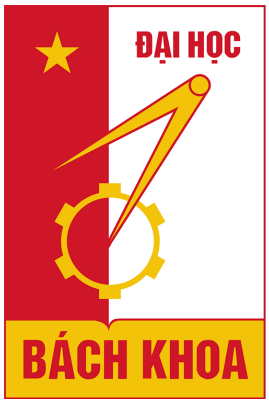
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG



Báo Cáo Bài Tập Lớn

Đề tài: Xây dựng ứng dụng Calendar với tính năng AI Scheduling

Học phần: Project 1

Mã học phần: IT3150

Giảng viên hướng dẫn: Nguyễn Thanh Hùng

Sinh viên thực hiện: Đỗ Ngọc Hoàng Hải 20230026

Hà Nội, Ngày 10 tháng 2 năm 2026

Mục lục

Lời nói đầu

Trong bối cảnh chuyển đổi số và nhịp sống ngày càng bận rộn, nhu cầu quản lý thời gian cá nhân một cách thông minh và hiệu quả trở nên cấp thiết hơn bao giờ hết. Đặc biệt với sinh viên, việc cân bằng giữa lịch học, deadline bài tập lớn, và các hoạt động cá nhân là một thách thức thường nhật. Mặc dù đã có nhiều ứng dụng lịch phổ biến như Google Calendar, hầu hết đều chỉ dừng lại ở mức *ghi nhận* sự kiện thiếu đi khả năng *chủ động đề xuất* cách phân bổ thời gian cho người dùng.
Báo cáo này trình bày quá trình xây dựng và phát triển ứng dụng **Calendar** một nền tảng quản lý lịch cá nhân full-stack tích hợp trí tuệ nhân tạo (AI), lấy cảm hứng từ Google Calendar nhưng bổ sung tính năng lên lịch tự động sử dụng **Google Gemini AI**. Hệ thống không chỉ hỗ trợ quản lý sự kiện và công việc mà còn có khả năng phân tích lịch trình hiện tại, tìm các khoảng trống phù hợp, và đề xuất thời gian học/làm việc tối ưu cho sinh viên.
Hệ thống được thiết kế dựa trên kiến trúc **Client–Server** hiện đại với hai nhóm chức năng chính:

- **Quản lý lịch trình**: Hỗ trợ đầy đủ CRUD cho sự kiện (với lặp lại, mã màu) và công việc (với mức độ ưu tiên, deadline, trạng thái). Cung cấp nhiều chế độ xem: ngày, tuần, tháng, năm.
- **AI Scheduling**: Tự động phân tích khoảng thời gian trống, xây dựng prompt tối ưu cho Gemini AI, đề xuất lịch học/làm việc dựa trên ưu tiên và deadline, kèm theo giải thích (Explainable AI) cho mỗi đề xuất.

Thông qua đề tài, em kỳ vọng người đọc có thể hiểu được cách xây dựng một ứng dụng web full-stack hoàn chỉnh với kiến trúc rõ ràng, đồng thời thấy được tiềm năng ứng dụng AI sinh tạo (Generative AI) vào bài toán quản lý thời gian thực tế.

Mã nguồn: https://github.com/HaiDNH20230026/Project1

Chương 1

Giới thiệu đề tài

1.1 Đặt vấn đề

Sinh viên đại học thường phải đối mặt với khối lượng công việc lớn: lịch học trên lớp, deadline bài tập, ôn thi, dự án nhóm, và các hoạt động ngoại khóa. Việc quản lý thời gian thủ công dễ dẫn đến:

- Quên deadline hoặc chồng chéo lịch trình
- Không biết phân bổ thời gian hợp lý cho các task dài hạn (ví dụ: bài tập lớn cần 10–20 giờ)
- Trì hoãn do không có kế hoạch cụ thể

Các ứng dụng lịch hiện tại chỉ đóng vai trò *ghi nhận thụ động* người dùng phải tự quyết định khi nào làm gì. Đề tài này đề xuất giải pháp bổ sung thành phần **AI chủ động lên lịch**, giúp tự động tìm thời gian trống phù hợp và đề xuất phiên làm việc dựa trên mức độ ưu tiên, deadline, và thói quen sinh hoạt.

