Product Requirements Document: MaverickAI – Multi-Agent Platform for Enterprise Fresher Onboarding and Training Management

1. Executive Summary

1.1 Project Abstract

The enterprise landscape for fresher onboarding and training is currently undergoing a systemic shift, moving away from static, human-intensive management toward dynamic, AI-driven orchestration. MaverickAI represents a pioneering intervention in this domain, designed as a centralized web-based onboarding and training management system supported by a coordinated open-source multi-agent framework. The platform's core mission is to decompose the complex, multifaceted lifecycle of fresher training—encompassing Foundation and Role-Specific phases—into specialized, autonomous agents capable of executing tasks with speed, precision, and personalization that far exceeds manual capabilities.

By leveraging an event-driven architecture orchestrated by n8n, MaverickAI addresses the critical operational inefficiencies plaguing modern enterprises: fragmented communication channels, manual assessment bottlenecks, data silos, and a lack of real-time visibility into workforce readiness. The solution replaces the traditional "pipeline" model of Learning Management Systems (LMS) with a "ecosystem" model, where agents for Onboarding, Assessment, Profiling, Analytics, and Reporting collaborate continuously to optimize the trainee experience and provide actionable intelligence to leadership. This document outlines the comprehensive functional, technical, and strategic requirements necessary to build, deploy, and scale MaverickAI, ensuring it meets the rigorous demands of enterprise security, scalability, and user experience.

1.2 Strategic Alignment and Business Value

The deployment of MaverickAI aligns with the broader strategic imperative of the "Autonomous Enterprise," where intelligent systems do not merely assist humans but actively manage complex workflows. The shift from single-model AI to multi-agent systems (MAS) allows for the specialization of tasks, ensuring that a "Profile Agent" can focus entirely on data integrity while an "Assessment Agent" focuses on pedagogical

feedback, mimicking a cross-functional human team. This specialization drives measurable organizational impact:

- **Operational Efficiency:** Automation of routine administrative tasks (scheduling, grading, reporting) reduces the manual burden on Training Managers by an estimated 40-60%, allowing them to focus on high-value mentorship.
- **Data-Driven Decision Making:** Real-time consolidation of assessment data enables "Course Correction" at the individual and cohort level, reducing the "Time-to-Productivity" for new hires.
- **Scalability:** The decoupled nature of the agentic architecture allows the system to scale horizontally, supporting cohorts ranging from 10 to 10,000 freshers without a linear increase in infrastructure costs.

---

2. Strategic Context and Market Analysis

2.1 The Evolution of Enterprise Onboarding

Historically, enterprise onboarding has been a linear, static process defined by checklists and manual supervision. The "Foundation Training" phase—covering functional, behavioral, and technical skills—often suffers from a "one-size-fits-all" approach that fails to account for individual learning speeds or prior knowledge. As organizations scale, the administrative overhead of managing these programs grows exponentially. Traditional LMS platforms act as passive repositories of content, requiring human administrators to push assignments, pull reports, and manually intervene when a fresher falls behind.

The emergence of Agentic AI changes this dynamic fundamental. Unlike passive software, AI agents possess the capability to perceive, reason, and act. In the context of MaverickAI, this means the system does not just *record* a failing grade; it *understands* the failure, *adjusts* the schedule, and *notifies* the relevant stakeholders immediately. This transition from "Passive Management" to "Active Orchestration" is the core differentiator of the MaverickAI platform.

2.2 The Case for Multi-Agent Systems (MAS)

Single-agent systems, often manifesting as simple chatbots, lack the complexity to handle end-to-end enterprise workflows. They are typically generalists, prone to context

loss and "hallucinations" when overloaded with diverse tasks. In contrast, a Multi-Agent System (MAS) architecture, as proposed for MaverickAI, employs a "Society of Agents" approach.

Feature	Single-Agent System	MaverickAI Multi-Agent System
Architecture	Monolithic LLM handling all tasks	Specialized Agents (Onboarding, Assessment, etc.)
Scalability	Limited by context window and token limits	Horizontal scaling; agents operate in parallel
Reliability	Single point of failure; prone to drift	Redundant pathways; cross-agent verification
Task Complexity	Handles simple, linear queries	Manages complex, multi-step workflows with handoffs
Data Handling	Centralized, often commingled context	Decoupled; agents access only relevant data schemas

The decision to utilize frameworks like **LangGraph, CrewAI, or AutoGen** is driven by the need for robust orchestration. These frameworks allow for the definition of strict "Graph" flows where agents must complete specific sub-tasks (e.g., "Grade Code") before passing state to the next agent (e.g., "Update Profile"), ensuring deterministic outcomes even within a probabilistic AI system.

