NYCPS Transportation Management System - Step-by-Step Execution Plan (RFP R1804)

Introduction

This document provides a high-level, step-by-step guide for executing the development and deployment of the NYCPS Transportation Management System on AWS GovCloud (US). It outlines the major phases, key activities, and considerations involved, assuming an agile development methodology within each phase. This plan aligns with the Solution Design and Architecture documents and aims to meet the 12-month timeline outlined in the Project Plan.

Note: This is a strategic execution plan. Detailed task breakdowns, resource assignments, and specific sprint planning will occur within each phase.

Phase 0: Setup & Foundation

Objective: Establish secure and compliant AWS GovCloud environment and core project infrastructure.

1. **AWS GovCloud Account Setup:**
    - Establish necessary AWS GovCloud (US) accounts (potentially separate for Dev/Test/Prod).
    - Configure billing, cost management alerts, and organizational structure (e.g., using AWS Organizations).
    - Engage AWS account managers and solution architects for GovCloud best practices.

2. **Identity & Access Management (IAM):**
    - Define and implement IAM roles, groups, and policies based on the principle of least privilege.
    - Set up MFA for all console/API users.
    - Configure identity federation (SAML 2.0) with NYCPS Identity Provider if feasible.
    - Establish break-glass access procedures.
3. **Networking Foundation:**
    - Design and provision VPCs spanning multiple AZs for Dev, Test, and Prod environments.
    - Configure subnets (public/private), route tables, NAT Gateways, and Internet Gateways.
    - Implement Security Groups and Network ACLs baseline rules.
    - Set up VPC Endpoints for private access to AWS services.
    - Establish secure connectivity (Direct Connect/VPN) to NYCPS data centers/systems.
4. **Security Baseline Configuration:**
    - Enable CloudTrail, Config, GuardDuty, Security Hub across all accounts/regions.
    - Configure centralized logging to S3.
    - Set up baseline CloudWatch alarms for security events.
    - Deploy WAF and configure initial rulesets.
    - Configure KMS keys (CMKs).
5. **CI/CD Pipeline Setup:**
    - Set up source code repositories (e.g., CodeCommit).
    - Configure CodePipeline, CodeBuild (including security scanning steps - SAST/DAST), and CodeDeploy/ECS/EKS deployment processes.
    - Define deployment strategies (e.g., blue/green).
6. **IaC Foundation:**
    - Select and configure IaC tool (CloudFormation/CDK/Terraform).

- Develop initial templates/modules for core infrastructure (VPC, IAM roles, security groups).

Phase 1: Core Infrastructure & Data Ingestion

Objective: Build the foundational data pipelines and storage for receiving and handling GPS data.

1. **Data Streaming Pipeline:**
    - Provision Kinesis Data Streams or MSK cluster.
    - Develop/Configure ingestion endpoints (API Gateway + Lambda/Fargate or IoT Core rules).
    - Implement initial data validation and transformation logic (Lambda/Fargate).
    - Set up S3 buckets for raw data storage with appropriate lifecycle policies.
2. **Core Databases:**
    - Provision RDS PostgreSQL instance(s) with PostGIS extension (Multi-AZ). Configure backups, security groups.
    - Provision DynamoDB tables for real-time location/status data. Configure backups, capacity mode (provisioned/on-demand).
    - Provision ElastiCache cluster (Redis/Memcached) for caching.
3. **Basic Device Management:**
    - Set up database schema/tables for device inventory.
    - Develop basic APIs for device registration and status updates.
4. **Archival Storage:**
    - Configure S3 lifecycle policies to transition data to Glacier/Deep Archive for 7-year retention.
5. **Testing:** Unit test ingestion endpoints, data transformations. Test database connectivity and basic CRUD operations. Test data archival policies.

Phase 2: Foundational Modules & Device Deployment

Objective: Develop minimal viable backends for Driver and Admin modules, deploy initial devices, and test end-to-end GPS data flow.

