

# QuickCab - PROJECT REPORT

## 1. INTRODUCTION

### 1.1 Project Overview

**QuickCab** is a full-stack cab booking web application developed using the **MERN stack** (MongoDB, Express.js, React.js, and Node.js). It delivers a seamless and responsive platform that enables users to book cabs efficiently while offering administrators the tools to manage ride operations, drivers, and customer data in real time.

The application is designed with a strong emphasis on **usability, scalability, and performance**. It provides a clean, mobile-responsive UI for users and a powerful backend interface for admins. With real-time updates, secure authentication, and intelligent booking logic, QuickCab aims to make cab services more accessible and reliable.

The platform incorporates the following key functionalities:

- **User Registration and Authentication:** Secure sign-up/login with role-based access (user/admin)
- **Cab Booking System:** Pickup and drop location selection, fare estimation, and booking confirmation
- **Real-Time Ride Tracking:** Users can track assigned cabs and get live ETAs using integrated maps
- **Admin Dashboard:** Manage drivers, view live rides, monitor user activity, and oversee system analytics
- **Ride History Module:** Stores past ride data with filters and export options for better user and admin insights
- **Responsive Design:** Accessible across desktop, tablet, and mobile devices

QuickCab's scalable design allows it to be deployed in small towns or large cities and easily expanded to accommodate additional features like driver assignment, payment gateways, and analytics tools.

## 1.2 Purpose

The primary purpose of QuickCab is to **modernize the cab booking process** by replacing outdated and manual systems with an intuitive, tech-enabled platform that enhances user experience and operational control.

The system addresses key inefficiencies such as lack of real-time communication, booking delays, poor data visibility, and limited ride history. By digitizing the end-to-end workflow, QuickCab makes cab services more **efficient, user-friendly, and transparent** for all stakeholders.

### Specific Goals:

- **Simplify the booking process** through an intuitive UI and quick ride confirmation
- **Enable real-time cab tracking** using integrated mapping solutions
- **Secure user data** through JWT authentication and encrypted communication
- **Empower administrators** with a dashboard to manage rides, users, and drivers
- **Improve customer experience** with features like ride history, notifications, and accurate ETAs
- **Ensure scalability** so that the platform can evolve with new features like driver onboarding, automated fare calculations, and payment integrations
- QuickCab aims to bridge the gap between modern ride expectations and the limitations of traditional cab services, offering a smart solution that adapts to both user and business needs.

## **2. IDEATION PHASE**

### **2.1 Problem Statement**

In many urban and suburban regions, public transportation systems are either inconsistent or insufficient, pushing daily commuters to rely heavily on cab services. However, the current landscape of cab booking—particularly in small cities and developing areas—still involves outdated processes like manual dispatching or phone-based requests, which introduce a range of inefficiencies.

Users face uncertainty around whether a cab is available, when it will arrive, and how much it will cost. Without real-time updates, users are left waiting blindly or making repeated calls to drivers or operators. For administrators, there's often no

centralized platform to monitor ride requests, track drivers, or analyze ride data, which leads to delays, errors, and mismanagement.

These limitations create frustration, erode user trust, and result in lost opportunities for both riders and service providers.

- **Key Problems Identified:**

- Lack of real-time cab availability information
- Inability to track driver location and ride progress
- Manual booking processes prone to errors and delays
- No centralized system for admin oversight and driver coordination
- Limited ride history, analytics, or digital payment support.

	<b>I am (Customer)</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
1	A daily office commuter	Book a cab quickly to reach work on time	Cabs are not always available when needed	There's no live availability or real-time tracking	Frustrated, late, and uncertain
2	A college student	Get an affordable cab for short-distance travel	Fare estimates vary across apps	Most apps show surge pricing and don't offer student-friendly options	Confused and cost-sensitive
3	A working professional	Track the cab after booking	There is no reliable way to know where the driver is	Existing platforms often lack live tracking or send delayed updates	Anxious and helpless
4	A city transport administrator	Monitor ride activity and user requests	No centralized system for ride or driver management	Manual processes cause inefficiencies and data gaps	Overwhelmed and unable to manage effectively
5	A parent booking for a child	Ensure the ride is safe and the driver is trusted	Limited information about	Security features and driver details are not	Concerned and cautious

			the driver before or during the ride	prominently visible	
--	--	--	---	------------------------	--

## 2.2 Empathy Map Canvas

### Think & Feel

Users want their experience with QuickCab to be simple, reliable, and fast. They often think about whether they'll reach their destination on time, if the driver is trustworthy, and whether the price matches their expectations. They care about safety and smooth communication throughout the ride. Concerns about delays, hidden costs, or unresponsive apps make them feel anxious and cautious while booking.

### Hear

Common feedback that users hear from friends, family, or other users includes:

- “Cabs are always booked when I need one.”
- “I had to wait too long and there was no update.”
- “The app doesn't show where the driver is.”

This type of feedback sets a negative expectation, influencing users to approach cab apps with skepticism or hesitation—especially during urgent or high-demand times.

### See

Users see competing platforms that offer modern designs and advanced options yet often fall short in consistency and user experience.

They witness:

- Long wait times displayed on booking screens
- Last-minute cancellations

- Inaccurate route maps or lagging location pins
- Public complaints or negative reviews online

Despite digital advancements, users frequently see a gap between the promise and the actual delivery of services.

## **Say & Do**

Users actively compare apps before confirming a ride. They often say:

- “Let me check the fare on another app first.”
- “This driver has low ratings, I’ll cancel.”
- “It’s taking too long—I’ll book something else.”

Behaviorally, they tend to:

- Check estimated fares and arrival times
- Read driver reviews
- Cancel rides if they're delayed
- Switch to alternatives if booking becomes inconvenient

These actions reveal how critical speed, trust, and transparency are in shaping user decisions.

