

StudyPilot: PRODUCT README2

StudyPilot – Full Tech Specification & System Design (v2.0)

Project Overview

StudyPilot is a university productivity platform designed for students. It enables easy uploading and access of departmental syllabi categorized by Faculty → Department → Semester, and provides intelligent study scheduling with reminders and progress tracking.

Tech Stack Overview

Layer	Tech / Tool
Frontend	Next.js, TypeScript, Tailwind CSS, Axios
Backend	Java Spring Boot (Maven), Apache Commons FileUpload, Apache PDFBox, Quartz Scheduler
Auth	Supabase (email/password)
Storage	Supabase (PDF and metadata)
Database	Supabase/PostgreSQL
Deployment	Render (Java), Vercel (Next.js)
Sync	Google Calendar API (optional via OAuth2)

User Type

- **Student:** Only role in the system.
- All users can upload syllabi (no admin layer required).

Key Features (MVP)

1. Syllabus Upload & Storage

- Upload PDF only.
- Parse metadata (course title, topics, etc.) using Apache PDFBox.
- Categorize under: Faculty → Department → Semester.
- Store metadata + file in Supabase.
- Searchable/filterable repository.

2. Study Plan Generation

- Auto-distributes topics evenly across days until exam.
- Accepts:
 - Busy hours (study block preferences)
 - Study start & end date (exam or revision deadline)
 - Topic/course prioritization (optional)
- Output:
 - Daily plan with sessions
 - Suggested start/end times per session

3. Reminders & Calendar Sync

- Daily session reminders
- Optional sync with Google Calendar via OAuth2
- Daily email reminders (optional)

4. Progress Tracker

- Completion checkbox for each topic/session
- Engagement tracker (e.g., did user view/open session)
- Progress bars & stats on dashboard

5. Dashboard

- Upcoming sessions

- Progress overview
- Reminders / missed tasks
- Quick view of syllabus & study plan

6. Authentication

- **Supabase Auth**
 - Email & password login
 - Signup fields: Full name, Email, Department, Password, Confirm Password
 - Session stored securely with cookies or local storage

Study Plan Algorithm (Pseudocode / Java Concept)

```
// User input
List<Topic> allTopics = syllabus.getTopics(); // parsed from PDF
Map<DayOfWeek, TimeRange> busyHours = user.getBusyHours(); // excluded
LocalDate start = LocalDate.now();
LocalDate end = examDate;

int totalDays = (int) ChronoUnit.DAYS.between(start, end);
int availableDays = countAvailableDays(busyHours, start, end);
int topicsPerDay = (int) Math.ceil((double) allTopics.size() / availableDays);

// Assign topics avoiding busy hours
for (LocalDate date : eachDayBetween(start, end)) {
    if (isBusy(date, busyHours)) continue;
    assignNextTopics(date, topicsPerDay);
}
```

 You can refine this with weights if prioritization is specified.

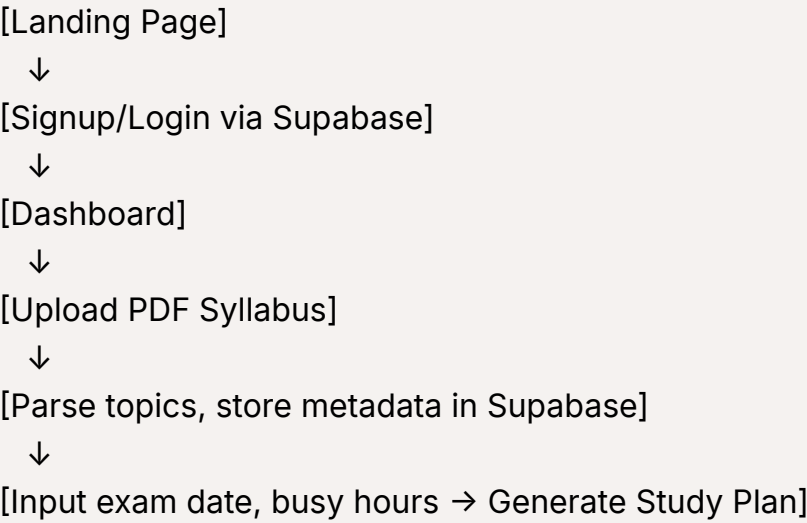
Backend Endpoints (Spring Boot)

Method	Route	Description
POST	/api/syllabus/upload	Upload PDF & metadata
GET	/api/syllabus	List all syllabi (filters: faculty, dept, semester)
POST	/api/study-plan/generate	Create study plan
GET	/api/study-plan	Retrieve user study plan
PUT	/api/progress/{sessionId}	Mark session/topic as completed
POST	/api/reminders/sync	Sync study sessions to external calendar
POST	/api/auth/signup	Supabase handles auth flow

Frontend Route Map (Next.js)

Route	Page
/	Landing page
/login / /signup	Auth pages
/dashboard	Main dashboard (sessions + stats)
/upload	PDF syllabus upload
/planner	Study plan view & edit
/progress	Visual progress tracking

System Flow (Simplified)





[Study Plan Created → Reminders Set → Calendar Synced (optional)]



[Progress Tracked on Dashboard]

UI Guidelines

- **Theme:** White and Blue (clean, university-like aesthetic)
- **Frameworks:** Tailwind CSS + ShadCN UI for clean components
- **Calendar & Progress:**
 - `FullCalendar` for scheduling
 - `Chart.js` or custom bars for progress
- **Transitions:** Use `Framer Motion` for a fluid UX