Software Engineering Project Plan

Project title: Study planner

Problem Identification

Students struggle to balance heavy course loads, shifting class schedules and extracurricular commitments. Current tools are often fragmented or not easy enough that they themselves add an extra task on the students end. Students often find themselves using different apps for different responsibilities. Calendars as reminders, to-do list as a tracking service and so on depending on the needs of the student. This lack of integration leads to:

- Missed Deadlines**:** Without a unified view of exam dates and assignment due dates, high-priority tasks often fall through the cracks.
- Inefficient Study Habits**:** Students often default to cramming because they lack a realistic daily breakdown of the work required to meet long-term goals .
- Hectic planning: Students spend more time organizing their to-do lists than actually studying. The process is too slow and confusing.

Intended Audience/Users :

The main target audience is university students as we build this but there is always a chance to expand this to all forms of students, ranging from

Students primarily need a simple and clear way to plan their study sessions , keeping track of the deadlines and assignments, and also optimize their progress without spending too much time.

Main Features/Components:

1.Login (email) / Google authentication

2.User profile creation

3.Task creation

4.Reminder setting

5.Deadline setting

6.Mark task status & priority

7.Date importing (class schedules) from personal calendar

8.Joint task creation with other members (optional)

9.AI study plan(based on tasks,deadlines & priority)(optional)

Project Objectives

The objectives of this Project are :

- **Develop an Integrated Platform:** Create a single application that allows students to manage tasks, deadlines, reminders, and study plans efficiently, reducing the need for multiple apps.
- **Enhance Productivity:** Provide a clear structure for students' daily, weekly, and long-term study goals to minimize inefficient study habits.
- **User-Friendly Design:** Ensure an user friendly interface using React and Tailwind CSS, which will be easier to use by students making the app accessible for all users.
- **Optional AI Integration:** Include an AI-based study plan feature that can prioritize tasks, deadlines, and student preferences, enhancing personalized productivity.
- **Seamless Functionality:** Maintain smooth integration between frontend, backend, and database components, ensuring reliable performance and scalability for future features.

Scope and Deliverables

Scope:
The project focuses on building a core task and study management system for university students, with optional AI-enhanced features. The system will provide:

- Task creation, reminders, and deadline management
- User authentication via email and Google
- Calendar integration for importing class schedules
- Optional collaborative task management
- Optional AI-generated personalized study plans

Deliverables:

- **Frontend Development:** Fully functional interface built with React and Tailwind CSS, covering login, profile, task creation, and dashboard views.
- **Backend Development:** RESTful API and business logic implemented using Java (Spring Boot) for task handling, authentication, and data management.
- **Database Setup:** Scalable database model supporting tasks, deadlines, user profiles, and optional AI study plans.
- **Integration:** Smooth coordination between frontend, backend, and database to ensure full functionality.

- **Testing & Quality Assurance:** Unit tests (JUnit 5 for backend, Jest for frontend), integrity tests, and user acceptance testing.
- **Documentation:** JavaDoc for backend, and a proper in-app user manual covering all functionalities.
- **Deployment:** Final deployment-ready application, packaged and ready for submission.

Project Timeline (Trello)

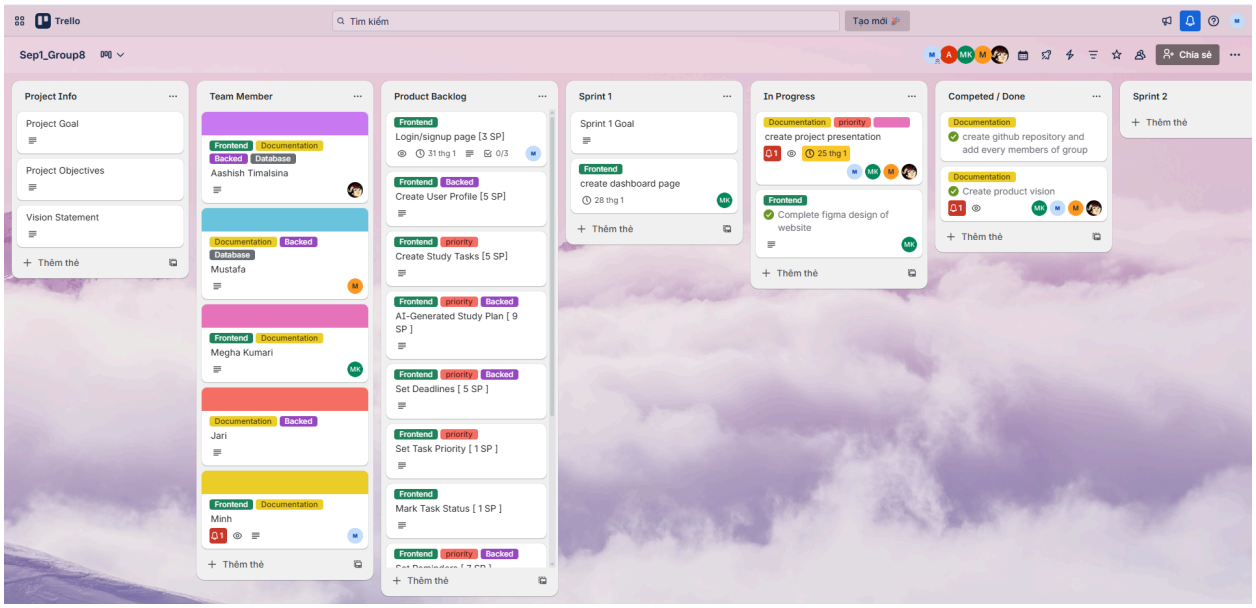
Sprint 1 – Planning & Design
- Define project goals, scope, and requirements
- Form project groups and assign roles
- Create user stories and initial backlog
- Design UI/UX mockups using Figma

