NYCPS TMS: Prescriptive Human Resources & Team Management Strategy

I. Introduction: Philosophy, Goals & Scope

This document mandates the comprehensive Human Resources (HR) and Team Management Strategy for the NYCPS Transportation Management System (TMS) project. Recognizing that the success of this complex, large-scale, geographically distributed project hinges critically on its people, this strategy establishes best-in-class processes for team structuring, talent acquisition, onboarding, development, performance management, collaboration, knowledge retention, and fostering a high-performance, positive, and compliant team culture.

Philosophy: We believe in empowering autonomous, cross-functional teams, fostering psychological safety, promoting continuous learning, demanding accountability through clear expectations and objective metrics, and ensuring seamless collaboration despite geographical separation. This strategy integrates HR and management practices directly into the Agile/DevSecOps workflow.

Goals:

- Attract, onboard, and retain top-tier engineering, product, and support talent globally.
- Ensure every team member possesses the necessary skills and clear understanding of their role, responsibilities, and project objectives.
- Foster a highly collaborative, inclusive, and productive environment across time zones.
- Implement objective, data-driven performance management focused on both output and outcomes (quality, reliability, value delivery).
- Guarantee knowledge continuity throughout the project lifecycle, minimizing impact from personnel changes.
- Ensure strict adherence to all NYCPS HR policies, labor laws, and compliance regulations.

- Achieve elite DORA metrics and project KPIs through effective team structure and management.

Scope: This plan covers all personnel directly assigned to the NYCPS TMS project (estimated 50-60 members, vendor and potentially designated NYCPS counterparts), including developers, QA, SRE/Ops, security, architects, PMs, POs, BAs, UX designers, trainers, support tiers, and leadership, from project initiation through operational handover and potential offboarding.

II. Team Structure, Roles & Responsibilities (RACI)

We will organize the 50-60 person team into cross-functional, domain-aligned Agile (Scrum) teams supported by specialized platform/central teams to optimize focus, collaboration, and delivery flow.

A. Proposed Team Structure (Example - To be finalized)

- **Feature Teams (Multiple):** Aligned with major functional domains/microservice groups (e.g., Team Apollo - Routing Engine; Team Juno - Parent/Student Apps; Team Minerva - Ridership & Driver App; Team Vulcan - Admin Consoles & Reporting).
    - Composition: Scrum Master, Product Owner (NYCPS designee liaison), Tech Lead, Backend Devs, Frontend/Mobile Devs, QA Engineers/SDETs.
- **Platform Teams (Centralized):** Provide foundational services and expertise across feature teams.
    - **DevOps/SRE Team:** Manages CI/CD pipelines, AWS infrastructure (via IaC), monitoring/alerting, incident response, operational reliability.
    - **Security Team (DevSecOps):** Manages security tooling, performs security reviews/testing, consults on secure design, manages compliance posture.
    - **Data Platform/GIS Team:** Manages core data stores (DBA functions), data integration pipelines, GIS platform maintenance, reporting/analytics infrastructure.

- **Architecture Guild:** Cross-team group of senior architects/leads providing architectural guidance, standards, and review (not a full-time team).

- **Project Leadership & Support:**
  - Project Managers (Vendor & NYCPS), Program Manager (Overall Lead), Core UX Design Team, Business Analysts, Technical Writers, Training Team, Tiered Support Teams (L1/L2/L3 - see Ops Strategy).

Implementation How-To:

1. Finalize team structure and domain alignment based on architecture and initial backlog priorities during project initiation.
2. Clearly document team charters, missions, and primary responsibilities in Confluence.
3. Assign initial team members based on skills and experience.

Responsibility: Project Leadership, Management.

B. Detailed Roles, Responsibilities & Expectations

Clear expectations are crucial. Below are key roles and their core responsibilities/expectations within our framework:

- **Project Manager (Vendor/NYCPS):** Overall project execution, schedule management, budget tracking, risk/issue management facilitation, change management process ownership, stakeholder communication, governance reporting, resource coordination. **Expectation:** Proactive planning, clear communication, rigorous adherence to PM processes, ensures alignment between vendor/NYCPS goals.
- **Product Owner (NYCPS Designee):** Owns/prioritizes Product Backlog, defines user story acceptance criteria, represents business/user needs, accepts/rejects completed work in Sprint Reviews, available for team clarification. **Expectation:** Decisive prioritization, clear communication of requirements, active participation in Agile ceremonies.
- **Scrum Master:** Facilitates Scrum ceremonies, removes team impediments, coaches team on Agile/Scrum practices, shields team from external distractions, tracks sprint metrics. **Expectation:** Servant leadership, process expert, effective impediment remover, fosters team self-organization.