1.2 Mục tiêu đề tài

1. Xây dựng ứng dụng web quản lý lịch cá nhân **full-stack** hoàn chỉnh với frontend React và backend Spring Boot.
2. Tích hợp **Google Gemini AI** để tự động đề xuất lịch trình cho các task có deadline, kèm giải thích cho mỗi đề xuất.
3. Triển khai hệ thống xác thực an toàn với **JWT + OAuth2 (Google Sign-in)**.

1.3 Phạm vi đề tài

Trong phạm vi	**Ngoài phạm vi**
Quản lý sự kiện (CRUD, recurring, color)	Chia sẻ lịch (shared calendar)
Quản lý task (priority, scale, status)	Push notification
AI đề xuất lịch cho task DEADLINE	Đồng bộ với Google Calendar API
Xác thực JWT + Google OAuth2	Quản lý nhóm/tổ chức
Deploy AWS (production)	Mobile app (chỉ có web responsive)

Bảng 1.1: Phạm vi đề tài

Chương 2

Công nghệ và kiến trúc hệ thống

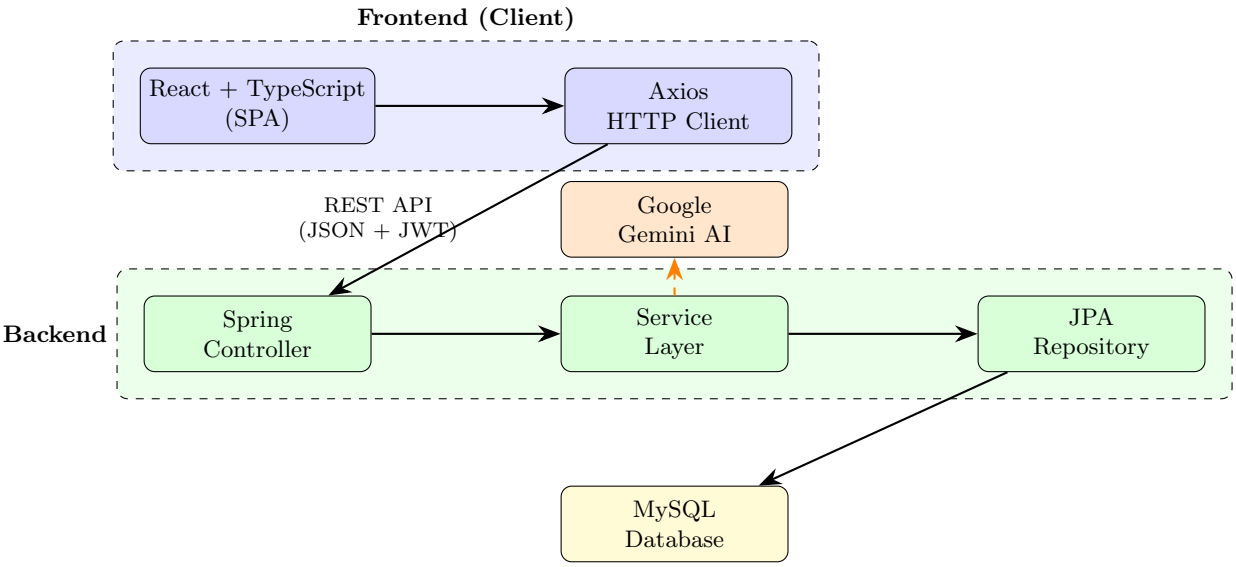
2.1 Công nghệ sử dụng

Tầng	**Công nghệ**	**Vai trò**
Frontend	React 19 + TypeScript	UI framework
	Material UI (MUI) 7	Component library
	Axios	HTTP client
	React Router DOM 7	Client-side routing
Backend	Java 17 + Spring Boot 3.4	Application framework
	Spring Security + JWT	Authentication & authorization
	Spring Data JPA	ORM / database access
	Google Gemini SDK 1.0	AI scheduling engine
	Gradle	Build tool
Database	MySQL	Relational database

Bảng 2.1: Bảng tổng hợp công nghệ

2.2 Kiến trúc hệ thống

Hệ thống theo kiến trúc **Client–Server** với 3 tầng rõ ràng:



2.3 Kiến trúc backend (Package Structure)

Backend tuân theo mô hình phân tầng **Controller → Service → Repository → Entity** chuẩn Spring Boot:

