About me

Gourav Agrawal

Native Place : Jabalpur, M.P

Current Location : Bangalore

Graduation : B.E. Electronics & Communication, RGPV, Bhopal, India, Year-2007

Experience : 15 Years of experience in IT industry as full stack developer
Investment bank like SocGen, Wells Fargo
Automation industry, Honeywell

Current Org : JioMart as full stack developer and an architect

Responsibility : Leading and ownership of order management flow
70-80 % time on coding and testing as IC using Java and React JS
Architecture design & POCs
Guiding Junior team members, code review, code merge

Tech Stack : Java, Spring Boot, Microservices, REST API
React JS, Redux, Axios
Radis Cache, Kafka, TIBCO, MySQL/MongoDB

Certification : Certified Solution Architect Expert in Azure Cloud Platform
Six Sigme Green Belt

Recent major accomplishments

Architeture desig and implimention of post order processing module

Tech used –

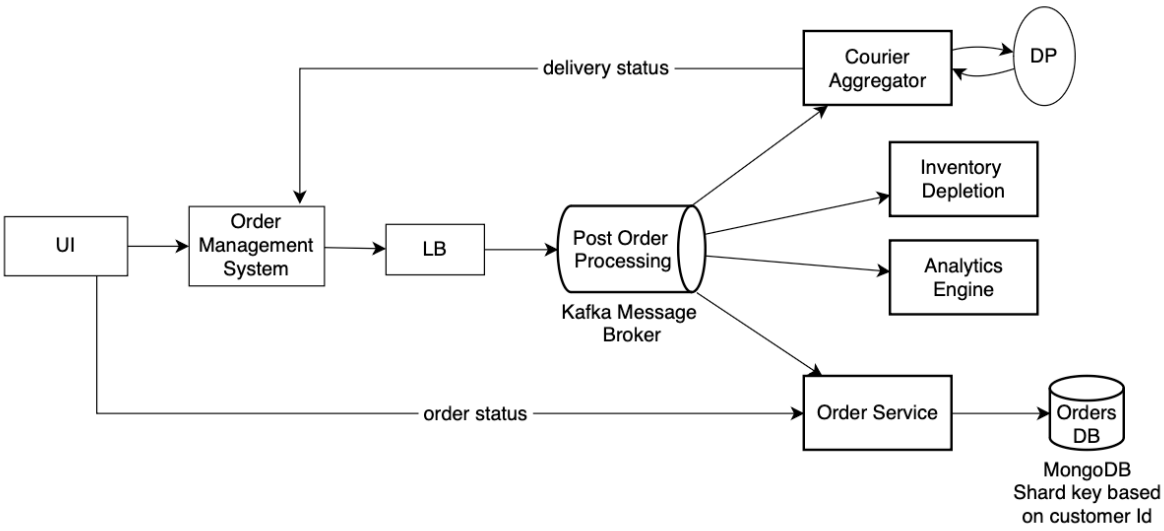
Java spring boot, Kafka, Mongo DB, Azure cloud

Problem statement –

Once the order has been created successfully, there are multiple sub process that needs to be handled like sending the shipment details to courier partner for delivery, tracking, analytics engine and inventory depletion etc.

Solution –

Decouple multiple process from bulky monolith service into separate components and scalaing, managing like feature enhancements, bug fixes of those components independently



Craft Demonstration

Rent A Bike App –

At the start of the application user can see neary by available list of pickup zones(Mobility Centers) within 1 km of radius. User's mobile device should have Bluetooth option and location access enabled in app settings. Location access will be used to identity user's current latitude/longitude to get the list of nearest pickup zones and Bluetooth wil be used to communicate with the bike to lock/unlock the vehicle. Mobility zones are completely automated and does not required human intervention.

User would visit the nearest zone and scan or enter vehicle Id, to unlock the vehicle and start the ride. Once user reached the drop zone, he/she can stop the ride and app would generate the bill based on the distance travelled and time elapsed.

Functional Requirements

Rent A Bike app supports following functionality -

1. User can search and select nearest pickup zone based on the current location
2. Enter destination address, search and select drop zone near to the destination address
3. User can see the ride ETA, estimated fare, navigation option for pickup/drop location, booking and cancellation option
4. Scan or enter vehicle Id at the pickup center to rent a bike and start the ride via bluetooth communication between mobile app and bike
5. End ride at the selected drop zone, and generate the receipt for the actual distance travelled and time elapsed
6. Bikes and Zones to be added in the inventory and its mapping via Admin panel from operation side.