- **Technical Lead (Feature Team / Platform):** Guides technical design/implementation within the team/domain, mentors developers, ensures adherence to architecture/coding/security standards, leads code reviews, resolves technical blockers. **Expectation:** Deep technical expertise, strong communication/mentoring skills, ownership of technical quality for their domain.
- **Software Engineer (Backend/Frontend/Mobile):** Designs, develops, tests (Unit, Integration, API/Component), and maintains code according to user stories and standards. Participates actively in code reviews, Agile ceremonies, and troubleshooting. Owns quality of their deliverables. **Expectation:** High-quality secure code, comprehensive automated tests, collaborative attitude, continuous learning, adherence to process.
- **QA Engineer / SDET:** Develops/maintains test strategy & plans, builds/maintains automated test frameworks (API, E2E, Perf), performs exploratory testing, manages defect tracking/triage, facilitates UAT, certifies release quality. **Expectation:** Strong testing/automation skills, critical thinking, quality advocacy, collaboration with Dev/PO.
- **DevOps Engineer:** Builds/maintains CI/CD pipelines, automates infrastructure provisioning (IaC), manages configuration, implements monitoring/alerting tooling. **Expectation:** Expertise in GitLab CI/CD, Terraform, AWS, scripting, automation focus.
- **SRE Engineer:** Focuses on production reliability, availability, performance. Defines/monitors SLOs, manages incident response, leads post-mortems, automates operational tasks (toil reduction), capacity planning, DR testing. **Expectation:** Deep operational expertise, strong troubleshooting, automation skills, focus on reliability metrics.
- **Security Engineer (DevSecOps):** Integrates/manages security tools (SAST/DAST/SCA), performs security reviews (code, architecture), consults on secure design, triages security findings, coordinates pen tests, monitors security posture. **Expectation:** Expertise in application/cloud security, secure coding, security tools, risk assessment.
- **Cloud Architect:** Defines/governs overall AWS GovCloud architecture, sets technical standards, reviews major designs, advises on service selection and cost optimization.
- **UX/UI Designer:** Creates user flows, wireframes, mockups, prototypes. Conducts usability testing. Ensures consistent and accessible user experience based on NYCPS guidelines.
- **Business Analyst:** Elicits, analyzes, documents detailed requirements (user stories, acceptance criteria). Facilitates communication between stakeholders and development teams.
- **Technical Writer:** Creates/maintains system documentation, user manuals, API documentation, KB articles.

- **Trainer:** Develops and delivers training materials/sessions for different user groups.
- **Support Engineers (L1/L2/L3):** Follow defined processes for incident/request handling, troubleshooting, escalation, and resolution (as per Ops Strategy).

C. RACI Matrix (Responsible, Accountable, Consulted, Informed)

A detailed RACI matrix will be developed and maintained in Confluence to clarify responsibilities for key project activities and deliverables, reducing ambiguity.

Implementation How-To:

1. Create a matrix template in Confluence.
2. List key project activities/deliverables down the rows (e.g., Budget Approval, Requirements Sign-off, Architecture Design, Code Review, Unit Testing, E2E Test Execution, Production Deployment Approval, Incident Resolution, Risk Mitigation Plan).
3. List key roles across the columns.
4. Collaboratively fill in the matrix cells with R, A, C, or I for each activity/role intersection, ensuring only one 'A' per activity.
5. Review and approve the RACI matrix with project leadership. Refer to it regularly to resolve responsibility questions.

Responsibility: Project Managers, Team Leads.

