**Technical Specification (TS)**

**For the development of a platform for ordering intercity trips and cargo transportation "AutoPort"**

**Version:** 2.0 (Consolidated)  
**Date:** October 25, 2023

**Table of Contents:**

1. Introduction
2. Project Goals and Objectives
3. Target Audience
4. General System Requirements  
   4.1. System Architecture  
   4.2. Platforms  
   4.3. Language Support
5. Functional Requirements  
   5.1. "Passenger" Module  
   5.2. "Driver" Module  
   5.3. "Administrator" Module (Web Panel)  
   5.4. Common Functions for All User Applications  
   5.5. Specific Functions (Intercity)
6. Security Requirements (Policies and Functionality)  
   6.1. User Verification  
   6.2. Trip Safety  
   6.3. Fraud Protection  
   6.4. Personal Data Processing Policy
7. Non-functional Requirements  
   7.1. Performance  
   7.2. Reliability and Availability  
   7.3. Scalability  
   7.4. Usability  
   7.5. Maintainability
8. Design Requirements (UI/UX)  
   8.1. General Concept  
   8.2. Color Palette  
   8.3. Typography  
   8.4. Iconography  
   8.5. Animations and Transitions
9. Technology Stack and Technical Specifications  
   9.1. Backend  
   9.2. Mobile Applications (iOS and Android)  
   9.3. Telegram Mini App (TMA)  
   9.4. Administrator Web Panel  
   9.5. Infrastructure and DevOps  
   9.6. Technical Security Aspects  
   9.7. Performance and Scalability Targets  
   9.8. Code Quality and Documentation
10. Integrations
11. Development Stages and Acceptance
12. Glossary

**1. Introduction**

This document describes the requirements for the development of a software and hardware complex (hereinafter referred to as the Platform) "AutoPort". The Platform is designed to organize intercity passenger and cargo transportation in the Republic of Uzbekistan. The Platform aims to replace outdated methods of finding transportation (so-called "pyataki" or gathering points) and spontaneously organized Telegram groups, providing a modern, convenient, and safe tool for interaction between drivers and passengers/cargo senders.

**2. Project Goals and Objectives**

* **Main Goal:** Create a leading digital platform in the Uzbekistan market for ordering intercity trips and related services (cargo transportation).
* **Objectives:**
  + Provide convenient and quick search/offering of trips and cargo transportation services.
  + Improve intercity transportation safety through user verification and trip tracking.
  + Provide a transparent pricing and review system.
  + Optimize logistics for passenger pickup and cargo delivery.
  + Create a unified information space for drivers and passengers.
  + Ensure platform monetization through commissions and additional services.

**3. Target Audience**

* **Passengers:** Residents of Tashkent, regional centers, and other cities of Uzbekistan who regularly make intercity trips (students, business travelers, tourists, etc.). They value comfort, safety, and predictability. Active smartphone and Telegram users.
* **Drivers:** Private car owners making intercity transportation on a commercial basis or to compensate for expenses ("going home"). Interested in a flow of orders and convenient business management.
* **Cargo Senders/Recipients:** Individuals and legal entities who need fast and inexpensive delivery of small-sized cargo/documents between cities.

**4. General System Requirements**

**4.1. System Architecture:**

* Client-server architecture.
* **Microservice architecture** for the backend: Each domain service (users, trips, payments, notifications, geolocation, chat, etc.) is implemented as a separate, independently deployable microservice.
  + **Communication between services:** Synchronous (RESTful API through API Gateway or gRPC for internal communication) and asynchronous (through message brokers).
* **API Gateway:** A single entry point for all client requests, providing routing, authentication, authorization, rate limiting, caching.
* **API-first approach** for interaction between the backend and client applications.
* A single logical database (physically can be distributed) for all client applications and the administrative panel.

**4.2. Platforms:**

* **Mobile application for Passenger:** iOS, Android.
* **Mobile application for Driver:** iOS, Android.
* **Telegram Mini App (TMA):** Functionality duplicating the main scenarios of mobile applications for Passenger and Driver. Implemented as a web application that opens within Telegram.
* **Administrator Web Panel:** Access through a browser, responsive design.

**4.3. Language Support:**

* Interfaces of all applications and administrator panel: Russian, Uzbek (Latin and Cyrillic scripts - with switching).
* Possibility of easily adding other languages.

