MAVERICKAI

Multi-Agent Platform for Enterprise Training Management

System Architecture, Workflows & Implementation Guide

Version 1.0
Date: February 11, 2026

Table of Contents

---

1. System Architecture Overview

MaverickAI implements a **microservices-based, event-driven architecture** orchestrated through n8n workflow automation. The platform leverages a multi-agent system (MAS) approach where specialized AI agents collaborate to manage the complete fresher training lifecycle.

1.1 Architectural Principles

- **Microservices Architecture**: Each agent operates as an independent, containerized service with well-defined APIs and responsibilities, enabling independent scaling, deployment, and maintenance
- **Event-Driven Communication**: Asynchronous message passing through n8n ensures loose coupling and fault tolerance across services
- **Polyglot Persistence**: PostgreSQL for structured transactional data, MongoDB for unstructured agent logs and context, ensuring optimal data storage per use case

- **Containerization**: Docker containers ensure consistency across development, staging, and production environments
- **Stateless Agent Design**: All agent state is persisted to databases, enabling horizontal scaling and recovery from failures

1.2 High-Level Architecture Layers

Layer	Components	Responsibilities
Presentation	Next.js Frontend	User interfaces, real-time dashboards, WebSocket connections
API	Express Gateway + FastAPI	Authentication, routing, rate limiting, request validation
Orchestration	n8n Workflow Engine	Event routing, workflow automation, agent coordination
Agent Services	5 Specialized Agents (LangGraph)	Onboarding, Assessment, Profile, Analytics, Reporting
Data	PostgreSQL + MongoDB	Structured transactional data + unstructured logs/context
External	HRIS, LMS, Calendar, Email	Third-party integrations via n8n connectors

Component Flow:
Frontend → API Gateway → n8n Orchestrator → Agent Services → Data Layer → External Integrations

---

2. Core System Components

2.1 Orchestration Layer: n8n Workflow Engine

Role: Central nervous system that coordinates all agent interactions, manages event routing, and orchestrates complex multi-step workflows

Key Capabilities:

- Visual workflow designer enabling non-technical L&D teams to modify business logic
- Event-driven triggers (webhooks, schedules, database changes)
- Native integrations with 400+ services (HRIS, LMS, communication tools)
- Error handling and retry mechanisms with dead letter queues
- Human-in-the-loop approval gates for critical decisions
- Comprehensive logging and audit trails for compliance

Technical Specifications:

Specification	Details
Deployment	Docker container (self-hosted)
Persistence	PostgreSQL (workflow definitions, execution history)
Scalability	Horizontal scaling via queue workers
Monitoring	Prometheus metrics, custom webhooks
Security	OAuth2, API keys, encrypted credentials vault

2.2 Multi-Agent Framework: LangGraph

Framework Selection Rationale: LangGraph chosen for its graph-based state management, enabling deterministic workflows essential for training operations

LangGraph Advantages:

- Cyclic graph support for iterative assessment-feedback loops
- Explicit state transitions preventing non-deterministic agent behavior
- Built-in checkpointing for long-running agent tasks
- Human-in-the-loop integration points
- Streaming support for real-time progress updates

Technical Specifications:

Specification	Details
Language	Python 3.11+
Base Framework	LangChain + LangGraph
LLM Provider	Anthropic Claude 3.5 Sonnet / GPT-4o
Vector Store	Pinecone / pgvector (Phase 3)
Agent Communication	JSON over REST APIs / Message Queue
State Persistence	MongoDB (agent context and memory)

2.3 Frontend Layer

Technology Stack:

Component	Technology
Framework	Next.js 14+ (React)
Language	TypeScript
Styling	Tailwind CSS + shadcn/ui components
State Management	React Context API + TanStack Query
Data Visualization	Recharts / Chart.js
Real-time Updates	WebSocket (Socket.io)
Authentication	Auth0 / NextAuth.js

Key Frontend Features:

- Server-side rendering (SSR) for initial page loads under 2 seconds
- Progressive Web App (PWA) capabilities for mobile access
- Real-time dashboard updates via WebSocket connections
- Responsive design supporting desktop, tablet, and mobile devices

2.4 Backend API Layer

Technology Stack:

Component	Technology
Framework	FastAPI (Python) for agent services
Language	Python 3.11+
API Gateway	Node.js Express (high-concurrency routing)
Protocol	REST APIs + WebSocket
Authentication	JWT tokens (OAuth2 flow)
Rate Limiting	Redis-based token bucket
API Documentation	OpenAPI/Swagger auto-generated

API Gateway Responsibilities:

- Authentication and authorization enforcement
- Request validation and sanitization
- Rate limiting per user/role
- Load balancing across agent service instances
- Request/response logging for audit trails

2.5 Data Layer: Polyglot Persistence

Database	Data Types	Primary Use Cases
PostgreSQL	Structured/Relational	User accounts, schedules, assessment scores, certifications, curriculum definitions
MongoDB	Unstructured/Document	Raw code submissions, AI feedback logs, agent context/memory, audit trails
Redis	In-Memory Cache	Session management, rate limiting counters, real-time leaderboard data
AWS S3	Object Storage	Code submission files, generated reports (PDFs), profile images
Pinecone (Phase 3)	Vector Database	Semantic search over training content, RAG for chatbot queries

3. The Five Specialized Agents

Each agent is a specialized microservice with dedicated responsibilities, inputs, outputs, and AI reasoning capabilities. Agents communicate through standardized JSON messages via the n8n orchestration layer.

3.1 Onboarding Agent: The Architect

Responsibility: Creating, managing, and dynamically adapting personalized learning schedules for each fresher

Inputs:

- User profile data from HRIS
- Master curriculum structure
- Academic calendar (holidays, weekends)
- Assessment results and pass/fail status
- Remedial task triggers from Assessment Agent

Outputs:

- Personalized JSON schedule object (daily tasks)
- Calendar events (Google Calendar / Outlook integration)
- Email/push notifications for schedule updates
- Remedial task insertions

AI Logic & Algorithms:

- Constraint satisfaction algorithms to balance workload
- Prerequisite dependency resolution
- Dynamic schedule adjustment based on performance
- Holiday and resource availability consideration

Performance Requirements:

Metric	Target
Schedule Generation	< 5 seconds per user
Dynamic Adaptation	Real-time (< 1 minute from trigger)
Calendar Sync	< 10 seconds
Notification Delivery	< 30 seconds

3.2 Assessment Agent: The Evaluator

Responsibility: Providing objective (unit test-based) and subjective (LLM-based) feedback on code submissions, quizzes, and project work

Inputs:

- Code submissions (Python, Java, JavaScript)
- Quiz responses (MCQ, short answer)
- Assignment rubrics and grading criteria
- Unit test suites for automated testing

Outputs:

- Assessment object (score, pass/fail, timestamp)
- Qualitative feedback text (code style, efficiency, best practices)
- Skill tag updates for profile synchronization
- Flagged submissions requiring human review

AI Logic & Algorithms:

- Code execution in sandboxed Docker container
- Unit test execution and result parsing
- LLM analysis for code quality (GPT-4o / Claude 3.5 Sonnet)
- Semantic similarity checking for plagiarism detection
- Regression evaluation suite to ensure grading consistency

Performance Requirements:

Metric	Target
Code Execution	< 30 seconds (sandbox timeout)
LLM Feedback Generation	< 60 seconds
Quiz Grading	< 5 seconds (immediate)
Feedback Accuracy	> 90% alignment with human graders

3.3 Profile Agent: The Librarian

Responsibility: Maintaining real-time, accurate fresher profiles aggregating data from all assessment activities and external certifications

Inputs:

- Assessment results from Assessment Agent
- Certification documents and verification data
- System activity logs (login frequency, engagement)
- External skill endorsements

Outputs:

- Unified user profile object (JSON)
- Skill radar chart data (proficiency levels)
- Historical performance timeline
- Certification records with verification status

AI Logic & Algorithms:

- Event-driven data aggregation from multiple sources
- Skill proficiency calculation algorithms
- Data deduplication and conflict resolution
- Real-time profile update triggers

Performance Requirements:

Metric	Target
Profile Update Latency	< 5 seconds from assessment completion
Data Consistency	100% (event sourcing pattern)
Query Performance	< 100ms for profile retrieval

3.4 Analytics Agent: The Strategist

Responsibility: Performing cohort-level analysis, identifying performance patterns, predicting at-risk students, and recommending content improvements

Inputs:

- Aggregated profile data across active cohorts
- Historical performance baselines
- Behavioral data (login patterns, submission timing)
- Content effectiveness metrics

Outputs:

- Cohort health report (averages, distributions)
- At-risk student alerts (high/medium/low priority)
- Skill gap identification (topics causing difficulty)
- Content improvement recommendations
- Predictive dropout risk scores

AI Logic & Algorithms:

- Statistical analysis (mean, standard deviation, percentiles)
- Pattern recognition using machine learning models
- Time-series analysis for trend detection
- Anomaly detection for early warning systems

Performance Requirements:

Metric	Target
Cohort Analysis	Nightly batch processing
Risk Alert Latency	< 1 hour from detection
Prediction Accuracy	> 80% for dropout risk

3.5 Reporting Agent: The Communicator

Responsibility: Generating formatted PDF reports and dashboards for stakeholders, automating the weekly status reporting process

Inputs:

- Analytics report data (cohort statistics)
- Individual profile summaries
- Report template configurations

- Stakeholder distribution lists

Outputs:

- Formatted PDF reports with charts and tables
- HTML email summaries
- Automated distribution to stakeholders
- On-demand custom reports

AI Logic & Algorithms:

- HTML-to-PDF conversion with template rendering
- Chart generation (matplotlib / Chart.js)
- Email composition and SMTP integration
- Report scheduling and distribution management

Performance Requirements:

Metric	Target
Report Generation	< 30 seconds
PDF Quality	Publication-ready formatting
Delivery Success	> 99% email delivery rate

---

4. Workflow Orchestration & Event Flow

This section details the critical workflows implemented in n8n that orchestrate agent interactions and data flow through the platform.

4.1 Workflow: Daily Learning Schedule Activation

Purpose: Automatically generates and distributes daily task lists to all active freshers at 6:00 AM local time

Workflow Steps:

Step	Node Type	Description
1	Schedule Trigger	Cron expression: 0 6 * * * (daily at 06:00 AM)
2	Database Query	Fetch all users with status = 'Training' from PostgreSQL
3	Split in Batches	Process users in batches of 50 to avoid rate limits
4	Onboarding Agent API	POST /api/onboarding/generate-schedule for each user
5	Database Write	Insert daily_task_list into Daily_Schedules table
6	Google Calendar	Create/update calendar events via Google Calendar API
7	Notification Service	Send 'Your Day is Ready' push notification + email
8	Error Handler	Catch failures, log to MongoDB, alert admin if critical

Implementation Notes:

- Batch processing prevents API rate limit violations
- Idempotency checks prevent duplicate schedule creation
- Timezone-aware scheduling using user profile location data
- Failure of one user does not block processing for others

Workflow Diagram:

```
[Schedule: 6AM] → [DB: Get Active Users] → [Batch: 50 users] →
[For Each User]:
  → [Onboarding Agent] → [Generate Schedule] →
  → [DB: Save Schedule] → [Google Calendar API] →
  → [Send Notification]
→ [Error Logging] → [Admin Alert if Critical]
```

4.2 Workflow: Assessment Evaluation Pipeline

Purpose: End-to-end flow from code submission to graded feedback delivery

Workflow Steps:

Step	Node Type	Description
1	Webhook Trigger	Event: submission_received from Frontend API
2	Request Validation	Validate JWT token, submission format, file size limits
3	MongoDB Write	Store raw submission in Assessment_Logs collection
4	Assessment Agent API	POST /api/assessment/evaluate with submission payload
5	Switch Node	Route based on evaluation status (success / error / timeout)
6	Profile Agent API	POST /api/profile/update-skill with new score
7	Analytics Check	POST /api/analytics/check-risk-status
8	Conditional Branch	If at-risk detected, trigger Manager Alert workflow
9	WebSocket Push	Send 'Grading Complete' signal to Frontend via Socket.io
10	Email Notification	Send detailed feedback email to student

Implementation Notes:

- Asynchronous processing allows immediate user acknowledgment
- Timeout handling (30s sandbox, 60s LLM) prevents hanging workflows
- Dual-write pattern ensures data integrity (MongoDB + PostgreSQL)
- Real-time WebSocket update provides instant user feedback

Workflow Diagram:

```
[User Submits Code] → [Webhook Received] → [Validate Request] →
[Save to MongoDB] → [Assessment Agent]:
  → [Docker Sandbox: Execute Code + Unit Tests] →
  → [LLM Analysis: Code Quality] →
  → [Combine Results: Score + Feedback]
→ [Save Score to PostgreSQL] → [Update Profile Agent] →
→ [Check Risk Status]:
  → If At-Risk → [Alert Manager]
  → If OK → [Continue]
→ [WebSocket: Push to Frontend] → [Email: Send Feedback]
```

4.3 Workflow: Weekly Status Report Generation

Purpose: Automated generation and distribution of comprehensive training status reports every Friday at 4:00 PM