2.3 The Role of Workflow Automation (n8n)

While agents provide the "Intelligence," **n8n** provides the "Nervous System." The choice of n8n for workflow orchestration is strategic. It allows for a low-code/no-code interface that bridges the gap between technical complexity and administrative usability. n8n facilitates:

- **Event-Driven Triggers:** Workflows are not just scheduled; they are triggered by events (e.g., a file upload, a webhook from an LMS, a specific date).

- **System Integration:** n8n's extensive library of nodes allows MaverickAI to connect seamlessly with existing HRIS (BambooHR, Workday), Communication Tools (Slack, Teams), and Databases (PostgreSQL, MongoDB) without writing custom API wrappers for every connection.
- **Human-in-the-Loop:** n8n workflows can be designed to pause for human approval (e.g., before sending a failing report to a manager), ensuring that AI autonomy remains supervised.

---

3. Problem Statement and Opportunity Definition

3.1 Current Operational Pain Points

The "Problem Statement" defined in the source material highlights several critical inefficiencies in the current state of fresher training. These can be categorized into three primary domains: **Execution**, **Data**, and **Visibility**.

3.1.1 Execution Friction

- **Inconsistent Communication:** Freshers receive instructions via email, chat, and verbal meetings, leading to confusion regarding deadlines and expectations. There is no "Single Source of Truth" for the daily schedule.
- **Manual Grading Constraints:** While objective quizzes are automated, high-value assessments like coding challenges and project submissions often sit in a queue waiting for manual review by senior developers or trainers. This delay prevents the "tight feedback loop" required for effective learning.
- **Scheduling rigidity:** Creating bespoke schedules for hundreds of freshers is impossible manually. As a result, all freshers follow the same track, causing boredom for advanced learners and anxiety for those who need remedial help.

3.1.2 Data Fragmentation

- **Siloed Systems:** Assessment results are scattered across disparate platforms—HackerRank for code, Moodle for quizzes, Excel for attendance. Consolidating this data to form a holistic view of a "Fresher Profile" is a manual, error-prone weekly task.

- **Stale Profiles:** Because data consolidation is manual, the "Fresher Profile" is rarely up-to-date. A manager looking at a profile on Wednesday might be seeing data from the previous Friday, rendering proactive intervention impossible.

3.1.3 Visibility Gaps

- **Lagging Indicators:** Managers typically rely on "Lagging Indicators" (e.g., failed final exam) rather than "Leading Indicators" (e.g., declining engagement in daily quizzes). This results in reactive "damage control" rather than proactive coaching.
- **Reporting Burden:** Generating the weekly status report for leadership is a "Friday Afternoon Nightmare," involving the manual stitching of CSVs and charts. This reduces the time managers can spend on actual strategy or mentorship.

3.2 The MaverickAI Opportunity

MaverickAI proposes to solve these problems by creating a **"Continuously Operating Training Ecosystem"**.

- **From Manual to Autonomous:** The **Onboarding Agent** automates the scheduling logistics, ensuring every fresher knows exactly what to do every day.
- **From Delayed to Instant:** The **Assessment Agent** provides immediate, qualitative feedback on subjective work, enabling instant learning.
- **From Siloed to Unified:** The **Profile Agent** acts as a real-time aggregator, ensuring the fresher profile is always a "Live" representation of skills.
- **From Reactive to Predictive:** The **Analytics Agent** identifies skill gaps and performance trends *before* they become failures.

---

4. User Personas and Requirements Analysis

To ensure the platform meets the diverse needs of its user base, we have defined three primary personas. These personas drive the user stories and functional requirements.

4.1 Persona A: "Riya" – The Fresher (Maverick)

Role: Graduate Trainee, Software Engineering Track. **Psychographics:** Eager to learn, anxious about performance, tech-savvy, appreciates transparency. **Primary Goal:** To successfully complete the Foundation and Role-Specific training and get deployed to a project. **Pain Points:** Unclear daily objectives, delayed feedback on code, fear of "silent failure" (not knowing she is falling behind until it's too late).