Conceptual RACI Snippet:

```
| Activity | PM | PO | Tech Lead | Developer | QA | SRE/Ops | Security | SteeringCo | |---
-------------------------------|--------|--------|-----------|-----------|--------|-------
--|----------|------------| | Define User Story Accept. Criteria | C | **A** | R/C | I | C
| I | I | I | | Develop Feature Code | I | I | R/C | **R** | I | I | I | I | | Write Unit
Tests | I | I | R/C | **R** | C | I | I | I | | Conduct Code Review | I | I | **A** | R |
C | C | C | I | | Execute E2E Tests | I | C | C | I | **R** | C | I | I | | Approve
Production Deployment | C | C | C | I | C | R | C | **A** | | Resolve SEV1 Production
Incident | C | I | R/C | R/C | C | **A** | C | I | | ... | ... | ... | ... | ... | ... |
... | ... | ... |
```

III. Geographical Distribution Strategy

Managing a 50-60 member globally distributed team requires specific strategies to maintain alignment, collaboration, and productivity across time zones.

Implementation How-To:

1. **Core Collaboration Hours:** Define a daily window of 3-4 hours where maximum overlap occurs across major geographical locations (e.g., US East Coast, Europe, India). Schedule critical synchronous meetings (Sprint Planning, Reviews, key design sessions) during these hours where possible.
2. **Asynchronous Communication First:** Emphasize detailed communication via Jira comments, Confluence documentation, well-documented MRs, and structured Slack/Teams channels over reliance on real-time meetings for non-urgent matters. Set clear expectations for response times within asynchronous channels.
3. **Meeting Recording & Minutes:** Mandate recording of key meetings (with consent) and prompt distribution of detailed minutes with action items to ensure those unable to attend synchronously stay informed. Store centrally in Confluence.
4. **Tooling for Visibility:** Leverage Jira dashboards, GitLab activity feeds, and shared Confluence spaces to provide asynchronous visibility into progress and status across time zones.
5. **Rotate Meeting Times:** Where feasible, rotate timings for recurring meetings (e.g., stand-ups for cross-geo teams, TRB meetings) to distribute inconvenience across time zones fairly.
6. **Cultural Awareness Training:** Provide training/resources on effective cross-cultural communication and collaboration.
7. **Travel Budget (Targeted):** Allocate budget for periodic co-location of key leads or teams for critical planning, design, or integration workshops if deemed necessary (recognizing RFP constraints on T&E reimbursement).

8. **US Presence Requirement:** Ensure all personnel requiring access to AWS GovCloud resources or potentially sensitive NYCPS data physically reside within the continental United States, as per RFP requirements and potential compliance constraints. Clearly define which roles have this constraint.

Responsibility: Project Management, Team Leads, Scrum Masters.

IV. Detailed Team Onboarding Plan

A structured, comprehensive onboarding process is essential to rapidly integrate new team members into this complex project and distributed team, ensuring they become productive quickly and understand project context, tools, and processes.

A. Onboarding Goals:

- **Provide necessary project context, domain knowledge (NYCPS OPT), and architectural overview.**
- **Ensure proficiency with required tools (GitLab, Jira, Confluence, AWS GovCloud basics, IDE setup).**
- **Instill understanding of and adherence to defined development processes (Agile/Scrum, Git workflow, coding standards, testing requirements, DevSecOps pipeline).**
- **Integrate new members into their specific team and establish key relationships (Manager, Buddy, Mentor, Team Lead).**
- **Enable contribution to project tasks within the first 2-4 weeks.**

B. Structured Onboarding Program (First 4 Weeks):

Implementation How-To:

1. **Week 1: Foundation & Setup**
    - **HR Onboarding:** Standard company HR processes completed.
    - **Welcome & Introductions:** Introductions to Manager, assigned Buddy, Mentor (if applicable), and immediate team. Team welcome message/virtual coffee chat.
    - **Project Overview:** Dedicated session(s) with PM/Lead covering project goals, scope, history, key stakeholders, high-level architecture, and current status. Review core project documentation (Charter, Roadmap).
    - **Tool Access & Setup:** Grant access to Jira, GitLab, Confluence, Slack/Teams, AWS GovCloud (DEV environment initially). Guided setup of local development environment (IDE, Git, Docker, AWS CLI, project repositories) using documented procedures. Troubleshoot setup issues with Buddy/DevOps support.
    - **Process Introduction:** Overview of Agile/Scrum ceremonies, Git workflow, Code Review process, CI/CD pipeline basics, Defect Management. Review key Confluence spaces.
    - **Security & Compliance Briefing:** Mandatory initial briefing on data privacy (FERPA, NY Ed Law 2-d), secure coding basics, AWS GovCloud restrictions, and acceptable use policies.
    - **Initial Task:** Assign a simple, well-defined introductory task (e.g., fix a minor bug, add a unit test, update documentation) paired with Buddy for guidance.
2. **Week 2: Domain & Technical Deep Dive**
    - **Domain Knowledge Sessions:** Focused sessions on NYCPS OPT operations, terminology, key user personas

(drivers, parents, admins), core business processes (routing, ridership).