## **Pain Points**

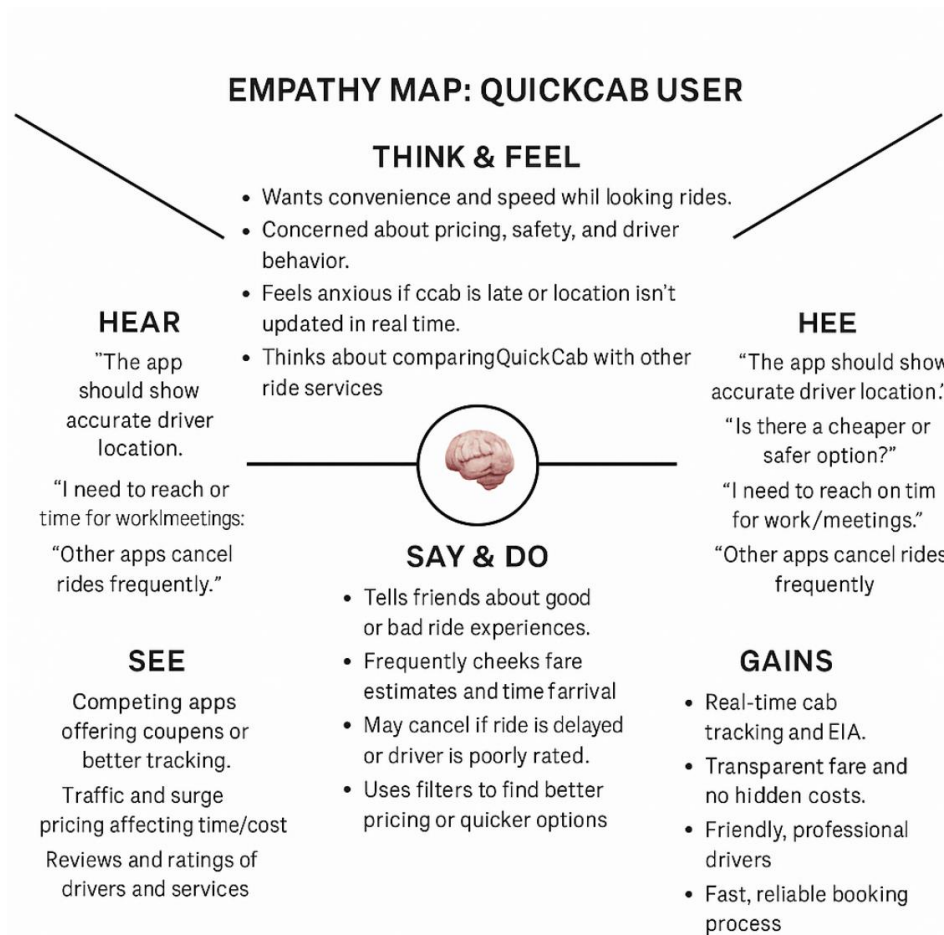
- Failed or delayed bookings, especially in peak hours
- Lack of driver updates or real-time tracking
- Poor or inconsistent fare transparency
- Communication gaps between users and drivers
- No central help or fast response from customer support
- Uncertainty and stress when cabs don't arrive as expected

These pain points lead to frustration, reduced trust, and a higher chance of users abandoning the app altogether.

## Gains

- Reliable, real-time tracking of the driver's location
- Clear and upfront fare estimates—no surprises
- Easy and fast booking process with just a few taps
- Consistent notifications and ride updates
- Access to ride history, receipts, and driver information
- A sense of control and confidence throughout the experience

These gains are what drive loyalty and satisfaction, making users more likely to return to QuickCab for future rides.




## 2.3 Brainstorming

- To identify and implement practical solutions for the core challenges users face while booking cabs, the QuickCab team conducted a structured brainstorming session. The goal was to enhance usability, improve operational efficiency, and build a feature-rich, scalable platform.
- The session was conducted in three phases:

### Step 1: Problem Framing

- The team began by clearly defining the key user and admin pain points, such as booking delays, lack of ride tracking, weak data security, and the absence of centralized ride management. This helped align everyone's focus around solving the most impactful problems.

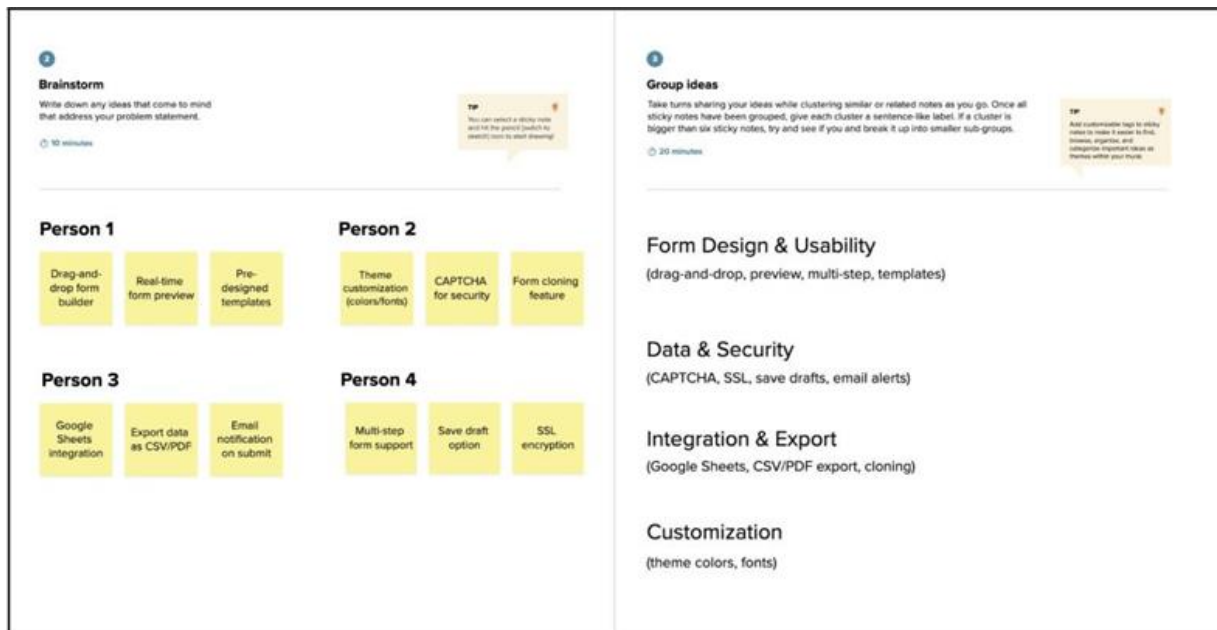
 <h3>Brainstorm &amp; idea prioritization</h3> <p>Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.</p> <p>⌚ 10 minutes to prepare 👥 1 hour to collaborate 👤 2-8 people recommended</p>	<h4>➕ Before you collaborate</h4> <p>A little bit of preparation goes a long way with this session. Here's what you need to do to get going.</p> <p>⌚ 10 minutes</p> <hr/> <p>➡ <b>Team gathering</b> Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.</p> <p>➡ <b>Set the goal</b> Think about the problem you'll be focusing on solving in the brainstorming session.</p> <p>➡ <b>Learn how to use the facilitation tools</b> Use the Facilitation Superpowers to run a happy and productive session.</p> <p><a href="#">Open article</a> ➡</p>	<h4>1 Define your problem statement</h4> <p>What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.</p> <p>⌚ 5 minutes</p> <div><p><b>PROBLEM</b></p><p>How might we make it easier and more secure for users to create and manage online forms through Medley Pharma?</p></div>
---	---	---

### Step 2: Idea Generation and Grouping

- Each team member contributed individual suggestions based on user journeys, competitor analysis, and personal experiences with ride-booking platforms. These ideas were then categorized into the following themes:
- **Booking Flow Optimization:** Ideas to streamline ride requests, improve interface responsiveness, and minimize steps to confirm a ride.



- **Real-Time Ride Visibility:** Suggestions focused on integrating Google Maps, live driver tracking, and ETA updates.
- **User Data Security:** Emphasis on JWT-based authentication and encrypted communication to protect sensitive information.



- **Admin Oversight:** Features such as ride history management, user activity logs, and driver control panels were proposed to ensure effective admin-level operations.

