Software Architecture and Design Specification (SAD)

Project: Peer-to-Peer Tutoring Scheduler

Version: 1.0

Authors: Bhavini Shrutha M(PES1UG23CS145), Chethana K(PES1UG23CS164), Dareddy Devesh Reddy(PES1UG23CS171), Chiyedu Vishnu(PES1UG23CS169)

Date: 18-09-2025

Status:Draft / Review

Revision History

Version	Date	Author	Change Summary
1.0	18-09-2025	Team P2P	Initial SAD aligned with SRS v1.0 and Test Plan v1.0

Approvals

Role	Name
QA Lead	Dareddy Devesh Reddy
Dev Lead	Chethana K
Product Owner	Chiyedu Vishnu
Core Developer	Bhavini Shrutha M

1. Introduction

1.1 Purpose

This SAD specifies the architecture and design of the Peer-to-Peer Tutoring Scheduler. It translates the functional, non-functional, and security requirements defined in the SRS and ensures alignment with the Test Plan.

1.2 Scope
Covers core system services: user authentication, slot creation & editing, calendar browsing & booking, conflict resolution, notifications & reminders, booking cancellation & history, admin management, reporting & oversight. Excludes integration with external video-conferencing tools.

1.3 Audience
Students, Tutors, Administrators, Developers, QA engineers, Security Auditors, System Architects, Maintainers.

1.4 Definitions
Slot, Booking, Reminder Job, Conflict Resolution, RBAC, TLS, OAuth2.

2. Document Overview

2.1 How to use this document
Provides UML diagrams, chosen architecture pattern, technology stack, security model, and API design.

2.2 Related Documents
*Peer-to-Peer Tutoring Scheduler SRS v1.0
*Test Plan v1.0
*Requirements Traceability Matrix (RTM)

3. Architecture

3.1 Goals & Constraints

- **Goals:** Reliable scheduling, secure & conflict-free booking, scalability to 1000+ concurrent users, responsive UI, ≥99.5% uptime, accessible (WCAG 2.1 AA).

- **Constraints:** Dependency on external email APIs, requires stable internet, limited to supported browsers.

3.2 Stakeholders & Concerns

Students: ease of booking, timely reminders.

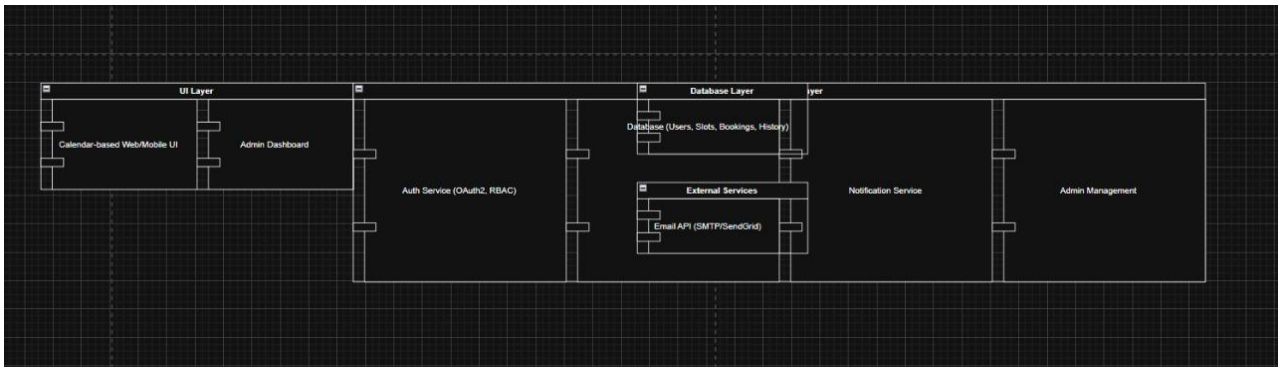
Tutors: flexible slot management, recurring availability.

Admins: oversight, reporting, conflict resolution.

Developers: maintainability, modularity.

Regulators: privacy & security compliance.

3.3 Component (UML) Diagram



3.4 Component Descriptions

UI Layer: Calendar-based web and mobile-friendly interface.

Auth Service: OAuth2 login, role-based access control.

Booking Engine: Slot creation, conflict resolution, booking limits.

Notification Service: Email confirmations, reminders, cancellations.

Admin Dashboard: User and slot management, conflict handling.

