

Case Study

Building a Ride-Sharing Platform with User and Admin Functionalities

Name : Aditya Sahani(B22CS003)

College : IIT Jodhpur

Tech-Stack :

1. NodeJS
2. ExpressJS(Backend)
3. MongoDB(Database)
4. ReactJS(Frontend)
5. Tailwind(Styling Framework)

Introduction :

Our platform caters to the needs of Travelers, their Companions, and Administrators. We focus on seamless ride-sharing experiences, top-notch security, and scalability to meet the ever-growing urban transportation demands.

Project Overview:




The primary objective of this project is to create a seamless and user-friendly platform that facilitates efficient ride-sharing experiences while ensuring the highest standards of security, reliability, and scalability.


Functionalities:

1. Traveler:

- Empower travelers with the ability to share ride details during their journeys effortlessly.
- This includes critical information such as Trip-Id, Driver Name, Driver Phone Number, and cab details through popular communication channels like WhatsApp or SMS.

- Additionally, travelers should be able to review the audit trail of their shared rides for a comprehensive post-trip analysis.

 ShareRide Home About Services  



Lucknow

Hyderabad

Search

Trip Details




Trip ID:
66b23b3534b5a039fc058c06

Driver Name: Ram Suthar

Cab Number: UP 32 MH 2024

Driver Phone: 9621232324

Book Cab

 ShareRide Home About Services  

RIDE DETAILS

Trip ID: 66b23b3534b5a039fc058c06

User ID: 66b23ab834b5a039fc058c03

Driver Name: Ram Suthar

Cab Number: UP 32 MH 2024

Driver Phone: 9621232324

Start Date and Time : 2024-08-06T15:03:17.315Z

Start Location : Lucknow

End Location : Hyderabad

Destination Reached

Share Ride



ShareRide

Home

About

Services



TripId : 24223

Date : lucknow

Driver Name : Aditya

Cab_Number : lucknow

See Details

TripId : 24223

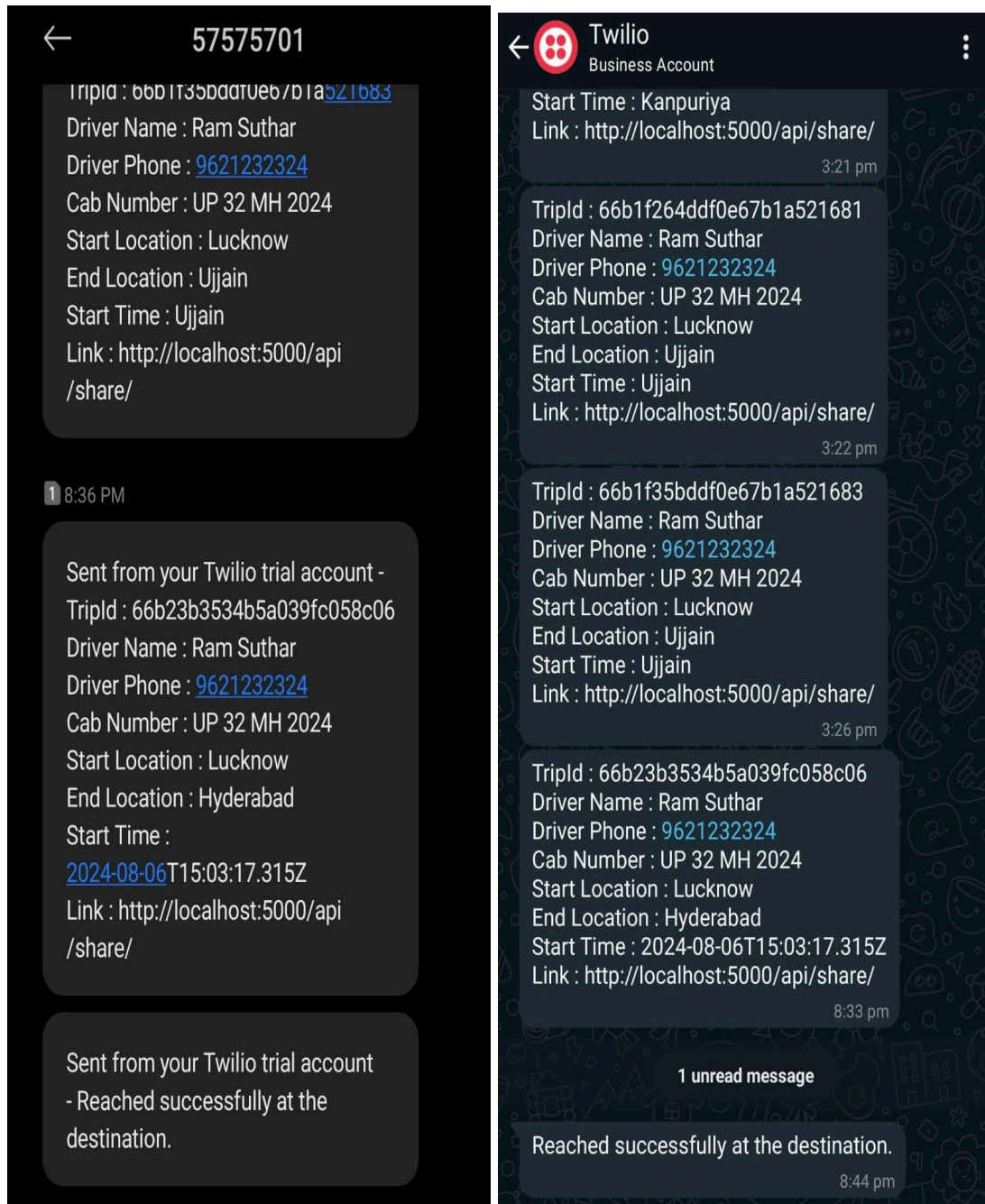
Date : lucknow

Driver Name : Aditya

Cab_Number : lucknow

See Details

View Trip Details

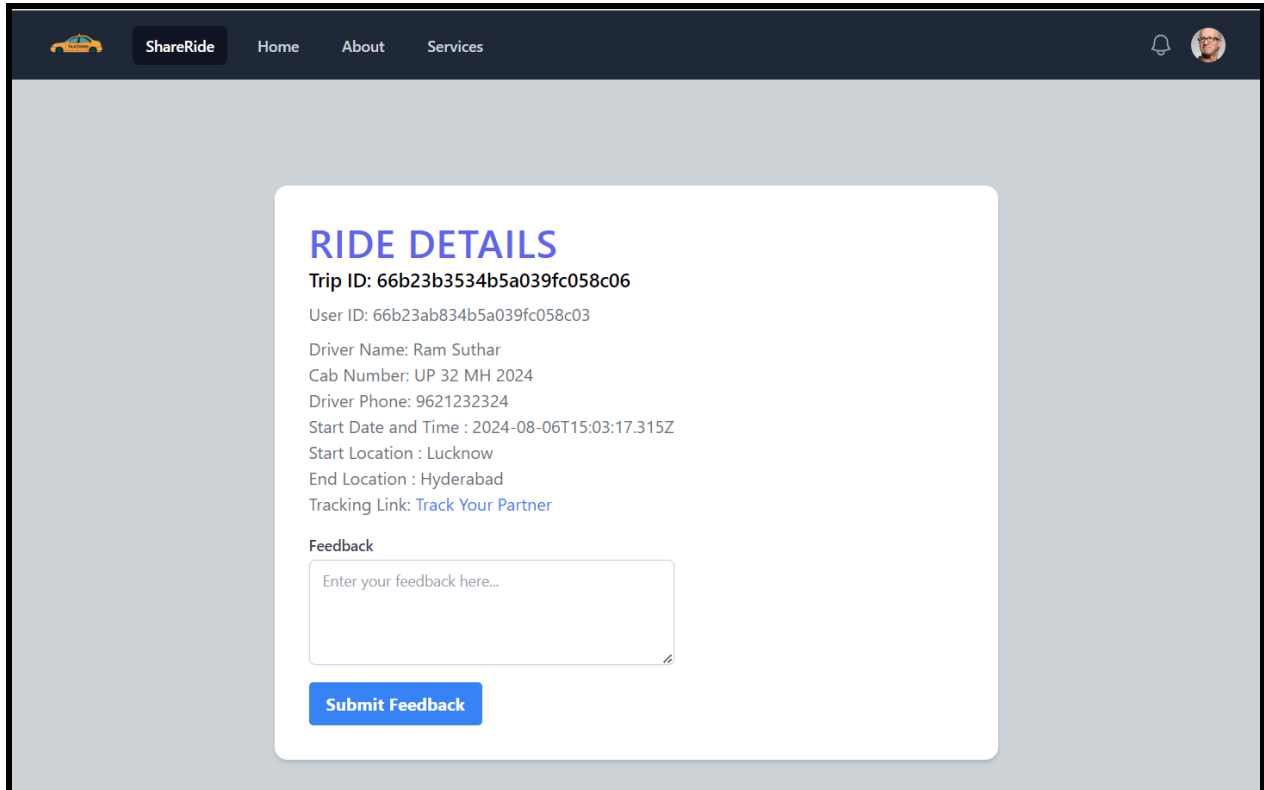


WhatsApp and SMS sent using Twilio's API.

2. Traveler Companion:

- Enhance the travel experience for companions by tracking the traveler's real-time ride.

- Traveler companions receive timely notifications, such as trip completion alerts and geofence triggers, when the cab approaches the traveler's drop location.
- Moreover, the platform enables companions to share valuable feedback on the overall travel experience with the Admin.