User Story ID	Requirement Description	Success Metric
US-M-01	As Riya, I want a personalized, dynamic daily schedule so that I know exactly what modules, quizzes, and projects are due today.	Dashboard loads schedule < 2s; Schedule updates daily.
US-M-02	As Riya, I want instant, detailed feedback on my coding challenges so that I can understand my mistakes immediately.	Feedback received < 1 min after submission; Includes logic & syntax analysis.
US-M-03	As Riya, I want to view my real-time progress and skill proficiency graph so that I know where I stand compared to the cohort.	Skill Graph reflects latest assessment score immediately.
US-M-04	As Riya, I want to ask the system for help on a specific topic without waiting for a human mentor.	Chatbot interface available 24/7; Responses context-aware to current module.

4.2 Persona B: "David" – The Training Manager

Role: L&D Lead, Enterprise Technology Division. **Psychographics:** Overwhelmed by administrative tasks, data-driven, protective of his cohort's reputation. **Primary Goal:** To ensure high pass rates, minimize washout, and report success to leadership with minimal effort. **Pain Points:** Spending 40% of time on spreadsheets, lack of visibility into "at-risk" students, difficulty in standardizing grading across different human mentors.

User Story ID	Requirement Description	Success Metric
US-TM-01	As David, I want an automated "At-Risk" alert system so that I can intervene with struggling freshers before they fail.	Alerts generated when performance dips below defined threshold.
US-TM-02	As David, I want to generate a comprehensive, formatted PDF report for the leadership team with a single click.	Report generation < 30s; Contains all key KPIs and charts.
US-TM-03	As David, I want to see a cohort-level "Skill Heatmap" to identify if a specific topic (e.g., Recursion) is causing widespread difficulty.	Heatmap visualization available on Analytics Dashboard.
US-TM-04	As David, I want the system to automatically assign remedial tasks to students who fail a specific module.	Remedial tasks appear on student schedule automatically.

4.3 Persona C: "Sarah" – The System Administrator

Role: DevOps / IT Infrastructure Lead. **Psychographics:** Security-conscious, focused on reliability and uptime, wary of "Shadow AI." **Primary Goal:** To ensure the platform is secure, compliant (SOC2/GDPR), and scalable. **Pain Points:** Unclear AI data usage policies, managing API costs, debugging complex agent interactions.

User Story ID	Requirement Description	Success Metric
US-SA-01	As Sarah, I want granular logs of all agent activities and decision pathways so that I can audit system behavior.	Comprehensive Logging (MongoDB); Traceability of AI reasoning.

US-SA-02	As Sarah, I want to enforce strict Role-Based Access Control (RBAC) so that freshers cannot access each other's data.	Penetration testing confirms data isolation.
US-SA-03	As Sarah, I want the system to handle PII securely, ensuring compliance with data privacy regulations.	Encryption at rest/transit; PII masking before AI processing.

---

5. Functional Specifications: The Multi-Agent Ecosystem

The core of MaverickAI is its five specialized agents. This section defines the functional requirements for each, outlining their specific inputs, processing logic, and outputs.

5.1 Onboarding Agent

Role: The Architect. Responsible for the structural planning of the trainee's journey.

- **Functionality:**
  - **Profile Ingestion:** Reads the fresher's profile data (education, background, role) from the HRIS via n8n integration.
  - **Schedule Generation:** Generates a structured "Learning Plan" mapping the master curriculum to the calendar days, accounting for holidays and weekends.
  - **Dynamic Adaptation:** Listens for "Module Failure" events from the Assessment Agent. If a student fails "Java Basics," the Onboarding Agent dynamically inserts a "Remedial Java" day and shifts the subsequent schedule.
- **Key Inputs:** User Profile, Master Curriculum, Academic Calendar, Assessment Results.
- **Key Outputs:** Personalized JSON Schedule Object, Calendar Events (Google/Outlook), Notification Triggers.
- **AI Logic:** Uses constraint satisfaction algorithms (via LLM reasoning) to balance workload and ensure prerequisites are met before advanced topics are scheduled.

5.2 Assessment Agent

Role: The Evaluator. Responsible for providing objective and subjective feedback.