- **Architecture Deep Dive:** Sessions covering specific microservices, data models, APIs, and infrastructure components relevant to the new member's team/role.
- **Tooling Deep Dive:** Hands-on sessions/labs on using specific features of Jira (workflows, reporting), GitLab (MRs, pipelines, security scans), testing frameworks, debugging techniques.
- **Coding Standards Review:** Detailed review of language-specific style guides and secure coding practices. Practice via code katas or reviewing existing MRs with Buddy.
- **Continue Initial Task:** Progress on the initial task, participate in team stand-ups and code reviews (as observer initially).
- **1:1 Check-ins:** Regular check-ins with Manager and Buddy.

3. Weeks 3-4: Integration & Contribution

- **First "Real" Task/Story:** Assign a small, well-defined user story or task from the current sprint backlog, ideally for pair programming with Buddy or another team member.
- **Full Ceremony Participation:** Actively participate in Daily Stand-ups, Sprint Planning (task breakdown), Backlog Grooming (asking questions), Sprint Reviews (observing/potentially demoing small part), Retrospectives.
- **First Code Review/MR:** Submit first Merge Request with support from Buddy/Lead. Participate in reviewing

others' MRs.

- **Shadowing (Role-Specific):** Shadow relevant activities (e.g., QA shadows exploratory testing, DevOps shadows pipeline monitoring, Support shadows ticket triage).
- **Refine Understanding:** Use 1:1s and team interactions to clarify remaining questions about process, architecture, or domain.
- **Onboarding Feedback:** Provide feedback on the onboarding process itself.

C. Supporting Mechanisms & Artifacts

Implementation How-To:

1. Onboarding Portal (Confluence Space):** Create a dedicated space containing:
    - Welcome Message & Team Directory (with photos/roles).
    - Onboarding Checklists (Week 1, Week 2, etc.).
    - Links to Key Project Documentation (Charter, Architecture, Process Guides, Style Guides).
    - Tool Setup Guides & Access Request Links/Forms.
    - Schedule of introductory meetings.
    - Links to recorded training sessions/workshops.
    - FAQ section.
2. Onboarding Ticket Workflow (Jira/ADO):** Create an "Onboarding" issue type/workflow to track progress through key setup and training milestones

for each new hire. Assign tasks to relevant parties (Manager, IT, Security, Buddy).

3. Buddy System:** Assign an experienced team member (not necessarily senior) as a go-to peer resource for day-to-day questions, environment setup help, and navigating team norms during the first month. Define Buddy responsibilities clearly.

4. Mentorship Program (Optional but Recommended):** Assign a more senior engineer/lead as a longer-term mentor (distinct from direct manager) focused on career development, technical growth, and navigating broader organizational context.

5. Regular Check-ins:** Schedule mandatory check-ins: Daily quick sync with Buddy (first week), Weekly 1:1 with Manager, Bi-weekly 1:1 with Mentor (if applicable).

Responsibility: Hiring Manager (Overall process), HR (Formalities), Buddy/Mentor (Guidance), DevOps/IT/Security (Access/Tooling), PM/Leads (Project Context), New Hire (Active participation).

V. Skills Management & Continuous Training Plan

We will proactively manage team skills to ensure we possess the necessary expertise across diverse technologies (AWS GovCloud, specific languages/frameworks, security, testing tools) and address gaps through targeted training and development.

A. Skills Matrix Development & Maintenance

Implementation How-To:

1. Define Required Skills:** Based on the project's technology stack and roles, create a comprehensive list of critical technical skills (e.g., Terraform, Python/FastAPI, React Native, PostgreSQL/PostGIS, Kinesis, ECS/Fargate,

GitLab CI, Cypress, AWS Security Services) and essential soft skills (Communication, Collaboration, Problem Solving, Agile Practices). Categorize by role where appropriate.