The screenshot displays the 'ShareRide' app interface. At the top, there is a dark blue navigation bar with a taxi icon, the 'ShareRide' logo, and links for 'Home', 'About', and 'Services'. On the right side of the bar are a notification bell and a user profile icon. The main content area has a light gray background and features a white card titled 'RIDE DETAILS' in blue. Below the title, the following information is listed: Trip ID: 66b23b3534b5a039fc058c06, User ID: 66b23ab834b5a039fc058c03, Driver Name: Ram Suthar, Cab Number: UP 32 MH 2024, Driver Phone: 9621232324, Start Date and Time: 2024-08-06T15:03:17.315Z, Start Location: Lucknow, End Location: Hyderabad, and a Tracking Link: [Track Your Partner](#). Below this information is a 'Feedback' section with a text input field labeled 'Enter your feedback here...' and a blue 'Submit Feedback' button.

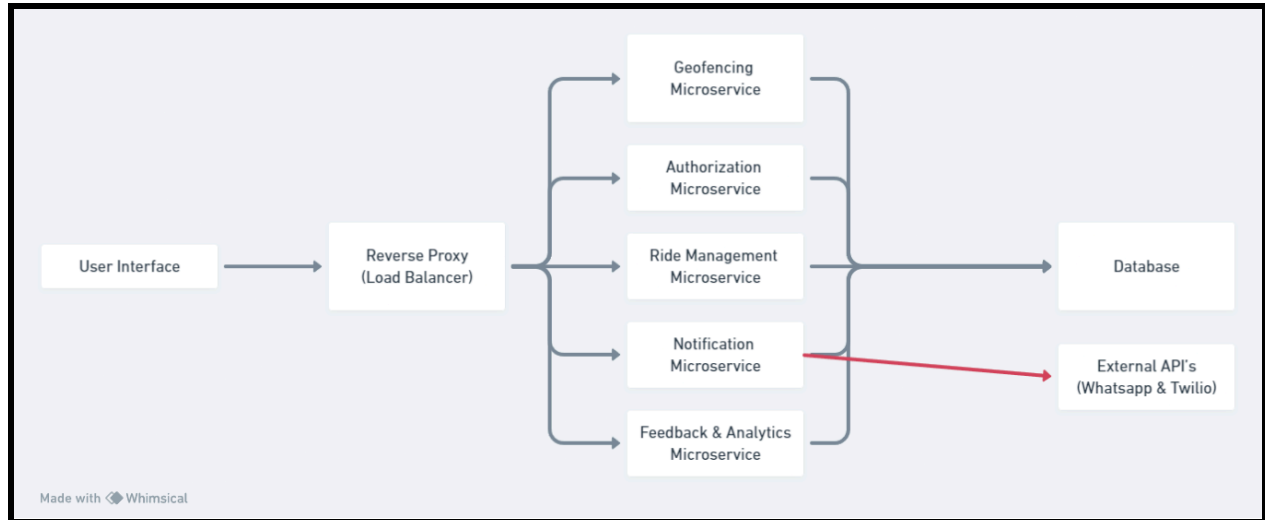
3. Admin:

- Empower administrators with comprehensive oversight and management capabilities.
- Admins have the authority to review all rides users share on the platform.
 - Additionally, they can gain insights into the overall user experience through aggregated feedback, allowing for informed decision-making and continuous platform improvement.

Shared Rides	Feedbacks
's Ride Trip ID: ajmertomumbai Start: End:	
66b11ecb087ba44fb45571a8's Ride Trip ID: Start: Lucknow End: Kanpur	
66b11ecb087ba44fb45571a8's Ride Trip ID: Start: Lucknow End: Kanpur	
66b13a4fc3f4876ebc15b2cf's Ride Trip ID: Start: Lucknow End: Kanpur	
66b13a4fc3f4876ebc15b2cf's Ride	

Shared Rides	Feedbacks
aditya sahani's Feedback dfokjdpognkkf	
Aditya Sahani's Feedback brtg rgrrterter	

System Architecture:



Microservices:

1. Geofencing Microservice:

- a. Functionality: Manages geofence-related operations, tracking the location of cabs and notifying relevant parties when they enter predefined geographical areas.
- b. Key Responsibilities: Geofence creation, monitoring, and triggering notifications based on cab movement.

2. Authorization:

- a. Functionality: Handles user authentication and authorization, ensuring secure access to the platform's features.
- b. Key Responsibilities: Authentication of users, authorization checks, and management of access tokens.

3. Ride Management:

- a. Functionality: Oversees the core ride-sharing operations, including trip creation, tracking, and completion.
- b. Key Responsibilities: Trip creation, real-time ride tracking, and managing the lifecycle of rides.