Extended Requirement

1. Bike catalog selection option - premium, EV bike
2. Advance booking, weekly/monthly subcription

Assumptions

1. Bikes can be rented on demand in real time. Advance booking and reservation options are not considered
2. Only one type of petrol vehicle is available for rent and are at the same price. Catalogue selection has not been considered so users would not have the option to select the type of vehicle like Honda activa, Yamaha FZ bike etc.
3. Booking option is available in app at the pickup center. User can see in app how many vehicle are available at nearest pickup center
4. User has already registered, the license and KYC document are verified and the user has sufficient threshold balance in the specific mode of payment like e-wallet etc

Non-Functional Requirements

1. Should be extandable to other cities and to the different geographical locations

   Distributed system across multiple nodes to handle user requests.
   Database storage needs to be partitioned based on the userId as shard key, consistent hashing is used for sharding algorithm.

   Georeduntant replicas would be created to multiple data centers across the geographical location, so that network travel time could be minimized.

2. **Fault tolerant system** by multiple replicas are created with DR environment to handle the fault tolerant with five 9's in azure cloud platform.

   Rate limiting is configured in Azure API gateway will be configured to avoid DDos attack intentioinal or unintentional with flexible sliding window algorithm for rate limiting.

3. **CAP** – for partition tolerance, availability is considered over consistency
   User should be able to see near by zones with best performace, so eventual consistency is considered instead of strong consistency

4. Back of the envelop calculations - considering extendabiltiy in future across multiple cities and globe following are the rough estimates and future plans to scale the system –

   If there are 10 million DAU with 1/10 ratio - 1 user book the ride out of 10 user use the app to check the availability -

   Performance – Azure API Management service is used for measuring different matrices and KPIs

   Read request around = 100 RPS
   Write request around = 10 RPS
   Throughput 100 ms
   Latency 10 ms

   In actual production considering 60% of the CPU threshold will be used with multiple replicas, we would need 100's of servers to handle each request efficiently

   Storage requirement – Systems looks to be read heavy for user journey with the usage of the App but in actual we would need to track each ride travel time and distance coordinates to calculate the fare and ride history.

   Different types of storage and capacity requirement
   - User profile information, booking etc – MySQL used for maintain ACID property
   - Search near by pickup zone - MongoDB used which supports Geospatial querries
   - Tracing travel distance – Cassandra DB used for writing in O(1)

   We would need high storage in PBs for Cassandra D, so archieving every year between the following storage would make sense to save the cost
   Hot Storage
   Cold Storage

5. Security – API gateway azure service would be used to authenticate the user with 2-factor authentication via OTP. User request would be redirected to Authentication server and once the request if authorized it will be redirected to the resource server. JWT token would be used for the subsequest request

6. Leverage from existing 'where is my cab' system

User profile module should be used with SSO, existing customers are onboarded without signon or registration

Destination address search module can be reused

For near by pickup zone functionality, existing system based on lat/lng can be reused
Vehicle tracking system should be reused