1. **Backend Service Development (Microservices):**
   - Develop initial microservices for Driver Module backend (login, route association, basic GPS data forwarding). Deploy using Lambda/Fargate.
   - Develop initial microservices for OPT Admin Module backend (basic device/bus map view, user management stubs). Deploy using Lambda/Fargate.
   - Set up API Gateway endpoints for these services.
2. **Bus Driver App (Core):**
   - Develop core native app functionality (iOS/Android): Secure login, route/vehicle association, background GPS tracking/transmission.
3. **Initial Device Provisioning & Deployment:**
   - Procure initial batch of GPS-enabled devices (tablets/phones).
   - Configure devices (OS hardening, install Driver App, MDM enrollment if used).
   - Coordinate with pilot SBCs for initial installations and setup (mounting, power).
4. **End-to-End Testing (Pilot):**
   - Test GPS data flow from pilot devices through ingestion pipeline to databases and basic OPT Admin map view.
   - Verify device status reporting and basic inventory tracking.
   - Test driver login and route/vehicle association process.

5. **Security Review:** Conduct initial security review of deployed components and data flows.

Phase 3: Routing & Ridership Core

Objective: Develop the core logic for the routing engine and ridership recording, integrating essential data sources.

1. **GIS Integration:**
    - Load and configure NYC LION ArcGIS data into RDS PostgreSQL/PostGIS or dedicated ESRI environment.
    - Develop services/functions for geocoding, reverse geocoding, and basic map data queries.
2. **Routing Engine (Core Logic):**
    - Develop algorithms for initial route generation based on static data (student locations, stops, schedules, basic constraints). Deploy on EC2/ECS/EKS.
    - Implement shortest path calculations and basic ETA estimation.
    - Develop services for creating/updating/retrieving route plans.
3. **Ridership Backend:**
    - Develop microservices to process boarding/alighting events (received via API Gateway).
    - Implement logic to associate events with students/stops/routes using data from Student Management.
    - Store ridership records in DynamoDB/RDS.
4. **Student Data Integration (Initial):**
    - Establish secure connections/APIs to pull necessary student roster, location, and basic needs data from relevant NYCPS systems (ATS, NPSIS, etc.).
    - Develop ETL processes (e.g., using Glue) or Lambda functions to ingest and synchronize student data into the Student Management backend database.
5. **Stop & Session Time Management:**

- Develop backend components to store and manage stop locations/rules and school session times/calendars.
- Integrate this data into the Routing Engine.

6. **Testing:** Test routing algorithm accuracy with sample data. Test ridership event processing. Validate student data integration. Test GIS query performance.

Phase 4: User Modules Rollout

Objective: Develop user-facing applications and enhance existing modules with core functionality.

1. **Parent/Student App Development:**
    - Develop native apps (iOS/Android) and optional web app.
    - Implement features: User signup/login, map view with real-time bus location/ETA, notification display, student QR code display, absence reporting (Parent), basic info update requests (Parent).
    - Integrate with backend APIs for data and notifications.
2. **School Admin Module Development:**
    - Develop web application interface.
    - Implement features: View routes/buses/students serving the school, receive relevant alerts, basic reporting dashboard.
    - Integrate with backend APIs.
3. **Driver App Enhancements:**
    - Implement turn-by-turn navigation feature (integrating with Routing Engine output).
    - Implement ridership scanning/manual entry interface (integrating with Ridership backend).
    - Implement two-way messaging/alerting interface.
4. **OPT Admin Module Enhancements:**
    - Enhance map view with ridership status, route details, filtering.

    - Implement comprehensive communication tools (messaging, alert configuration/initiation).
    - Develop initial KPI dashboards and reporting interfaces (stubbed).
    - Implement RBAC and user management interface.
5. **Testing:** UI/UX testing, functional testing of all module features, cross-platform testing (web/mobile), accessibility testing (WCAG).

Phase 5: Integration & Enhancement

Objective: Integrate with remaining external systems, enhance core engines with dynamic capabilities, and build out reporting.