**5. Functional Requirements**

**5.1. "Passenger" Module (iOS, Android, TMA)**

* **Registration and Authentication:** Phone number (SMS confirmation), full name, date of birth, document photo (passport/ID card), selfie with document (online verification). Optional: Google/Apple ID. Profile management.
* **Search and Booking Trip:** "From-To" (auto-suggestions), date/time, number of passengers. View trips (map/list). Filters (price, time, car, rating, amenities). Trip details (route, driver, car, price, cabin layout with seat selection - front, rear window/middle). Option to "Buy entire cabin". Booking. Confirmation.
* **Ordering Cargo/Document Transportation:** Separate order for cargo. Dimensions, weight, type, value, photo. Selection of drivers with the service. Separate pricing.
* **Trip Management:** Active/completed trips. Driver tracking. Booking cancellation (possible penalties). Chat/call to driver (through the platform).
* **Payment:** Integration with Uzcard, Humo, Click, Payme. Online card payment. Cash payment. Payment history.
* **Ratings and Reviews:** Rating of driver/trip (punctuality, politeness, driving, cleanliness). Text review.
* **Security (in the application):** SOS button (signal with geolocation to trusted contacts/support). Share trip details.
* **Other:** Favorite routes/drivers. Notifications. Promo codes, referral system. FAQ, support.

**5.2. "Driver" Module (iOS, Android, TMA)**

* **Registration and Verification:** Similar to passenger + photo of driver's license, vehicle registration, car photos (exterior, interior, trunk). Car data (make, model, year, color, license plate, amenities). Moderation by administrator.
* **Creating and Managing Trips:** Route, date/time, number of seats (max. 4), price per seat/cabin. "Cargo transportation" option with price. "Going home" mode. Management of active trips, accepting/declining bookings. Trip status.
* **Navigation and Logistics:** Integration with maps. Optimal pickup/dropoff route (basic).
* **Communication:** Chat/calls to passengers (through the platform).
* **Finances:** History of trips/earnings. Platform commission. Withdrawal of funds.
* **Driver Profile:** Car data management. Schedule. Ratings/reviews.
* **Security (in the application):** SOS button.
* **Other:** Notifications. FAQ, support.

**5.3. "Administrator" Module (Web Panel)**

* **User Management:** List, driver verification (documents/car), editing, blocking/unblocking. Action history.
* **Trip and Order Management:** Monitoring, details, dispute resolution, manual editing/cancellation.
* **Financial Management:** Commission setup. Transactions. Payments to drivers. Reports.
* **Content Management:** FAQ, news, rules. Promotional campaigns, promo codes.
* **Analytics and Reports:** Statistics (users, trips, routes). Income. Geoanalytics.
* **System Management:** Parameters. Administrator roles. Action logging.
* **User Support:** Ticket system. Inquiry history.

**5.4. Common Functions for All User Applications:**

* Geolocation. Push notifications. Offline mode (basic). Search.

**5.5. Specific Functions (Intercity):**

* **Seat Selection in the Car:** Visual cabin layout.
* **Car Filling Logic:** Booking 1-4 seats. Price per seat/cabin.
* **Cargo Transportation:** Separate service, pricing by driver.
* **Pickup/Dropoff Optimization:**
  + MVP: Driver sets pickup/dropoff points.
  + Prospect (v2.0+): AI/ML for dynamic optimization of pickup/dropoff route (minimizing time/deviations).

**6. Security Requirements (Policies and Functionality)**

**6.1. User Verification:**

* Thorough verification of driver documents (passport/ID, driver's license, vehicle registration) and cars.
* Verification of driver's photo with document and selfie with document.
* Similar verification of passengers (passport/ID, selfie with document).
* Automated (with AI elements) and manual verification.

**6.2. Trip Safety:**

* SOS button in applications for driver and passenger (transmission of geolocation and trip data to trusted contacts and/or platform dispatch service).
* "Share trip" function (tracking for third parties).
* Phone number anonymization (calls through the platform).
* "Blacklist" system for unscrupulous drivers/passengers.

**6.3. Fraud Protection:**

* Monitoring of suspicious activity (frequent cancellations, fake accounts).
* Anti-fraud system for payments.

**6.4. Personal Data Processing Policy:**

* Full compliance with the legislation of the Republic of Uzbekistan on personal data.
* Transparent privacy policy for users.

**7. Non-functional Requirements**

**7.1. Performance:**

* Interface response time: no more than 2 seconds for standard operations.
* Trip search time: no more than 5 seconds under average loads.
* Target indicators are detailed in section 9.7.

