# **Taxi Tap: Comprehensive Technical Design Document**

## **1. Project Overview**

**Taxi Tap** is a cloud-native, mobile-first platform designed to digitize South Africa’s minibus taxi industry. The system preserves the dynamic, flexible nature of taxis while modernizing the ecosystem with real-time GPS-based routing, vehicle tracking, and efficient communication between drivers and passengers. Built on a scalable, serverless architecture, Taxi Tap is optimized for low data usage, offline capabilities, and seamless user experience.

## **2. Technology Stack Justification**

### **Frontend: Expo (React Native with TypeScript)**

#### **Why Expo?**

* **Cross-platform Compatibility**: Code once, deploy to both Android and iOS.
* **Native Features**: Access to GPS, accelerometer, push notifications, offline storage, camera, QR scanning, etc.
* **Web Support**: Leverages Expo Web for rendering web-based dashboards and admin panels.
* **Live Reloading & Fast Iteration**: Expo Go provides hot reloading and rapid prototyping with a unified development experience.
* **Battery & Data Optimization**: React Native ecosystem provides fine-grained control over performance, reducing overhead.

### **Backend: Convex (TypeScript)**

#### **Why Convex?**

* **Truly Serverless**: No provisioning, no scaling headaches. Functions, database, and auth all run in one integrated environment.
* **Built-in Database**: Convex provides a powerful document-oriented database that supports relations, IDs, indexes, and real-time reactivity.
* **Type Safety**: Schema definition is in TypeScript, ensuring end-to-end type safety from backend to frontend.
* **Zero DevOps**: No need to manage infrastructure or containers. Deploy directly from your project.
* **Realtime Sync**: Built-in support for reactive queries allows passengers to see live taxi updates, seat availability, and ETA.

#### **Convex Database Architecture**

* **Document Store**: Convex uses collections of JSON-like documents, similar to MongoDB, but with built-in schema validation.
* **Indexes**: Automatic indexing on IDs and custom indexing for optimized query performance.
* **Relationships**: You can use Convex v.id() to reference documents between tables, ensuring referential integrity.
* **Realtime Subscriptions**: Query results update automatically when the underlying data changes.

### **Convex Free Tier (as of 2025)**

* **Compute**: Up to 1 million function calls/month.
* **Storage**: 1 GB document data storage.
* **Bandwidth**: 5 GB of egress.
* **Authentication**: Integrated with third-party auth providers (Firebase Auth, Clerk, etc.).
* **Deployment**: 1 Production Deployment and 1 Dev Deployment per project.

**Perfect for COS 301:** Within budget, no surprise bills, and production-grade scalability.

## 

## **3. Key Functional Modules & Implementation Plan**

### **3.1 User Management Subsystem**

* **Authentication**: Convex Auth with Clerk or Firebase integration.
* **Registration/Login**: Role-based registration (passenger or driver) with schema enforcement.
* **Profile Updates**: Mutation to update user document with profile fields.
* **Security**: JWT-based session validation, encryption at rest and in transit.

### **3.2 Location Services Subsystem**

* **Driver Location**: Periodic GPS updates using Expo Location API.
* **Passenger Location**: One-time or continuous tracking during trip.
* **Proximity Alerts**: Triggered from Convex using background function.
* **ETA Calculation**: Naive approach using Haversine distance + average speed (no Google Maps API due to cost).

### **3.3 Taxi Request Subsystem**

* **Request Workflow**:  
  + Passenger sends request with coordinates and optional destination.
  + Nearby drivers notified (push notification via Expo).
  + Driver accepts or rejects request.
  + Status changes handled in real time.

### **3.4 Route Management Subsystem**

* **Driver Route Declaration**: Input form for common route + destination.
* **Passenger View**: Map view of taxis on route + destinations.
* **Optimized Routing (Optional)**: Historical route optimization using stored patterns (stretch goal).