4. Notification: a. Functionality: Manages to send notifications to users and companions, including trip completion alerts and geofence triggers.

- b. Key Responsibilities: Notification generation, scheduling, and delivery based on various ride events.

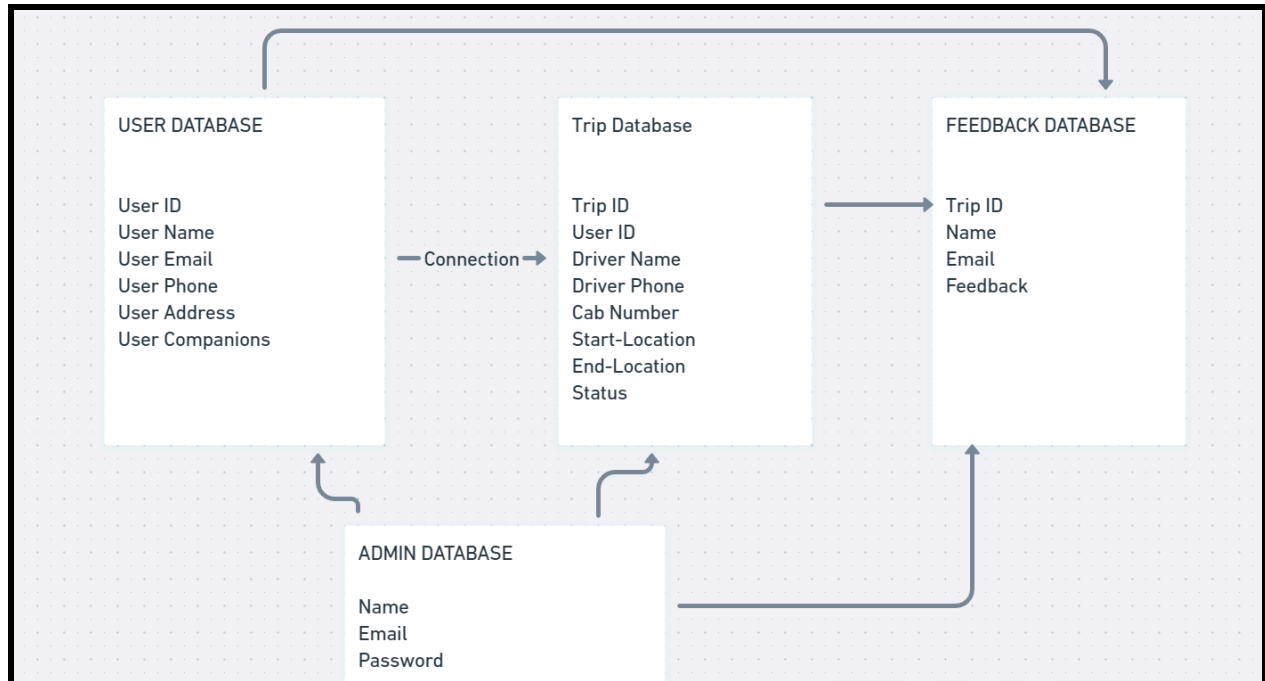
5. Feedback & Analytics:

- a. **Functionality:** Gathers and analyzes user feedback to enhance the platform's performance. Provides administrators with insights through analytics.
- b. **Key Responsibilities:** Collecting user feedback, analyzing trends, and presenting actionable insights to improve the ride-sharing experience.

External APIs:

1. **Twilio APIs :** Utilizes Twilio SMS and WhatsApp APIs service to send the ride details to the companion with details and notifications.

Database:



Authorization :

1. Using bcryptjs for Password Security

Hashing Passwords: Before storing a user's password in the database during the registration process, bcryptjs is used to hash the password. Hashing transforms the plain password into a fixed-length string that appears random and cannot

be reversed back into the original password. This ensures that even if the database is compromised, the actual passwords remain secure.

Salting: bcryptjs also incorporates a salt—random data added to the password before hashing. This prevents attackers from using precomputed hashes (rainbow tables) to reverse-engineer passwords.

Password Verification: When a user logs in, bcryptjs compares the submitted password (after hashing it) with the stored hash to verify the user's identity without ever needing to decrypt anything.

2. Using express-validator for Data Validation

Input Validation: express-validator is a middleware that integrates with Express.js to validate and sanitize user inputs. It ensures that the data received through user forms (like login and registration forms) adheres to expected formats and is safe to process. For example, checking if an email is properly formatted, or ensuring a password meets complexity requirements.

3. Using JWT for Managing Sessions and Authorization

Token Creation: Upon successful login or registration, the server generates a JWT that encodes a user's identity and possibly their role or permissions. This token is sent back to the client and typically stored in the client's local storage or cookies.

