High School – Digital Transformation

Technology Stack

- **Backend:** Java 17, Spring Boot 3.5.7
- **Frontend:** Angular 17
- **Database:** SQLite (with JPA/Hibernate)

Key Assumptions

1. Simple login for user identification (no actual authentication/authorization)

2. Master schedule covers one academic week (Monday–Friday), repeating for entire semester

3. **courses.hours_per_week** defines how many hours each course meets every week

4. Student eligibility is based on historical pass/fail records in the database; no need to mark current semester results

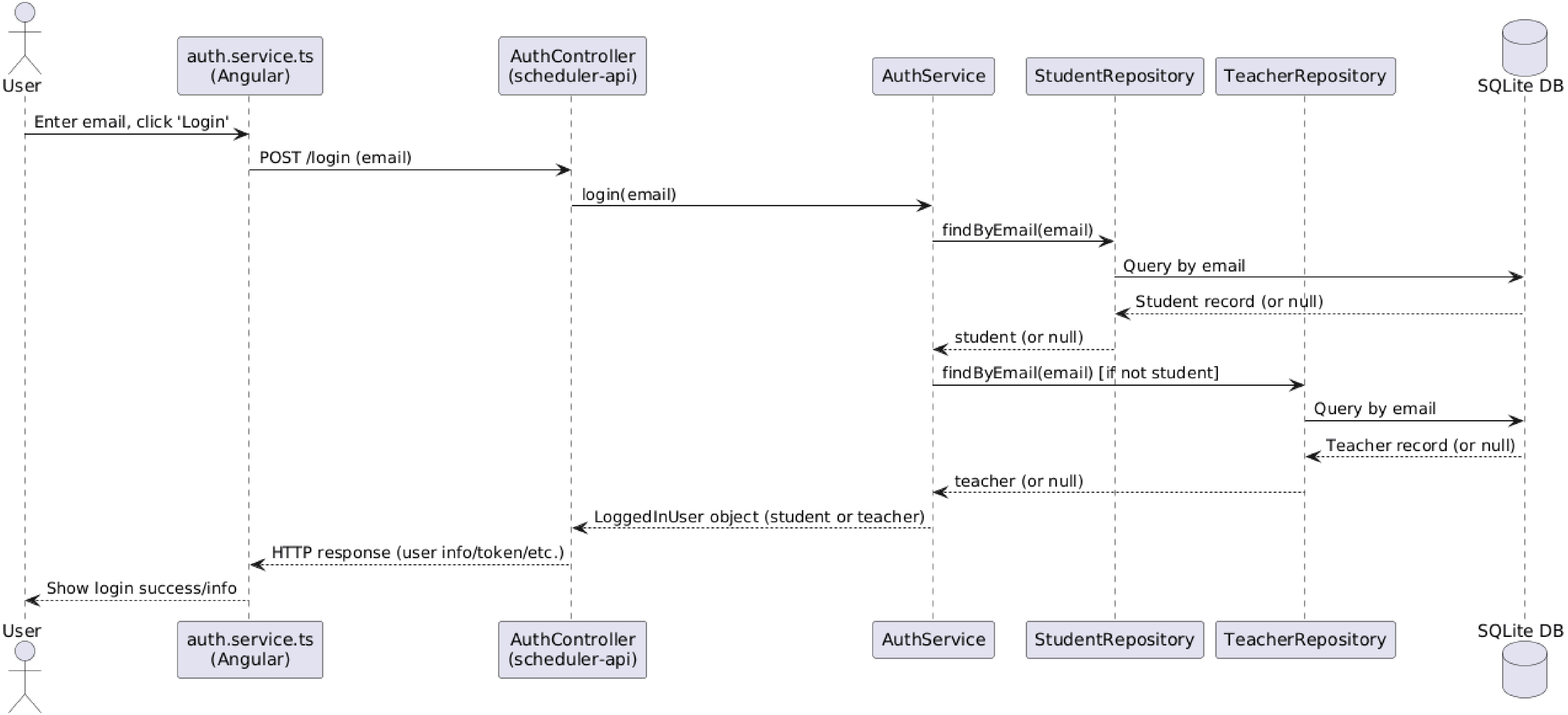
5. Room types (lab, classroom, etc.) are assigned as needed

6. Students see a filtered list of eligible courses up front—no trial-and-error selection

Challenge 1: Master Schedule Generator

Login

- Minimal login to identify the student or teach using email
- Quick login option
- Based on email address received, teacher / student portal is loaded
- Used to personalize the experience and filter data



User

auth.service.ts (Angular)

AuthController (scheduler-api)

AuthService

StudentRepository

TeacherRepository

SQLite DB

Enter email, click 'Login'

POST /login (email)

login(email)

findByEmail(email)

Query by email

Student record (or null)

student (or null)

findByEmail(email) [if not student]

Query by email

Teacher record (or null)

teacher (or null)

LoggedInUser object (student or teacher)

HTTP response (user info/token/etc.)