Database Layer: Stores users, slots, bookings, history.

3.5 Chosen Architecture Pattern & Rationale

Layered Architecture (Presentation, Business Logic, Data). Rationale: separation of concerns, scalability, testability.

3.6 Technology Stack & Data Stores

- Frontend: ReactJS
- Backend: Node.js + Express

- Database: MySQL / PostgreSQL
- Email Service: SMTP / SendGrid
- Security: TLS 1.2+, bcrypt password hashing

3.7 Risks & Mitigations

- Email delivery failure → fallback email provider
- High concurrency → load balancing & stress testing
- Security breaches → penetration testing, RBAC, audit logging

3.8 Traceability to Requirements

- P2P-F-001 → Booking Engine
- P2P-F-003 → Booking Engine + Admin Dashboard
- P2P-F-004 → Notification Service
- P2P-NF-001 → Performance Testing Layer
- P2P-SR-002 → Auth Service

3.9 Security Architecture

STRIDE Model:

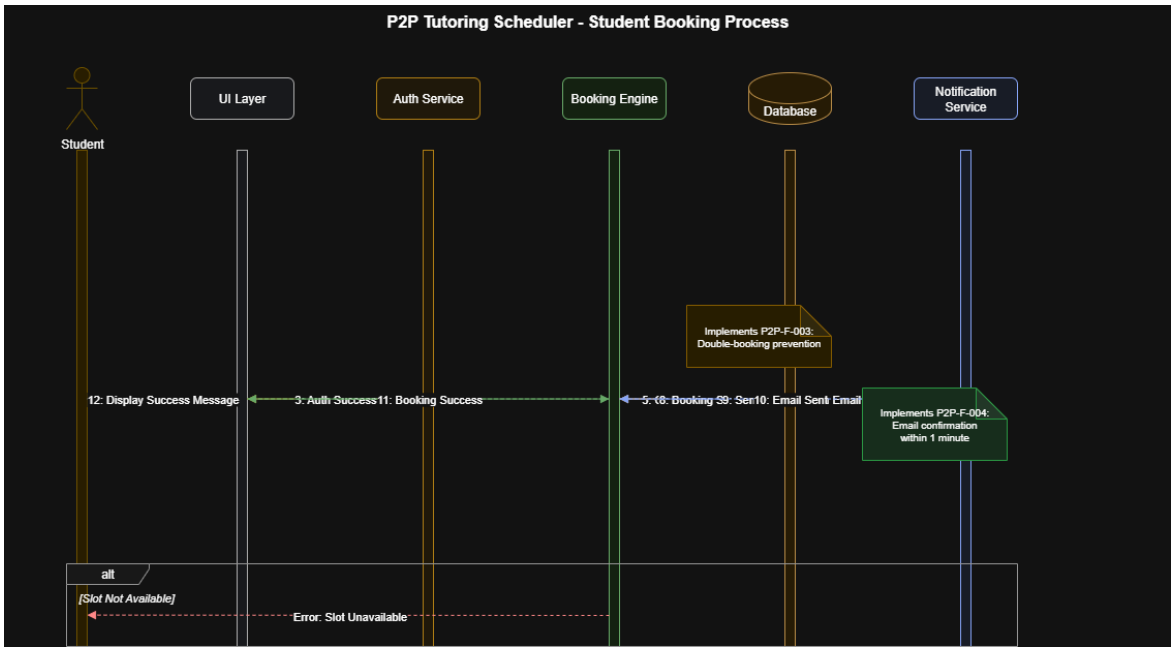
- **Spoofing:** OAuth2 login
- **Tampering:** TLS, hashed & salted passwords
- **Repudiation:** audit logs
- **Info Disclosure:** encryption
- **DoS:** load balancing

