Software Architecture and Design Specification

Project: Automated Performance Appraisal System (APAS)

Version: 1.0

Authors: Swathi D, Sneha Verma, Suraj Singh, Taha Hussain

Date: 30-09-2025

Status: Draft

Revision History

Version- 1.0 | Date- 30-09-2025 | Authors- Swathi D, Sneha Verma, Taha Hussain, Suraj Singh| Change Summary- Initial Draft

Approvals

Role	Name	Signature / Date
QA Lead	Swathi	[illegible]
Dev Lead	Sneha	[illegible]
Test Engineer	Taha	[illegible]
Product Owner	Suraj	[illegible]

1. Introduction

1.1 Purpose

This document specifies the architecture and design of the Automated Performance Appraisal System (APAS), including its components, modules, workflows, APIs, and security design.

1.2 Scope

The system provides automated employee appraisal services, including:

- Employee self-appraisal
- Manager review and rating
- 360-degree peer feedback
- KPI/goal tracking
- HR dashboard and analytics
- Notification and reporting

1.3 Audience

This document is intended for:

- Developers (implementation)
- QA Engineers (testing and verification)
- Security Auditors (compliance validation)
- HR Managers/Product Owners (functional oversight)
- Maintenance Teams

1.4 Definitions

APAS – Automated Performance Appraisal System

KPI – Key Performance Indicator

HRMS – Human Resource Management System

RBAC – Role-Based Access Control

UAT – User Acceptance Testing

2. Document Overview

2.1 How to use this document

This document provides architectural deliverables, including:

- UML diagrams (use-case, class, sequence)
- Component descriptions
- API design
- Threat model and security architecture
- Technology stack and risks

2.2 Related Documents
Software Requirements Specification (SRS)

Software Test Plan (STP)

Requirements Traceability Matrix (RTM)

3. Architecture

3.1 Goals & Constraints

Goals: Secure, reliable, scalable, and user-friendly appraisal system with ≥99.5% availability.

Constraints: Must comply with HR data policies, data privacy rules, and role-based access restrictions.

3.2 Stakeholders & Concerns

Employees: Easy-to-use, transparent, confidential appraisals.

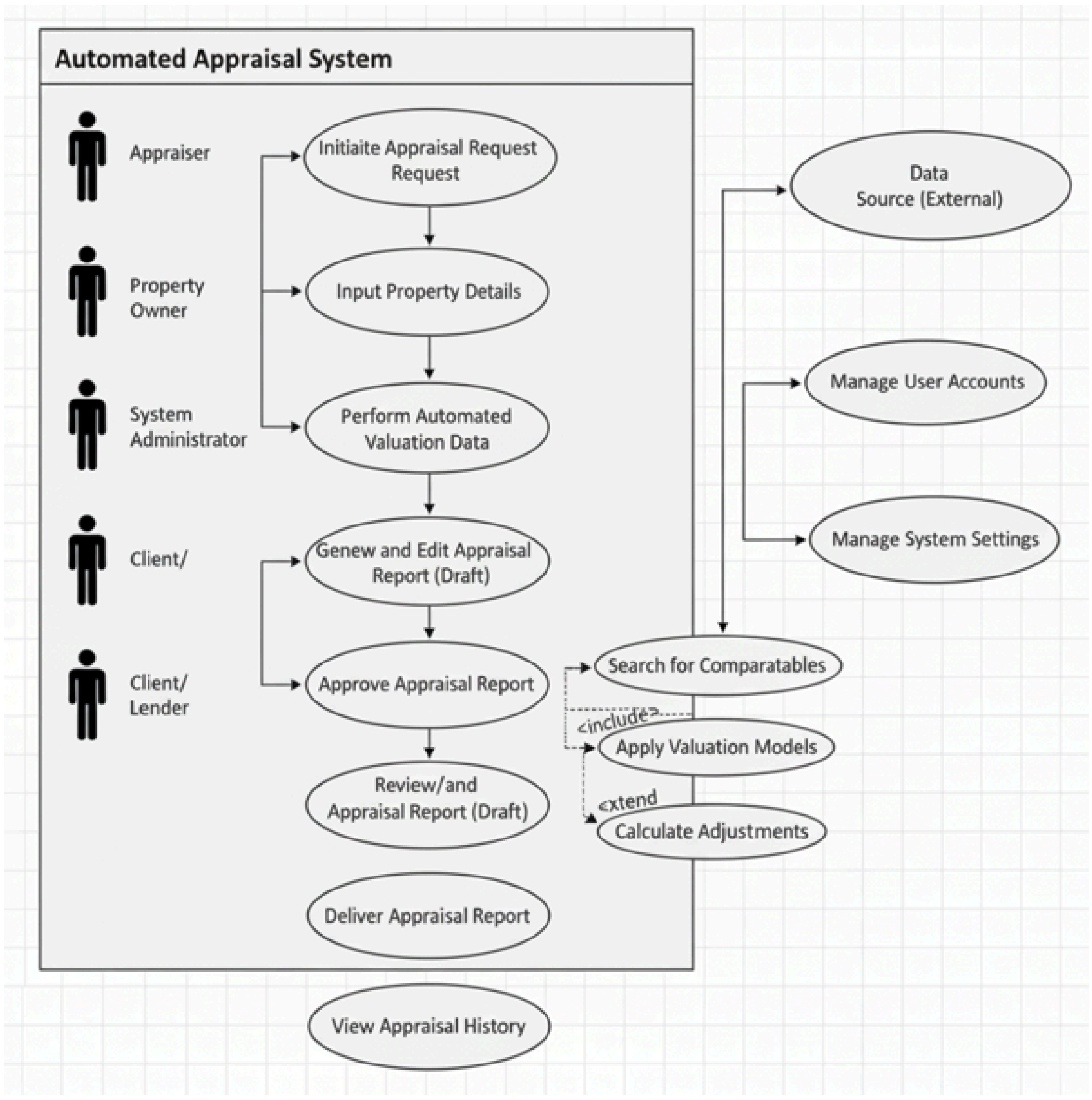
Managers: Efficient review tools and analytics.