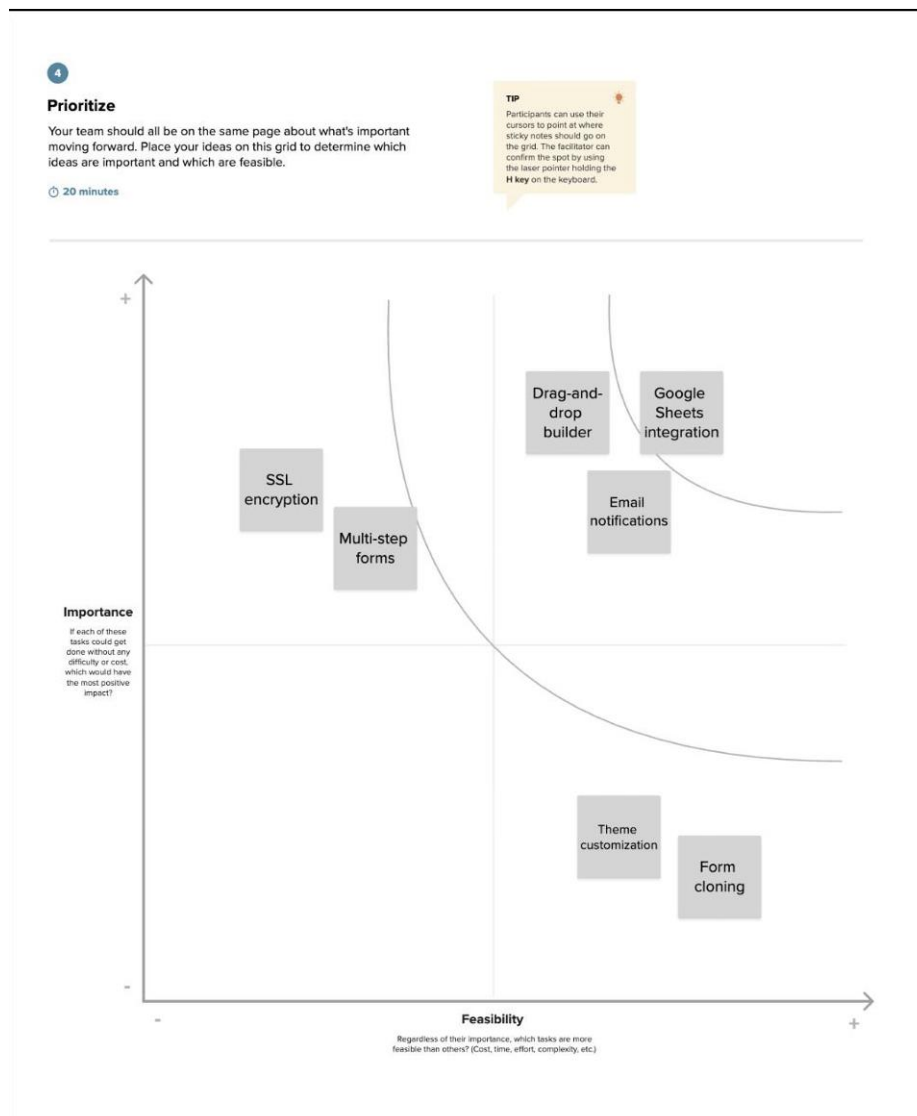
### Step 3: Prioritization of Feasible Features

- After evaluating each idea's feasibility and impact, the team selected a list of high-priority features to implement in the first development cycle.

### Final Chosen Ideas:

- **Map-Based Cab Selection:** Integration with **Google Maps API** to allow users to view available cabs, set pickup/drop points visually, and receive accurate ETAs.
- **Secure Login via JWT:** Implementation of token-based authentication to ensure secure and role-based access for users and admins.

- **Admin Dashboard:** A backend interface for administrators to view all active rides, manage user and driver data, and oversee system analytics.
- **Ride History Module:** A feature allowing users to view, filter, and export their past rides for better tracking and transparency.
- These ideas formed the foundation for QuickCab's MVP (Minimum Viable Product) and will evolve as the application scales.



## 3. REQUIREMENT ANALYSIS

### 3.1 Customer Journey Map

#### Customer Journey Map – QuickCab

Stage / Layer	ENTICE (Become Aware)	ENTER (Explore / Sign Up)	ENGAGE (Book Ride)	EXIT (During/After Ride)	EXTEND (Post-Ride)
Steps	Discover QuickCab via ads, referrals, or search	Sign up or log in through the homepage	Enter pickup/drop location and confirm booking	Ride begins, track driver live, complete trip	View ride history, download invoice, rate driver
Interactions	Friends, social media, app stores	Signup screen, chatbot, onboarding prompts	Driver interaction, fare calculator, route map	In-ride UI, notifications, contact options	History tab, rating popup, email follow-up
Goals	Find a reliable cab option	Easy login and quick access to features	Book quickly with accurate info	Reach safely, track progress	Review experience and get summary
Positive Moments	Attractive offer, quick loading site	Smooth signup, helpful interface	Fast booking, clear fare breakdown	Accurate tracking, timely arrival	Easy invoice access, rewarding feedback
Negative Moments	Unclear value, irrelevant ads	Slow signup, password issues	Driver delays, unclear ETA	No map updates, delayed alerts	No feedback option, poor history visibility
Opportunities	Promote unique features upfront	Offer social sign-in, skip login for trial	Improve fare prediction, better driver info	Enhance tracking and live support	Provide insights, discounts for ratings

## 3.2 Solution Requirement

### Functional Requirements Table

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	<ul style="list-style-type: none"><li>- Registration through Form</li><li>- Registration through Gmail</li><li>- Registration through LinkedIn</li></ul>
2	User Confirmation	<ul style="list-style-type: none"><li>- Confirmation via Email</li><li>- Confirmation via OTP</li></ul>
3	Form Creation	<ul style="list-style-type: none"><li>- Drag-and-drop form builder</li><li>- Add various fields (text, checkbox, dropdown, etc.)</li><li>- Apply validations</li></ul>
4	Form Submission & Storage	<ul style="list-style-type: none"><li>- Store responses securely</li><li>- Support file uploads</li><li>- Submission tracking and limitations</li></ul>
5	Analytics & Reporting	<ul style="list-style-type: none"><li>- Generate charts/graphs from form data</li><li>- Export responses in CSV/Excel</li><li>- Visualize trends</li></ul>
6	Integration & Notifications	<ul style="list-style-type: none"><li>- Connect with third-party tools (Google Sheets, Mailchimp)</li><li>- Set email alerts and automated actions on submission</li></ul>

