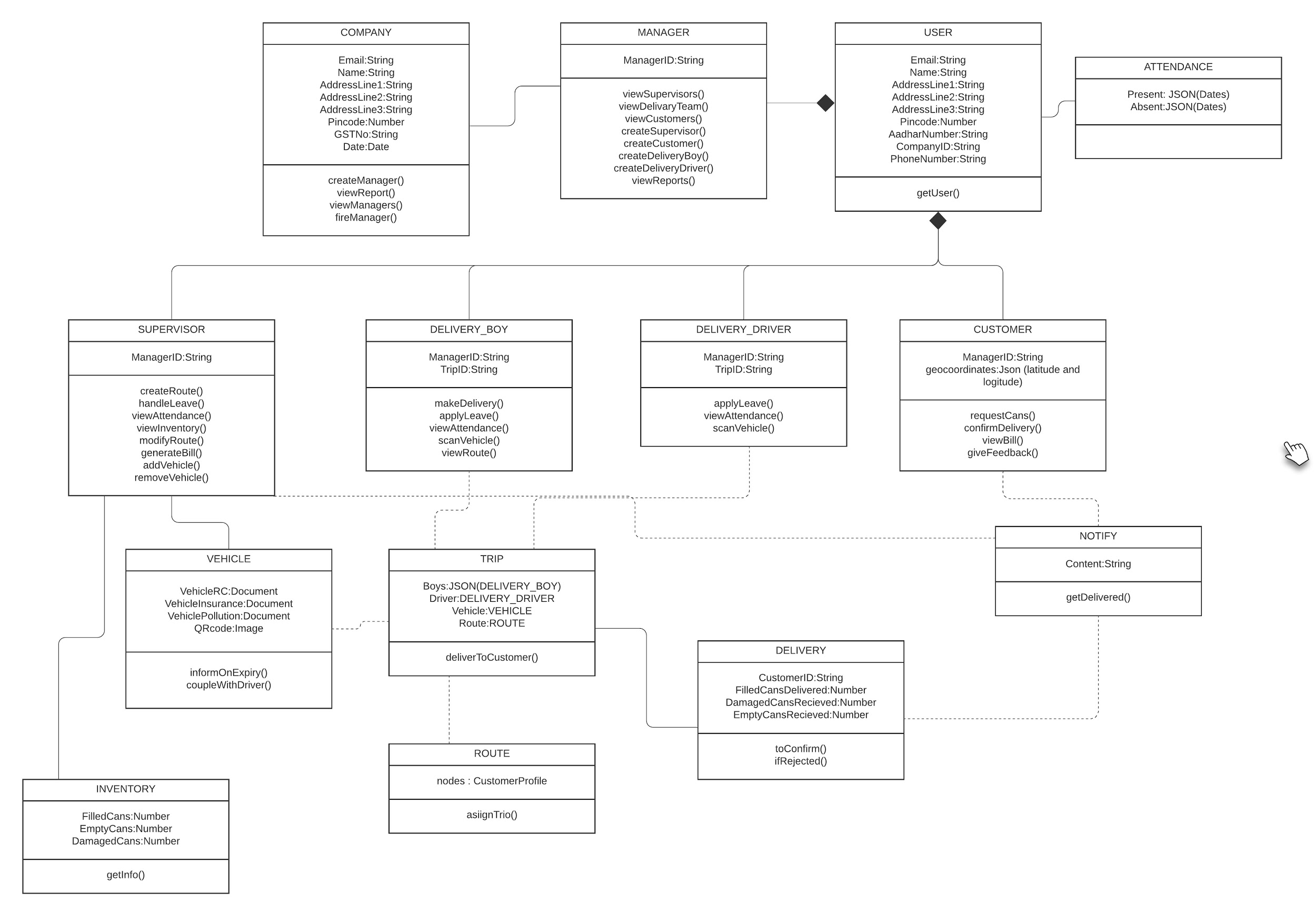
## 

## 

## 

## 

## **Model**



|  |  |
| --- | --- |
| Company | Class State -   * Email of the Admin/POC * Name of Company * Addresses of Company * Security PIN of Account * Associated GSTNo * Date of creation of account   Class Behaviour -   * Create new Manager accounts * View service report * Get a list of current Managers * Fire a manager |
| User | Class state   * This is a parent class which other classes like Manager, Customer, etc inherit from * Email address of account * Name of account holder * Addresses of user * Security PIN of account * Aadhar Number of user * Company ID * Contact Number * Users ID   Class behavior   * Get details of a user |
| Manager | Class state   * … from User * Manager’s ID   Class behavior   * … from User * Get all the Supervisors * View Delivery Teams * View Customer Info * Create Supervisor Accounts * Create Customer Accounts * Create Delivery Boy Accounts * Create Driver Accounts * View Report |
| Attendance | Class state   * Dates present * Dates absent   Class behavior   * N/A |
| Customer | Class state   * … from User * ID of Manager who created account * Location of drop off point for customer   Class behavior   * … from User * Request water cans * Confirm whether delivery has been completed * Get bill * Provide Feedback |
| Delivery Driver | Class state   * … from User * ID of Manager who created account * ID associated with current trip   Class behavior   * … from User * Apply for leave * View attendance * Scan the vehicle |
| Delivery Boy | Class state   * … from User * ID of Manager who created account * ID associated with current trip   Class behavior   * … from User * Apply for leave * View attendance * Scan the vehicle * Log a delivery * View delivery |
| Supervisor | Class state   * … from User * ID of Manager who created account   Class behavior   * … from User * Create a new route * Address issues/requests regarding leave * View attendance * View inventory * Modify created route * Generate a bill * Add a vehicle * Remove a vehicle |
| Notify | Class state   * Notification Content   Class behavior   * This class handles notifications that need to be sent |
| Vehicle | Class state   * Vehicles’ ID * The registration details for vehicle * Scan of the vehicles insurance * Pollution test results * QRcode associated with vehicle   Class behavior   * Remind supervisor near expiry of documents * Associate a driver with a vehicle |
| Trip | Class state   * List of delivery boys’ ID * Driver’s ID * Vehicles’ ID * Associated Route   Class behavior   * Deliver Bottle to customer |
| Delivery | Class state   * Custormer’s ID * Number of Filled cans delivered * Number of Damaged cans received * Number of Empty cans received   Class behavior   * Confirm Delivery * Reject/Disapprove delivery |
| Route | Class state   * Customer for each delivery point   Class behavior   * Assign to trip |
| Inventory | Class state   * Number of Filled cans * Number of Empty cans * Number of Damaged cans   Class behavior   * Get inventory info |