7. Needs to handle the usecase scenarios when real-time location info are not accurate

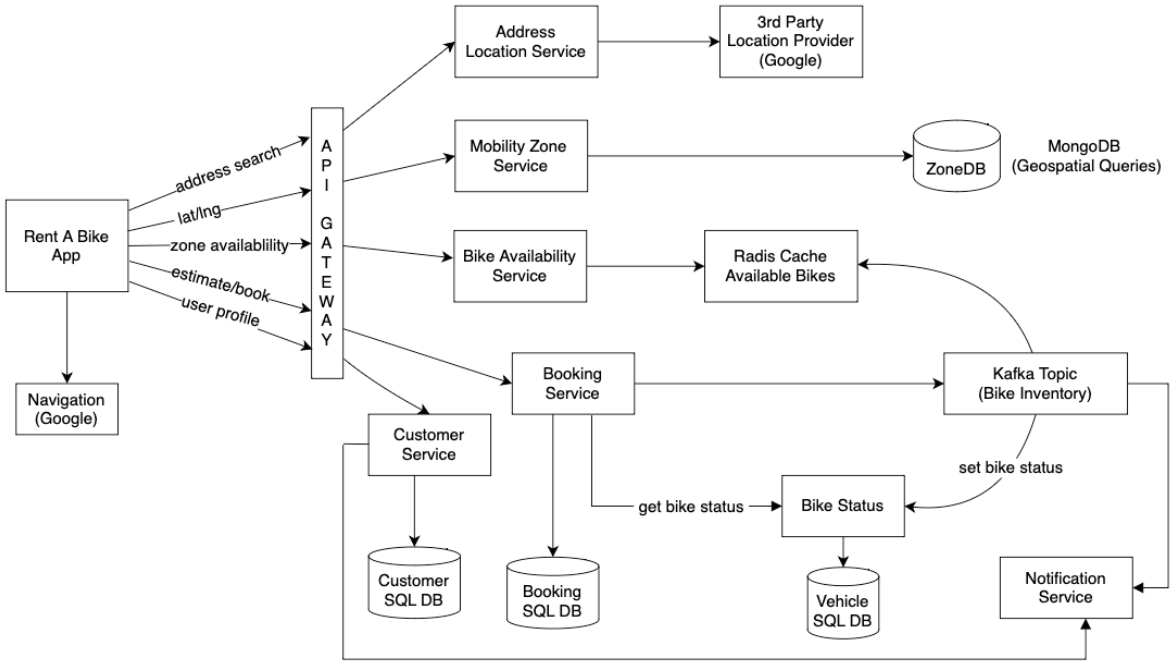
   For customer booking and end journey system is not dependent on the real-time location information.

   To avoid real-time accuracy issue - ride needs to be booked in app via mobile internet and Bluetooth communication with the bike at the zone, and Start/End ride at the pickup/drop zone will be used for bike availability at nearest pickup zones

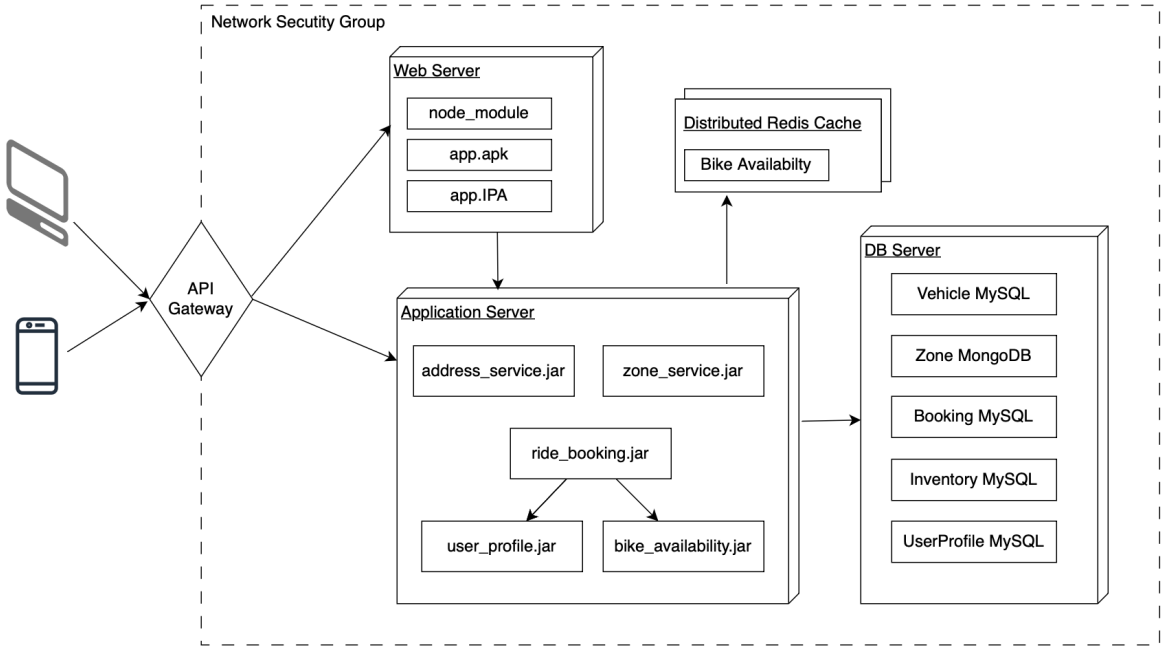
8. Should be able to perform A/B testing with faster results