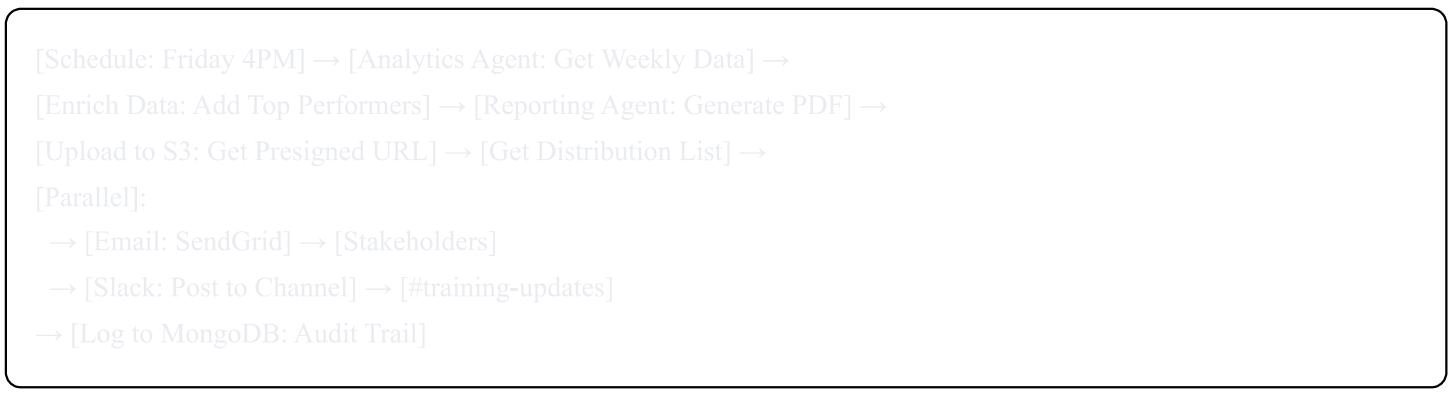
Workflow Steps:

Step	Node Type	Description
1	Schedule Trigger	Cron expression: 0 16 * * 5 (Fridays at 16:00)
2	Analytics Agent API	GET /api/analytics/weekly-summary
3	Data Enrichment	Join cohort data with individual top performer profiles
4	Reporting Agent API	POST /api/reporting/generate-pdf with summary data
5	S3 Upload	Store PDF in AWS S3 with expiring presigned URL
6	Distribution List Query	Fetch stakeholder emails from Configuration table
7	Email Send (SMTP)	Send PDF attachment + HTML summary via SendGrid
8	Slack Notification	Post summary message to #training-updates channel
9	Audit Log	Record distribution success in MongoDB for compliance

Implementation Notes:

- PDF generation uses headless Chrome for high-fidelity rendering
- Presigned URLs expire after 7 days for security
- Distribution list is configurable without code changes
- Comprehensive audit trail for SOC2 compliance

Workflow Diagram:



4.4 Workflow: At-Risk Student Intervention

Purpose: Triggered when Analytics Agent detects a student falling below performance thresholds

Workflow Steps:

Step	Node Type	Description
1	Event Trigger	Analytics Agent publishes 'student_at_risk' event
2	Risk Assessment	Fetch historical performance data for context
3	Severity Classification	Classify as High (2+ fails), Medium (1 fail), Low (trend)
4	Conditional Branch	Route based on severity level
5	High Severity Path	Immediate manager alert + schedule 1:1 meeting
6	Medium Severity Path	Assign remedial tasks + peer mentor pairing
7	Low Severity Path	Send motivational resources + skill-building tips
8	Onboarding Agent API	Adjust schedule to include remedial content
9	Notification Dispatch	Notify student, manager, and mentor as appropriate
10	Follow-up Scheduler	Set reminder for 3-day progress check

Implementation Notes:

- Multi-tier intervention strategy prevents alert fatigue
- Automated remedial task assignment reduces manager workload
- Follow-up tracking ensures no student falls through cracks
- Privacy-preserving alerts (managers see anonymized data initially)

Workflow Diagram:

[Analytics: Detect At-Risk] → [Fetch Student History] → [Classify Severity]:

→ If HIGH:
  → [Alert Manager: Urgent] → [Schedule 1:1 Meeting]
  → [Onboarding Agent: Add Remedial Tasks]

→ If MEDIUM:
  → [Assign Peer Mentor] → [Add Remedial Tasks]
  → [Notify Manager: Info]

→ If LOW:
  → [Send Resources] → [Monitor]
  → [Notify Manager: FYI]

→ [Update Schedule] → [Send Notifications] → [Schedule 3-Day Follow-up]

---

5. End-to-End Data Pipelines

This section maps complete data flows through the platform, from user action to final system response, illustrating cross-component interactions.