### **3.5 Notification System**

* **Technology**: Expo Notifications API.
* **Use Cases**:  
  + Taxi is approaching.
  + Ride accepted or declined.
  + Route changes or delays.
* **Offline Support**: Caching notifications locally using AsyncStorage.

### **3.6 Safety and Fare Management Subsystem**

* **QR Identification**: QR codes linked to taxi documents in Convex.
* **Reporting**: Anonymous incident reports saved to a secure Convex table.
* **Fare Estimate**: Static fare matrix per route (e.g., km-based fare slabs).
* **Payment**: Optional - integrate with SnapScan/Yoco for digital payments.

## **4. Architecture Overview**

### **Architectural Patterns**

* **Client-Server Pattern**: Expo App (Client) → Convex (Serverless Backend)
* **Repository Pattern**: Functions encapsulate data logic, mutations are business logic units.
* **Reactive Pattern**: Realtime reactivity with Convex reactive queries (like GraphQL subscriptions).

### **Quality Requirements**

* **Security**: End-to-end encryption, role-based auth, POPIA compliance.
* **Scalability**: Convex autoscales functions and DB storage seamlessly.
* **Performance**: Edge-deployed backend, low-latency function calls.
* **Maintainability**: Typed schemas and well-separated function files improve code clarity.
* **Accessibility**: Simple UI, multilingual support, clear icons and high contrast.

## **5. DevOps and CI/CD**

### **Version Control**

* GitHub repo with main and dev branches.
* Feature branches for each core module.

### **CI/CD Strategy**

* **Convex Deployment**: Triggered via GitHub Action or manual npx convex dev / convex deploy.
* **Expo Deployment**: Use eas build + eas submit for App Store/Play Store releases.
* **Linting & Tests**: Pre-commit lint checks with ESLint + Jest unit tests.

### **Testing Frameworks**

* **Backend**: Jest (unit and integration tests for Convex functions).
* **Frontend**: React Native Testing Library.
* **Manual Testing**: Device tests using Expo Go and emulators.

## 

## **8. Comparative Analysis of Tech Stack Choices**

### **Current Stack: Expo + Convex (TypeScript)**

* **Pros**:
  + Serverless simplicity = No infrastructure management.
  + Tight end-to-end TypeScript support.
  + Realtime reactivity out-of-the-box.
  + Blazing fast development cycle.
  + Fits student + startup context (free tier, fast iteration).
* **Cons**:
  + Less control than AWS/Firebase.
  + Not battle-tested at global enterprise scale yet.
  + Smaller ecosystem/community for advanced integrations.

### **Firebase (Google Cloud)**

* **Pros**:
  + Great auth, real-time database, Firestore, analytics.
  + Deep Google Cloud integrations (Maps, ML, Hosting).
  + Easy to prototype and scale to mid-size apps.
* **Cons**:
  + Pricing gets expensive at scale.
  + Vendor lock-in with proprietary tech stack.
  + Less type safety compared to Convex (JavaScript-centric).

### **Supabase (Open Source Firebase Alternative)**

* **Pros**:
  + SQL-based (PostgreSQL), great if you prefer relational models.
  + Open-source and self-hostable.
  + Integrates well with React.
* **Cons**:
  + No built-in real-time sync like Convex.
  + Requires more setup and backend expertise.
  + Less mobile-native SDK support than Firebase or Expo.

### **AWS (Native Services)**

* **Pros**:
  + Highly scalable and reliable.
  + Full control over compute, storage, networking.
  + Great for enterprise-grade deployment.
* **Cons**:
  + Overkill for early-stage/startup/student apps.
  + Requires managing IAM roles, VPCs, Lambda complexity.
  + DevOps burden is higher; learning curve is steep.

## **Summary**

Taxi Tap leverages modern serverless technology (Convex), native mobile power (Expo), and the strength of the React + TypeScript ecosystem to build a revolutionary solution tailored to South Africa’s unique transport challenges. With scalability, performance, and usability baked into the design, Taxi Tap is positioned not just as a project, but as a transformative platform for millions of users.

**Let's Git It Done !**