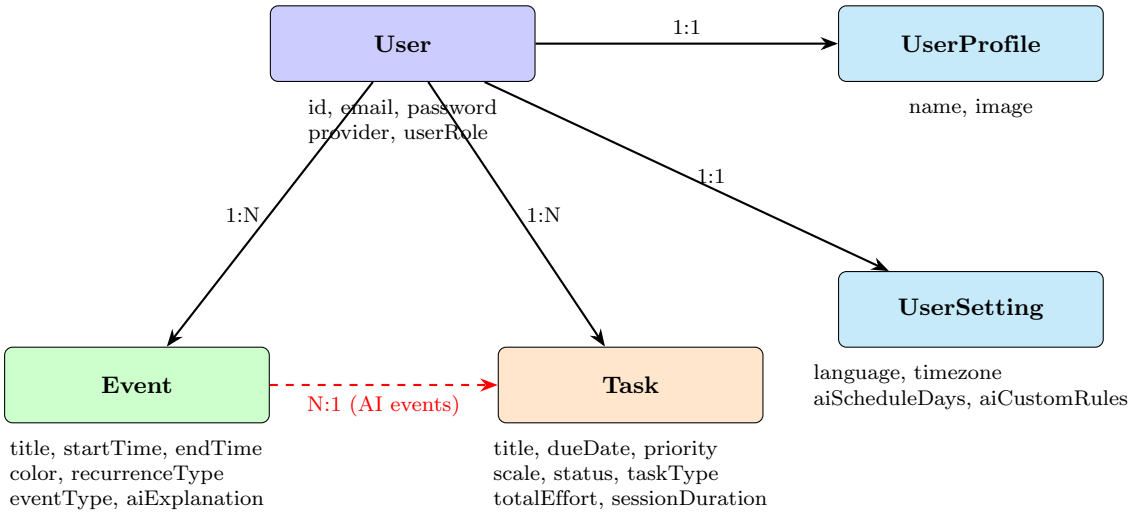
Package	Chức năng
`com.leun.auth`	Xác thực (JWT, OAuth2, Security config)
`com.leun.user`	Quản lý User, UserProfile, UserSetting
`com.leun.event`	Quản lý sự kiện (Event entity, CRUD)
`com.leun.task`	Quản lý công việc (Task, priority, status)
`com.leun.calendar`	API truy vấn lịch theo ngày/tuần/tháng/năm
`com.leun.ai`	AI Scheduling (Gemini integration, heuristics)
`com.leun.exception`	Xử lý lỗi toàn cục
`com.leun.health`	Health check endpoint

Bảng 2.2: Cấu trúc package backend

Chương 3

Thiết kế hệ thống

3.1 Mô hình dữ liệu (Entity Relationship)



Quan hệ đặc biệt: **Event → Task** (qua `sourceTask`) — mỗi sự kiện do AI tạo sẽ liên kết ngược về task gốc, cho phép theo dõi tiến độ sessions.

3.2 REST API Design

Hệ thống cung cấp RESTful API với các nhóm endpoint chính:

Nhóm	Method	Endpoint	Mô tả
Auth	POST	/v1/auth/login	Đăng nhập email/password
	POST	/v1/auth/google/login	Đăng nhập Google OAuth2
User	POST	/v1/user	Đăng ký tài khoản
	GET	/v1/user/profile	Lấy thông tin profile
Event Calendar	GET/POST	/v1/event	Lấy/Tạo sự kiện
	PUT/DELETE	/v1/event/{id}	Sửa/Xóa sự kiện
	GET	/v1/calendar/{unit}/{y}/{m}/{d}	Lấy lịch theo view
Task	POST	/v1/task	Tạo task
	PUT	/v1/task/{id}	Cập nhật task
	PUT	/v1/task/{id}/completion	Toggle hoàn thành
AI	POST	/v1/ai/schedule/propose/{taskId}	AI đề xuất lịch
	POST	/v1/ai/schedule/accept-all	Chấp nhận tất cả đề xuất
	POST	/v1/ai/schedule/accept	Chấp nhận 1 đề xuất

Bảng 3.1: Các endpoint REST API chính

3.3 Luồng xác thực (Authentication Flow)

1. Người dùng đăng nhập bằng **email/password** hoặc **Google OAuth2**.
2. Backend xác minh credentials, tạo **JWT token** (sử dụng thư viện JJWT, mã hoá HS256).
3. Frontend lưu JWT vào localStorage, gắn vào header `Authorization: Bearer <token>` cho mọi request qua Axios interceptor.
4. Backend có `JwtAuthenticationFilter` chạy trước mỗi request, giải mã và xác thực token.
5. Hệ thống hoàn toàn **stateless** không lưu session phía server.

