Maplewood Academic Scheduling System Backend Documentation

This document provides a comprehensive backend technical specification for the Maplewood Academic Scheduling System. It includes architecture, design principles, entity definitions, service responsibilities, and placeholders for diagrams.

1. System Overview

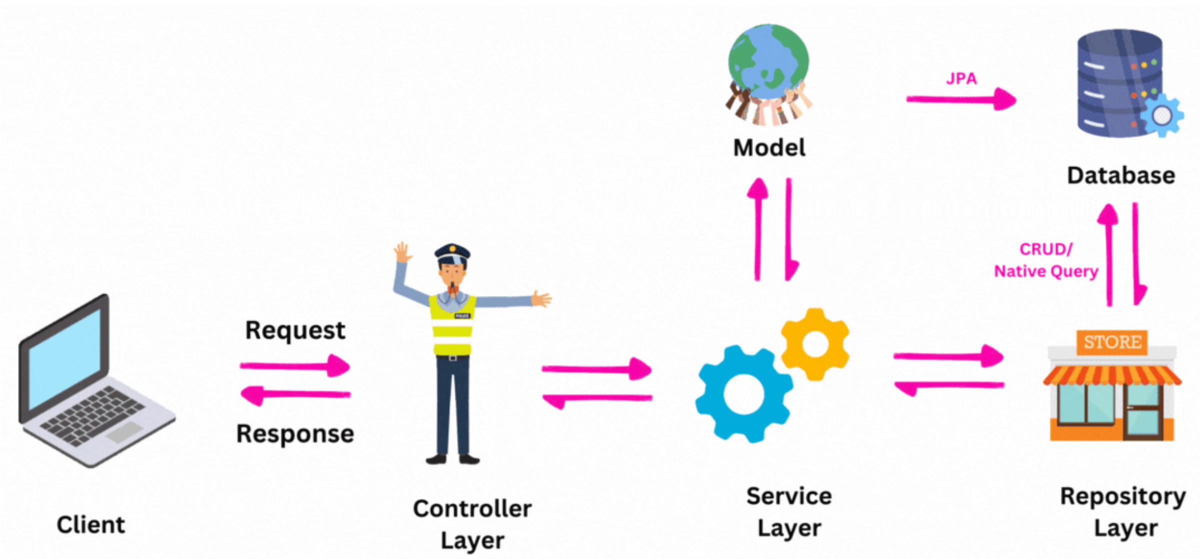
The backend is built using Java and Spring Boot, following a layered architecture. It manages academic scheduling, course offerings, student enrollment, teacher assignments, academic records, progress tracking and resource utilization.

2. High-level Architecture

The system follows the classic layered architecture:

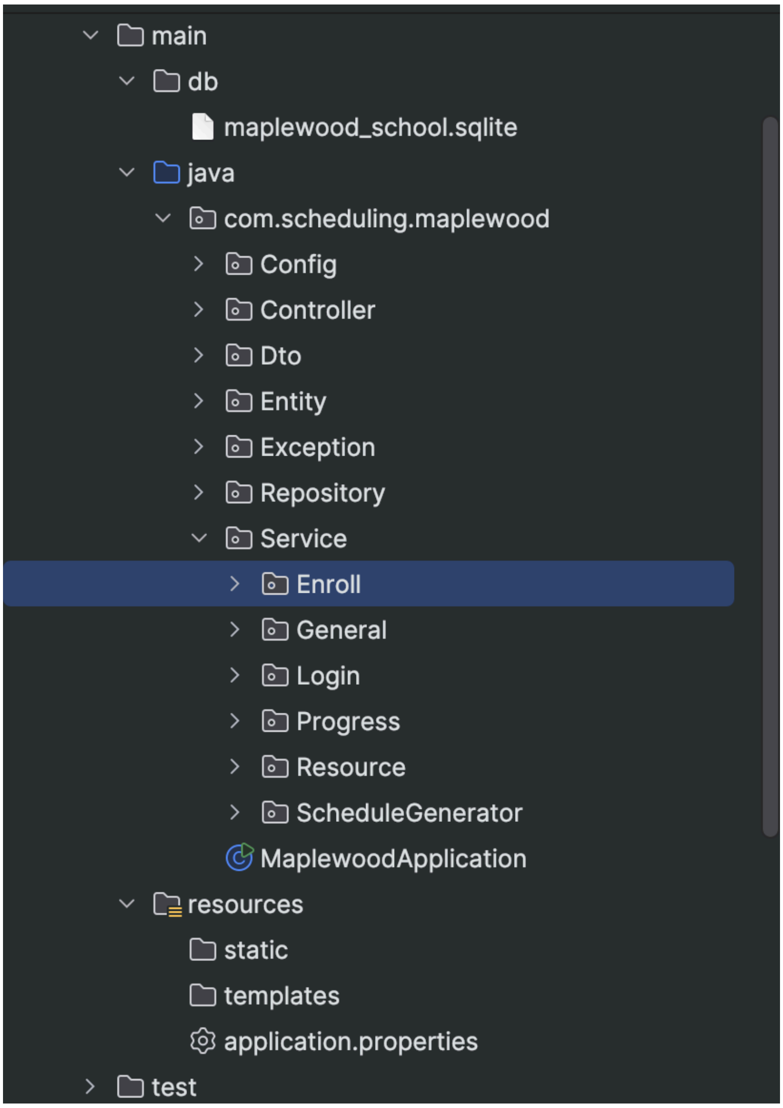
- Controller Layer — Handles HTTP requests and exposes REST APIs.
- Service Layer — Contains business logic and orchestration.
- Repository Layer — Interfaces with the database using Spring Data JPA.
- Persistence Layer — SQLite relational database.