Token Structure: A JWT consists of three parts—header, payload, and signature. The header specifies the token type and the hashing algorithm. The payload holds claims about the user, which includes user details and token expiration time. The signature is used to verify that the token wasn't changed on the way back to the server on subsequent requests.

Session Management: On each subsequent request after login, the client sends the JWT back to the server. The server then validates the signature using the secret key used to generate the token. If validation is successful, it extracts the user data from the payload, allowing the server to authenticate the request without needing to check the user's credentials again.

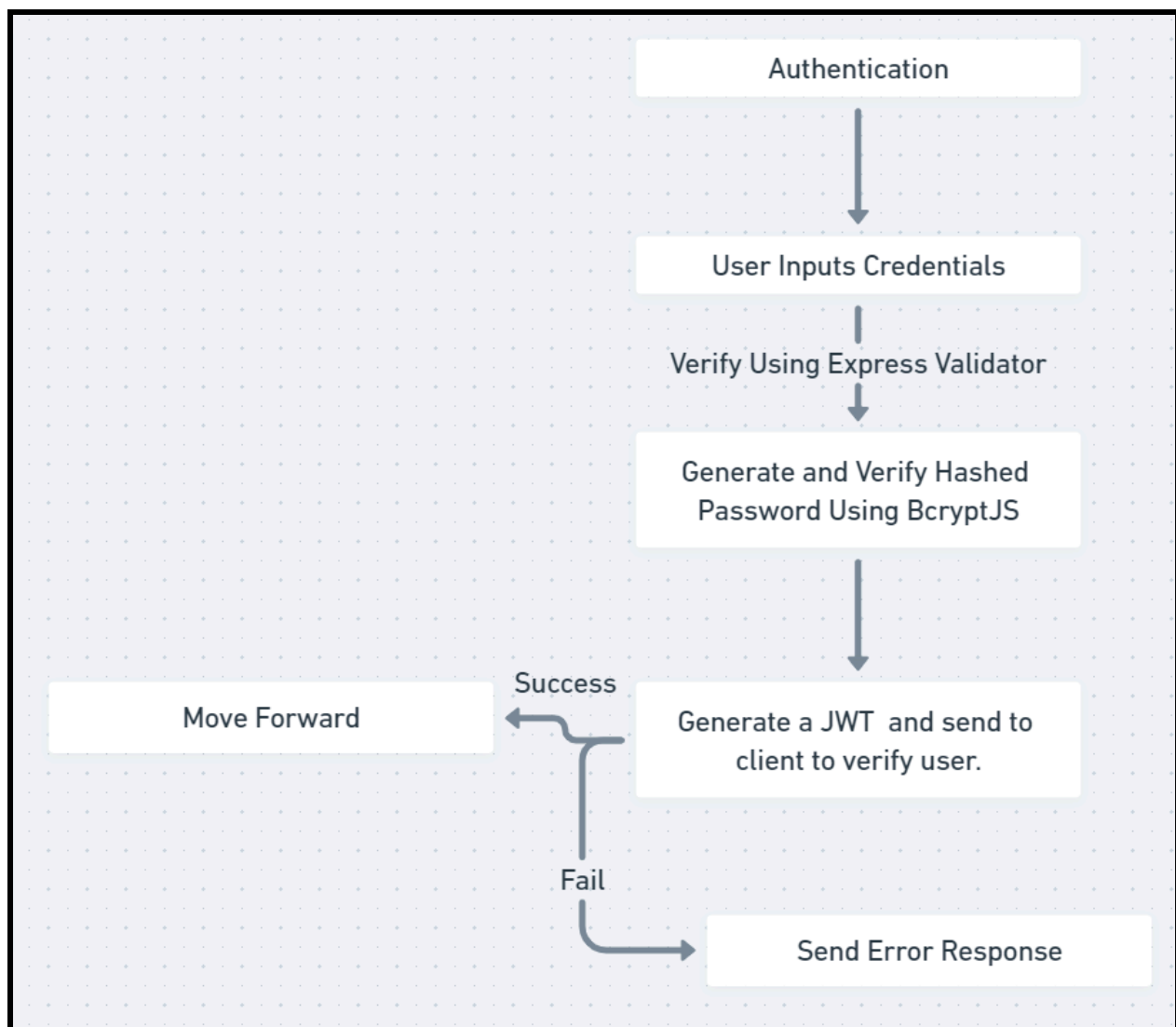
4. Secure Workflow

Registration: Validate user input data, hash the password, and store the user in the database.

Login: Validate credentials, compare the hashed password, generate a JWT if successful, and return it to the client.

Access Protected Routes: For routes that require authentication, middleware checks the presence and validity of the JWT. For routes requiring specific permissions, the JWT's claims are checked against the required permissions.