Chương 4

Tính năng AI Scheduling

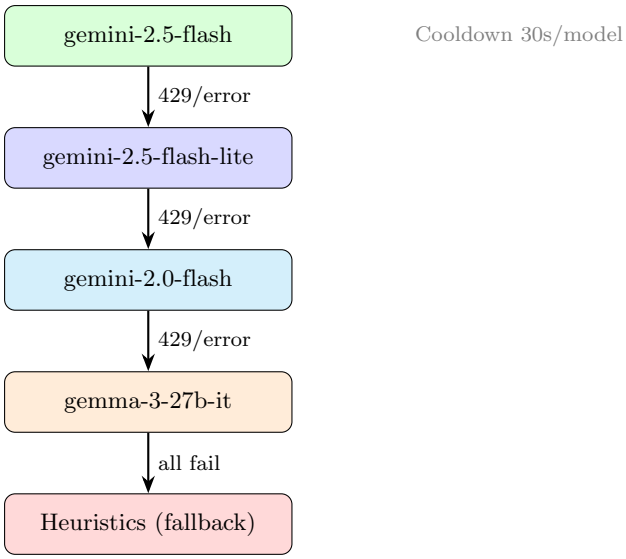
Đây là tính năng cốt lõi và khác biệt của đề tài, sử dụng **Google Gemini AI** (Generative AI) để tự động đề xuất lịch học/làm việc cho sinh viên.

4.1 Luồng hoạt động

1. Người dùng tạo task loại **DEADLINE** với các thuộc tính: tên, mô tả, deadline, mức ưu tiên (HIGH/MEDIUM/LOW), quy mô (QUICK/REGULAR/PROJECT), tổng thời gian cần thiết, thời lượng mỗi session.
2. Nhấn nút **"AI Schedule"** trên giao diện.
3. Backend thực hiện:
    a. Tải cài đặt người dùng (số ngày nhìn trước, custom rules).
    b. Truy vấn tất cả events hiện có để tìm **khoảng thời gian trống** (8:00–23:00).
    c. Tính toán số sessions cần đề xuất dựa trên deadline và effort còn lại.
    d. Xây dựng **prompt tối ưu** gửi cho Gemini AI, bao gồm: thông tin task, danh sách slots trống (tối đa 25 slots để tiết kiệm token), custom rules.
    e. Gemini AI trả về các đề xuất kèm lý do.
    f. Nếu Gemini fail → **fallback sang heuristics** (scoring-based).
4. Người dùng xem đề xuất (thời gian, điểm phù hợp, giải thích AI) và chọn chấp nhận.
5. Hệ thống tạo `AI_GENERATED` events liên kết với task gốc.

4.2 Multi-model Fallback

Để đảm bảo tính sẵn sàng cao, `GeminiService` triển khai cơ chế fallback qua 4 models:



Mỗi model bị rate limit (HTTP 429) sẽ được đánh dấu cooldown 30 giây, hệ thống tự động chuyển sang model tiếp theo. Nếu tất cả 4 models đều fail, fallback về thuật toán **heuristics** hoàn toàn local (không cần API).

4.3 Heuristic Scoring Algorithm

Khi Gemini AI không khả dụng, hệ thống sử dụng thuật toán scoring để đánh giá và chọn slot tốt nhất:

Khung giờ	Điểm	Lý do
8:00 – 11:30	+30	Buổi sáng — tập trung cao nhất
11:30 – 13:30	−10	Giờ ăn trưa — hiệu suất thấp
13:30 – 17:00	+20	Buổi chiều — phù hợp làm việc
17:00 – 19:00	−5	Giờ ăn tối
19:00 – 21:00	+10	Tối sớm — hiệu quả tự học
21:00 – 23:00	−15	Tối muộn — ảnh hưởng sức khỏe
Slot $\geq$ 90 phút	+10	Ưu tiên phiên dài

Bảng 4.1: Bảng scoring cho heuristic scheduling

Quy tắc bổ sung:

- Mỗi buổi (sáng/chiều/tối) chỉ tối đa **1 session**, trừ khi deadline $\leq$ 3 ngày (urgent).
- Duration linh động: chia đều effort còn lại cho số sessions, làm tròn lên bội 15 phút, trong khoảng [30, 150] phút.
- Slots được nhóm theo ngày + buổi, chọn slot dài nhất và điểm cao nhất trong mỗi buổi.