2. Create Matrix Template:** Develop a spreadsheet or Confluence table listing skills vs. team members. Define proficiency levels (e.g., 0-Not Applicable, 1-Awareness, 2-Beginner/Learning, 3-Intermediate/Competent, 4-Advanced/Proficient, 5-Expert/Mentor).
3. **Initial Assessment:** Conduct initial skills assessment via self-assessment validated by managers/leads during onboarding or project start. Populate the matrix.
4. **Regular Updates:** Update the matrix periodically (e.g., quarterly or semi-annually) as part of performance review cycles or when significant technology changes occur. Track skill development progress.
5. Analyze Gaps:** Regularly analyze the matrix to identify critical skill gaps within teams or across the project. Use this analysis to inform the training plan.

Tools: Confluence Tables, Shared Spreadsheet (Excel/Google Sheets), potentially specialized HR/Skills Management software.

> Responsibility: Project Management/HR, Team Leads/Managers (Assessment validation), All Team Members (Self-assessment).

B. Targeted Training & Development Plan

Implementation How-To:

1. Individual Development Plans (IDPs):** As part of performance management, managers work with individuals to create IDPs focusing on developing skills needed for their role and career growth, referencing the Skills Matrix gaps.
2. Training Needs Analysis:** Aggregate identified skill gaps from the matrix and IDPs to determine project-wide training priorities.

3. Training Delivery Methods (Blended Approach):**
    - **Internal Workshops/Brown Bags:** Schedule regular sessions led by internal SMEs on project-specific architecture, tools, processes, or new technologies being adopted. Record sessions and store in Confluence.
    - **Pair Programming / Shadowing:** Encourage pairing for complex tasks or knowledge transfer between senior/junior engineers or across domains.
    - **External Training Courses:** Identify and budget for relevant online courses (e.g., AWS Training, Pluralsight, Udemy) or instructor-led training for critical skills (especially AWS GovCloud, specific security certs, advanced testing frameworks).
    - **Certifications:** Support and potentially fund relevant certifications (AWS certifications, Security certs, Scrum Master/PO certs).
    - **Documentation & Self-Study:** Encourage use of official documentation, tutorials, and allocate some time for self-directed learning. Maintain curated list of relevant resources in Confluence.
4. Tracking & Budgeting:** Allocate a specific training budget. Track training participation and effectiveness (e.g., via post-training surveys, skill assessment updates).

Responsibility: Project Management/HR (Budget/Coordination), Team Leads/Managers (Identifying needs, IDPs), Internal SMEs (Delivering internal training), All Team Members (Active participation).

VI. Performance Management & KPIs

We will implement a performance management system based on clear expectations, continuous feedback, and objective data (KPIs) tied to both individual contributions and team/project outcomes.

A. Defining Expectations

Implementation How-To:

1. **Document clear Role Descriptions (as outlined in Section II.B) detailing core responsibilities and expected competencies for each role on the project.**
2. **Set specific, measurable, achievable, relevant, time-bound (SMART) goals for individuals and teams aligned with project milestones and objectives during performance planning cycles (e.g., quarterly or semi-annually). Goals should cover both "what" (deliverables, KPI targets) and "how" (collaboration, adherence to process, skill development).**

Responsibility: Management, Team Leads, HR.

B. Continuous Feedback & Formal Reviews

Implementation How-To:

1. **Regular 1:1 Meetings:** Mandatory bi-weekly 1:1s between managers/leads and direct reports for ongoing discussion of progress against goals, feedback (positive and constructive), challenges, and development needs. Document key discussion points.**
2. **Peer Feedback (Code Reviews):** Utilize GitLab MR comments as a primary mechanism for specific, timely technical feedback. Promote constructive review culture.**
3. **Sprint Retrospectives:** Provide a forum for team-level process feedback.**

4. **Formal Performance Reviews:** Conduct formal reviews (e.g., semi-annually) documenting performance against goals, utilizing KPI data, peer feedback (potentially via 360-degree feedback tools/process if implemented), and manager assessment. Link review outcomes to compensation/promotion cycles and update IDPs.

Responsibility: Managers/Team Leads, All Team Members (giving/receiving feedback).