- **Functionality:**
    - **Code Evaluation:** Receives code submissions. Executes them in a sandboxed environment (Docker) to check for correctness (Unit Tests) and performance (Time Complexity).
    - **Semantic Analysis:** Uses a Code-Specific LLM (e.g., specialized GPT-4 or Claude model) to analyze code style, readability, and adherence to best practices (e.g., DRY principles). Provides "Human-like" qualitative feedback (e.g., "Your logic is correct, but using a global variable here is unsafe").
    - **Quiz Grading:** Auto-grades multiple-choice and short-answer questions, providing immediate score calculation.
- **Key Inputs:** Submission Data (Code/Text), Assignment Rubric, Unit Test Suite.
- **Key Outputs:** Assessment Object (Score, Pass/Fail Status, Qualitative Feedback Text), Skill Tag Updates.
- **Quality Assurance:** Implements a "Regression Eval" suite to ensure grading consistency. If the LLM grade deviates significantly from the Unit Test result (e.g., LLM gives 90% but tests fail), the system flags it for human review.

5.3 Profile Agent

Role: The Librarian. Responsible for the integrity and currency of the Fresher Profile.

- **Functionality:**
    - **Data Aggregation:** Continuously listens for events from the Assessment Agent. When a score is generated, the Profile Agent updates the specific skill record in the fresher's profile.
    - **Certification Management:** Verifies and logs external or internal certifications.
    - **Skill Graphing:** Maintains a dynamic "Skill Graph" for each user. It translates discrete assessment scores into proficiency levels (e.g., "Score > 90 in 3 Arrays assessments" = "Intermediate Proficiency in Arrays").
- **Key Inputs:** Assessment Results, Certification Artifacts, System Activity Logs.
- **Key Outputs:** Unified User Profile Object (JSON), Skill Radar Chart Data.

5.4 Analytics Agent

Role: The Strategist. Responsible for high-level pattern recognition and prediction.

- **Functionality:**

    - **Cohort Analysis:** Aggregates data across all active profiles to calculate cohort averages, standard deviations, and distribution curves.
    - **Gap Identification:** Identifies systemic issues. (e.g., "40% of the cohort failed 'Microservices Architecture' – potential issue with training content").
    - **Predictive Risk Modeling:** Analyzes behavioral patterns (login frequency, submission latency) to predict "Dropout Risk" or "Performance Dip." Triggers alerts to the Training Manager.
- **Key Inputs:** Aggregated Profile Data, Cohort Metadata, Historical Performance Baselines.
- **Key Outputs:** Cohort Health Report, Risk Alerts (High/Medium/Low), Content Improvement Recommendations.

5.5 Reporting Agent

Role: The Communicator. Responsible for translating data into stakeholder-ready documents.

- **Functionality:**
    - **Automated Report Generation:** Generates PDF status reports on a schedule (e.g., Weekly Business Review) or on-demand.
    - **Customization:** Allows managers to select specific modules or metrics to include (e.g., "Show me only the 'At-Risk' list").
    - **Distribution:** Utilizing n8n's email nodes, it distributes reports to a pre-defined distribution list (Training Managers, HR Heads).
- **Key Inputs:** Analytics Report Object, User Profile Summaries, Template Config.
- **Key Outputs:** Formatted PDF Document, HTML Email Summary.

---

6. Technical Architecture and Data Strategy

6.1 Architecture Overview

MaverickAI employs a microservices-based, event-driven architecture. The system is containerized using Docker to ensure consistency across development, testing, and production environments.

6.1.1 The Orchestration Layer: n8n

At the heart of the system lies **n8n**, a workflow automation tool that manages the "nervous system" of the platform.

- **Role:** n8n acts as the central bus for all events. It handles webhooks from the frontend, triggers agent workflows, and manages the flow of data between the database and the AI models.
- **Why n8n?** It provides a visual, low-code environment for defining complex agent interactions (e.g., "If Grade < 50, Trigger Remedial Workflow") without hard-coding business logic, allowing L&D teams to modify workflows easily.

6.1.2 The Agent Framework

- **Framework Selection:** The platform supports **LangGraph**, **CrewAI**, or **AutoGen**. For the initial implementation, **LangGraph** is recommended due to its graph-based state management, which is ideal for the deterministic workflows required in training (e.g., Assessment -> Profile Update -> Notification).
- **Inter-Agent Communication:** Agents communicate via a standardized JSON schema over the internal message bus (n8n or direct API calls).

6.1.3 The Frontend Layer

- **Technology: React** or **Next.js** framework.
- **Styling: Tailwind CSS** for rapid, responsive UI development.
- **Visualization: Recharts** or **Chart.js** for rendering complex data (Skill Radars, Heatmaps).