- **Privilege Escalation:** RBAC

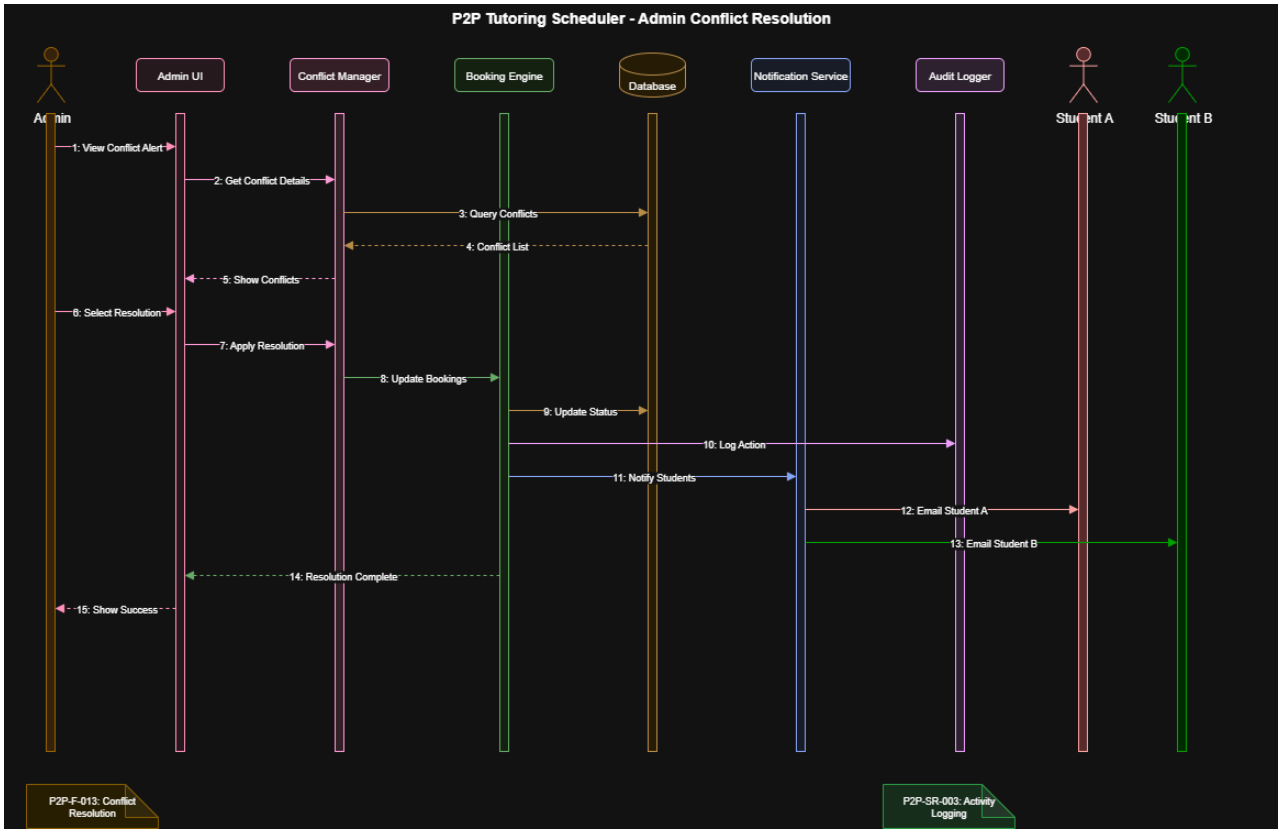
4. Design

4.1 Design Overview
Layered modular design: UI → Business Logic (Booking Engine, Notifications) → Data Layer (Database).

4.2 UML Sequence Diagrams
1. Student Booking Flow



2. Admin Conflict Resolution Flow



4.3 API Design

- `/slots/create` → POST {tutorId, time, subject} → {slotId, status}
- `/slots/book` → POST {slotId, studentId} → {bookingId, status}
- `/slots/cancel` → POST {bookingId} → {status}
- Errors: 401 Unauthorized, 403 Limit exceeded, 409 Conflict.

4.4 Error Handling, Logging & Monitoring

- Invalid login → 401 Unauthorized
- Double booking attempt → 409 Conflict
- Expired slots → auto-cancel with log entry
- Monitoring: system uptime, booking failures, email delivery success

4.5 UX Design

Responsive calendar UI, accessibility (WCAG 2.1 AA), mobile-friendly design, high contrast mode, keyboard navigation.

4.6 Open Issues & Next Steps

- Integration with third-party video-conferencing
- Push notification support
- AI-based tutor recommendations

5. Appendices

5.1 Glossary
Slot, Booking, Reminder Job, Conflict Resolution, TLS, OAuth2.

5.2 References
IEEE 42010, OWASP, WCAG 2.1, Peer-to-Peer Scheduler SRS v1.0, Test Plan v1.0.

5.3 Tools
PlantUML, Draw.io, Swagger, Postman, JMeter.