   Parallel production environment will be used with 25% of traffic redirected for A/B testing for the set of users based on the customer Id hashing algorigthm

   Using analytics engine in Admin panel, product team can view the user conversion rate and take the business decision accordingly



Architecture Diagram



High Level Deployment Diagram

Schema Design

User

user_id	phone_number	address	kyc_status	kyc_id	status
123	900000789	BTM, Banglaore	InProgres	12300XYZ	active
456	812345001	Jaynagar, Banglaore	Completed	45067ABC	disabled

Zone (mobility center)

id	Address	latitude	longitude	status
1	mobility center, #103, bangalore	6.101213	7.101123	active
2	mobility center, #512, bangalore	6.101213	7.101123	inactive

Bike

Id	registration_number	status	manufacturing_date
1	KA01-ET1024	Available	12/02/18
2	KA56-AK0102	Not Available	25/05/19

Inventory

zone_id	Available Quantity
1	78
2	0

Booking

id	customer_id	bike_id	status	pickup_center_id	drop_center_id	charges
123	5	3	InProgress	1	2	200
456	7	2	Completed	3	1	80

REST API endpoints

GET /mobility/rental/address/search?text={searchText}

GET /mobility/rental/zones/nearBy?lat={userLatitude}&lng={userLongitude}

GET /mobility/rental/vechile/vehicleDetail?id={vehicleId}

GET /mobility/rental/ride/rideDetail?id={rideId}

GET /mobility/rental/ride/estimates?pickup={centerId}&drop={centerId}

POST /mobility/rental/ride/start

POST /mobility/rental/ride/end

POST /mobility/rental/ride/pause

POST /mobility/rental/ride/resume

POST /mobility/rental/ride/update

Pickup and Drop module demo and code walkthrough

Select pickup and drop mobility zone to rent a bike

Search near by pickup mobility zone based on user's current location latitude/longitude and select one of the desired pickup zone
Search near by destination address drop mobility zones and select one of the desired drop zone.
User can change pickup and zone until booking is confirmed.

Tech stack used

Backend module –
Java 17
Spring boot
Junit5
Mockito
Mongo DB
Maven to manage dependency

Frontend module –
React JS
Bootstrap
Axios

Find destination address suggestions and get the latitude/longitude of the selected address

Google search address API
- Auto Complete – consider lat/lng bounds
- SearchBox – particular country

Find near by zone based on the user's latitude/longitude and zones latitude/longitude with 1 KM of radius using geospatial queries of Mongo DB

Places entity in Mongo DB

```
db.places.insertMany( [
   {
      name: "ecospace zone 1",
      address: "outer ring road, bangalore, karnataka",
      location: { type: "Point", coordinates: [ -73.97, 40.77 ] },
      category: "zone"
   },
   {
      name: "ecospace zone 2",
      address: "outer ring road, bangalore, karnataka",
      location: { type: "Point", coordinates: [ -73.9928, 40.7193 ] },
      category: "zone"
   },
   {
      name: "silkboard zone",
      address: "silkboard, bangalore, karnataka",
      location: { type: "Point", coordinates: [ -73.9375, 40.8303 ] },
      category: "zone"
   }
] )
```

Indexing on places entity

```
db.places.createIndex( { location: "2dsphere" } )
```

Search near by zone query based on lat/lng and radius within 1 km

```
db.places.find(
   {
     location:
       { $near:
          {
            $geometry: { type: "Point",  coordinates: [ -73.9667, 40.78 ] },
            $minDistance: 0,
            $maxDistance: 1000
          }
       }
   }
)
```

GET /mobility/rental/address/search?text=abc

Response

```
{
        zones : [
                {
                        name : 'ABC'
                        complete_address : 'XYZ',
                        latitude : 6.123456,
                        longitude : 7.123456

                },
                {
                        name : 'ABC'
                        complete_address : 'XYZ',
                        latitude : 6.123456,
                        longitude : 7.123456

                },
        ]
}
```

