

Sanjay — Module Documentation

1. System Overview

RapidRide simulates a real ride-hailing ecosystem with distributed design.

Core objectives include:

- Stable real-time communication
- Predictable scaling behavior
- Modular subsystem separation

System Highlights

- Distributed Services
- Scalable Data Flow
- Event-driven Coordination

2. Functional Architecture

Each module works semi-independently but maintains clear interaction boundaries.

Major functional flows:

- Ride request & validation
- Driver availability broadcast
- ETA calculation & ML inference

Core Flows

- Request Intake
- Service Coordination
- Result Dispatch

3. Technical Stack Breakdown

A combination of modern technologies ensures responsiveness:

- Node.js for backend orchestration
- FastAPI for ML-heavy tasks
- Redis for in-memory caching
- ElasticSearch for analytics

Tech Purpose

- Orchestration Layer
- ML Processing

- State Caching
- Log & Search Engine

Role Responsibilities

This member is responsible for:

- Frontend flows
- WS client consumption
- Map UI
- State management

Sanjay Overview

- Frontend flows
- WS client consumption
- Map UI
- State management

4. Integration Flow

Interaction order between major modules:

- Frontend → Backend (API request)
- Backend → FastAPI (ML/ETA)
- Backend → Redis (State lookup)
- Backend → ES (Trip logs)

Sequential Flow

- Input ← User
- Processing → Backend
- Decision → Matching
- Output → Notifications

5. Future Scope

Possible long-term improvements:

- Heatmap-based driver analysis
- Predictive surge pricing
- Log-based behavioral ML models

Roadmap Ideas

- Analytics Dashboards
- Distributed caching clusters
- Auto-scaling observability