Authentication Flow :



Ride Sharing Functionality :

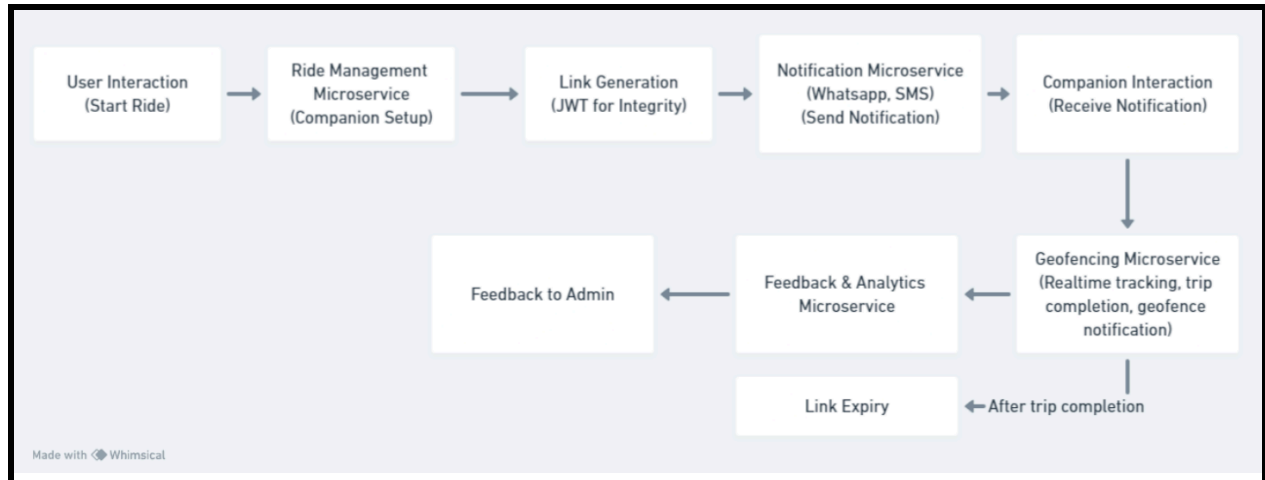
Assuming Ride has already been created, we are implementing the ride-sharing functionality.

1. User Interaction: The Traveler initiates the ride-sharing process through the user interface by selecting a companion and starting a ride.
2. Ride Management Microservice: The Ride Management microservice facilitates the ride-sharing process and interfaces with other components.
3. Companion Setup: Before starting a ride, the Traveler sets a companion by providing the companion's phone number through the Ride Management microservice.
4. Link Generation: When the Traveler starts a ride, the Ride Management microservice generates a unique link or token associated with that ride. This link contains details such as the trip ID and is signed using JWT for integrity.
5. Notification Sending: The Notification Microservice is triggered to notify the designated companion through WhatsApp or SMS. The notification includes the unique link and information about the ride.
6. Companion Interaction: The Traveler Companion receives the notification and opens the link, gaining access to real-time tracking and additional features.
7. Real-time Tracking: The link allows the Traveler Companion to track the ride in real-time using the Geofencing Microservice.
8. Trip Completion Notification: The Geofencing Microservice sends a notification to the Traveler Companion when the trip is complete, indicating a successful ride.
9. Geofence Notification: The Geofencing Microservice monitors the cab's location and sends nearby notifications to the Traveler Companion when the cab approaches the geofence of the traveler's drop location.
10. Feedback Submission: The Traveler Companion can share feedback about the ride experience through a feedback form submitted to the Feedback & Analytics Microservice.
11. Link Expiry: The link generated for ride-sharing has a time-limited validity. The link expires once the ride is complete, ensuring that access to ride details is restricted after the trip.
12. Feedback to Admin: The Feedback & Analytics Microservice records the feedback and forwards it to the Admin.

Security Measures :

2. Secure Notification Delivery: Notifications containing links are sent securely through WhatsApp or SMS, utilizing encryption protocols provided by these services.

3. Link Expiry: The time-limited validity of the link ensures that even if intercepted, it becomes useless after the ride is complete.



Real-Time Tracking Functionality :