Sprint 2 – Backend Development
- Set up backend project structure
- Design and implement database schema
- Develop core business logic
- Implement required API(s)
- Handle validation and error handling

Sprint 3 – Frontend Development
- Set up frontend project structure
- Implement UI components based on Figma designs
- Connect frontend to backend APIs
- Implement navigation, forms, and state management

Sprint 4 – Integration, Testing & Deployment
- Integrate frontend and backend
- Fix bugs and edge cases
- Perform testing and refactoring
- Deploy the application
- Final review and documentation

*Sprint 2 and 3 can also be done simultaneously since the teams for them are separate.



Resource Allocation

This project will be developed by a small multidisciplinary team, with responsibilities divided based on technical expertise to ensure efficient progress and clear ownership.

Team Members and Roles

Aashish Timalsina

- Oversees the overall project development
- Contributes to both frontend and backend development
- Responsible for the database design and management
- Participates in code reviews and ensures integration between system components

Megha Kumari

- Figma Design (UI/UX)
- Frontend Developer
- Focuses on designing intuitive and user-friendly interface using React and Tailwind CSS

Mhin

- Frontend Developer
- Works closely with Megha on layout design, usability and responsiveness

Jari

- Backend developer
- Responsible for implementing core backend logic using Java, including task handling and authentication features

Mustafa

- Backend developer
- Supports backend feature development and API implementation
- Participates in code review and quality assurance

This division of roles helps reduce overlap, improves productivity and ensures that all parts of the system are well maintained.

Software, Hardware and Tools

To implement the project successfully, the following software, hardware and tools will be used:

Software and Development Tools

- Frontend framework: React, Tailwind Css
- Backend language: Java (Spring Boot)
- Database: MariaDB
- DevOPs: Docker, Kubernet
- Version control: Git and GitHub
- Authentication: Email Login and Google authentication
- AI services (optional): OpenAI API, Hugging Face or Gemini
- Development environments: Visual Studio Code, Intellij IDEA
- Project management and communication: WhatApp's, In Person and Trello

Hardware

- Personal computers or laptops capable of running development
- Internet connection for cloud services, API usage and collaboration

---

External Resources or Support

- AI APIs: External AI services may be used to generate personalized study plans based on tasks and deadlines
- Google Services: Used for Google authentication and possible calendar integration
- Documentation and Online Resources: Official documentation for React, Java, MariaDB and AI APIs
- University Support: Guidance and feedback from instructors and peers during testing and evaluation phases.

Risk Management

This section identifies potential risks that could affect the success of the project, along with their likelihood, impact and mitigation strategies.

Risk 1: AI Integration Complexity

- Description: Implementing AI-based study planning may be technically challenging or time-consuming.
- Likelihood: Medium
- Impact: Medium
- Mitigation Strategies: Treat AI features as optional and prioritize core functionality first. AI integration will be added only if time and stability allow.

Risk 2: Frontend and Backend Integration Issues

- Description: Miscommunication or mismatched API contracts between frontend and backend may cause delays.
- Likelihood: Medium
- Impact: High
- Mitigation Strategies: Define API specification early and maintain regular coordination between frontend and backend developers.

Risk 3: Limited Development Time

- Description: The project timeline may not allow all planned features to be completed.
- Likelihood: Low
- Impact: low
- Mitigation Strategies: Prioritize essential features such as task creation, deadlines, reminders and authentication before optional features.

Risk 4: Database Design Issues

- Description: An inflexible database structure may limit future feature expansion.
- Likelihood: Low
- Impact: Medium
- Mitigation Strategies: Design scalable database schemas early and review them before full implementation

---

Testing and Quality Assurance

Unit testing will be the first form of testing created, this is to make sure key aspects work as they're intended. First making sure that individual components work before starting to test their functionality together. This will ensure a strong foundation that'll be easy to build upon.

Integrity testing will be used, because there will be main components composed of multiple subcomponents. It is crucial to test these many part components thoroughly because they will be the core of the application and core way of applications functionality.
User acceptance testing will be performed near the end of finishing the project, to ensure the wanted outcome and smooth user experience. Possible code adjustments might present themselves during this phase and will be taken into count to improve the overall experience.

Aim is to have the foundational tests to be completed with high accuracy, point is to make foundation strong and not to lean on exploratory testing. Testing will be marked successful if the core functionalities are passed.

JUnit 5 will be used to create the tests on the Java side of the project. Jest will be used to create the tests on the react side of the project. In both of the cases core functionalities are the main priorities to be passed and new tests are expected on the user acceptance phase.

Documentation and Reporting

Documentation will be done via IntelliJ IDEAs feature JavaDoc, on the Java side. TypeDoc will be used to handle the documentation on the React side of the project. These two documentations will be published together under one site.

Only short and concise user manuals will be provided, there won't be a need for a comprehensive user manual, due to the applications straightforward use case. The user manual will be covered inside the application with a simple text box, detailing through the correct use of intended functionalities of the application.

Reports will be shared weekly between team members. Occasional meetings/contact between members is expected between weekly reports, this is to tackle challenging tasks that interfere with the progression of the project. In both of these cases it will be done through Whatsapp or Discord. The frontend and the backend teams within their sub-teams will of course be sharing more resources and time.