GET /mobility/rental/zones/nearBy?lat={latitude}&lng={longitude}

Response

```
{
        selected_zone_id : 231,
        zones : [
                {
                        zone_id : 231,
                        name : 'ABC'
                        complete_address : 'XYZ',
                        latitude : 6.123456,
                        longitude : 7.123456
                        vechile_count : 8,
                        available_always : true,
                        start_time : 00:00:00,
                        end_time : 24:00:00,

                },
                {
                        zone_id : 231,
                        name : 'ABC'
                        complete_address : 'XYZ',
                        latitude : 6.123456,
                        longitude : 7.123456
                        vechile_count : 8,
                        available_always : true,
                        start_time : 00:00:00,
                        end_time : 24:00:00,

                },

        ]
}
```

Engineering Excellence

Logging (Splunk)
Monitoring and Health check (Azure)
GIT - source code version control, code review, code merge, CI/CD pipelines
Code coverage of unit test cases and memory leaks
Documentations – architecture diagram, use cases flow diagram
KPIs and PT testing

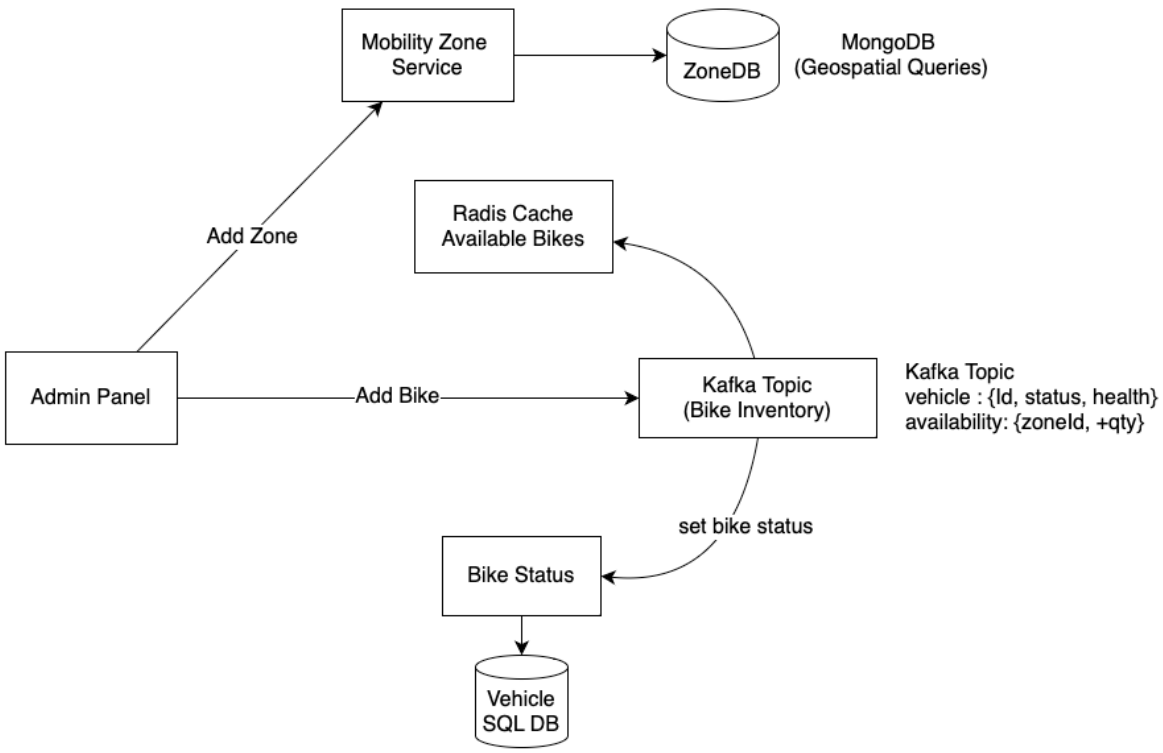
Operational Excellence

Management of vehicle – adding and removing bikes in the vehicle repository via Admin planel user interface