Architecture Diagram



3. Folder Structure

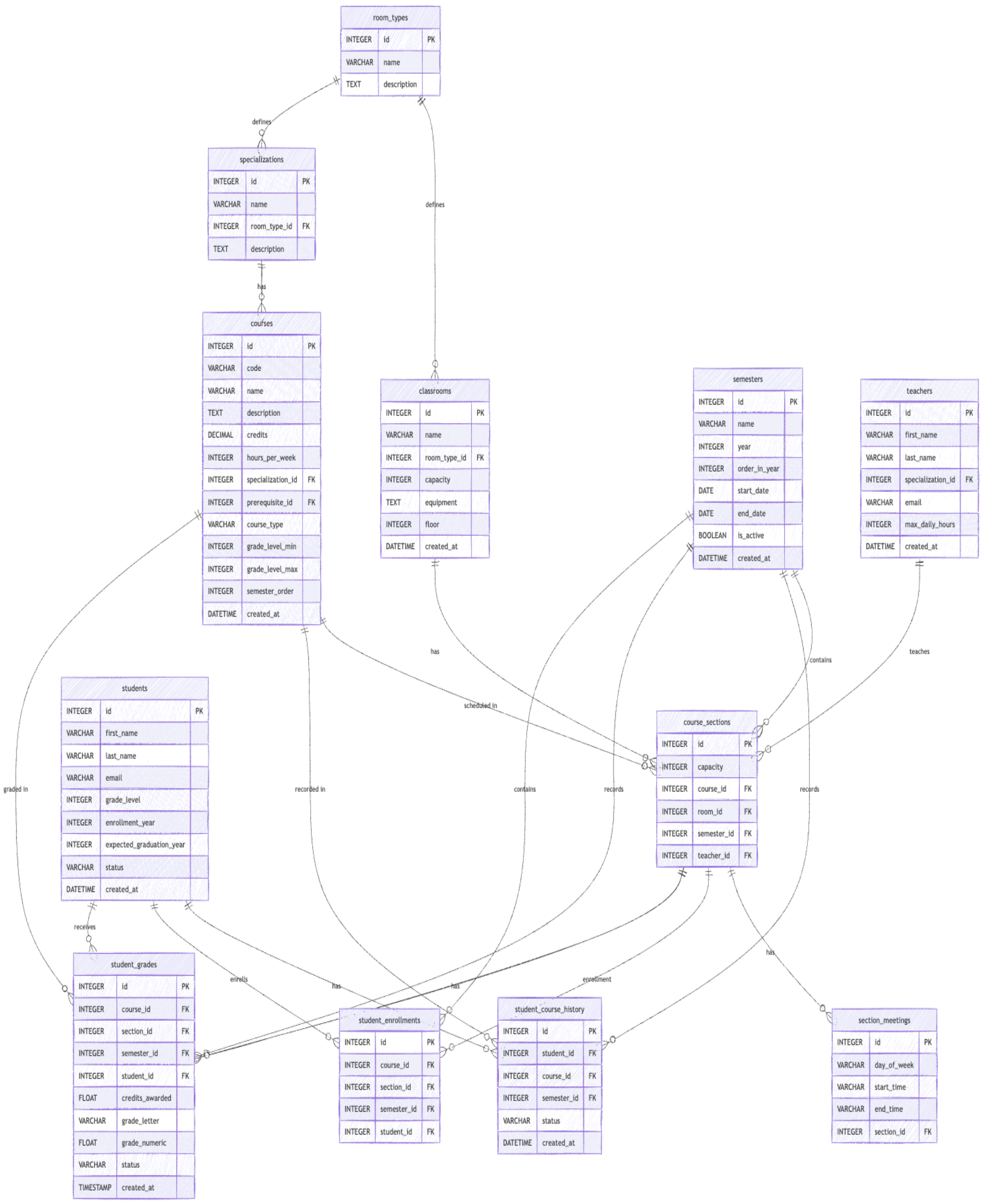
• Config — Application configuration (CORS, Swagger, initialization)
• Controller — REST controllers to perform functionalities
• DTO — Request/response objects
• Entity — JPA entities mapped to DB tables
• Repository — Spring Data repositories for DB access
• Service — Business logic
• Exception — Custom exceptions
• Resources — Application configuration files