HR/Admins: Configurability, reports, compliance.

Regulators: Data protection compliance.

Developers: Maintainable and modular system.

3.3 Component (UML) Diagram



3.4 Component Descriptions

Employee Portal: Submit self-appraisal, view goals, receive notifications.

Manager Portal: Review employee appraisals, give ratings, track KPIs.

360 Feedback Module: Collect anonymous peer reviews.

HR Dashboard: Configure cycles, rating scales, goals, and policies.

Reports Engine: Generate appraisal reports and analytics.

Notification Service: Sends reminders and alerts (email/portal).

Database: Stores employee data, appraisals, goals, feedback.

3.5 Chosen Architecture Pattern and Rationale

A **layered architecture** is chosen:

- **Presentation Layer** → Web/Mobile UI
- **Application Layer** → Business logic (appraisal workflows)
- **Data Layer** → Database, storage
- **Integration Layer** → APIs with HRMS/Notification services

Rationale: Clear separation of concerns, scalability, easier maintenance.

3.6 Technology Stack & Data Stores

Frontend: ReactJS / Angular

Backend: Node.js / Express

Database: MongoDB / PostgreSQL

Security: TLS, JWT authentication, RBAC

Notification: Email/SMS gateway

3.7 Risks & Mitigations

Risk: Sensitive employee data leak → **Mitigation:** End-to-end encryption + RBAC

Risk: Delays in review cycle → **Mitigation:** Auto-reminder notifications

Risk: Poor adoption due to usability → **Mitigation:** Simple, responsive UI

3.8 Traceability to Requirements

R1 (Employee Login): → Auth Service

R2 (Self-Appraisal): → Employee Portal

R3 (Manager Review): → Manager Portal

R4 (360 Feedback): → Feedback Module

R5 (Report Generation): → Reports Engine

3.9 Security Architecture

Threat Modeling (STRIDE):

- **Spoofing:** Prevented via strong authentication (SSO/JWT).
- **Tampering:** Audit trails & DB integrity checks.
- **Information Disclosure:** TLS + AES-256 encryption.
- **Denial of Service:** Rate limiting + monitoring.
- **Elevation of Privilege:** Enforced RBAC.