### Non-Functional Requirements Table

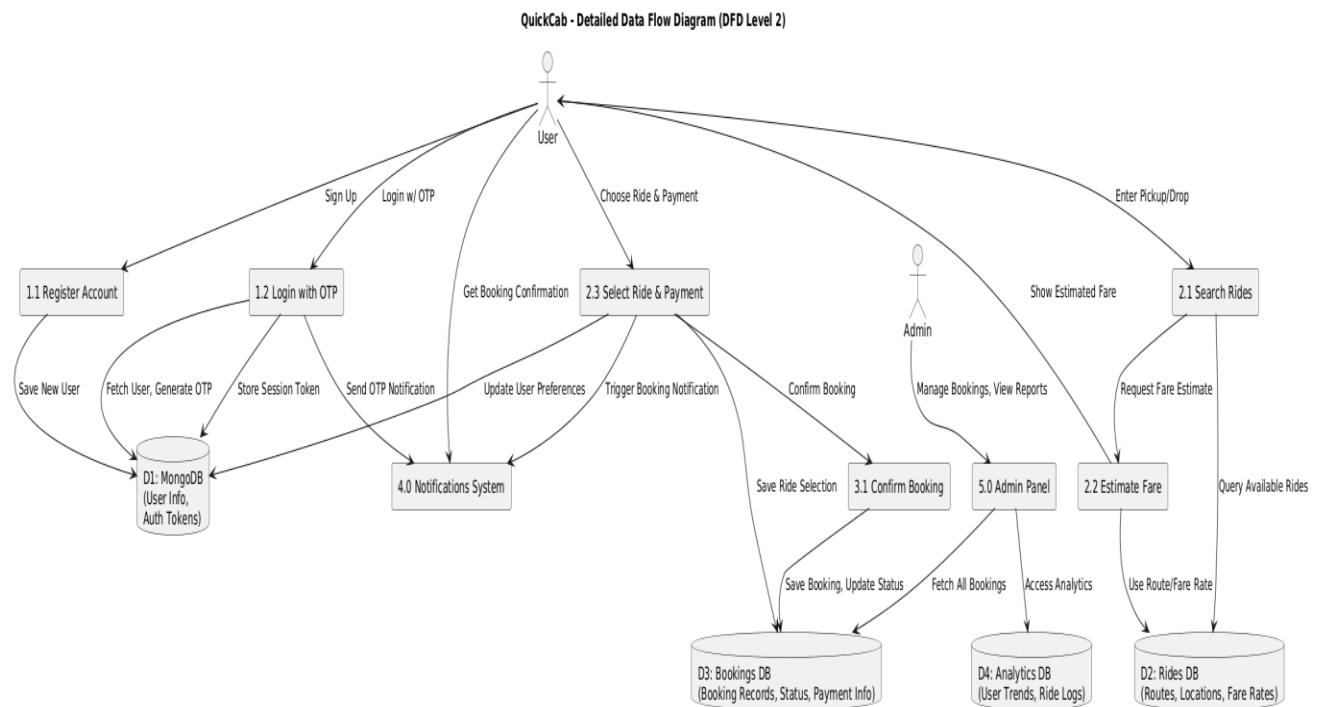
FR No.	Non-Functional Requirement	Description
1	Usability	Intuitive drag-and-drop interface for easy form creation without technical knowledge
2	Security	SSL encryption for data, CAPTCHA to prevent spam, secure authentication methods
3	Reliability	System should ensure zero data loss and smooth submission handling under load
4	Performance	Fast form load time, quick processing of submissions, optimized data storage
5	Availability	24/7 uptime with minimal downtime for maintenance or updates

6	Scalability	Capable of handling large number of users/forms and high submission volumes
---	-------------	---

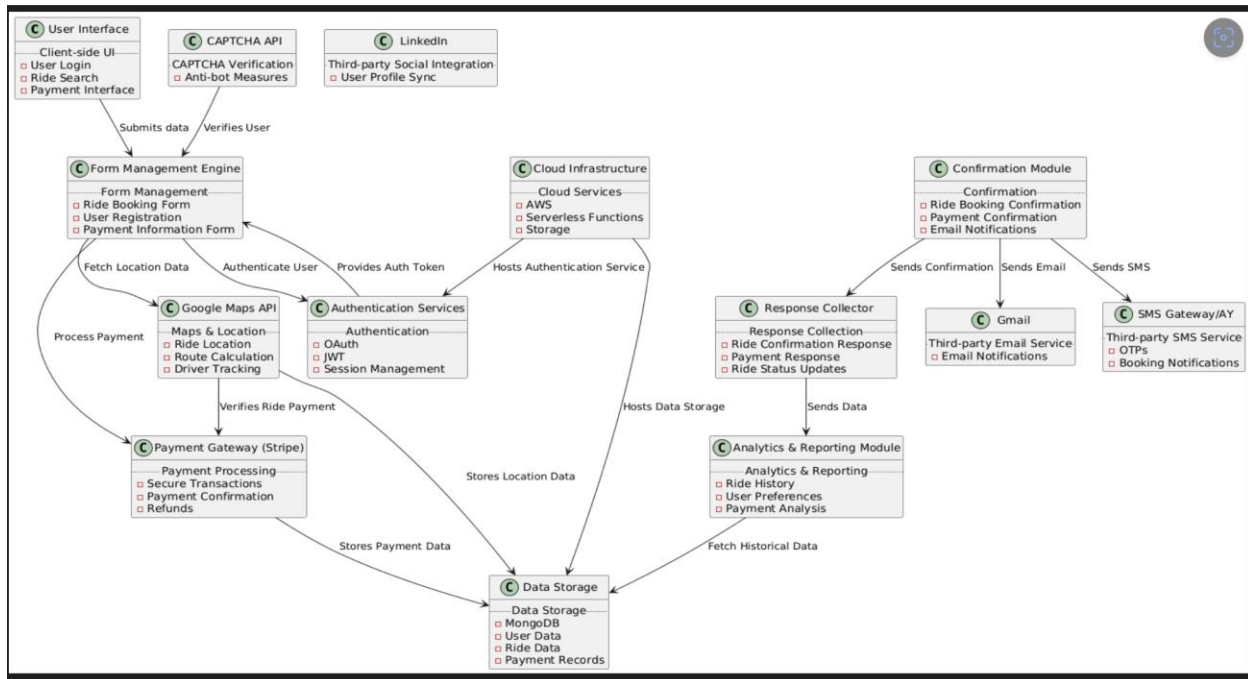
### 3.3 Data Flow Diagram (DFD)

**User → Login/Register → Book Ride → Confirm Ride → Track Ride → View History**

- MongoDB: Store user data, booking records
- Admin: Fetch ride list, manage data, track analytics



### 3.4 Technology Stack



**Table-1: Technical Architecture**

S. No	Component	Description	Technology
1	User Interface	How users interact with the platform to create and manage forms (drag-and-drop UI)	HTML, CSS, JavaScript, ReactJS / Angular
2	Application Logic-1	Handles form creation (drag and drop interface, layout customization)	JavaScript / ReactJS, Node.js
3	Application Logic-2	Manages form submissions, validations, and CAPTCHA integration	Express.js / Node.js, reCAPTCHA API
4	Application Logic-3	Handles analytics, report generation (charts, graphs) from responses	Chart.js / D3.js / Python (Flask or FastAPI for back-end)
5	Database	Stores user profiles, form metadata, and submitted data	MySQL / MongoDB
6	Cloud Database	Cloud-hosted storage for form data for scalability	IBM DB2 / IBM Cloudant / Firebase Realtime DB
7	File Storage	For storing uploaded files from form submissions	IBM Cloud Object Storage / Firebase Storage
8	External API-1	For email notifications on form submissions	SendGrid API / SMTP APIs

9	External API-2	For third-party integration and automation (e.g., Google Sheets, Slack)	Zapier API / Google Sheets API
10	Machine Learning Model	(Optional/Future) Smart suggestions for form field types based on title/content entered	ML Model using Python & scikit-learn (if implemented)
11	Infrastructure (Server/Cloud)	Application hosted on cloud platforms with secure deployment	IBM Cloud / Kubernetes / Docker / Firebase Hosting