Show login success/info

Course Section Generation

The API generates course sections for all courses in the active semester (based on semester order).

When creating course sections for each course:

- The load is distributed equally among all teachers with the required specialization.
- The load is distributed equally among all rooms with the specialization.
- Day and time slots are used evenly (from Monday 9am to Friday 5pm).

The least-loaded teacher, room, and time slot are selected for each course section.
Course sections are distributed across the week and this schedule repeats throughout the semester.

Course Section Generation

Constraints Enforced

Room Capacity: Max 10 students per section.

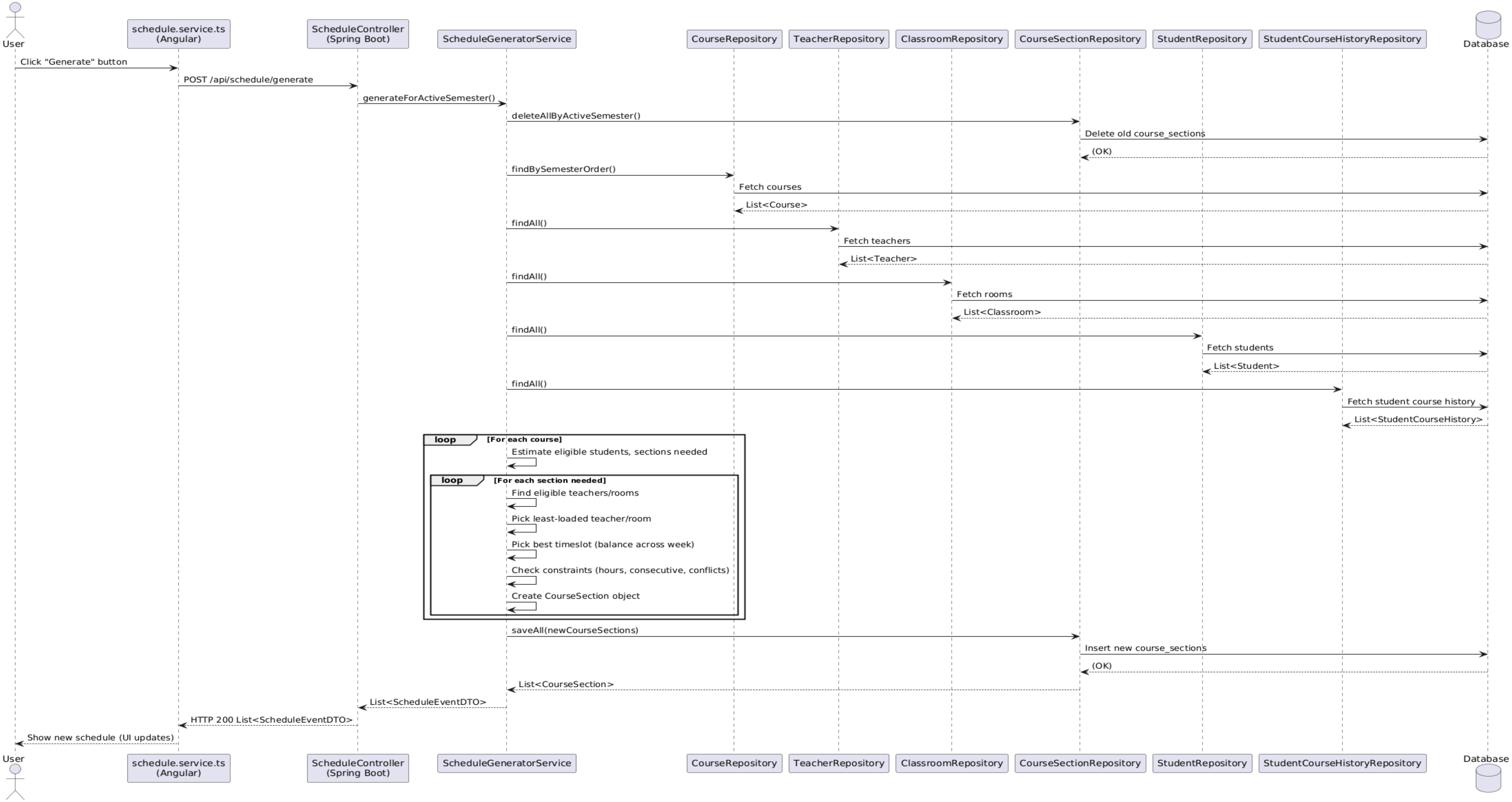
Teacher Limits: Max 4 teaching hours per day; max 2 consecutive hours.