6.1.4 The Backend & API Layer

- **Technology: FastAPI** (Python) or **Node.js**. Given the heavy reliance on AI libraries (LangChain/LangGraph), a Python-based FastAPI backend is the preferred choice for the Agent Services, while Node.js is suitable for the high-concurrency API Gateway.
- **Protocol:** REST APIs for frontend-backend communication; Webhooks for n8n integration.

6.2 Data Strategy and Schema

The platform utilizes a polyglot persistence strategy to handle different data types efficiently.

6.2.1 Data Stores

1. **Relational Database (PostgreSQL):** Used for structured, transactional data that requires ACID compliance.
    - *Users Table:* Stores Fresher/Manager credentials, roles, and static profile data.
    - *Schedules Table:* Stores the mapping of users to curricula.
    - *Certifications Table:* Stores verified achievements.
2. **Document Store (MongoDB):** Used for unstructured or semi-structured data.
    - *Assessment_Logs:* Stores the raw code submissions, full text of AI feedback, and conversation history.
    - *Agent_Memory:* Stores the context/state of long-running agent tasks.

6.2.2 Data Flow Example: Assessment Submission

1. **User Action:** Fresher submits code via Frontend.
2. **API Gateway:** Receives payload, validates auth token.
3. **n8n Webhook:** Triggered by API.
4. **n8n Workflow:**
    - Saves raw submission to **MongoDB** (`Assessment_Logs`).
    - Calls **Assessment Agent** (FastAPI Service).
    - **Assessment Agent** executes code, calls LLM, generates JSON result.
    - n8n receives JSON result.
    - Updates **PostgreSQL** (`Scores` table).
    - Triggers **Profile Agent** to recalculate skill average.
    - Pushes notification to Frontend via WebSocket.

---

7. UI/UX and Dashboard Specifications

The user interface is the primary touchpoint for engagement. It must be intuitive, responsive, and data-rich.

7.1 Fresher Dashboard

Objective: Provide clarity and motivation.

- **"Today's Mission" Panel:** Prominent display of the day's tasks (Videos, Quizzes, Labs) generated by the Onboarding Agent.
- **Skill Radar Chart:** A visualization of the student's proficiency across 5-6 core dimensions (e.g., Coding, Debugging, Theory, Soft Skills).
- **Feedback Inbox:** A unified view of all qualitative feedback received from the Assessment Agent, allowing the student to review past advice.
- **Leaderboard (Gamification):** Anonymized ranking based on "XP" (Experience Points) earned from completing tasks, designed to drive engagement.

7.2 Training Manager Dashboard

Objective: Provide observability and control.

- **Cohort Health Monitor:** A "Traffic Light" system (Red/Amber/Green) showing the overall status of the cohort based on completion rates and average scores.
- **Risk Radar:** A list of "At-Risk" students flagged by the Analytics Agent, with specific reasons (e.g., "Low Engagement," "Failed 2 consecutive assessments").
- **Drill-Down View:** Ability to click on any student to view their "Fresher Profile," seeing the exact same data the Profile Agent maintains.
- **Report Generator:** A control panel to configure and trigger the Reporting Agent.

---

8. Workflow Orchestration Logic (n8n)

The n8n orchestration layer is the operational backbone of MaverickAI. This section details specific workflows that need to be implemented.

8.1 Workflow: "Daily Learning Schedule Activation"

- **Trigger:** Schedule Trigger (Every day at 06:00 AM Local Time).
- **Node 1 (Database):** Fetch all active users with `status = 'Training'`.
- **Node 2 (Split in Batches):** Process users in batches of 50 to avoid rate limits.
- **Node 3 (Onboarding Agent):** For each user, check the master curriculum and current progress. Generate the `daily_task_list`.
- **Node 4 (Database):** Write the `daily_task_list` to the `Daily_Schedules` table.

- **Node 5 (Notification):** Send a "Your Day is Ready" push notification/email to the user.

8.2 Workflow: "Assessment Evaluation Pipeline"

- **Trigger:** Webhook (Event: `submission_received`).
- **Node 1 (Assessment Agent):** Send submission data to the AI service.
- **Node 2 (Switch):** Check `status` of evaluation.
    - *If Success:* Proceed to Node 3.
    - *If Error (e.g., Sandbox timeout):* Trigger "Error Handling" workflow (Alert Admin).
- **Node 3 (Profile Agent):** Update user's skill profile with new score.
- **Node 4 (Analytics Agent):** Check if this score puts the user in "At-Risk" category.
    - *If Yes:* Send Alert to Manager Dashboard.
- **Node 5 (Frontend):** Send "Grading Complete" signal to user UI.