5.1 Complete Onboarding Pipeline

Flow: From new hire data entry to first day schedule delivery

Stage	Component	Details
Data Entry	HRIS (BambooHR/Workday)	HR team enters new hire profile with start date, role, background
Integration Sync	n8n Webhook Receiver	HRIS sends webhook to n8n on new hire creation event
Data Validation	API Gateway	Validate required fields (name, email, role, start_date)
Profile Creation	PostgreSQL Users Table	Insert user record with unique ID, role, initial status = 'Pending'
Curriculum Assignment	Onboarding Agent	Match user role to appropriate curriculum track (Dev/QA/DevOps)
Schedule Generation	Onboarding Agent	Generate 60-day learning plan with daily task breakdown
Calendar Integration	Google Calendar API	Create recurring calendar events for live sessions
Welcome Email	SendGrid SMTP	Send welcome email with login credentials and Day 1 agenda
Dashboard Prep	Frontend Cache	Pre-load dashboard data for fast first login experience
Status Update	PostgreSQL	Update user status = 'Training' on start date

Timeline: 5-10 minutes from HRIS entry to welcome email delivery

Data Flow Visualization:

5.2 Code Submission & Assessment Pipeline

Flow: Real-time evaluation flow from student code upload to graded feedback

Stage	Component	Details
User Action	Frontend (React)	Student submits Python code via Monaco editor
File Upload	API Gateway	Upload code file to S3, receive presigned URL
Webhook Trigger	n8n	API Gateway triggers 'submission_received' webhook
Raw Storage	MongoDB	Store original code, timestamp, metadata in Assessment_Logs
Sandbox Execution	Assessment Agent	Spin up Docker container, execute code against unit tests
Unit Test Results	Assessment Agent	Parse test output: 8/10 tests passed = 80% correctness score
LLM Analysis	Claude 3.5 Sonnet	Analyze code style, efficiency, best practices adherence
Feedback Synthesis	Assessment Agent	Combine unit test + LLM feedback into comprehensive JSON object
Score Logging	PostgreSQL	Insert into Assessments table: user_id, assignment_id, score, timestamp
Profile Update	Profile Agent	Recalculate skill averages for 'Data Structures' competency
Risk Check	Analytics Agent	If score < 50%, flag for 'At-Risk' workflow
Real-time Notification	WebSocket (Socket.io)	Push 'grading_complete' event to Frontend
Dashboard Update	Frontend	Display feedback modal with score, comments, suggested improvements
Email Delivery	SendGrid	Send detailed feedback email for offline review

Timeline: 30-90 seconds from submission to feedback display

Data Flow Visualization:

```
[Student: Submit Code]
  ↓
[Frontend: Upload to S3] → [API Gateway: Trigger Webhook]
  ↓
[n8n Orchestrator]:
  → [MongoDB: Save Raw Submission]
  → [Assessment Agent]:
      → [Docker: Execute Code]
      → [Unit Tests: 8/10 Pass]
      → [LLM: Analyze Quality]
      → [Combine: Score + Feedback]
  ↓
[PostgreSQL: Log Score]
  ↓
[Profile Agent: Update Skills]
  ↓
[Analytics Agent: Check Risk]
  → If At-Risk → [Trigger Alert Workflow]
  ↓
[Parallel Delivery]:
  → [WebSocket: Push to Frontend] → [Display Modal]
  → [SendGrid: Email Feedback] → [Student Inbox]
```

5.3 Analytics Aggregation & Reporting Pipeline

Flow: Batch processing flow for cohort analytics and executive reporting

Stage	Component	Details
Scheduled Trigger	n8n Cron	Daily at 02:00 AM - batch analytics processing begins
Data Extraction	PostgreSQL	Query last 24h assessments, attendance, engagement metrics
ETL Processing	Analytics Agent	Transform raw data: calculate averages, percentiles, distributions
Pattern Detection	Analytics Agent	Run ML models to identify: skill gaps, at-risk students, content issues
Insight Storage	MongoDB	Store analytics results in Analytics_Reports collection
Alert Generation	Analytics Agent	Generate at-risk student list with severity classification
Manager Notification	n8n Email Node	Send daily digest to Training Managers with action items
Weekly Aggregation	Analytics Agent (Friday)	Compile 7-day summary with trends, top performers, challenges
Report Generation	Reporting Agent	Render PDF report using HTML template + Chart.js visualizations
Report Distribution	SendGrid + Slack	Email PDF to stakeholders, post summary in Slack channel
Dashboard Sync	PostgreSQL	Update materialized views for fast dashboard queries