4. Database Summary

The SQLite database schema contains tables for classrooms, room types, courses, course sections, section meetings, semesters, specializations, students, teachers, student course history, student enrollments, and student grades.

ERD Diagram



5. Key Backend Services

The system contains several domain-specific services:

1. Login Service – Handles authentication logic
2. ScheduleGeneratorService — Automated scheduling
3. Enrollment Service - Handles course enrollment logic
4. AcademicProgress Service – Computes academic metrics
5. Transcript Service – Generates academic trabscripts
6. Resource Utilization Service – Tells Staff and Room utilization details

6. Service Summary

6.1 ScheduleGeneratorService

Purpose

Generates the full semester schedule by:
Creating course sections
Assigning available teachers and classrooms
Scheduling weekly meeting times
Returning master/teacher/course schedules

Design Principles

- SRP – Only responsible for schedule creation & formatting
- Separation of Concerns – Availability, assignment, and time scheduling handled by separate services
- Dependency Injection – Clean, testable, loosely coupled
- Transactional Integrity – Ensures schedule generation completes fully or rolls back

Design Patterns

- Service Layer Pattern – Business logic in a dedicated service
- Repository Pattern – Data access through Spring repositories
- Mapper Pattern – Methods converting entities → response maps
- Strategy-like Delegation – Teacher/room selection delegated to AssignmentService

Algorithms Used

1. Course Scheduling Loop – Iterates over all courses and builds sections Constraint-based Assignment – Picks teachers and rooms based on availability rules
2. Greedy Time Scheduling – Places meetings in the earliest valid time slots
3. Sorting Algorithm – Orders meetings by weekday
4. Grouping – Groups sections by course code for course schedule response

6.2 EnrollmentService

Purpose

Handles student enrollment into course sections, validating prerequisites, capacity, schedule conflicts, and enrollment rules.

Design Principles

- SRP – Only manages enrollment logic
- Validation before mutation – All checks performed before saving
- Transactional integrity – Enrollment happens atomically
- Separation of Concerns – Data access delegated to repositories

Design Patterns

- Service Layer Pattern
- Repository Pattern
- Validation Pattern (series of rule checks)
- Recursive Pattern (prerequisite checking)

Algorithms Used

1. Section Selection Algorithm – Picks first available section with capacity
2. Validation Pipeline – Executes multiple enrollment rules sequentially
3. Prerequisite Checking (Recursive) – Walks prerequisite chain
4. Conflict Detection Algorithm – Compares meeting time ranges
5. Counting Algorithm – Counts courses taken in semester and enrolled students

6.3 EligibilityService

Purpose

Determines which course sections a student is eligible to enroll in based on history, prerequisites, capacity, and schedule conflicts.

Design Principles

- SRP – Dedicated to eligibility checks
- Separation of Concerns – Mapping, validation, and formatting isolated
- Fail-fast filtering – Quickly eliminates ineligible sections

Design Patterns

- Service Layer Pattern
- Repository Pattern
- Mapper Pattern (section → response item)
- Recursive Pattern (prerequisite validation)

Algorithms Used

1. Eligibility Filtering Algorithm – Sequential filtering of sections
2. Prerequisite Validation (Recursive)
3. Time Conflict Detection – Time overlap comparison
4. Sorting Algorithm – Orders meetings by weekday
5. Capacity Check Algorithm

6.4 ScheduleService

Purpose

Generates a student's full semester schedule, formatting each enrolled section with course info, teacher, room, and meeting times.