C. Key Performance Indicators (KPIs) & Measurement Automation

Implementation How-To:

We will track a balanced set of KPIs across different dimensions, automating data collection wherever possible.

1. **KPI Selection & Definition:** Finalize the specific KPIs, targets, calculation methods, and data sources for each category below. Document definitions in Confluence.
2. **Automation Implementation:**
    - **Configure Jira/ADO dashboards and REST API queries (using scripts e.g., Python with `requests` and `jira` library) to extract Sprint/Flow metrics, Defect metrics.**
    - **Configure GitLab Insights or use GitLab APIs (via scripts) to extract DORA metrics, Pipeline metrics, Code Coverage, Security Scan results.**
    - **Configure CloudWatch Metrics/Alarms/Dashboards and potentially APM tools to track Operational/Reliability metrics and SLO adherence. Extract data via AWS SDK/CLI or CloudWatch API.**
    - **Develop scripts/processes to aggregate data from these sources into a central reporting dashboard (e.g., built in QuickSight, Grafana, or a custom web app) or directly into MBR/Status Report templates.**

3. **Reporting Cadence:** Integrate automated KPI reporting into Weekly Status Reports, Sprint Reviews, and Monthly Business Reviews as appropriate for the audience.

Responsibility: Project Managers, Tech Leads, DevOps/SRE Team (Automation), QA Lead, Security Lead.

KPI Categories & Examples (To be finalized with specific targets):

Category	KPI Example	Data Source (Potential)	Automation	Primary Audience
Velocity & Flow	Team Velocity (Avg Story Points/Sprint)	Jira/ADO	High (Jira Reports/API)	Team, Leads, PMs
	Sprint Commitment Reliability (%)	Jira/ADO	High (Jira Reports/API)	Team, Leads, PMs
	Cycle Time (Avg time from 'In Progress' to 'Done')	Jira/ADO	High (Jira Reports/API)	Team, Leads
DORA Metrics	Deployment Frequency	GitLab CI/CD Logs/API	High	Leads, PMs, SteeringCo
	Lead Time for Changes (Commit to Prod)	GitLab CI/CD Logs/API, Git History	High	Leads, PMs, SteeringCo
	Change Failure Rate (%)	GitLab CI/CD Deploy Logs, Incident Data (Jira/PagerDuty)	Medium (Requires correlation)	Leads, PMs, SRE, SteeringCo
	Mean Time To Restore (MTTR)	Incident Management System (PagerDuty/Jira)	Medium (Requires incident data)	SRE, Leads, PMs, SteeringCo
Quality	Unit Test Code Coverage (%)	GitLab CI (Coverage Reports)	High	Dev Team, Leads, QA
	CI Pipeline Success Rate (%)	GitLab CI/CD Logs/API	High	Dev Team, DevOps, Leads
	Defect Escape Rate (Prod Defects / Total Defects per Release)	Jira/ADO	Medium (Manual calculation	QA, Leads, PMs

Category	KPI Example	Data Source (Potential)	Automation	Primary Audience
			from Jira data)	
	Automated Test Pass Rate (%) (E2E, Integration)	GitLab CI/CD Test Reports	High	QA, Dev Team, Leads
Security	Open Critical/High Vulnerabilities (SAST/SCA/DAST/Container) Count & Age	GitLab Security Dashboard, Jira/ADO (Security Defects)	High	Security Team, Leads, PMs
	Time to Remediate Critical/High Vulnerabilities	Jira/ADO (Security Defects)	High	Security Team, Leads
Operations & Reliability	Service Availability / Uptime (%) vs SLA	CloudWatch Metrics/Alarms, APM Tool	High	SRE/Ops, Leads, PMs, SteeringCo
	SLO Adherence / Error Budget Burn Rate	CloudWatch Metrics/Alarms, APM Tool	High	SRE/Ops, Dev Team, Leads
	Incident Count by Severity & MTTR	PagerDuty/Opsgenie, Jira/ADO	Medium (Requires correlation)	SRE/Ops, Leads, PMs
Individual (Examples)	Code Review Feedback Quality/Timeliness	GitLab MRs (Manual Assessment)	Low	Manager/Lead
	Automated Test Contribution (Coverage / Tests Written)	Code Coverage Reports, Git History (Manual Assessment)	Medium	Manager/Lead
	On-time Task Completion (%)	Jira/ADO (Manual Assessment based on estimates/sprint commitments)	Low	Manager/Lead

VII. Ways of Working & Collaboration

Establishing clear norms and leveraging tools effectively is crucial for a large, distributed team.