**Table-2: Application Characteristics**

S . No	Characteristics	Description	Technology Used
1	Open-Source Frameworks	Uses open-source frameworks for building responsive UI and scalable backend services	ReactJS / Node.js / Express.js / Chart.js / MongoDB
2	Security Implementations	Data security is ensured through SSL encryption for form submissions, CAPTCHA to prevent bots, and role-based access control	SSL, reCAPTCHA, HTTPS, SHA-256 Hashing, JWT (Token Auth)
3	Scalable Architecture	Built with a modular design using a 3-tier architecture (Presentation, Logic, and Data Layer) enabling independent scaling of each layer	Node.js backend, React frontend, MongoDB database, Docker
4	Availability	Ensured via cloud hosting with distributed architecture and use of load balancers for high uptime and fault tolerance	IBM Cloud / Firebase Hosting, Load Balancers, Docker Swarm
5	Performance	Designed for high performance using CDN for static assets, data caching, and optimized queries to support multiple form submissions per second	CDN (Cloudflare), Redis (Caching), MongoDB indexing

## 4 Project Design

### 4.1 Problem-Solution Fit

QuickCab - Problem Solution Fit		
<b>1. Customer Segments (CS)</b> <ul style="list-style-type: none"><li>• Daily commuters</li><li>• Students and working professionals</li><li>• Tourists and travelers</li><li>• Corporate employees</li><li>• Elderly users needing assisted travel</li></ul>	<b>5. Available Solutions (AS)</b> <ul style="list-style-type: none"><li>• Uber, Ola, Rapido</li><li>• Public transport and autos</li><li>• Pros: Established user base, wide driver network</li><li>• Cons: High surge pricing, inconsistent service</li></ul>	<b>6. Customer Constraints (CC)</b> <ul style="list-style-type: none"><li>• Limited internet access in remote areas</li><li>• Budget sensitivity among students</li><li>• App complexity for older users</li><li>• Lack of multilingual support</li></ul>
<b>2. Jobs-to-be-Done / Problems (J&amp;P)</b> <ul style="list-style-type: none"><li>• Need reliable and safe cab services</li><li>• Difficulty finding rides during peak hours</li><li>• Lack of pricing and driver transparency</li><li>• Inconvenient booking experiences</li><li>• Need ride scheduling and real-time tracking</li></ul>		<b>7. Behaviour (BE)</b> <ul style="list-style-type: none"><li>• Booking via phone calls or traditional methods</li><li>• Comparing prices on multiple apps</li><li>• Avoiding digital bookings due to trust issues</li><li>• Using known local drivers</li></ul>
<b>3. Triggers (TR)</b> <ul style="list-style-type: none"><li>• Frustration with surge pricing</li><li>• Need for on-time travel</li><li>• Competitor apps lack support or ease</li><li>• Word-of-mouth recommendations</li></ul>	<b>4. Emotions: Before / After (EM)</b> <ul style="list-style-type: none"><li>• <b>Before:</b> Anxiety, hidden charges, poor UI</li><li>• <b>After:</b> Confidence, satisfaction, trust</li></ul>	<b>8. Channels of Behaviour (CH)</b> <ul style="list-style-type: none"><li>• <b>Online:</b> App, website, maps integration</li><li>• <b>Offline:</b> Kiosks, call centers, help desks</li></ul>
<b>9. Problem Root Cause (RC)</b> <ul style="list-style-type: none"><li>• Lack of region-specific affordable solutions</li><li>• Poor optimization for user trust &amp; transparency</li><li>• Limited real-time updates and scheduling</li></ul>	<b>10. Your Solution (SL)</b> <ul style="list-style-type: none"><li>• Real-time ride matching &amp; tracking</li><li>• Transparent pricing with digital receipts</li><li>• Google Maps integration</li><li>• Secure Stripe-based payments</li><li>• Driver/rider rating system</li><li>• Multilingual UI, voice assist</li><li>• Admin panel for analytics &amp; support</li><li>• Offline booking &amp; trip logging</li></ul>	

QuickCab was meticulously conceptualized to alleviate urban transportation inefficiencies, particularly those prevalent in underserved or congested metropolitan environments. The platform was designed to identify, analyze, and ultimately remedy the prevalent gaps in traditional cab booking systems—such as lack of real-time ride visibility, ambiguous ride statuses, and inefficient communication between users, drivers, and administrators.

QuickCab strategically streamlines the entire cab booking lifecycle by offering an intuitive, data-driven, and responsive digital experience. It addresses customer pain points by synthesizing advanced frontend interactivity with robust backend logic, ensuring seamless ride booking, tracking, and management.



The application implements and orchestrates a real-time, map-based cab discovery mechanism that minimizes booking delays and eliminates manual dispatch errors. Through the integration of Google Maps API and secure JSON Web Token (JWT) authentication, QuickCab empowers users to interact with a transparent, trustworthy, and location-aware system.

Furthermore, it enables administrators to oversee all operations from a centralized dashboard, where they can monitor driver statuses, manage user data, and optimize route logistics. These capabilities not only enhance the user experience but also foster operational efficiency, scalability, and safety.

In essence, QuickCab does not merely digitize an existing service—it revitalizes the urban cab dispatch paradigm by formulating a modern, mobile-first, and user-centric ride booking platform. It empowers stakeholders, facilitates quick decision-making, reduces uncertainty, and enhances service reliability across the board.

## 4.2 Proposed Solution – QuickCab

S.No	Parameter	Description
1	<b>Problem Statement (Problem to be solved)</b>	Users often face unreliable ride services, hidden pricing, and lack of user-friendly interfaces. There's also a need for real-time tracking, secure payments, and regional customizations in ride-booking platforms.
2	<b>Idea / Solution Description</b>	QuickCab is a MERN stack-based smart cab booking platform offering an intuitive interface, real-time driver tracking, transparent fare calculation, Stripe-powered payments, multilingual support, and Google Maps integration.

3	<b>Novelty / Uniqueness</b>	QuickCab goes beyond standard apps by integrating region-specific fare logic, complete transparency, built-in analytics, ride history, and offline booking options. It is designed with a modular architecture for future enhancements.
4	<b>Social Impact / Customer Satisfaction</b>	The platform empowers users from all walks of life, including elderly and rural users, with reliable transport access. By eliminating pricing confusion and tech barriers, it improves satisfaction and promotes digital inclusivity.
5	<b>Business Model (Revenue Model)</b>	QuickCab can adopt a Freemium model—offering basic ride services for free with premium features (priority booking, driver selection, etc.) under subscription. Additional revenue channels include in-app ads, enterprise tie-ups, and commissions from drivers.
6	<b>Scalability of the Solution</b>	With cloud-based deployment and a scalable MongoDB backend, QuickCab can efficiently manage thousands of concurrent users and drivers. Its modular design supports easy scaling across multiple cities or regions.

## 4.3 Solution Architecture

**QuickCab** is an intelligent, real-time cab booking platform developed using the **MERN (MongoDB, Express.js, React.js, Node.js)** stack. Its architecture is crafted to bridge the gap between modern transportation needs and technology by offering a secure, scalable, and user-centric ride-booking experience.