Design Principles

- SRP – Only responsible for building schedule responses
- Separation of Concerns – Mapping logic separated into helper functions
- DRY – Unified formatting for schedule items

Design Patterns

- Service Layer Pattern
- Repository Pattern
- Mapper Pattern
- Comparator Pattern (custom ordering of days)

Algorithms Used

1. Enrollment Lookup Algorithm – Retrieves sections for the student
2. Sorting Algorithm – Orders meetings by weekday
3. DTO Mapping Algorithm – Converts section data into schedule items
4. Capacity Calculation – Computes remaining seats

6.5 AcademicProgressService

Purpose

Calculates a student's academic progress by computing earned credits, remaining credits, GPA, passed core courses, and predicting semesters needed to graduate.

Design Principles

- SRP – Dedicated only to progress calculation and reporting
- Separation of Concerns – GPA, remaining core, and credit calculations isolated in helper methods
- DRY – Repeated course lookup and filtering kept minimal

Design Patterns

- Service Layer Pattern
- Repository Pattern
- Mapper Pattern (building output maps)

Algorithms Used

1. Credit Summation Algorithm – Computes earned/attempted credits
2. GPA Calculation Algorithm – Weighted quality-points formula
3. Core Course Completion Algorithm – Counts passed core requirements
4. Remaining Core Detection – Filters out passed courses
5. Projection Algorithm – Predicts semesters needed using average load

6.6 AcademicTranscriptService — Short Documentation

Purpose

Builds a complete academic transcript for a student, listing every course taken with credits, semester, grade status, and timestamps.

Design Principles

- SRP – Handles transcript generation only
- Separation of Concerns – Data retrieval, formatting, and mapping kept independent
- Fail-safe Lookups – Handles missing course/semester records gracefully

Design Patterns

- Service Layer Pattern
- Repository Pattern
- Mapper Pattern (each course history → transcript row)

Algorithms Used

1. History Iteration Algorithm – Loop through past courses
2. DTO Mapping Algorithm – Converts history entries → transcript rows
3. Semester Labeling Algorithm – Formats “Fall 2024” style names

6.7 ResourceUtilizationService

Purpose

Analyzes teacher and classroom utilization by calculating workload hours, daily loads, usage percentages, and detecting scheduling conflicts.

Design Principles

- SRP – Focused solely on workload/utilization analytics
- Separation of Concerns – Workload, conflicts, and day-based grouping in separate helpers
- Reusability – Shared time-calculation logic used across teacher and room analytics

Design Patterns

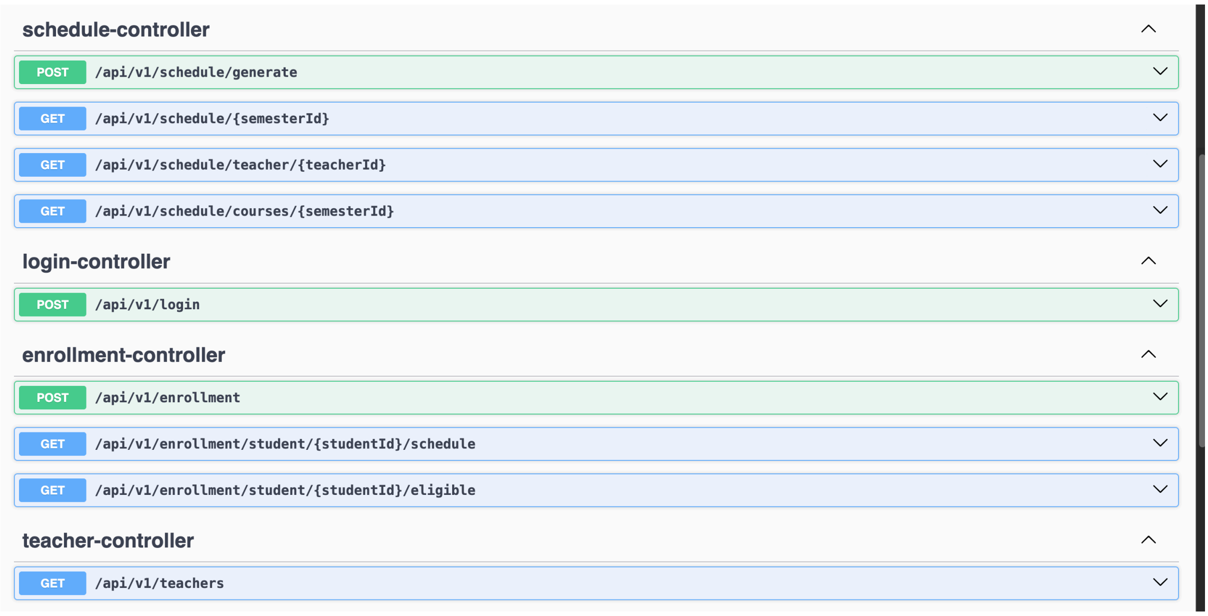
- Service Layer Pattern
- Repository Pattern

