## **System Architecture**

### **High-Level Architecture Overview**

The Ride Pooling System implements a hybrid architecture combining microservices design patterns with blockchain integration. This approach enables scalability, maintainability, and the security benefits of distributed ledger technology.

### **Architecture Layers**

#### **1. Client Layer**

The client layer comprises the user-facing applications that provide the interface for riders and drivers to interact with the system.

**Mobile Application (Flutter):**

* Cross-platform mobile application built with Flutter framework
* Implements Material Design principles for Android and Cupertino design for iOS
* Native features for location tracking, push notifications, and QR code scanning
* Offline capability with data synchronization when connectivity returns
* Local secure storage for user credentials and temporary ride data

**Web Application (React):**

* Single-page application built with React.js
* Responsive design supporting desktop, tablet, and mobile browsers
* Progressive Web App (PWA) features for improved mobile experience
* WebSocket integration for real-time updates on ride status
* Integration with browser geolocation APIs

#### **2. API Gateway Layer**

Acts as the single entry point for all client requests, handling routing, load balancing, and common cross-cutting concerns.

**API Gateway:**

* JWT-based authentication validation
* Rate limiting to prevent abuse
* Request routing to appropriate microservices
* Response caching for frequently accessed data
* Request/response logging for debugging and analytics
* API version management

#### **3. Application Services Layer**

Core business logic implemented as independent microservices, each responsible for a specific domain function.

**User Service:**

* User registration and profile management
* Authentication and authorization
* Driver verification process
* User reputation system
* Privacy preferences management

**Ride Service:**

* Ride planning and scheduling
* Route optimization algorithms
* Driver-passenger matching
* Fare calculation
* Ride status management

**Geolocation Service:**

* Real-time location tracking
* Geocoding and reverse geocoding
* Route calculation and traffic analysis
* Proximity search for nearby rides
* Geofencing for pickup/dropoff zones

**Notification Service:**

* Push notifications for mobile devices
* Email notifications
* In-app messaging system
* Real-time alerts for ride status changes
* Scheduled reminders

**Payment Service:**

* Traditional payment processing (credit cards, bank transfers)
* Payment gateway integrations
* Invoicing and receipts
* Refund processing
* Driver payout management

#### **4. Blockchain Layer**

Specialized services for interacting with the Cardano blockchain network.

**Blockchain Service:**

* Middleware between application services and blockchain network
* Transaction creation and submission
* Block exploration and transaction verification
* Event monitoring for smart contract updates
* Transaction fee estimation

**Smart Contracts:**

* Ride verification contracts
* Escrow payment contracts
* Driver-passenger agreement contracts
* Dispute resolution mechanisms

**ADA Wallet Integration:**

* Wallet creation and management
* Address generation and validation
* Transaction signing
* Balance checking
* Transaction history

#### **5. Data Layer**

Storage solutions for persistent data handling across the system.

**User Database:**

* User profiles and credentials
* Driver verification information
* User preferences and settings
* Rating and review history
* Session data

**Ride Database:**

* Ride details and history
* Route information
* Matching data
* Pricing information
* Ride status tracking

**Transaction Database:**

* Payment records
* Blockchain transaction references
* Invoice and receipt data
* Refund information
* Financial reporting data

**Cache Layer (Redis):**

* Session caching
* Frequently accessed ride data
* Geospatial data caching
* Rate limiting counters
* Real-time ride status

#### **6. External Services**

Third-party integrations providing specialized functionality.

**Map Provider API:**

* Mapping visualization
* Geocoding services
* Turn-by-turn navigation
* Traffic information
* Points of interest

**Cardano Node:**

* Blockchain network access
* Transaction broadcasting
* Smart contract execution
* Consensus participation
* Network synchronization

**Cloud Storage:**

* Document storage (ID verification, etc.)
* Media storage
* Backup and disaster recovery
* CDN integration for static assets
* Temporary file storage

**Analytics Platform:**

* Business intelligence gathering
* User behavior analysis
* System performance monitoring
* Market analytics
* Reporting dashboards

### **Architecture Viewpoints**

#### **Logical View**

The system is organized into domain-specific microservices that encapsulate specific business capabilities. Each service maintains its own data storage and exposes well-defined APIs for communication with other services.

#### **Process View**

The system handles several key processes:

1. **Ride Offering Process**: Driver registration → route definition → ride offering → passenger matching → ride execution → payment collection → ride completion verification
2. **Ride Booking Process**: Passenger search → route selection → booking request → driver confirmation → ride tracking → payment processing → rating submission
3. **Blockchain Verification Process**: Ride agreement creation → smart contract deployment → milestone verification → payment release → transaction recording

#### **Development View**

The system's codebase is organized into distinct repositories:

* Client applications (Mobile and Web) in separate repositories
* Microservices each in their own repository with independent CI/CD pipelines
* Shared libraries for common functionality
* Infrastructure as code for deployment configuration
* Smart contract repository with formal verification tools

#### **Physical View**

The production deployment will utilize cloud infrastructure with:

* Containerized microservices running in Kubernetes clusters
* Regional deployment for low-latency user experience
* Database clusters with appropriate replication strategies
* Content delivery networks for static assets
* Dedicated Cardano stake pool for blockchain transaction processing

### **Communication Patterns**

#### **Synchronous Communication**

* REST APIs for direct request-response interactions

#### **Asynchronous Communication**

* WebSockets for real-time updates to client applications
* Event sourcing for tracking state changes in critical business processes
* Blockchain event monitors for smart contract state changes

### **Fault Tolerance and Recovery**

* Circuit breakers to prevent cascade failures
* Retry mechanisms with exponential backoff
* Fallback mechanisms for critical services
* Data replication across multiple availability zones
* Regular automated backups with point-in-time recovery
* Disaster recovery procedures for various failure scenarios

### **Security Architecture Considerations**

* Isolation of sensitive components (payment processing, user data)
* Defense-in-depth approach with multiple security layers
* End-to-end encryption for all user communication
* API authentication using OAuth 2.0 and OpenID Connect
* Role-based access control for administrative functions
* Rate limiting to prevent brute force and DoS attacks
* Regular security audits and penetration testing