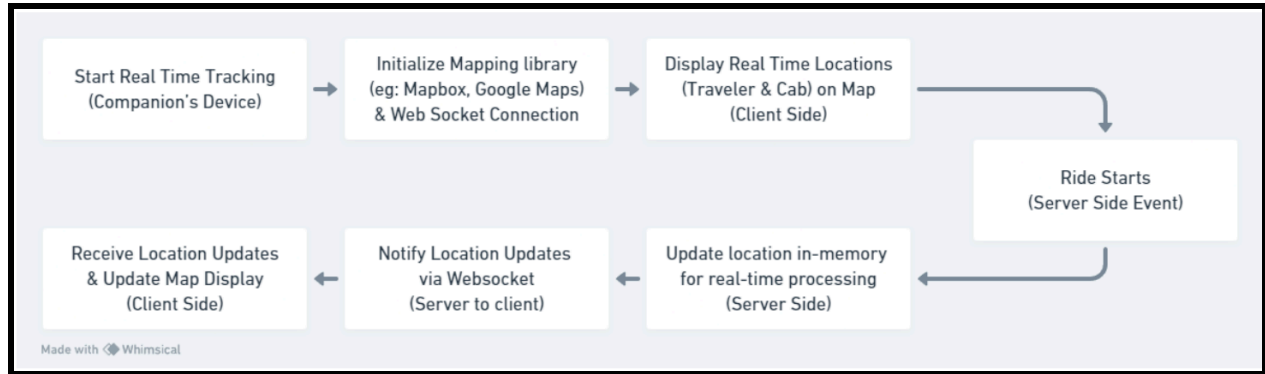
Client Side: • Use a mapping library (e.g., Mapbox, Google Maps) to display the traveler's and cab's real-time location on the Traveler Companion's device.

• Utilize a WebSocket connection to continuously update the traveler's and cab's location on the map.

Server Side:

• When the ride starts, update the traveler's and cab's locations on the server regularly.

• Use WebSocket to notify the Traveler Companion's device of location updates.



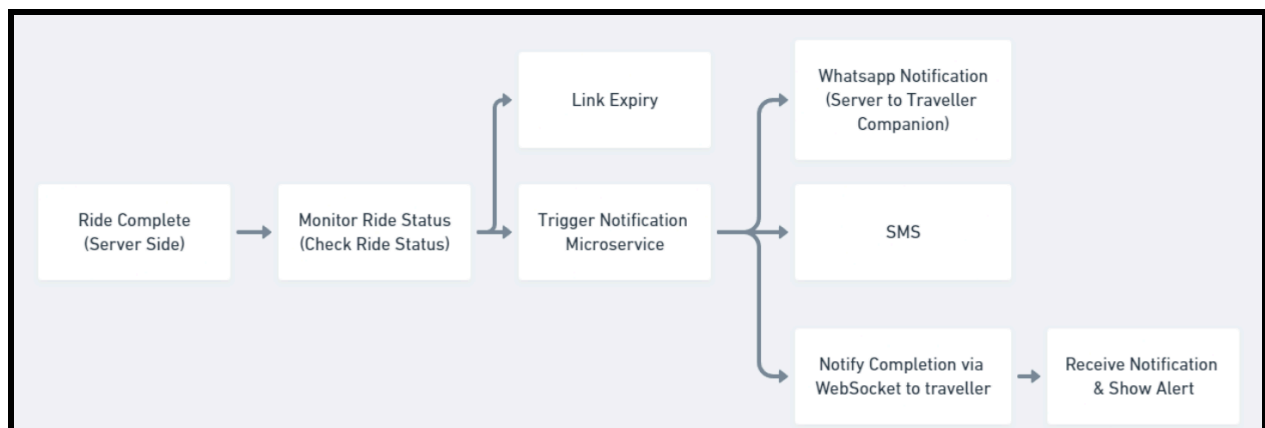
Trip Completion Notification:

Server Side:

- Monitor the ride's status, and when it's marked as complete, trigger a notification event.

Client Side:

- Use WebSocket to listen for trip completion events.
- When the event occurs, show a notification on the Traveler Companion's device indicating that the trip is complete.



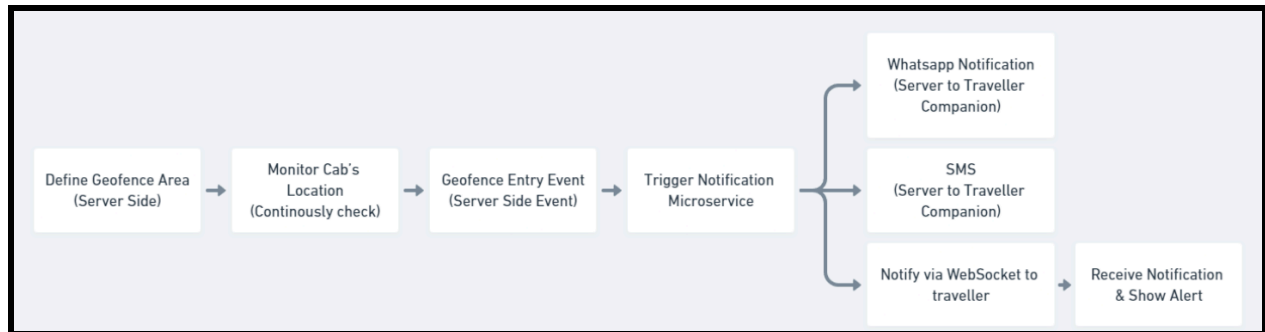
Geofence Notification :

Server Side:

- Define a geofence area around the traveler's drop location.
- Continuously monitor the cab's location.
- When the cab enters the geofence area, trigger a geofence entry event.