## 1. Finding the Best Tech Solution for Business Problems

**QuickCab** addresses key challenges in the cab booking industry such as:

- Lack of transparency in pricing and driver assignment.
- Difficulty in finding reliable rides, especially in peak hours or remote areas.
- Need for real-time tracking, scheduling, and multilingual support.
- Secure and seamless digital payments.

### **MERN Stack Justification:**

- **MongoDB:** NoSQL database structure allows for flexible storage of rides, users, drivers, payments, and ratings.
- **Express.js & Node.js:** Handle robust backend operations, including route matching, user authentication, and fare calculation.
- **React.js:** Enables a fast, interactive frontend for booking rides, tracking drivers, and accessing ride history.

## 2. Describing the Software Architecture to Stakeholders

**QuickCab** adopts a **3-tier architecture**:

- **Presentation Layer (React.js):**
  - Handles user interface and user interactions (booking, map views, profiles).
  - Google Maps integration for live tracking and route visualization.
- **Application Layer (Node.js + Express.js):**
  - Manages user authentication, ride booking logic, fare estimation, and admin dashboards.
  - Communicates with third-party services (e.g., Stripe for payments).
- **Data Layer (MongoDB):**
  - Stores users, drivers, ride details, payments, reviews, and ride history.

### **Additional Features:**

- Stripe-based payment integration.
- Driver and user rating system.
- Ride scheduling and real-time trip tracking.

- Multilingual UI and voice assist for accessibility.

### 3. Defining Features, Development Phases, and Requirements

#### Key Features:

- User and driver authentication & dashboards.
- Real-time cab availability and booking system.
- Admin panel for managing users, drivers, and transactions.
- Notifications (email/SMS) for ride status and updates.
- Secure online payments and digital receipts.

#### Development Phases:

1. **Phase 1** – React.js-based frontend for ride booking and live map.
2. **Phase 2** – Express.js backend with APIs for user/driver modules.
3. **Phase 3** – Integration with Google Maps and Stripe for tracking and payments.
4. **Phase 4** – Ratings, reviews, and scheduling system.
5. **Phase 5** – Deployment, performance optimization, and offline booking support.

#### Requirements:

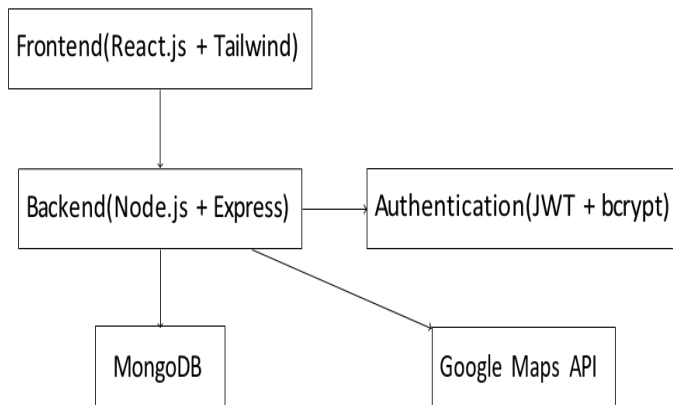
- Cross-platform responsive design (web/mobile).
- Role-based access control (User, Driver, Admin).
- Secure and scalable cloud hosting.
- Integration with mapping and payment APIs.

### 4. Providing Specifications for Managing and Delivering the Solution

**QuickCab** follows modern DevOps practices for reliability and scalability:

- **Version Control:** Git & GitHub for source management and collaboration.
- **CI/CD Pipelines:** For seamless code integration, testing, and deployment.
- **Containerization:** Docker used for consistent deployment across environments.

- **Cloud Hosting:** Deployed on Firebase, AWS, or DigitalOcean with auto-scaling support.
- **Monitoring & Logging:**
  - Real-time logs for ride and payment operations.
  - Analytics dashboards for user activity, driver performance, and system uptime.



## 5 Project Planning & Scheduling

### 5.1 Project planning

#### Product Backlog and Sprint Schedule for QuickCab

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	User Registration & Login	USN-1	As a user, I can register by entering my name, email, and password.	2	High
Sprint-1	User Registration & Login	USN-2	As a user, I receive a welcome email after registration.	1	High
Sprint-1	User Registration & Login	USN-3	As a user, I can log in using email and password.	1	High
Sprint-2	Booking	USN-4	As a user, I can enter pickup and drop locations to search for a ride.	3	High

Sprint-2	Booking	USN-5	As a user, I can schedule a ride in advance.	2	Medium
Sprint-2	Maps Integration	USN-6	As a user, I can view my route on Google Maps.	2	High
Sprint-3	Driver Module	USN-7	As a driver, I can register and log in to the system.	2	Medium
Sprint-3	Driver Module	USN-8	As a driver, I can accept or decline ride requests.	2	High
Sprint-3	Ride Management	USN-9	As a user, I can view my ride status and driver location in real-time.	3	High
Sprint-4	Payments	USN-10	As a user, I can pay securely using Stripe.	3	High
Sprint-4	Ratings & Reviews	USN-11	As a user, I can rate and review a completed ride.	2	Medium
Sprint-4	Notifications	USN-12	As a user/driver, I receive real-time SMS/email notifications.	2	Medium

Project Tracker, Velocity & Burndown Chart

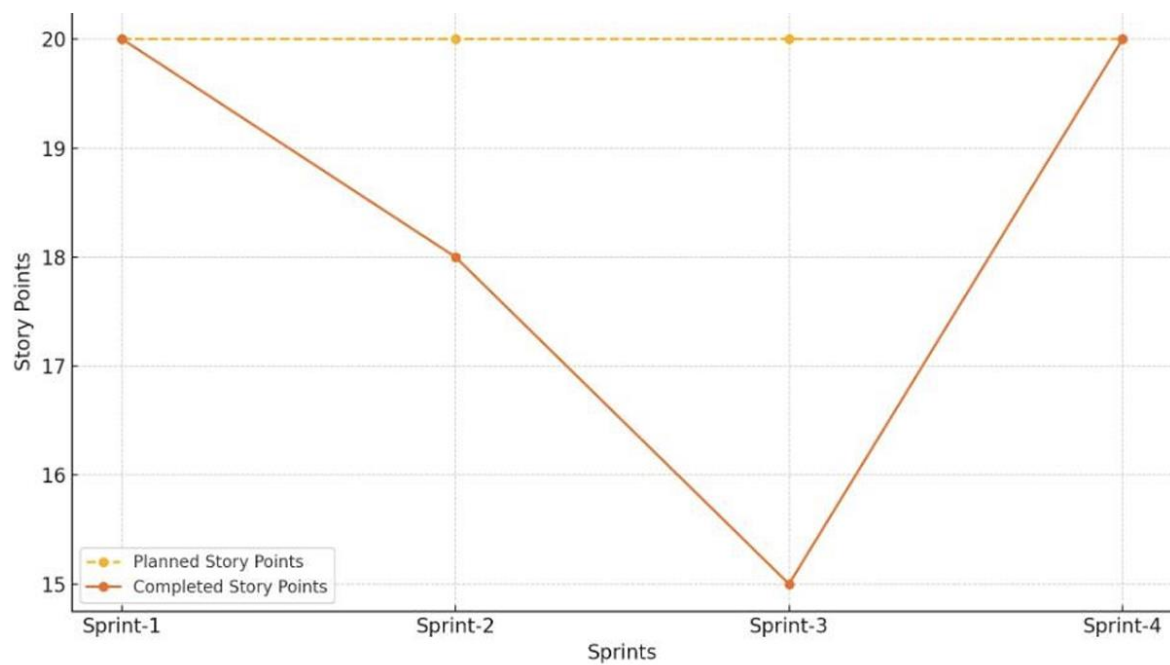
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	01 Apr 2025	06 Apr 2025	20	06 Apr 2025
Sprint-2	20	6 Days	07 Apr 2025	12 Apr 2025	18	13 Apr 2025
Sprint-3	20	6 Days	14 Apr 2025	19 Apr 2025	15	20 Apr 2025
Sprint-4	20	6 Days	21 Apr 2025	26 Apr 2025	20	26 Apr 2025