Time Slots: School hours are 9AM–5PM; no classes during lunch (12–1PM); only on weekdays.

Specialization Matching: Teachers and rooms must match the course's requirements.

Prerequisites: Only students who have passed prerequisites (from historical data) are counted as eligible for a course.

No Double Booking: Teachers and rooms cannot be scheduled in two places at the same time.



User

schedule.service.ts (Angular)

ScheduleController (Spring Boot)

ScheduleGeneratorService

CourseRepository

TeacherRepository

ClassroomRepository

CourseSectionRepository

StudentRepository

StudentCourseHistoryRepository

Database

Click "Generate" button

POST /api/schedule/generate

generateForActiveSemester()

deleteAllByActiveSemester()

Delete old course_sections

(OK)

findBySemesterOrder()

Fetch courses

List<Course>

findAll()

Fetch teachers

List<Teacher>

findAll()

Fetch rooms

List<Classroom>

findAll()

Fetch students

List<Student>

findAll()

Fetch student course history

List<StudentCourseHistory>

loop [For each course]

Estimate eligible students, sections needed

loop [For each section needed]

Find eligible teachers/rooms

Pick least-loaded teacher/room

Pick best timeslot (balance across week)

Check constraints (hours, consecutive, conflicts)

Create CourseSection object

saveAll(newCourseSections)

Insert new course_sections

(OK)

List<CourseSection>

List<ScheduleEventDTO>

HTTP 200 List<ScheduleEventDTO>

Show new schedule (UI updates)

User

schedule.service.ts (Angular)

ScheduleController (Spring Boot)