**7.2. Reliability and Availability:**

* Service availability: 99.9%.
* Regular database backup (RPO < 15 min, RTO < 1 hour).
* Mechanisms for quick recovery after failures.

**7.3. Scalability:**

* Horizontal scaling for backend services and DB.
* Architecture should allow a 10-fold increase in load without major redesign.
* Target indicators are detailed in section 9.7.

**7.4. Usability:**

* Intuitive interface, minimal number of steps for key actions.
* Responsive design for the administrator web panel.
* Consistent style and UX across all platforms following guidelines.
* Good readability, contrast, feedback on actions.

**7.5. Maintainability:**

* Detailed technical documentation.
* Ease of application updates.
* Error and event logging. Monitoring tools.

**8. Design Requirements (UI/UX)**

**8.1. General Concept:** Modern, clean, minimalistic design. Trust, reliability. Emphasis on convenience. Adaptation to cultural characteristics (without compromising modernity).

**8.2. Color Palette:**

* Primary: Calm blue (#2A6FDB) or deep green (#1A8A5D).
* Accent: Bright orange (#FF7A00) or warm yellow (#FFC107) for CTA.
* Secondary: Light gray (#F0F2F5) for background, dark gray (#333333, #555555) for text, white.

**8.3. Typography:** Modern sans-serif fonts (Inter, Roboto, Open Sans, SF Pro, Noto Sans). Clear hierarchy. Comfortable spacing.

**8.4. Iconography:** Unified style (Material Design Icons, Feather Icons, or custom). Intuitive. Clear.

**8.5. Animations and Transitions:** Smooth, unobtrusive. Loading animations, transitions, interactive elements. Without slowdown.

**9. Technology Stack and Technical Specifications**

**9.1. Backend**

* **Language/Framework:**
  + **Recommendation 1: Python + FastAPI** (High performance, type hints, auto-documentation OpenAPI, ASGI, Pydantic).
  + **Recommendation 2: Node.js + NestJS (TypeScript)** (Event model, TypeScript, modularity, DI).
  + **Alternative (high-load microservices): Go (Golang)** (Performance, parallelism).
* **API Gateway:** Kong, Nginx (with Lua), or cloud solutions (AWS API Gateway, Google Cloud Endpoints).
* **Message Broker:** RabbitMQ (flexibility, AMQP) or Apache Kafka (streaming, high throughput).
* **Database (OLTP): PostgreSQL (latest stable version)**.
  + **PostGIS** extension for geodata. ACID. JSONB. Replication (Master-Slave).
* **Caching: Redis (latest stable version)**. In-memory. Redis cluster.
* **Full-text search: Elasticsearch or OpenSearch**.
* **Real-time communication: WebSockets** (Python websockets, Node.js Socket.IO).

**9.2. Mobile Applications (iOS and Android) - "AutoPort"**

* **Approach: Cross-platform.**
  + **Recommendation 1: Flutter (Dart)**. SDK: latest stable. Architecture: BLoC/Cubit or Riverpod. Navigation: go\_router. Network: dio. Local storage: sqflite/shared\_preferences/isar. Maps: yandex\_mapkit (preferred for Uzbekistan) or google\_maps\_flutter. Push: firebase\_messaging.
  + **Recommendation 2: React Native (TypeScript)**. SDK: latest stable. State: Redux Toolkit/Zustand. Navigation: React Navigation. Network: axios. Maps: react-native-maps. Push: react-native-firebase.
* **General Specifications:**
  + Min OS: iOS 13.0+, Android 6.0+.
  + Offline mode (basic). Deep Linking. Responsive UI.
  + Testing: Unit, Widget/Component, Integration, E2E (Appium/Detox).
  + Crash collection/analytics: Firebase Crashlytics/Analytics, Sentry.
  + Performance optimization (60 FPS), resources, application size.

**9.3. Telegram Mini App (TMA) - "AutoPort"**

* **Technologies:** Web technologies + Telegram Web Apps SDK.
  + JS Framework: React.js, Vue.js, or Svelte (with TypeScript).
  + Bundler: Vite or Webpack.
  + UI: Lightweight UI libraries or custom, @twa-dev/sdk.
  + CSS: Tailwind CSS, CSS Modules.
  + Specification: Integration with Telegram UI/API, fast loading, secure interaction (initData).

**9.4. Administrator Web Panel - "AutoPort"**

* **Technologies:**
  + JS Framework: React.js (with TypeScript). Alternatives: Angular, Vue.js.
  + UI library: Material-UI (MUI) or Ant Design.
  + State: Redux Toolkit/Zustand. Charts: Chart.js/Recharts.
  + Specification: Responsive design, RBAC, audit, performance with big data.