Velocity Chart (Summary)

Sprint	Story Points Completed
Sprint-1	20

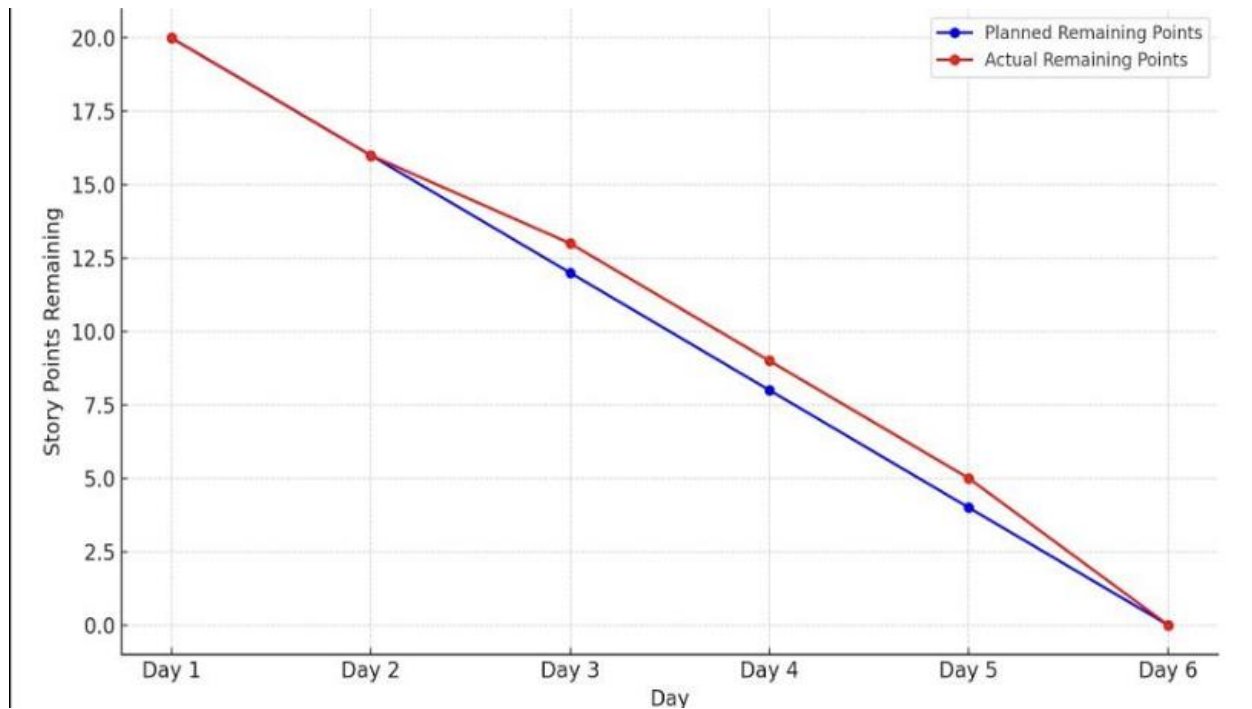
Sprint-2	18
Sprint-3	15
Sprint-4	20

**Average Velocity =  $(20 + 18 + 15 + 20) / 4 = 18.25$  Story Points per Sprint**



### **Burndown Chart Data**

Day	Planned	Actual
Day 1	20	20
Day 2	16	16
Day 3	12	13
Day 4	8	9
Day 5	4	5
Day 6	0	0



## Sprint 1 (5 Days)

**Epic:** *User Registration and Form Builder UI*

Task	Story	Story Points	Complexity
Design registration page (email/password)	Story 1	2	Easy
Implement user registration logic	Story 2	3	Moderate
Develop basic form builder UI (drag & drop)	Story 3	5	Difficult

**Total Story Points (Sprint 1): 10**

## Sprint 2 (5 Days)

**Epic:** *Form Input Features & Security*

Task	Story	Story Points	Complexity
------	-------	--------------	------------



Add support for dropdowns, checkboxes, file uploads	Story 4	3	Moderate
Implement field validation (text, number, required, etc.)	Story 5	2	Easy
Integrate CAPTCHA & SSL encryption	Story 6	5	Difficult
Set submission limits and form access control	Story 7	2	Easy

**Total Story Points (Sprint 2): 12**

## Velocity Calculation

Metric	Value
Total Story Points	$10 + 12 = 22$
Number of Sprints	2
<b>Velocity</b>	<b><math>22 / 2 = 11</math> Story Points per Sprint</b>

# 6. Functional and Performance Testing

## 6.1 Performance Testing

### Overview:

QuickCab is a MERN stack-based cab booking web app that provides a seamless experience for users to book cabs in real-time, with secure login, driver assignment, and ride history.

**Key Features:**

User Authentication: Registration, login, password recovery

Cab Booking: Pickup and drop selection, cab availability, fare estimation

Ride History: View past rides and details

Driver Allocation: Real-time driver assignment

**Technology Stack:**

Frontend: React.js

Backend: Node.js, Express.js

Database: MongoDB

External Services: Google Maps API, Notification API

**Project Version:** Version 1

**Testing Period:** 8/04/25 to 10/04/25

## 6.2 Testing Scope

**Features and Functionalities to be Tested:**

User Registration and Login

Booking a Cab

Viewing Ride History

Driver Assignment

Notifications

**User Stories to be Tested:**

As a user, I want to register and log in

As a user, I want to book a cab with pickup and drop points

As a user, I want to view my past rides

As a user, I want to receive cab assignment notifications

### Testing Environment:

URL: <http://localhost:3000/home>

Credentials: [quickcabuser@gmail.com](mailto:quickcabuser@gmail.com)

## 6.3 Test Cases

Test ID	Scenario	Test Steps	Expected Result	Actual Result	Stat
TC-01	User Registration	Enter details and submit	User is registered successfully	Registration successful	Pass
TC-02	Login	Enter credentials and log in	Redirected to dashboard	Successfully redirected	Pass
TC-03	Book Cab	Enter pickup/drop, confirm	Cab booked and driver assigned	Booking confirmed	Pass
TC-04	Ride History	Navigate to history page	Ride history is displayed	History shown correctly	Pass
TC-05	Notification	Book cab, wait for driver	Notification received	Notification received	Pass