Implementation How-To:

1. **Agile Ceremonies:** Mandatory participation as defined by role. Strict timeboxing enforced by Scrum Masters. Focus on defined purpose of each ceremony.
2. **Tool Usage Standards:**
    - **Jira/ADO:** Single source of truth for *work items* (Stories, Tasks, Bugs, Risks, CRs). Keep status updated daily. Log time accurately if required. Link related issues.
    - **Confluence/SharePoint:** Single source of truth for *documentation* (Plans, Designs, ADRs, Minutes, KBs, Runbooks, How-Tos). Keep pages organized and up-to-date. Use templates.
    - **GitLab:** Single source for code, MRs, CI/CD pipelines, registries. Use MR descriptions effectively. Respond to code review comments promptly.
    - **Slack/Teams:** For *transient*, quick communication. Use threads. Avoid making key decisions in chat; summarize and move to Jira/Confluence if needed. Respect focus time; use DND status appropriately. Adhere to channel purpose guidelines.
    - **Email:** For formal external communication or broad internal announcements (supplemented by Confluence).
3. **Meeting Etiquette (Mandatory for Video Calls):**
    - **Cameras ON by default to improve engagement.**

- **Mute when not speaking. Use clear audio (headset recommended).**
- **Be prepared and on time. Stick to the agenda.**
- **Facilitators ensure inclusive participation.**
- **Minimize distractions.**

4. **Asynchronous Communication Best Practices:** Write clearly and concisely. Provide sufficient context. State desired outcome or question clearly. Allow reasonable response time (consider time zones).**
5. **Cross-Team Collaboration:** Define clear points of contact between teams. Use shared Slack channels for cross-team topics. Document dependencies clearly in Jira/ADO. Proactively communicate potential impacts to other teams.**
6. **Definition of Ready (DoR) & Done (DoD):** Rigorous adherence to DoR before starting work and DoD before closing work ensures clarity and quality handoffs. DoD *must* include passing all required automated tests and code review approval.**

Responsibility: All Team Members, Scrum Masters (Facilitation), Team Leads (Reinforcement).

VIII. Knowledge Management & Transfer Plan

We will implement robust practices to capture, organize, share, and retain project knowledge, minimizing reliance on individuals and ensuring continuity.

Implementation How-To:

1. **Central Knowledge Hub (Confluence/SharePoint):** Maintain the well-structured project space as the single source of truth for all documentation (Requirements, Design, Architecture, ADRs, Test Plans, Runbooks, Meeting

Minutes, How-Tos, Onboarding Materials, Process Guides). Mandate documentation creation/updates as part of the Definition of Done for relevant tasks.

2. Code Documentation:**
    - **Comments:** Mandate clear comments for complex logic, algorithms, non-obvious code sections, and public APIs/functions. Keep comments concise and up-to-date.
    - **READMEs:** Require comprehensive README files at the root of each repository/microservice detailing: purpose, setup instructions, local testing commands, key architectural decisions, dependencies, deployment process, and points of contact. Use standardized template.
3. Architecture Decision Records (ADRs):** Document significant architectural decisions (e.g., choice of database, communication pattern, major library) using a simple ADR template (Context, Decision, Consequences). Store versioned ADRs in Git or Confluence.
4. **Knowledge Sharing Sessions:** Schedule regular (e.g., bi-weekly/monthly) internal sessions:
    - **Tech Talks / Brown Bags:** Team members present on new technologies, architectural patterns, complex features they built, or lessons learned.
    - **Cross-Team Demos:** Feature teams demo progress or specific technical implementations to other teams.
    - **Recorded Sessions:** Record key sessions and store them in Confluence for asynchronous viewing / onboarding.
5. Pair Programming:** Encourage pair programming, especially for complex tasks, onboarding new members, or cross-skilling between frontend/backend/ops.
6. Code Review as KT:** Leverage the mandatory code review process as a primary mechanism for knowledge sharing about code changes.

7. **Handover Process (Internal):** When tasks or component ownership transitions between team members, require a documented handover including a brief meeting and updates to relevant Confluence pages/READMEs.