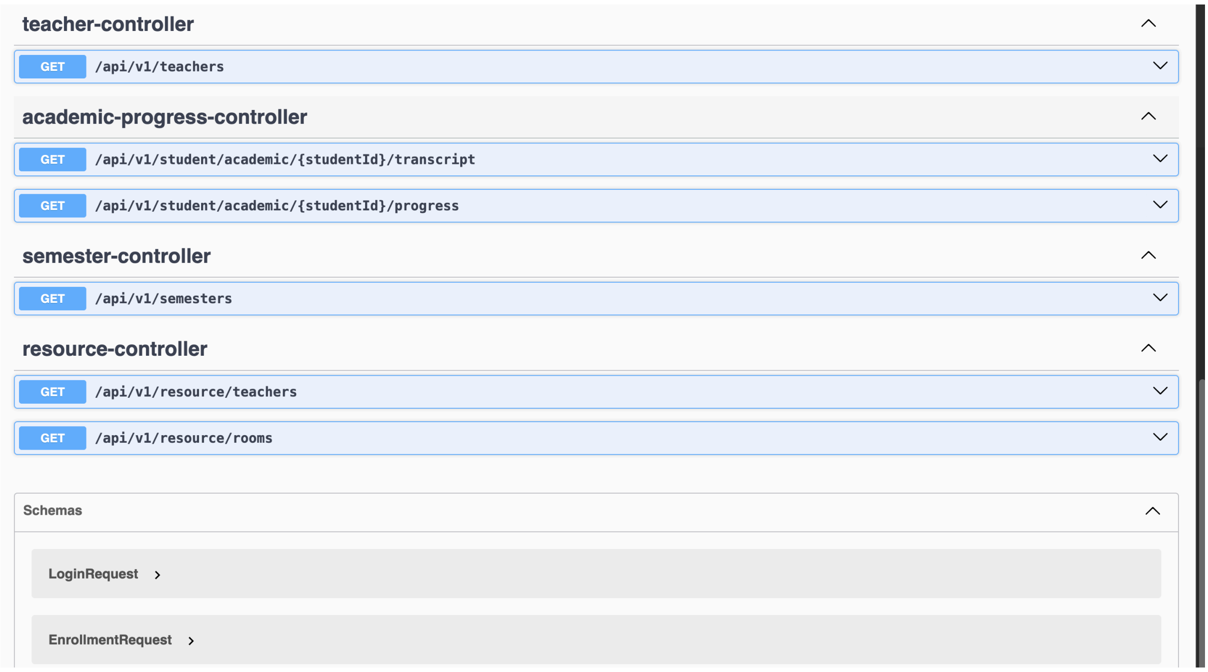
- Aggregator Pattern (collects meetings grouped by teacher/room)
- Mapper Pattern (formats workload output)

<u>Algorithms Used</u>

1. Weekly Hours Calculation – Summation of (end – start) for all meetings
2. Daily Load Aggregation – Groups meeting hours by weekday
3. Utilization Percentage Calculation – (weeklyHours / totalAvailableHours) * 100
4. Conflict Detection Algorithm – $O(n^2)$ comparison of meeting overlaps
5. Grouping Algorithm – Maps sections → teacher or classroom

7. API Documentation





8. Error Handling Strategy

- Custom exceptions include EnrollmentException and others.
- Error responses follow a unified ApiResponse DTO.
- Typical behaviors include:
  Validations
  Exception mapping
  Standardized HTTP responses

9. Security Design

Security considerations:

- Authentication handled by LoginService.
- Authorization rules applied based on user role (student/teacher/admin).
- Input validation in controllers.
- SQL injection prevented through ORM.