## 6.4 Bug Tracking

Bug ID	Description	Steps to Reproduce	Severity	Status	Notes
BG-001	Delay in driver assignment	Book cab in low availability area	High	Open	Driver allocation takes over 1 min
BG-002	Ride history not loading	Click on history after booking	Medium	Resolved	API delay observed
BG-003	Notification not received	Book cab, turn off notifications	Low	Open	Works after re-fresh

## 7 Results

# Welcome to QuickCab

Book a cab in minutes and travel anywhere in India

[Book a Cab Now](#)

## Real-time Tracking

Track your cab in real-time and know exactly when it will arrive



## Affordable Prices

Competitive pricing with transparent fare calculation



## Safe Rides

All our drivers are verified and trained for your safety



## Login to QuickCab

Email Address

Password

☐ Remember me

[Forgot password?](#)

Login

Or continue with



Driver Login



Admin Login

Don't have an account? [Sign up now](#)

### Admin Dashboard

Total Users  
25  
[View All](#)

Total Drivers  
10  
[View All](#)

Total Rides  
0  
[View All](#)

Active Rides  
0  
[View All](#)

Total Revenue  
₹0

Recent Bookings

No bookings found.  
[View All Bookings](#)

Quick Actions

Add New User

Add New Driver

Create New Booking

Export Reports

### Driver Dashboard

Your Status

Name: rahul

Vehicle: swift (up32fa2130)


Rating: 5.0 ★

Total Rides: 0

Go Offline

Available

Your Current Location



### Book a Cab

Pickup City

Destination City

Find Cabs

Show Map

Pickup City

Bhopal, Madhya Pradesh

Destination City

Indore, Madhya Pradesh

Find Cabs

Hide Map

I



## Ride Details

Confirmed - Driver On The Way

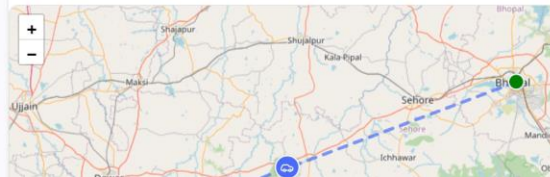
End Ride



Rahul Kumar

Maruti Swift - DL-01-AB-3184

Live Driver Location



# 8 Advantages and Disadvantages

## Advantage

1. **Full-Stack MERN Implementation**  
Leverages modern technologies (MongoDB, Express, React, Node.js) for scalable and maintainable development.
2. **User Role Segmentation**  
Supports three distinct user roles — passengers, drivers, and admins — with customized features for each.
3. **Real-Time Ride Booking Experience**  
Offers features like map integration and dynamic fare calculation for an intuitive booking process.
4. **Responsive and Modern UI**  
Built with Tailwind CSS and React for a clean, mobile-first responsive design.
5. **Secure Authentication**  
Implements JWT and bcrypt for session management and password protection.
6. **Modular and Scalable Architecture**  
RESTful APIs and component-based structure allow for easy updates and future feature integrations.
7. **Extensive Feature Set**  
Includes ride history, driver info, ratings, admin dashboards, and more — mimicking real-world apps like Uber/Ola.
8. **Open to Cloud Deployment**  
Compatible with cloud platforms like Vercel, Heroku, AWS, and MongoDB Atlas.

## Disadvantages

1. **No Real-Time Communication Layer**  
Lacks WebSocket or Socket.io implementation for true live updates (ride status, availability, chat, etc.).
2. **Missing Payment Integration**  
Doesn't currently support online payments, limiting real-world usage.
3. **Basic Driver Matching Logic**  
No intelligent ride-matching algorithm (e.g., proximity or ETA based driver selection).
4. **Limited Security Hardening**  
No advanced protection features like rate limiting, captcha, or advanced threat detection.
5. **No Offline Support or Native App**  
As a web app, it doesn't support offline booking or offer mobile app functionality.
6. **Lack of Deep Analytics**  
Admin dashboard provides basic info but lacks detailed insights or data visualizations.



#### 7. **Scalability Bottlenecks**

Current monolithic backend may face performance issues under high concurrent load.

#### 8. **Manual Verification Needed**

Driver validation and approval must be done manually, with no automated verification system.

## 9 Conclusion

The QuickCab application represents a full-featured, scalable, and modern cab booking

solution built on the powerful MERN (MongoDB, Express, React, Node.js) stack. It successfully simulates the functionality of industry leaders like Uber and Ola, offering tailored experiences for passengers, drivers, and administrators through dedicated interfaces.

With real-time ride booking, fare calculation, profile management, and a secure authentication system, QuickCab demonstrates best practices in both frontend and backend development. The modular architecture, clean UI, and integration-ready design make it an excellent foundation for a production-grade mobility platform.

While the current version covers all essential workflows, future enhancements such as payment gateway integration, push notifications, real-time communication, and advanced analytics can significantly boost the platform's capability and user experience. Known limitations, like basic driver matching and lack of real-time alerts, are clearly identified and form the basis for future development priorities.

Overall, QuickCab serves as a robust, extensible, and practical application that showcases modern web development techniques, secure architecture, and a real-world use case — making it a great candidate for further evolution into a commercially viable product.

Let me know if you want this packaged into a polished README or project report. I can also help format it for a PDF, portfolio submission, or GitHub project.

## 10 Future Scope

The following enhancements can significantly boost the platform's usability, scalability, and feature richness:

## ✦ Functional Enhancements

- **Payment Gateway Integration:** Integrate Razorpay, Stripe, or PayPal for seamless fare transactions.
- **Push Notification System:** Implement Firebase Cloud Messaging (FCM) for real-time alerts.
- **Real-Time Communication:** Use WebSockets or Socket.io for live ride updates, request notifications, and chat support.
- **Advanced Ride Matching:** Add geospatial querying (e.g., MongoDB's GeoJSON) for smarter driver allocation.
- **In-App Chat System:** Allow secure messaging between driver and passenger.

## ✦ User Experience

- **Multi-language Support:** Use i18n libraries to support different languages based on user preference.
- **Dark Mode:** Add a toggle for light/dark UI themes for better accessibility.
- **Driver Verification System:** Enable KYC, license upload, and admin approval workflows.
- **Promo Codes & Loyalty Programs:** Offer discounts, cashback, and reward points.

## ✦ Admin & Analytics

- **Advanced Dashboard:** Include charts, heatmaps, and ride trends using tools like Chart.js or Recharts.
- **Role-based Permissions:** Add granular access control for different admin staff roles (e.g., analytics viewer, operations manager).

## ✦ Technical Enhancements

- **Microservices Architecture:** Refactor backend into scalable services (auth, ride, user, analytics).
- **Caching Layer:** Introduce Redis or similar for caching frequently accessed data (like cab types, fare rates).
- **Logging and Monitoring:** Use tools like ELK Stack or Prometheus + Grafana for real-time system monitoring.
- **Load Balancing:** Implement NGINX or AWS ALB for better traffic distribution.

## 11 Appendix

GitHub Link - <https://github.com/Anshag4567/MERN-Project-SmartInternz-QuickCab.git>