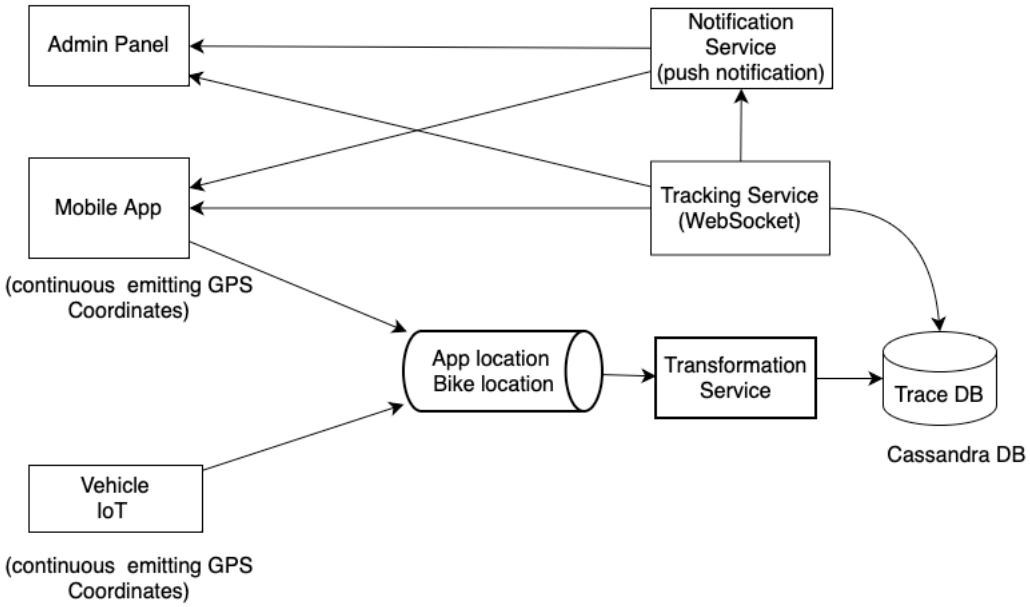
Management of zones – Adding and removing zone in the zone DB via Admin panel

Bikes availability management – whenever new bike is added or removed it needs to be in sync with the zone and bike availability or rather bike zone mapping needs to be in sync via Admin panel

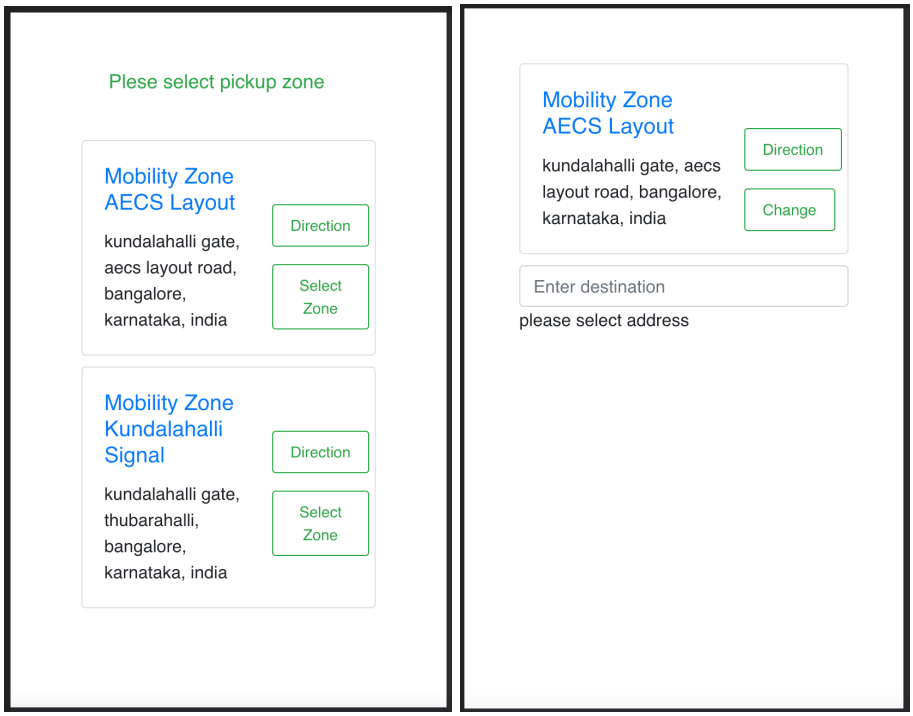
Vehicle tracking system – Bikes should be GPS enabled and current location location of the bikes should be pushed to the central DB. In case of bike breakdown, lost, theft scenarios each and every vehile should be able to track via Admin panel