8.3 Workflow: "Weekly Status Report"

- **Trigger:** Schedule Trigger (Every Friday at 16:00 PM).
- **Node 1 (Analytics Agent):** Generate `Weekly_Cohort_Summary` (Pass rates, attendance, top performers).
- **Node 2 (Reporting Agent):** Convert Summary JSON into PDF using a pre-defined HTML template.
- **Node 3 (Email):** Send PDF to the "Stakeholders" distribution list via SMTP.

---

9. Key Performance Indicators (KPIs) & Success Metrics

To validate the efficacy of the MaverickAI platform, the following KPIs will be tracked. These align with industry standards for onboarding and training effectiveness.

KPI Category	Metric Name	Definition	Target	Data Source
Efficiency	**Time-to-Productivity**	Days from Day 1 to passing the final "Role-Ready" Capstone.	< 45 Days	Profile Agent

Engagement	**Daily Active Usage (DAU)**	% of active freshers who log in and interact with the platform daily.	> 95%	System Logs
Quality	**First-Pass Yield**	% of freshers passing assessments on the first attempt (indicates effective content).	> 70%	Assessment Agent
Retention	**Training Retention Rate**	% of freshers who successfully graduate the Foundation phase.	> 90%	HRIS / Profile
Impact	**Skill Gap Closure Speed**	Average time (days) taken to resolve a flagged skill gap.	< 5 Days	Analytics Agent
Satisfaction	**NPS (Net Promoter Score)**	Monthly survey score from freshers regarding the training experience.	> 50	Survey Tool

---

10. Non-Functional Requirements (NFRs)

10.1 Security and Compliance

- **Data Residency:** All data persistence (PostgreSQL/MongoDB) must reside within the enterprise's Virtual Private Cloud (VPC) to meet data sovereignty laws.
- **Encryption:** Data must be encrypted at rest (AES-256) and in transit (TLS 1.3).
- **SOC2 Compliance:** The system must maintain a comprehensive audit trail of all actions. This is particularly critical for the Assessment Agent—every grade assigned by AI must be traceable to the specific input and model version used.
- **GDPR/Privacy:** PII must be segregated. The "Profile Agent" is the only entity with access to raw identity data. Analytics must be performed on anonymized datasets.

10.2 Scalability and Reliability

- **Horizontal Scaling:** The microservices architecture allows individual agents to scale. If the "Assessment Agent" experiences a high load during a hackathon,

Kubernetes can spin up additional pods for that specific service without scaling the entire platform.
- **Rate Limiting:** API Gateway must implement rate limiting to prevent abuse and manage LLM cost/token quotas.
- **Uptime:** Target availability of 99.9% during business hours.

10.3 Performance

- **Dashboard Latency:** Critical dashboards (Fresher Home, Manager Overview) must load in < 2 seconds.
- **Assessment Speed:** Automated code grading should complete in < 60 seconds to provide the "instant feedback" value proposition.

---

11. Implementation Roadmap

11.1 Phase 1: MVP & Proof of Concept (Weeks 1-8)

- **Objective:** Validate the Multi-Agent architecture and n8n orchestration.
- **Deliverables:**
    - Synthetic Dataset creation (Mock Freshers, Mock Submissions).
    - Basic n8n workflow: Ingest Profile -> Generate Schedule -> Assess Submission.
    - Simple React UI for Fresher view.
    - Deployment via Docker Compose on a local server.
- **Success Criteria:** End-to-end flow works for 1 synthetic user without manual intervention.

11.2 Phase 2: Pilot Deployment (Weeks 9-16)

- **Objective:** Live test with a small cohort (e.g., 20 freshers).
- **Deliverables:**
    - Integration with Live HRIS (Read-Only).
    - Full Dashboard functionality (Manager & Fresher).
    - Implementation of "Human-in-the-Loop" for Assessment Agent (Manager approves grades).

    - Security Hardening (RBAC, Encryption).
- **Success Criteria:** System handles 20 users; Feedback received is accurate >90% of the time.

