Project Proposal: Student Assignment & Deadline Tracker

1. Project Title

Student Assignment & Deadline Tracker

2. Problem Statement

Students frequently miss assignment deadlines or mismanage time across multiple courses and group projects. This leads to last-minute work, lower grades, and stress. Existing tools are often generic and lack course-aware or collaborative features tailored for students.

3. Proposed Solution

A web application that helps students create and manage assignments, track progress, collaborate on group projects, visualize deadlines on a calendar, and receive intelligent, AI-driven reminders and start-date suggestions.

4. Objectives

- Provide an easy-to-use interface for adding and organizing assignments by course and priority.
- Offer collaborative boards for group projects with assignment ownership and comments.
- Display deadlines on a calendar and allow drag-and-drop rescheduling.
- Send smart reminders based on urgency, difficulty, and student workload.
- Ensure secure authentication and role-based access for students and group members.

5. Scope

In scope (MVP):

- User registration and authentication (JWT / Firebase Auth)
- CRUD for tasks/assignments
- Task status and progress indicator
- Group project boards with member assignment
- Calendar view and Google Calendar sync (optional)

- Basic AI suggestion engine for recommended start dates
- Notifications via in-app alerts and optional email reminders

Out of scope (for later iterations):

- Full mobile apps (only responsive web UI)
- Advanced analytics (beyond basic productivity score)
- Multi-institution integrations

6. Features / Modules

1. **Authentication** — Signup/Login, JWT or Firebase Auth, password reset.
2. **Task Creation & Management** — Add title, description, subject, deadline, priority, attachments.
3. **Progress Tracking** — Status states, progress bars, productivity score, daily/weekly summaries.
4. **Group Project Board** — Create groups, assign members, assign tasks, comments, file sharing.
5. **Calendar Integration** — Monthly/weekly views, drag-and-drop rescheduling, optional Google Calendar sync.
6. **AI Reminder Engine** — Suggest optimal start dates and reminder schedule using simple heuristics or lightweight ML trained on user history.
7. **Notifications** — In-app notifications and configurable email reminders.

7. Non-functional Requirements

- **Security:** Secure authentication, authorization, input validation.
- **Performance:** Reasonable response times for frontend operations; DB indexing for task queries.
- **Usability:** Clean, minimal UI suitable for students; responsive layout.
- **Maintainability:** Modular code, clear documentation, use of Git for version control.

8. High-level Architecture

- **Frontend:** React (single-page application) — components for Dashboard, Task Editor, Calendar, Group Board.
- **Backend:** Node.js + Express REST API — routes for auth, tasks, groups, notifications.
- **Database:** MongoDB (preferred for flexible task documents) or MySQL if relational schema preferred.
- **Auth:** JWT-based sessions or Firebase Auth if using managed auth.
- **AI Reminder:** Backend microservice or module that computes suggested start dates and sends reminders.

9. Technology Stack (confirmed by team)

- Frontend: **React**
- Backend: **Node.js + Express**
- Database: **MongoDB** (or **MySQL** if the team prefers relational)
- Authentication: **JWT** (or **Firebase Auth** as an alternative)
- Version Control: **Git / GitHub**
- Deployment: **Heroku / Render / Vercel (frontend)**

10. Development Plan & Timeline (12 weeks mapped to course timeline)

- **Week 1–2 (Proposal & Requirements):** Finalize proposal, gather detailed requirements, create user stories and use cases.
- **Week 3–4 (Design):** High-level design, system architecture diagrams, database schema, UI wireframes.
- **Week 5–6 (Implementation — Part 1):** Auth, task CRUD, basic frontend components.
- **Week 7 (Refinement):** Requirements refinement, change log, integrate feedback.
- **Week 8–9 (Implementation — Part 2):** Group board, calendar view, AI suggestion engine, notifications.
- **Week 10 (Testing):** System testing, bug fixes, user acceptance testing.

- **Week 11–12 (Finalization):** Prepare final report (5–10 pages), user manual, and presentation slides.

11. Deliverables

- Project Proposal (Week 2)
- System Requirements Specification (Week 4)
- High-Level Design Document (Week 6)
- Implementation Code (Week 8)
- Final Project Report (Week 11)
- Presentation Slides (Week 12)

12. Risk & Mitigation

- **Risk:** Scope creep. **Mitigation:** Define MVP clearly and keep feature additions to a backlog.
- **Risk:** Team coordination issues. **Mitigation:** Weekly standups, use GitHub issues and project board.
- **Risk:** Tight timeline for AI component. **Mitigation:** Start with heuristic-based reminders; replace with ML if time permits.

13. Success Criteria

- All core MVP features implemented and demonstrable.
- Clean UI and working calendar integration.
- Basic AI reminders functioning and reasonable for sample users.
- Well-documented code and final report meeting course requirements.