1. **External System Integrations:**
    - Develop and test integrations with NYCPS Ticketing System, payment processing systems (if applicable), external messaging platforms, etc., using the API integration layer.
    - Integrate real-time traffic data feeds into the Routing Engine.
2. **Routing Engine Enhancements:**
    - Implement dynamic route adjustment logic based on real-time GPS, traffic, and ridership data.
    - Refine ETA calculations using real-time data.
    - Implement conflict detection logic and alert generation.
    - Build out "what-if" scenario planning capabilities.
3. **Notification System Enhancements:**
    - Implement full range of configurable automated notifications.
    - Integrate robocall functionality (e.g., via Amazon Connect).
    - Refine multi-language support.
4. **Reporting & Analytics Build-out:**
    - Develop required canned reports (NYC Council compliance).
    - Build interface for custom report generation.

    - Populate dashboards with relevant KPIs.
    - Set up data lake/warehouse structures (S3, Glue, Redshift) and ETL processes.
5. **Testing:** Integration testing with external systems, end-to-end testing of dynamic routing scenarios, report validation.

Phase 6: Testing & Hardening

Objective: Ensure system robustness, security, performance, and compliance before full rollout.

1. **Comprehensive Integration Testing:** Test all components and integrations together.
2. **Performance & Load Testing:** Simulate peak load conditions (e.g., start/end of school day) to validate performance against RFP requirements (response times, throughput). Identify and remediate bottlenecks.
3. **Security Testing:**
    - Conduct independent third-party penetration testing (external and internal).
    - Perform thorough vulnerability scanning (SAST, DAST, IAST integration in CI/CD).
    - Remediate all critical/high vulnerabilities.
4. **User Acceptance Testing (UAT):** Engage OPT staff, pilot school admins, drivers, and potentially parents/students to test functionality and usability. Gather and address feedback.
5. **Accessibility Audit:** Conduct third-party WCAG 2.0 AA compliance audit and remediate findings.
6. **Disaster Recovery Testing:** Perform planned DR failover test to validate RPO/RTO capabilities.
7. **Documentation Finalization:** Finalize technical documentation, user manuals, training materials, SOPs, Business Continuity Plan, Disengagement Plan.

Phase 7: Phased Deployment & Training

Objective: Roll out the system incrementally to the full user base and conduct comprehensive training.

1. **Finalize Rollout Plan:** Refine the phased rollout strategy (e.g., by borough, by SBC, by student type) based on pilot results and UAT feedback, in coordination with OPT.
2. **Hardware Deployment:** Complete procurement, configuration, and installation of all required devices across the entire fleet according to the rollout plan.
3. **Software Deployment:** Deploy finalized application versions to production environments.
4. **Training Delivery:**
    - Execute comprehensive training plan for all user groups (drivers, attendants, OPT staff, dispatchers, school admins, parents, students) using agreed methods (virtual, in-person, train-the-trainer).
    - Distribute user manuals and support materials.
5. **Phased Go-Live:** Execute the rollout plan, closely monitoring system performance, user adoption, and support channels during each phase. Provide hypercare support during initial rollout stages.

Phase 8: Operations & Optimization

Objective: Transition to steady-state operations, ongoing monitoring, maintenance, and continuous improvement.

1. **Operational Handover:** Formally transition system monitoring and Tier 1/2 support responsibilities (as defined in SLA) to relevant NYCPS/OPT teams, supported by vendor Tier 3.

2. **Ongoing Monitoring & Maintenance:** Continuously monitor system health, performance, and security using CloudWatch, Security Hub, etc. Apply patches and updates per maintenance schedule.
3. **Performance Tuning:** Analyze performance data and optimize database queries, caching strategies, and algorithm parameters.
4. **Iterative Improvement:** Gather ongoing user feedback. Plan and deploy regular updates with minor enhancements and bug fixes based on feedback and operational data.
5. **Reporting & Review:** Generate regular operational and compliance reports. Conduct periodic reviews with OPT stakeholders to assess system effectiveness and plan future enhancements.
6. **Knowledge Transfer & Disengagement Prep:** Continue knowledge transfer to NYCPS teams per maintainability requirements. Keep Disengagement Plan updated.