Electric Freight Booking System

Software Development Ballpark Proposal

Fixed Cost: INR 9,50,000 + GST
Timeframe: 3 months

A detailed software requirement spec and UI/UX will be defined before the start of coding, to ensure the team delivers a product exactly as expected.

Technical Specifications

1. A framework will be developed and made open source to facilitate freight bookings
2. Framework offers a ready facility to on-board, integrate & operate a fleet of electric delivery bikes
3. A web based administration and management dashboard is made available to the admins, for real-time monitoring of electric bikes. Includes location, and available range monitoring.
4. A mobile app will be designed and developed for riders, wherein they can see the pickup and drop locations.
5. Drivers must use the mobile app only to receive bookings. One cannot do so using the web interface.
6. The mobile application will be developed using React Native framework and published on both Android and iOS stores. React Native is a build once deploy anywhere framework, so there is no additional cost for developing iOS, even though the primary user base is likely to be on Android.
7. The application will be open sourced on Github, with clear and easy to follow build and run instructions.
8. Ready integrations will be in place for services like Google Maps which are integral to the whole operation. The users of the system will have to buy their license key and plug it in.

Key Application Features

1. Individual drivers / electric vehicle freight owners can register their vehicle on the portal by using the android / iOS mobile application.
2. Registration of a vehicle will require them to upload proper KYC documents, and approval by an admin of the KYC. The document submission can be done online via the app, but all approvals are manually reviewed by authorized system admins.

3. Once on-boarded, the driver must decide whether to keep their status as active or offline. The driver will receive a booking only if the status as set in the application is active.
4. The live location of the driver will be continuously captured when status is set to active.
5. The driver will receive only those bookings that are nearby for the start location and for which the driver has enough charge to reach the destination location.
6. The system works on a just in-time booking system basis. As soon as a booking request is received, the system notifies potential drivers and requests them to accept the booking.
7. The bookings are placed via an API integration, or manually by the administrator using the web administration interface.
8. The buyer mentions a starting bid and the maximum he is willing to pay for the booking. The system conducts an automated bidding amongst the available riders and appoints a rider that accepts the lowest fare.
9. The option to accept / decline a booking will only be flashed on screens of drivers that have sufficient charge to complete the trip.
10. Once a booking is selected, the driver is required to immediately reach the origin point. The driver will be able to view a Google Maps route on the app to reach the origin point.
11. Once the driver reaches the origin, he can pick up the parcel and start the trip. He will then see the exact destination location.
12. Once the trip is started, the driver will then see a real-time route to the destination. The driver is expected to keep the app powered on using the application itself for routing to destination.
13. For any integrated vehicles, the driver can use the in-bike system (for Android based in-bike screens), to view the route, and view and accept / decline the booking. This feature can be used on Android Auto supported bikes, and on bikes that have directly integrated with the system.
14. The application will provide a unique web based real-time updated tracking link, where the end customer can view the real-time location of the rider. Location can be displayed with Google Maps view (like Swiggy), or as an ETA value without Map view (like Zepto).
15. The real-time location of the rider can be updated via the app, or via an integration with the vehicle's real-time location.
16. Billing reports per rider are shown to the admins. The admins must settle the bill of the rider offline and manually mark the settlement in the system for ease of tracking.

Potential Users and Use Cases

Bike rental companies like Yulu can spin up this ready framework, integrate with their bikes, and make a last mile electric delivery service available to the likes of home-bakers, restaurants and eCommerce stores. It would be expected that such companies ship a pre-installed copy of the rider app on the in-bike computer of the bike.

Restaurant chains that have multiple outlets, like Foo, Socials, Smoke House Deli etc, can use the system to manage their own fleet of electric delivery scooters, with all their outlets within a single city being integrated together to manage quick delivery for their customers.

Chemist shops that have a laptop within the store premises, can use the system to manage their own fleet of electric bikes for self fulfillment of orders.

With the system, individual restaurants can offer electric self-fulfillment for orders received from Swiggy / Zomato, as is done by Dominos. A Dominos order placed on Swiggy for example is delivered by Dominos and not Swiggy. Other restaurant chains can offer similar self-delivery options on electric bikes, presented as a choice to the end customer on Swiggy / Zomato.

Open APIs and Integrations

The system will offer open integrations to electric bike manufacturers. Electric bike companies such as Revolt, Oben, Odysse, Ola etc can build and publish plug-and-play integrations with the system for sharing real-time mileage and location information. Such an integration can make the electric bike very attractive for delivery agents. The onboard computer of such bikes, which is typically android based, can run an in-bike application (pre-installed rider app) to show orders and routing information, making it super safe and convenient for delivery personnel to route to their next location.

Public docs will be released on readme.io, along with release of a Postman API collection. The Postman collection allows for ready integration code to be generated in several common programming languages such as Python, Java, NodeJS etc, which makes for easy integration.

The rider app will be open-sourced, allowing bike manufacturers to customize and pre-install the apps on their vehicles.

General Terms

1. A detailed Software Requirement Specifications will be defined before the start of coding. The SRS documentation phase is likely to take 15 days.
2. The cost of a UX designer is included within the quote. The UI/UX will be finalized within the first month itself.
3. The functional architecture of the software will be completed within the first month.
4. Once SRS and UX are finalized, the coding work will be started. Any changes in SRS or UX post start of coding may result in changes in price and timelines.
5. Currently quote is provided based on high level understanding of requirements. The SRS phase will determine whether the quote is appropriate for the SRS and UX that is agreed upon in writing by the end of month 1.

Rough Timeline

Phase	M1	M2	M3	M4
SRS documentation				
UI / UX				
Software & API Architecture				
Software Development				
Public Docs				
Testing, Demo & Open Source Release				