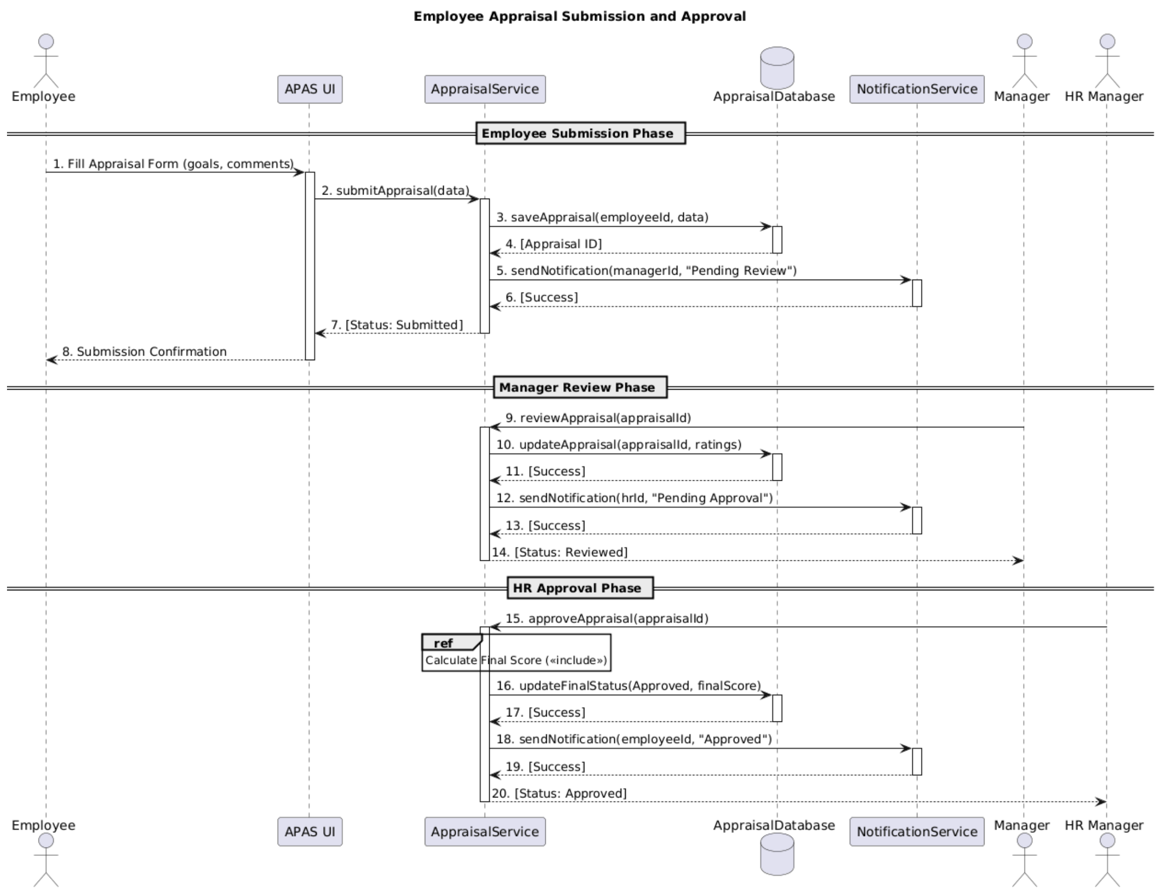
4. Design

4.1 Design Overview

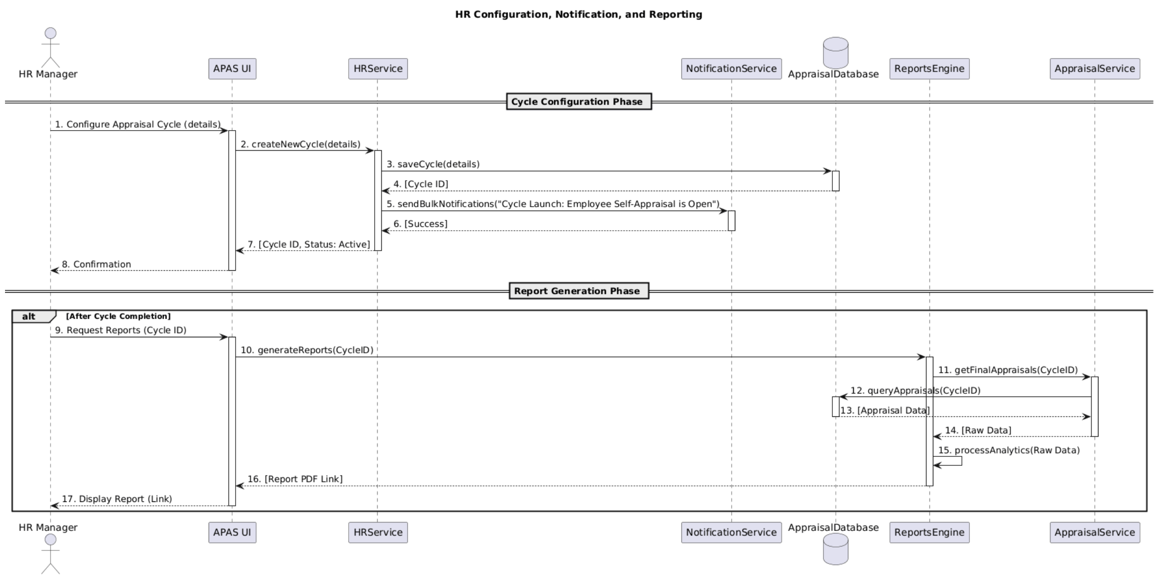
APAS is designed as a multi-tier web application with modular services for self-appraisal, manager evaluation, HR admin, and analytics.