.

# 

# 

# 

# 

# **Sequence Diagram(s)**

# 

# **Design Rationale**

**Mobile App:**

Our main concern while thinking about the design of the Mobile app was simplicity, as this app is meant to be used by the delivery personnel who may not necessarily be well versed with using complex applications. This meant that we tried to reduce the learning curve of interacting with the UI as smooth as possible. To this end, we gave simplicity priority over features. With this in mind, the layout was designed to be as minimalistic as possible.

**Customer Identification:** Initially we had considered logging deliveries by having the user enter the details after each delivery into the app. Our client suggested that it would be faster and simpler if we could accomplish the same function by scanning a QR code. This idea has been incorporated into the design

**Transaction Details:** Initially a text box to input log details was floated , but the likelihood of typos was high, and a scroll down menu was the next option, but the options inside a dropdown would be small in size , and would be liable to misclicks. Hence a dial to input number for logging was decided.

**Navigation:** A sidebar for navigation was always decided as the primary mode of navigation, as most users would be already familiar with this method.

**Login:** Options for passwords were considered, but on discussion with client, a pin entry was decided as it is quicker to input and easier to remember.

**Build options for app:**

Initial options considered were

* Flutter for its drag and drop approach to app creation and high performance specs,
* React-Native, due to its lightweight nature, and extensive support and documentation
* and Kotlin due to its popularity among android developers

Kotlin was eliminated as we aimed to build a cross platform app, and its focus on android apps was undesirable.

While flutter provided a cross platform option, it’s coding style was not focussed on modularity, and considering the short time frame of this project, and future plans to add more functionality by outside developers, this would be a hindrance.

**React-native**, provided a coding style already known to us through the DASS course, provided modularity, was cross platform, and had great support API’s and libraries available , and is well maintained by FB devs. **Hence it seemed like the best option considering future developments**

**Web-App:**

**Regular Features:**

1.) Login/Register Company

2.) Create all types of profiles

3.) Login for each type of profile

4.) For the Supervisor, we have this API which will let him design the daily routes that are to be followed by the delivery team.

5.) Supervisor, Manager and the Company Owner can view the bills (may subject to change)

6.) Delivery Team’s attendance is captured, conflicts are also resolved

7.) Get Information about the Stock Point’s inventory

**Build Options:**

1.) MERN – Stack : MongoDb, ExpressJs, ReactJs and NodeJs.

NodeJs, MongoDb, ExpressJs to write Backend

ReactJs to write frontend