Client Side:

- Use WebSocket to listen for geofence entry events.
- When the event occurs, notify the Traveler Companion's device that the cab is nearby.



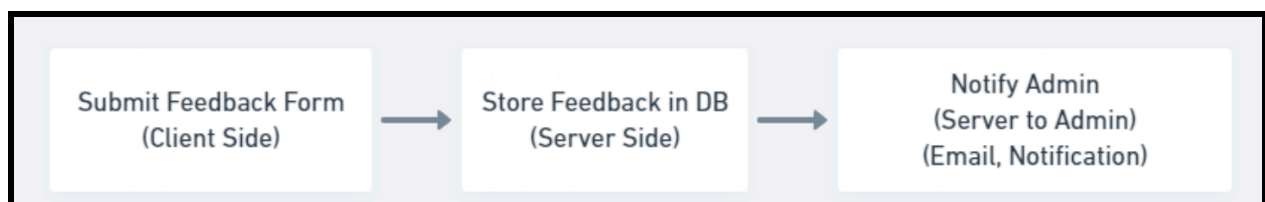
Feedback Functionality :

Server Side:

- Provide an API endpoint for submitting feedback.
- Fetch and serve feedback data from the database.
- The server forwards feedback to the Admin for analysis.

Client Side:

- Implement a feedback form in the client application.
- Users submit feedback about their ride experience.
- Collected feedback is sent to the server.



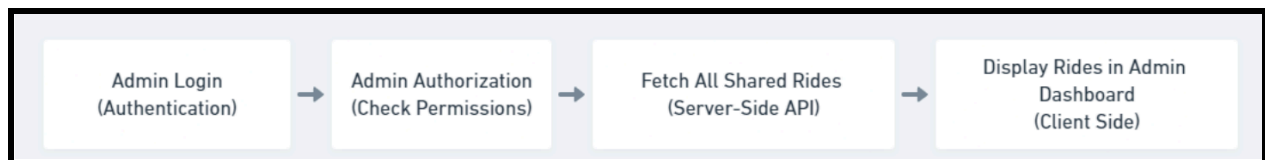
View All Shared Rides(Admin) :

Server Side:

- Provide an API endpoint for admin authentication and authorization.
- Fetch and serve ride data from the database.

Client Side:

- Admins access the ride information through a secure admin dashboard.



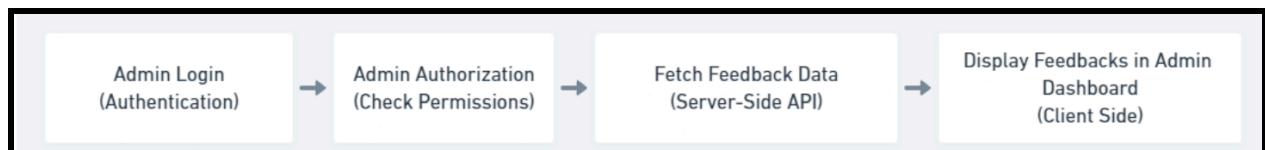
View All Feedbacks(Admin) :

Server Side:

- Aggregate and process user feedback data.
- Make feedback data available through a secured API endpoint.

Client Side:

- Admins access the feedback overview through the admin dashboard.



Time and Space Estimation :

1. Insertion, Deletion, Updation in Database (Time)

Criteria	Insertion	Deletion	Updation
Normal	$O(1)$	$O(n)$	$O(n)$
Index Based	$O(\log n)$	$O(\log n)$	$O(\log n)$

UserBase Scaling :

1. Horizontal and Vertical Scaling

Horizontal Scaling (Scaling Out/In):

Add more servers to your pool to handle increased traffic. This is effective for handling more parallel requests and is generally more flexible and resilient than vertical scaling.

Vertical Scaling (Scaling Up/Down):

Upgrade the existing servers with more powerful CPUs, more RAM, or faster storage. This can be a quicker and simpler short-term solution but often has limitations and higher long-term costs.

2. Cloud-Based Solutions

Use cloud services that offer elasticity, automatically scaling your resources up or down based on demand. Services like AWS EC2, Google Compute Engine, and Azure Virtual Machines provide scalable server options.

Conclusion :

In conclusion, our ride-sharing platform successfully integrates essential features such as robust user authentication, efficient ride-sharing functionalities, real-time tracking for traveler companions, and a streamlined feedback mechanism for administrators. The implementation of WhatsApp and SMS notifications enhances the user experience. Leveraging in-memory storage for location updates optimizes time and space complexity, ensuring swift and responsive performance. Scalability strategies have been systematically integrated, laying the foundation for accommodating an expanding user base. This comprehensive solution aligns with the project's goals, offering a secure, scalable, and user-centric ride-sharing experience.