Test update  
  
  
System Architecture

This system uses a Microservices Architecture to ensure high availability, scalability, and flexibility. It supports on-demand transportation services, targeted advertising, accessibility features, and AI-based optimization for predictive dispatching. The system comprises the following key modules:

1. Front-end System (User-end Web & Mobile)

The front-end system provides interactive interfaces for passengers and drivers, supporting multi-platform access (Mobile and Web).

1.1 Passenger App (Mobile & Web)

Booking System: Supports real-time and scheduled ride bookings.

Vehicle Tracking: Real-time GPS tracking of vehicles.

Payment Options: Supports online payments via PayPal, Google Pay, Credit Cards, and others.

Voice Control: AI-powered voice assistant minimizes manual operations.

Intelligent Recommendations: Personalized route suggestions based on user habits.

Accessibility Support: Options for wheelchair-accessible vehicles, in-vehicle alerts, and reminders.

Advertisement System: Personalized ad delivery based on user behavior.

Multilingual Support: English, Chinese, and other local languages.

1.2 Driver App (Mobile)

Order Management: Accept, update, and manage ride orders.

Navigation System: Real-time route optimization based on traffic conditions.

Passenger Management: Confirm passenger boarding and alighting.

Voice Notifications: Automated announcements about routes, stops, and urgent messages.

Emergency Assistance: Driver SOS button integrated.

2. API Gateway

Centralized management of front-end requests.

Authentication using OAuth2 and JWT tokens.

Traffic control and load balancing.

Unified logging and error handling.

3. Core Backend Services (FastAPI)

User Management: User registration, authentication, and profile management.

Order Management: Create, modify, cancel orders, and update statuses.

Dispatch Optimization: Vehicle matching and route optimization.

Advertising Management: Targeted in-app and onboard ads.

Payment Processing: Payments, refunds, wallet, and rewards system.

Data Analytics: User behavior analysis, predictive analytics for demand forecasting, and operational reporting.

4. AI Prediction Services

Demand Prediction: LSTM and K-Means clustering-based demand forecasting.

Route Optimization: A search algorithm for route planning.

Vehicle Scheduling: Real-time vehicle allocation based on predictive demand.

Profitability Optimization: Calculation of profitability for each order to optimize revenue.

Environmental Optimization: Calculate carbon emissions and incentivize through carbon credits.

5. On-board Vehicle System

Real-time GPS tracking.

Passenger Display Systems showing journey details and ads.

Voice broadcast system for alerts, stops, and emergencies.

Accessibility features like audio announcements and alerts.

5. Data Storage System

5.1 Databases (PostgreSQL + Redis)

PostgreSQL:

Stores orders, user information, vehicle statuses, and advertising data.

Ensures transactional integrity.

Redis (Cache):

Real-time caching of order statuses.

Vehicle location caching to improve query performance.

API rate-limiting and session management.

5.2 Publish-Subscribe System (Message Queues)

Kafka/RabbitMQ handling high-concurrency data flows:

Order status updates.

Vehicle status updates.

Real-time ad push notifications.

Passenger notifications (ride updates, payment confirmations).

Event-driven architecture facilitating efficient real-time communication.

6. Data Flow within System

6.1 API Communication Types

Front-end communicates with backend via RESTful APIs and WebSockets.

Inter-service communication via gRPC and RESTful APIs.

External interactions with APIs like Google Maps and Payment Gateways using REST APIs.

7. Logical Layer Separation

7.1 Layered Architecture

API Layer (HTTP Request/Response): Handling external requests, authentication, and rate limiting.

Business Logic Layer (Service Layer): Order processing, vehicle dispatch, passenger and driver management.

Data Access Layer (Repository Layer): ORM for database interactions, transaction handling, and caching.

Utility Layer: Encryption, logging, and external API adapters.

8. Data Models

8.1 Included Data Models

User Model: ID, name, roles, contact details, authentication.

Order Model: Order ID, user ID, driver ID, status, pricing details.

Vehicle Model: Vehicle ID, model, GPS location, operational status.

Payment Model: Payment ID, order ID, payment method, transaction status.

Ad Model: Ad ID, content, targeted users, delivery records.

8.2 Data Model Management

Managed via ORM (SQLAlchemy).

Validation performed at API layer (using Pydantic for data validation).

Business logic validation performed at the service layer.

Database migrations managed by Alembic.

9. Communication within Internal Components

RESTful API and WebSocket communication between front-end and back-end.

Event-driven communication within back-end components using Kafka.

ORM and transaction handling for database interactions.

Redis caching for reducing latency and database load.

10. Interaction with External Services

Integration with external APIs (Google Maps, payment gateways, weather services).

Email and SMS notifications via Twilio/SendGrid APIs.

Cloud Storage solutions (AWS S3 or Google Cloud Storage) for media file storage.

Technology stack summary

|  |  |  |
| --- | --- | --- |
| Component | |  | | --- | | Technology | |
| |  | | --- | | Frontend | | |  | | --- | | Flutter, React, Vue | |
| |  | | --- | | Backend | | FastAPI (Python) |
| Databases | PostgreSQL, Redis |
| Message Queue | Kafka, RabbitMQ |
| Authentication | OAuth2, JWT |
| AI Prediction | LSTM, A\* Search, K-Means |
| Payments | PayPal, Google Pay, Alipay, Stripe |
| Monitoring & Logging | ELK Stack, Prometheus, Grafana |
| External APIs | Google Maps, OpenWeather, Twilio, SendGrid |
| Deployment | AWS (EC2, S3, RDS, Lambda), Docker, Kubernetes |