That's an excellent project! EzyRide is a complex system combining real-time features (chat, live ride status) and critical business logic (payments, OTPs, posting/searching). This makes the Product Specification Document (PRD) absolutely essential for the testing tool to generate meaningful tests.

Here is a comprehensive draft PRD in Markdown format, specifically structured to guide an automated testing agent. You can copy and paste this text into a file named EzyRide_PRD.md and upload it to the configuration screen.

📝 EzyRide Carpool Service Web App PRD

1. Goal & Objective

The primary goal of EzyRide is to connect drivers (offering rides) with riders (seeking rides) in a secure, efficient, and cost-effective manner. The system must ensure data integrity, user security, and smooth transaction processing.

2. Core Features & Functional Requirements

The application is split into four main modules: User Lifecycle, Ride Management, Communication, and Payments.

2.1. User Lifecycle (Authentication & OTP)

Feature	Requirement	Testing Focus
Registration/Login	Users must register with a phone number and email. All sign-ins require a unique One-Time Password (OTP) sent to the registered phone number.	Must test OTP validity, expiration, and rate limiting for resends.
Profile Management	Users can update basic details (name, profile picture). Drivers must upload required vehicle documents.	Test input validation (e.g., max length, valid image format).

2.2. Ride Management (Post & Search)

Feature	Requirement	Testing Focus

Post Ride (Driver)	Drivers define the route (Origin/Destination), date/time, total seats available, and cost per seat.	Test edge cases like past dates, zero seats, and negative costs.
Search Ride (Rider)	Riders search by Origin, Destination, and Date. Results must display relevant rides sorted by time/relevance.	Test search performance and filtering accuracy based on proximity to the route.
Book/Accept Ride	Riders request a seat. Drivers must explicitly accept or decline the request. Booking reduces available seats.	Test concurrent booking requests on the same ride.
Live Ride Status	The system must track and update the ride status: Posted > Booked > Started > Completed/Cancelled.	Test transitions and ensure only the driver can update the Started/Completed status.

2.3. Communication (Live Chat)

- **Requirement:** Once a ride is booked, a private, persistent chat room must be created between the driver and the rider(s) for that specific ride.
- **Testing Focus:** Test message delivery, security (only participants can view), and persistence after a page refresh.

2.4. Payments

- **Requirement:** Payment integration must handle the secure transfer of funds from the rider to the driver, facilitated by a third-party payment gateway (e.g., Stripe/Razorpay).
- **Workflow:**
    1. Rider pays the full fare upon booking. Funds are held in escrow.
    2. Funds are released to the driver only after the ride status is updated to Completed.
- **Testing Focus:** Test payment success/failure scenarios, ensure funds are locked in escrow, and verify the correct release amount after ride completion.

3. Non-Functional Requirements (NFRs)

- **Performance:** Search results must be delivered in under **500ms**.
- **Security:** All API endpoints must be protected by **Bearer Token** authentication (as defined in the config).
- **Scalability:** The platform should support at least **1,000 concurrent users** during peak hours.