Timeline: 15-30 minutes for nightly batch; weekly report generated in < 2 minutes

Data Flow Visualization:

```
[Schedule: 2AM Daily]
  ↓
[PostgreSQL: Extract 24h Data]
  ↓
[Analytics Agent]:
  → [Transform: Calculate Statistics]
  → [Analyze: Identify Patterns]
  → [Predict: At-Risk Students]
  → [Recommend: Content Improvements]
  ↓
[MongoDB: Store Results]
  ↓
[Generate At-Risk Alerts] → [Email Managers]
  ↓
[Friday Only]:
  → [Compile Weekly Summary]
  → [Reporting Agent: Generate PDF]
  → [Upload to S3]
  → [Distribute via Email + Slack]
  ↓
[PostgreSQL: Update Dashboard Views]
```

---

6. Security Architecture & Compliance

6.1 Multi-Layer Security Model

Security Layer	Implementation
Network	AWS VPC with private subnets; Security groups restricting inbound traffic; WAF for DDoS protection
Application	RBAC (Role-Based Access Control): Fresher, Manager, Admin roles; JWT tokens with 1-hour expiration; OAuth2 flow for SSO integration
Data	Encryption at rest (AES-256) for PostgreSQL and MongoDB; TLS 1.3 for all API communication; PII field-level encryption
API	Rate limiting: 100 requests/minute per user; API key rotation every 90 days; Request signature verification

Security Layer	Implementation
Code Execution	Docker sandbox isolation for code evaluation; Resource limits (CPU: 0.5 core, Memory: 512MB, Time: 30s); Network disabled in sandbox
Monitoring	Real-time threat detection via AWS GuardDuty; Audit logging to immutable S3 bucket; Automated security scanning (OWASP Top 10)

6.2 Compliance & Audit Requirements

SOC2 Type II Compliance:

- Comprehensive audit logging of all system actions (who, what, when, from where)
- Immutable audit trail stored in MongoDB with 7-year retention
- AI decision traceability: every grade includes model version, prompt, confidence score
- Change management process with approval workflows for production deployments

GDPR Compliance:

- Data residency: all EU user data stored in EU-based AWS regions
- Right to erasure: automated PII deletion workflows
- Data minimization: only required fields collected and retained
- Consent management: explicit opt-in for analytics and profiling
- PII masking before any external API calls (including LLM providers)

Security Architecture Diagram:

```
[Internet]
  ↓
[WAF: DDoS Protection]
  ↓
[ALB: SSL Termination]
  ↓
[API Gateway]:
  → [JWT Validation]
  → [Rate Limiting]
  → [RBAC Check]
  ↓
[Private Subnet]:
  → [Agent Services]
  → [Databases: Encrypted at Rest]
  ↓
[Audit Trail]: All actions logged to immutable S3
```

---

7. Deployment Architecture & Infrastructure

7.1 Docker Container Architecture

Container	Base Image	Resources
Frontend (Next.js)	node:18-alpine	CPU: 1 core, Memory: 1GB
API Gateway (Express)	node:18-alpine	CPU: 2 cores, Memory: 2GB
n8n Orchestrator	n8nio/n8n:latest	CPU: 2 cores, Memory: 4GB
Agent Services (FastAPI)	python:3.11-slim	CPU: 2 cores, Memory: 4GB (per agent)
PostgreSQL	postgres:15-alpine	CPU: 4 cores, Memory: 8GB
MongoDB	mongo:7.0	CPU: 4 cores, Memory: 8GB
Redis	redis:7-alpine	CPU: 1 core, Memory: 2GB

7.2 Cloud Infrastructure (AWS)

Service	AWS Component	Configuration
Compute	EKS (Kubernetes)	3-node cluster (t3.xlarge), auto-scaling 3-10 nodes
Load Balancing	Application Load Balancer	Cross-AZ, SSL termination, health checks
Database	RDS PostgreSQL	db.r5.xlarge, Multi-AZ, automated backups
NoSQL Database	DocumentDB (MongoDB)	db.r5.large, 3-node replica set
Caching	ElastiCache Redis	cache.r5.large, cluster mode enabled
Object Storage	S3	Standard tier, versioning enabled, lifecycle policies
CDN	CloudFront	Edge caching for static assets, HTTPS only
DNS	Route 53	Latency-based routing, health checks

Infrastructure Diagram:

7.3 Monitoring & Observability

Category	Tool	Metrics
Infrastructure	Prometheus + Grafana	CPU, memory, disk, network utilization per container
Application	Custom Metrics	Agent response times, LLM token usage, API latency
Logging	ELK Stack	Centralized logs with full-text search, 30-day retention
Tracing	Jaeger	Distributed tracing across agent interactions
Alerting	PagerDuty	24/7 on-call rotation, escalation policies
User Analytics	Mixpanel	User engagement, feature adoption, funnel analysis

Monitoring Dashboard Example:

```
[Grafana Dashboard]:
  → Panel 1: System Health (CPU/Memory/Disk)
  → Panel 2: API Latency (P50/P95/P99)
  → Panel 3: Agent Performance (Response Times)
  → Panel 4: LLM Costs (Token Usage per Day)
  → Panel 5: User Activity (DAU/MAU)
  → Panel 6: Error Rates (4xx/5xx by Endpoint)
```

---

8. Implementation Roadmap

Phase	Timeline	Deliverables & Success Criteria
Phase 1: MVP	Weeks 1-8	• Core agent loop (Onboarding → Assessment → Profile) • Basic n8n workflows • Simple React dashboard • Docker Compose deployment **Success:** End-to-end flow works for 1 synthetic user
Phase 2: Pilot	Weeks 9-16	• Live 20-user cohort testing • HRIS integration (read-only) • Full dashboards (Manager + Fresher) • Human-in-the-loop grading approval

Phase	Timeline	Deliverables & Success Criteria
		• Security hardening (RBAC, encryption) **Success:** 90%+ feedback accuracy, no major incidents
Phase 3: Scale	Weeks 17-24	• Advanced Analytics (predictive models) • Gamification (leaderboards, badges) • Multi-tenant support (multiple tracks) • Remove human-in-the-loop for low-risk assessments • AWS production deployment **Success:** 500 concurrent users, < 2s dashboard load
Phase 4: Optimize	Weeks 25-32	• RAG chatbot (Pinecone vector search) • Mobile app (React Native) • Advanced reporting (custom dashboards) • Cost optimization (spot instances, caching) **Success:** < $1/user/month operating cost

Implementation Milestones:

```
Week 1-2:   Infrastructure setup (Docker, n8n, databases)
Week 3-4:   Onboarding Agent MVP
Week 5-6:   Assessment Agent MVP (basic grading)
Week 7-8:   End-to-end MVP testing
Week 9-10:  LLM integration (qualitative feedback)
Week 11-12: Profile + Analytics Agents
Week 13-14: Manager dashboard
Week 15-16: Pilot launch (20 users)
Week 17-20: Production infrastructure (AWS EKS)
Week 21-24: Scale testing + optimization
Week 25-28: Advanced features (RAG, mobile)
Week 29-32: Final testing + launch
```

---

9. Operational Procedures

9.1 Deployment Procedures

Standard Deployment Process:

1. Code review and approval via GitHub Pull Request
2. Automated testing (unit, integration, E2E) must pass 100%
3. Build Docker images and tag with semantic versioning

4. Deploy to staging environment for smoke testing
5. Production deployment during maintenance window (Saturday 02:00-04:00 AM)
6. Blue-green deployment with automated rollback on health check failure
7. Post-deployment monitoring for 1 hour before release approval

Deployment Command Example:

```bash
# Build and tag
docker build -t maverickai/assessment-agent:1.2.0 .

# Push to registry
docker push maverickai/assessment-agent:1.2.0

# Deploy to Kubernetes
kubectl set image deployment/assessment-agent \
  assessment-agent=maverickai/assessment-agent:1.2.0

# Monitor rollout
kubectl rollout status deployment/assessment-agent

# Rollback if needed
kubectl rollout undo deployment/assessment-agent
```

9.2 Backup & Disaster Recovery

Data Type	Backup Frequency	Retention & RTO
PostgreSQL	Continuous (WAL)	Point-in-time recovery, 30-day retention, RTO: 15 minutes
MongoDB	Daily snapshots	7 daily + 4 weekly backups, RTO: 1 hour
Code Submissions (S3)	Versioning enabled	Retain all versions for 90 days, RTO: 5 minutes
System State	Daily configuration backup	Infrastructure-as-Code (Terraform), Git versioned

Disaster Recovery Scenarios:

1. **Database Corruption:**
   - Restore from last clean snapshot
   - Replay WAL logs to current state

- Verify data integrity
- Resume operations

2. **Complete Region Outage:**
    - Failover to secondary AWS region
    - Update DNS to point to backup
    - Restore from cross-region replicas
    - RTO: 2 hours, RPO: 1 hour
3. **Security Breach:**
    - Isolate affected systems
    - Rotate all credentials
    - Audit logs for unauthorized access
    - Restore from known-good backup
    - Notify users per GDPR requirements