ScheduleGeneratorService

CourseRepository

TeacherRepository

ClassroomRepository

CourseSectionRepository

StudentRepository

StudentCourseHistoryRepository

Database

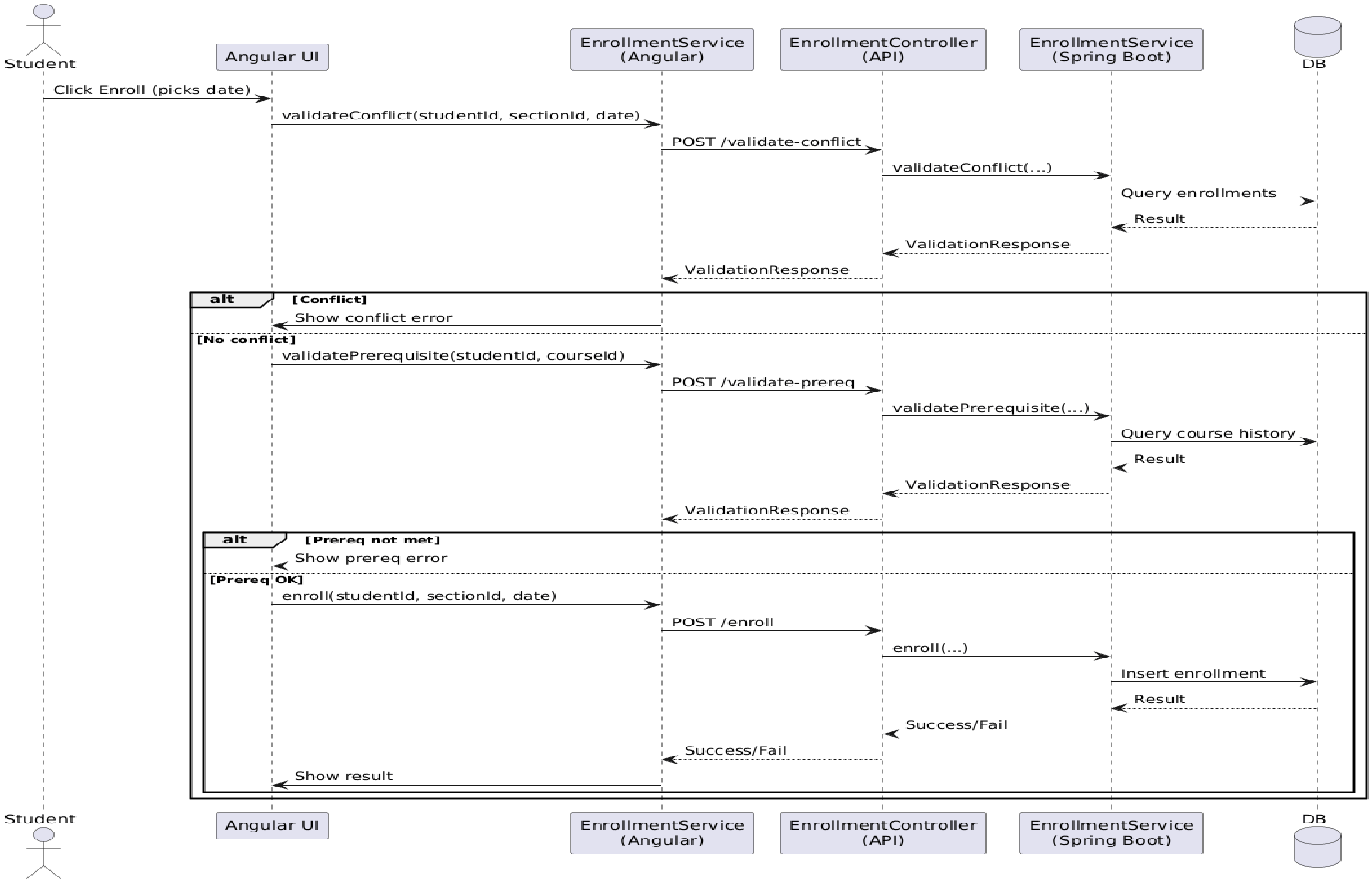
Utilization Charts

- The Charts are plotted in UI using char.js library
- Visualizes usage of:
    - Teachers Utilization (% of teachers utilized per week based on 20hrs)
    - Room Utilization (hours/week, occupation rate)
    - Time slots (which are busiest)
    - Other key metrics like most loaded / least loaded days and average utilization percentage

Challenge 2: Student Course Planning

Browse Available Courses

- Students see **only the courses they’re eligible for**:
  - Prerequisites satisfied
  - Not already passed or enrolled
  - Grade level appropriate
  - Section not full
- No “trial and error” — only eligible courses and dates are shown
- When enrolling, system checks for:
  - Time conflicts, Section capacity, Prerequisite satisfaction, Max 5 courses per semester


Student
Angular UI
EnrollmentService (Angular)
EnrollmentController (API)
EnrollmentService (Spring Boot)
DB
Click Enroll (picks date)
validateConflict(studentId, sectionId, date)
POST /validate-conflict
validateConflict(...)
Query enrollments
Result
ValidationResponse
ValidationResponse
alt
[Conflict]
Show conflict error
[No conflict]
validatePrerequisite(studentId, courseId)
POST /validate-prereq
validatePrerequisite(...)
Query course history
Result
ValidationResponse
ValidationResponse
alt
[Prereq not met]
Show prereq error
[Prereq OK]
enroll(studentId, sectionId, date)
POST /enroll
enroll(...)
Insert enrollment
Result
Success/Fail
Success/Fail
Show result
Student
Angular UI
EnrollmentService (Angular)
EnrollmentController (API)
EnrollmentService (Spring Boot)
DB

Semester Schedule

- The schedule calendar shows the course section for which the student enrolled for
- The course_sections are plotted based on the enrolled date

Student Progress

- Dashboard for tracking:
    - GPA (from past grades)
    - Credits earned vs. graduation requirement (30 credits over 4 years)
- Uses **historical records** for accurate progress calculation
- Shows past academic performance
- Courses opted for the current semester

TODO/Further Improvements

Optimize the scheduling algorithm for faster generation.
Industry-standard constraint-solving libraries such as OptaPlanner or Timefold can be used for more efficient and scalable scheduling.

Enhance the calendar view:

- Display 2-hour slots as a single course section card.
- Show student occupancy for each course section directly on the calendar.
- Limit the calendar view to only the actual semester start and end dates.

Enable student enrollment via the calendar view.
Students can select and enroll in available sections directly from the calendar interface.

Thank you