11.3 Phase 3: Enterprise Rollout (Weeks 17+)

- **Objective:** Scale to full department.
- **Deliverables:**
    - Advanced Analytics (Predictive Modeling).
    - Full Gamification (Leaderboards).
    - Multi-tenant support (Different tracks for QA, Dev, DevOps).
    - Removal of "Human-in-the-Loop" for low-risk assessments.

---

12. Risk Management

Risk	Impact	Likelihood	Mitigation Strategy
LLM Hallucination (Grading)	High	Medium	Implement "Regression Evals". Use deterministic unit tests as the primary grading mechanism (70% weight) and LLM only for style/feedback (30% weight).
Workflow Fragility	Medium	High	Use n8n's error handling nodes to retry failed API calls. Implement "Dead Letter Queues" for failed events so no data is lost.
User Trust	High	Medium	Transparency. Clearly label AI-generated feedback. Provide a "Dispute Grade" button that escalates to a human manager.

Data Privacy	Critical	Low	strict PII masking before sending data to any external model. Use Azure OpenAI / Bedrock private instances where data is not used for model training.

---

13. Conclusion

MaverickAI is not merely a software upgrade; it is an operational transformation. By embedding intelligence into the very fabric of the onboarding process, the enterprise can move from a model of *monitoring* freshers to *empowering* them. The proposed Multi-Agent System, orchestrated by n8n, provides the necessary flexibility, scalability, and intelligence to handle the complex, dynamic nature of human learning. This PRD serves as the blueprint for building that future—a future where every "Maverick" receives the personalized guidance they need to succeed, and every manager has the insights they need to lead.

1. UI/UX: Figma Wireframe Documentation

This section outlines the structure and key components for the primary screens required in the Figma prototype. The design philosophy focuses on "Clarity for Freshers" and "Observability for Managers."

1.1 Global Design System

- **Typography:** Inter or Roboto (Clean, readable sans-serif).
- **Color Palette:**
    - *Primary:* Deep Indigo (Professionalism/Trust).
    - *Secondary:* Electric Teal (Action items/Gamification).
    - *Status Colors:* Emerald (Pass/Safe), Amber (At-Risk/Warning), Rose (Fail/Critical).
- **Layout:** Sidebar navigation with a responsive grid content area.

1.2 Screen: Fresher Dashboard ("The Cockpit")

Goal: Answer "What do I need to do today?" immediately.

- **Frame 1: The Daily Mission Control (Top Center)**
    - *Component:* Horizontal card list.
    - *Elements:* "Today's Schedule" header, Dynamic cards for specific modules (e.g., "Java Streams - 09:00 AM"), Status indicators (Pending/In-Progress/Done).

    - *Interaction:* Clicking a card expands details (Zoom link, content URL).
- **Frame 2: Skill Radar Chart (Top Right)**
    - *Component:* Hexagonal Radar Chart (Recharts visualization).
    - *Elements:* Axes for Coding, Debugging, Communication, Architecture, Theory.
    - *Data Binding:* Real-time scores fetched from the **Profile Agent**.
- **Frame 3: Recent Feedback Stream (Bottom Left)**
    - *Component:* Scrollable list view.
    - *Elements:* "Latest Assessment" header, AI-generated summary text (e.g., "Great logic on the recursion task, but watch your time complexity").
- **Frame 4: Leaderboard Widget (Bottom Right)**
    - *Component:* Compact list with gamification elements.
    - *Elements:* "Cohort Rank," "XP Earned," "Next Badge" progress bar.

1.3 Screen: Training Manager Dashboard ("The Tower")

Goal: Provide high-level visibility and "management by exception."

- **Frame 1: Cohort Health Monitor (Top Band)**
    - *Component:* Aggregate KPI cards.
    - *Elements:* "Cohort Average Score," "Attendance Rate," "At-Risk Count" (Red Badge).
- **Frame 2: The Risk Radar (Main Left)**
    - *Component:* Data Grid / Table.
    - *Elements:* List of students flagged by the **Analytics Agent**. Columns: Name, Risk Level (High/Med), Reason (e.g., "Failed 3 consecutive quizzes"), Recommended Action (e.g., "Assign Mentor").
- **Frame 3: Skill Gap Heatmap (Main Right)**
    - *Component:* 2D Heatmap Grid.
    - *Elements:* X-axis (Topics), Y-axis (Student Clusters). Darker red cells indicate topics where the majority of the cohort is struggling (e.g., "Dynamic Programming").