Responsibility: All Team Members (Contributing), Tech Leads/Architects (Driving documentation standards/ADRs), Scrum Masters/PMs (Facilitating sharing sessions).

IX. Detailed Offboarding & Knowledge Retention Process

A formal offboarding process is mandatory to ensure knowledge continuity and security when a team member leaves the project.

Implementation How-To:

1. **Initiation:** Manager notifies PM, HR, Security, and relevant Leads upon confirmation of departure date.
2. **Knowledge Handover Plan:** Manager works with departing individual and relevant lead(s)/team members to create a specific handover plan covering:
    - **Current work in progress (status, next steps, handover target person).**
    - **Areas of unique knowledge or ownership.**
    - **Location of critical documentation.**
    - **Pending action items or decisions.**
3. **Documentation Completion:** Departing individual *must* ensure all owned documentation (Confluence pages, READMEs, code comments for WIP) is complete and up-to-date before last day. Manager/Lead verifies.

4. **Handover Sessions:** Schedule dedicated handover meetings between departing individual and receiving team member(s)/lead(s) as defined in the plan. Receiving member confirms understanding.

5. **Access Revocation Checklist:** Utilize a standardized checklist (managed potentially via Jira workflow or secure spreadsheet) tracked by Manager/HR/Security:
    - **Revoke access to GitLab repositories.**
    - **Revoke access to Jira/ADO project.**
    - **Revoke access to Confluence space.**
    - **Revoke access to Slack/Teams channels.**
    - **Revoke access to AWS GovCloud IAM user/roles.**
    - **Revoke VPN access.**
    - **Disable any service accounts tied to the individual.**
    - **Revoke access to any third-party tool licenses.**
    - **Confirm return of any physical assets (laptop, hardware tokens, ID badges).**

6. **Final Verification:** Manager confirms completion of knowledge handover and documentation updates. Security confirms completion of access revocation checklist. HR completes formal offboarding procedures.

Responsibility: Manager (Oversees process), Departing Individual (Execution), Receiving Team Member/Lead (Receives KT), Security Team (Access Revocation), HR (Formalities), PM (Awareness).

> ***Timely and complete access revocation upon departure is a critical security control. Maintain auditable records of offboarding checklist completion.***

X. Team Health, Culture & Well-being

A high-performing team requires a healthy, inclusive, and sustainable culture.

Implementation How-To:

1. **Psychological Safety:** Actively foster an environment where team members feel safe to speak up, ask questions, admit mistakes, and challenge ideas respectfully without fear of retribution. This is critical for effective retrospectives, risk identification, and innovation. Managers and Leads model this behavior.
2. **Blameless Culture:** Reinforce blameless post-mortems for incidents. Focus on systemic and process improvements, not individual fault.
3. **Recognition:** Implement mechanisms for both formal (e.g., performance reviews, project milestone celebrations) and informal (e.g., peer-to-peer shout-outs in Slack/stand-ups) recognition of contributions and effort.
4. **Workload Management:** Managers/Leads monitor team workload via sprint commitments, velocity trends, and regular 1:1s. Proactively address signs of burnout or unsustainable pressure. Encourage taking PTO.
5. **Diversity & Inclusion:** Promote respectful collaboration and actively work against bias. Ensure equitable opportunities for participation and growth.
6. **Team Building:** For distributed teams, intentionally schedule periodic virtual (or occasional in-person, budget permitting) team-building activities to foster camaraderie.

Responsibility: All Team Members, Managers/Leads (Setting the tone), Scrum Masters (Facilitating team dynamics), HR (Supporting policies).

XI. Conclusion: People as the Foundation for Success

This detailed Human Resources and Team Management Strategy provides the essential framework for building, managing, and sustaining the high-performance, globally distributed team required to deliver the NYCPS TMS project successfully. By implementing these prescriptive practices for structure, onboarding, skills development, performance management based on objective KPIs, collaborative ways of working, knowledge retention, and fostering a positive culture, we establish the human foundation necessary to meet the project's complex technical challenges, demanding timeline, and critical public sector mission. Consistent execution and continuous refinement of this strategy are paramount to maximizing our most valuable asset – our people – and ultimately delivering exceptional value to NYCPS.