4.2 UML Sequence Diagrams

1. **Employee submits self-appraisal → Manager reviews → HR approves**


Employee Appraisal Submission and Approval
Employee
APAS UI
AppraisalService
AppraisalDatabase
NotificationService
Manager
HR Manager
Employee Submission Phase
1. Fill Appraisal Form (goals, comments)
2. submitAppraisal(data)
3. saveAppraisal(employeeId, data)
4. [Appraisal ID]
5. sendNotification(managerId, "Pending Review")
6. [Success]
7. [Status: Submitted]
8. Submission Confirmation
Manager Review Phase
9. reviewAppraisal(appraisalId)
10. updateAppraisal(appraisalId, ratings)
11. [Success]
12. sendNotification(hrId, "Pending Approval")
13. [Success]
14. [Status: Reviewed]
HR Approval Phase
15. approveAppraisal(appraisalId)
ref
Calculate Final Score («include»)
16. updateFinalStatus(Approved, finalScore)
17. [Success]
18. sendNotification(employeeId, "Approved")
19. [Success]
20. [Status: Approved]

2. **HR configures appraisal cycle → Notifications sent → Reports generated**


HR Configuration, Notification, and Reporting
HR Manager
APAS UI
HRService
NotificationService
AppraisalDatabase
ReportsEngine
AppraisalService
Cycle Configuration Phase
1. Configure Appraisal Cycle (details)
2. createNewCycle(details)
3. saveCycle(details)
4. [Cycle ID]
5. sendBulkNotifications("Cycle Launch: Employee Self-Appraisal is Open")
6. [Success]
7. [Cycle ID, Status: Active]
8. Confirmation
Report Generation Phase
alt
[After Cycle Completion]
9. Request Reports (Cycle ID)
10. generateReports(CycleID)
11. getFinalAppraisals(CycleID)
12. queryAppraisals(CycleID)
13. [Appraisal Data]
14. [Raw Data]
15. processAnalytics(Raw Data)
16. [Report PDF Link]
17. Display Report (Link)
HR Manager
APAS UI
HRService
NotificationService
AppraisalDatabase
ReportsEngine
AppraisalService

4.3 API Design

Example API endpoints:

Endpoint 1: Submit Appraisal

- Method: POST
- Request: `{ employeeId, goals, ratings, comments }`
- Response: `{ status, appraisalId }`
- Errors: 400 Invalid Data, 401 Unauthorized

Endpoint 2: Get Employee Report

- Method: GET
- Request: `/report/{employeeId}`
- Response: `{ reportData }`
- Errors: 404 Not Found, 403 Forbidden

4.4 Error Handling, Logging & Monitoring

- Standardized error messages for users.
- No sensitive appraisal content logged.
- Monitoring: cycle completion rate, failed submissions, notification failures.

4.5 UX Design

- Simple employee dashboard with form-based appraisal submission.

- Manager dashboard with comparative analytics.
- HR dashboard with cycle progress and reporting.

- Accessibility: WCAG 2.1 compliance.

4.6 Open Issues & Next Steps

- Possible integration with Payroll/HRMS in future.
- Support for AI-driven appraisal suggestions.
- Mobile app version for employees.

---

5. Appendices

5.1 Glossary

- **KPI:** Key Performance Indicator
- **360 Feedback:** Peer review system
- **RBAC:** Role-Based Access Control
- **JWT:** JSON Web Token

5.2 References

- IEEE 42010: Systems & Software Architecture Description
- OWASP Top 10 Security Practices
- HR Appraisal Policy Handbook

5.3 Tools

- UML: PlantUML / Lucidchart / draw.io
- API Design: Swagger
- Monitoring: Grafana, ELK stack