4.4 Explainable AI

Mỗi sự kiện do AI tạo đều lưu trường `aiExplanation` — giải thích bằng tiếng Việt tại sao AI chọn khung giờ đó. Ví dụ:

> *"Session 2: Buổi sáng — thời điểm tập trung cao nhất. Thời lượng 90 phút."*

Điều này giúp người dùng hiểu quyết định của AI và tin tưởng hơn vào đề xuất.

Chương 5

Kết quả đạt được

5.1 Tổng quan chức năng

STT	Chức năng	Trạng thái
1	Đăng ký / Đăng nhập (Email + Google OAuth2)	✓
2	Quản lý sự kiện (tạo, sửa, xóa, xem chi tiết)	✓
3	Sự kiện lặp lại (daily, weekly, monthly, yearly, weekdays)	✓
4	Mã màu sự kiện (10 màu)	✓
5	Quản lý task (priority, scale, status, deadline)	✓
6	Chế độ xem: Ngày, Tuần, Tháng, Năm	✓
7	AI đề xuất lịch học (Gemini + fallback heuristics)	✓
8	Explainable AI (giải thích đề xuất bằng tiếng Việt)	✓
9	Multi-model fallback (4 models + cooldown)	✓
10	Cài đặt cá nhân (ngôn ngữ, timezone, AI rules)	✓
11	Dark/Light theme	✓
12	Mini calendar sidebar	✓

Bảng 5.1: Bảng tổng hợp chức năng đã hoàn thành

5.2 Thống kê mã nguồn

Thành phần	Số file	Công nghệ chính
Backend (Java)	∼40 files	Spring Boot 3.4, JPA, JWT
Frontend (React)	∼50 files	React 19, TypeScript, MUI
CSS styles	∼20 files	Custom CSS
Config/Build	∼10 files	Gradle, Webpack, YAML
Tổng	**∼120 files**	

Bảng 5.2: Thống kê mã nguồn

5.3 Đánh giá

5.3.1 Ưu điểm

- Kiến trúc rõ ràng, phân tách trách nhiệm tốt (Controller–Service–Repository).

- Tính năng AI scheduling là điểm khác biệt so với các ứng dụng lịch thông thường.
- Cơ chế fallback đảm bảo hệ thống luôn hoạt động ngay cả khi API AI gặp sự cố.
- Explainable AI giúp người dùng hiểu và tin tưởng đề xuất.

5.3.2 Hạn chế

- Chưa hỗ trợ chia sẻ lịch (shared calendar) và push notification.
- AI phụ thuộc vào Google Gemini API (cần API key, giới hạn quota).
- Chưa có unit test tự động cho backend.

Chương 6

Kết luận và hướng phát triển

6.1 Kết luận

Đề tài đã hoàn thành việc xây dựng ứng dụng **Calendar** — một nền tảng quản lý lịch cá nhân full-stack với tính năng AI Scheduling sử dụng Google Gemini. Cụ thể, các mục tiêu đã đạt được bao gồm:

- **Phương pháp**: Áp dụng kiến trúc Client–Server hiện đại (React + Spring Boot), thiết kế RESTful API, sử dụng JWT cho xác thực stateless, và tích hợp Generative AI (Google Gemini) với cơ chế multi-model fallback.
- **Kết quả**: Sản phẩm hoạt động đầy đủ với 13 chức năng chính.
- **Đóng góp**: Chứng minh tính khả thi của việc ứng dụng AI sinh tạo vào bài toán quản lý thời gian cá nhân, đặc biệt phù hợp với đối tượng sinh viên.

6.2 Hướng phát triển

1. **Shared Calendar**: Cho phép chia sẻ lịch giữa các thành viên trong nhóm.
2. **Push Notification**: Nhắc nhở trước sự kiện qua email hoặc browser notification.
3. **Đồng bộ Google Calendar**: Import/export sự kiện với Google Calendar API.
4. **Mobile App**: Phát triển ứng dụng mobile bằng React Native.
5. **AI nâng cao**: Tích hợp mô hình học từ hành vi người dùng (preference learning) để đề xuất chính xác hơn theo thời gian.
6. **Testing**: Bổ sung unit test và integration test cho backend.