**9.5. Infrastructure and DevOps**

* **Cloud Provider:** AWS, GCP, Azure (managed services).
* **Containerization: Docker.**
* **Orchestration: Kubernetes (K8s)** (managed EKS, GKE, AKS).
* **CI/CD:** GitLab CI/CD, GitHub Actions, Jenkins. Automated builds, tests, deployment (Blue/Green, Canary).
* **Monitoring/Logging:** Prometheus + Grafana (metrics). ELK/EFK Stack (logs). Sentry/Dynatrace (APM).
* **Infrastructure as Code (IaC): Terraform or Pulumi.**
* **CDN:** Cloudflare, AWS CloudFront. For static assets and DDoS protection.

**9.6. Technical Security Aspects**

* **Encryption:** TLS 1.3 (HTTPS, HSTS). Data encryption at rest (pgcrypto).
* **Authentication/Authorization:** OAuth 2.0 (JWT). Password hashing (Argon2/scrypt/bcrypt + salt). 2FA/MFA (mandatory for admins, optional for users).
* **API Protection:** OWASP Top 10. Validation. Rate limiting, brute force protection.
* **Vulnerability Scanning:** SAST/DAST.
* **Secret Management:** HashiCorp Vault.
* **Network Security:** Firewalls, VPC, segmentation.

**9.7. Performance and Scalability Targets**

* **API Response Time:** P95 < 200ms (key operations), P99 < 500ms.
* **Mobile App Load Time (cold start):** < 3 seconds.
* **TMA Screen Load Time:** < 2 seconds.
* **Throughput (RPS):** Start: 1000 RPS, scaling to 10,000+ RPS.
* **Concurrent Users (DAU):** Start: 10,000 DAU, with scaling.
* **DB Scaling:** Read replicas, readiness for sharding.

**9.8. Code Quality and Documentation**

* **Coding Standards:** Unified (PEP 8, Effective Dart, etc.).
* **Code Review:** Mandatory.
* **Test Coverage:** Unit > 80%, Integration > 60%.
* **Documentation:** API (Swagger/OpenAPI), architectural, code comments.

**10. Integrations**

* **Mandatory:**
  + Map services (Yandex Maps API / Google Maps API).
  + Payment gateways (Uzcard, Humo, Click, Payme).
  + SMS gateway (local provider).
  + Push notification services (FCM, APNS).
* **Desirable/Prospective:**
  + Analytics systems (Firebase Analytics, Amplitude/Mixpanel).
  + Government services for automatic document verification (if API is available).
  + Accounting systems.

**11. Development Stages and Acceptance**

The project is implemented iteratively (Agile/Scrum):

1. **MVP (Minimum Viable Product) - (~3-4 months):** Basic functionality for Passenger (registration, search, booking 1 seat, cash payment), Driver (registration, trip creation, viewing bookings), Admin panel (manual verification, trip viewing). One mobile platform (Android) + TMA.
2. **Iteration 2 - Functionality Expansion (~2-3 months):** Full functionality for Passenger/Driver (seat selection, online payment, cargo, ratings, SOS). Second mobile platform (iOS). Extended Admin panel (finances, analytics). Partially automated verification.
3. **Iteration 3 - Optimization and Additional Features (~2-3 months):** Logistics optimization. Promo codes, referral system. UI/UX improvement. Enhanced security. AI algorithm development (if decided).

**Acceptance of each stage:** Demonstration, testing (functional, load, usability, security), compliance with specifications, provision of source code and documentation.

**12. Glossary**

* **Platform:** Software and hardware complex "AutoPort".
* **TMA (Telegram Mini App):** Web application in Telegram.
* **Verification:** Authentication of data.
* **SOS button:** Emergency notification function.
* **CTA (Call To Action):** Call to action.
* **API:** Application Programming Interface.
* **RPO (Recovery Point Objective):** Maximum allowable period of data loss.
* **RTO (Recovery Time Objective):** Maximum allowable recovery time.
* **DAU (Daily Active Users):** Number of unique active users per day.
* **RPS (Requests Per Second):** Number of requests per second.

**Conclusion:**

This Technical Specification is the basis for the development of the "AutoPort" platform. During development, it is possible to clarify and supplement the requirements by agreement of the parties. A proactive approach from the development team, suggestions for optimal technical solutions, and high quality of the final product are expected.