9.3 Incident Response Procedures

Severity	Response Time	Actions
P0: Critical	< 15 minutes	Platform down, data loss risk. Immediate page, war room, hourly updates
P1: High	< 1 hour	Degraded service, partial outage. Team alert, 4-hour updates
P2: Medium	< 4 hours	Non-critical feature broken. Assigned owner, daily updates
P3: Low	< 24 hours	Minor bug, no user impact. Triaged in next sprint planning

Incident Response Workflow:

```
[Alert Triggered] → [PagerDuty: Page On-Call]
  ↓
[< 15 min]: Acknowledge incident
  ↓
[Assess Severity]: P0/P1/P2/P3
  ↓
[P0/P1 Path]:
  → [Create War Room (Zoom)]
  → [Update StatusPage: Investigating]
  → [Gather Logs (ELK)]
  → [Identify Root Cause]
  → [Implement Fix]
  → [Verify Resolution]
  → [Update StatusPage: Resolved]
  → [Post-Mortem (48h later)]
```

---

10. Performance Benchmarks & SLAs

Metric	Target	Measured
Dashboard Load Time (P95)	< 2 seconds	Lighthouse Performance Score
Code Assessment Latency	< 60 seconds	Agent monitoring logs
API Response Time (P99)	< 500ms	APM (Application Performance Monitoring)
Database Query Time (P95)	< 100ms	PostgreSQL slow query log
System Uptime	> 99.9%	Pingdom / StatusPage
Concurrent Users	1000+	Load testing (k6 / JMeter)
Agent Accuracy	> 90%	Human grader comparison study

Cost Optimization

- **LLM API costs:** Estimated at $0.50-$1.00 per student per month
- **AWS infrastructure costs:** $2,000-$5,000/month for 500 concurrent users
- **Auto-scaling policies:** Reduce costs by 40% during off-peak hours

Performance Testing Strategy:

```
[Load Test Scenarios]:

1. Normal Load (500 users):
   - Sustained for 2 hours
   - All endpoints < 500ms P99

2. Peak Load (1000 users):
   - Sustained for 30 minutes
   - Dashboard load < 3s P95
   - No error rate increase

3. Stress Test (2000 users):
   - Find breaking point
   - Verify graceful degradation
   - Confirm auto-scaling triggers

4. Spike Test:
   - 0 → 500 users in 1 minute
   - Verify system handles sudden load
   - Check cache effectiveness
```

Appendix A: External Integration Specifications

External System	Integration Method	Data Exchanged
BambooHR / Workday	REST API via n8n	New hire profile, start date, role, department
Google Calendar	Google Calendar API	Training session schedules, 1:1 meetings, deadlines
Slack	Slack Webhooks + Bot API	At-risk alerts, daily digests, report notifications
Microsoft Teams	Microsoft Graph API	Alternative to Slack for Teams-based organizations
SendGrid / AWS SES	SMTP API	Email notifications, feedback delivery, report distribution
HackerRank / Codility	Webhook integration	External coding challenge results ingestion
Zoom / Google Meet	Calendar integration	Automatic meeting link generation for live sessions
Auth0 / Okta	OAuth 2.0 / SAML	Single Sign-On (SSO) for enterprise authentication

Integration Architecture:

```
[MaverickAI Platform]
  ↓
[n8n Orchestrator]
  ↓
[Integration Nodes]:
  → [HRIS: BambooHR] → [Sync User Profiles]
  → [Calendar: Google] → [Sync Schedules]
  → [Chat: Slack] → [Send Alerts]
  → [Email: SendGrid] → [Send Notifications]
  → [LMS: HackerRank] → [Import Results]
  → [Auth: Auth0] → [SSO Login]
```

Conclusion

MaverickAI represents a comprehensive, production-ready architecture for enterprise-grade AI-powered training management. The multi-agent system, orchestrated through n8n workflows, provides the flexibility, scalability, and intelligence required to transform fresher onboarding from a manual, reactive process into an autonomous, proactive ecosystem.

Key Architectural Strengths:

- **Modular microservices design** enabling independent scaling and maintenance
- **Event-driven architecture** ensuring loose coupling and fault tolerance
- **Comprehensive security and compliance** built into every layer
- **Observable, monitorable system** with complete audit trails
- **Phased implementation approach** minimizing risk and enabling early validation

This architecture document serves as the blueprint for development, deployment, and operational teams to build a system that will handle thousands of freshers while maintaining sub-second response times, 99.9% uptime, and enterprise-grade security.

End of Document