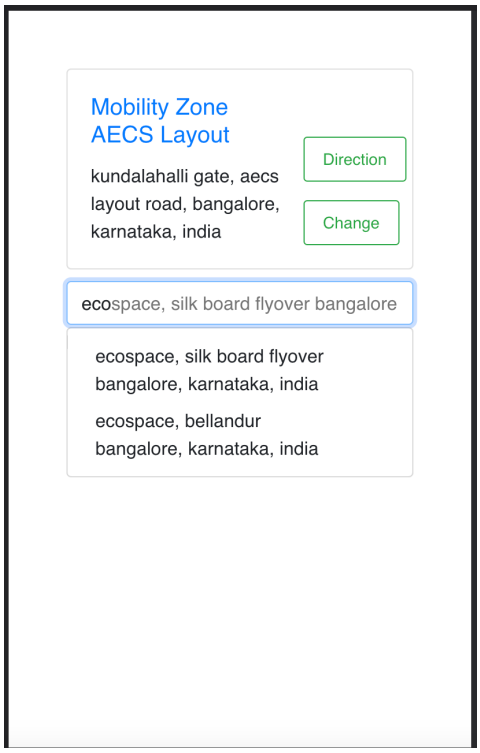


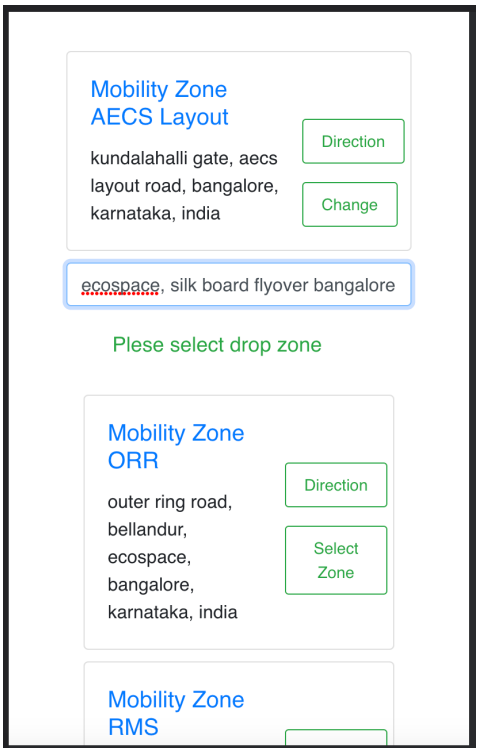
Admin Panel Architecture Diagram

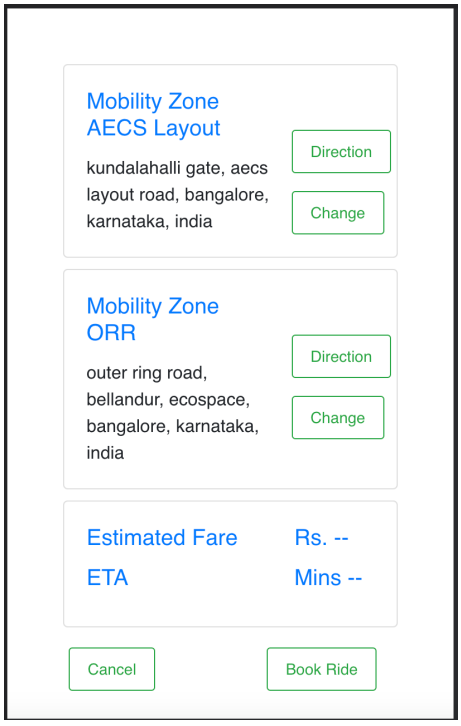


Vehicle Tracing System Architecture Diagram




Mobility Zone
AECS Layout
kundalahalli gate, aecs layout road, bangalore, karnataka, india
Direction
Change
ecospace, silk board flyover bangalore
ecospace, silk board flyover bangalore, karnataka, india
ecospace, bellandur bangalore, karnataka, india


Mobility Zone
AECS Layout
kundalahalli gate, aecs layout road, bangalore, karnataka, india
Direction
Change
ecospace, silk board flyover bangalore
Plese select drop zone
Mobility Zone
ORR
outer ring road, bellandur, ecospace, bangalore, karnataka, india
Direction
Select Zone
Mobility Zone
RMS


Mobility Zone
AECS Layout
kundalahalli gate, aecs layout road, bangalore, karnataka, india
Direction
Change
Mobility Zone
ORR
outer ring road, bellandur, ecospace, bangalore, karnataka, india
Direction
Change
Estimated Fare
Rs. --
ETA
Mins --
Cancel
Book Ride