---

2. Implementation Documentation (Priority-Based)

This roadmap prioritizes the "Minimum Viable Product" (MVP) to validate the core multi-agent loop before scaling to advanced analytics.

Phase 1: The Core Loop (Priority P0 - Weeks 1-4)

Objective: Enable a fresher to log in, view a schedule, submit code, and get a grade.

1. **Infrastructure Setup:**

    - Deploy **n8n** on a cloud instance (AWS EC2 / DigitalOcean) using Docker Compose.
    - Set up **PostgreSQL** for user/schedule data and **MongoDB** for unstructured agent logs.
2. **Onboarding Agent Implementation:**
    - *Workflow:* Connect n8n to Google Calendar API.
    - *Logic:* Create a workflow that reads a JSON curriculum file and generates calendar invites for a specific user email.
3. **Assessment Agent (Basic):**
    - *Workflow:* Create an API endpoint (FastAPI) that accepts code text.
    - *Logic:* Execute code against 3 predefined unit tests. Return Pass/Fail boolean.

Phase 2: The Feedback Loop (Priority P1 - Weeks 5-8)

Objective: Replace binary Pass/Fail with qualitative AI feedback.

1. **Assessment Agent (Advanced):**
    - *Integration:* Connect the Assessment Service to an LLM (e.g., GPT-4o or Claude 3.5 Sonnet) via **LangGraph**.
    - *Logic:* Prompt engineering to analyze code style and efficiency, not just correctness. Output JSON with "Score" and "Critique."
2. **Profile Agent:**
    - *Workflow:* Create an n8n listener for "Assessment Complete" events.
    - *Logic:* Update the User's "Skill Record" in PostgreSQL when a new score arrives.

Phase 3: Visibility & Reporting (Priority P2 - Weeks 9-12)

Objective: Give managers the tools to see what is happening.

1. **Dashboard Frontend:**
    - Build the React/Next.js frontend. Connect **Recharts** to the PostgreSQL reporting views.
2. **Analytics Agent:**
    - *Workflow:* Scheduled n8n workflow (Nightly).
    - *Logic:* Aggregate daily scores, calculate standard deviation, and flag users > 1 SD below the mean.
3. **Reporting Agent:**
    - *Workflow:* Weekly trigger. Generate HTML summary of the Analytics data, convert to PDF, and email to Manager list.

---

3. Tech Stack Documentation

This technical specification ensures scalability, ease of automation, and modularity.

3.1 Orchestration & Logic Layer

- **Workflow Engine: n8n** (Self-Hosted).
    - *Why:* Visual management of complex agent handoffs, native webhooks, and easy integration with external APIs (Slack, Email, Calendar) without writing boilerplate code.
- **Agent Framework: LangGraph** (Python).
    - *Why:* Unlike autonomous agent swarms (which can be unpredictable), LangGraph allows us to define *cyclic* and *stateful* graphs, ensuring a deterministic flow (e.g., Code -> Test -> Review -> Grade) critical for enterprise training.

3.2 Application Layer

- **Frontend: Next.js** (React Framework).
    - *Why:* Server-side rendering for fast dashboard loads; extensive library support for data visualization (Recharts).
- **Backend API: FastAPI** (Python).
    - *Why:* High performance; native support for async operations (essential for handling long-running AI inference tasks); auto-generation of Swagger documentation.

3.3 Data Layer

- **Structured Data: PostgreSQL**.
    - *Usage:* User auth, strict curriculum schemas, grade book, attendance records.
- **Unstructured Data: MongoDB**.
    - *Usage:* Storing raw code submissions, full text of AI feedback logs, and flexible agent "memory" states.
- **Vector Database (Optional Phase 3): Pinecone** or **pgvector**.
    - *Usage:* RAG (Retrieval-Augmented Generation) for the Onboarding chatbot to answer questions like "Where do I find the documentation for module X?"

3.4 DevOps & Infrastructure

- **Containerization: Docker** & **Docker Compose**.
    - *Usage:* Service isolation (Frontend container, Backend container, n8n container, DB containers).
- **Authentication: OAuth 2.0** / **Auth0**.
    - *Usage:* Secure, standard login for freshers and managers.
- **Monitoring: Prometheus** + **Grafana**.
    - *Usage:* Monitoring agent latency, token usage costs, and system uptime.