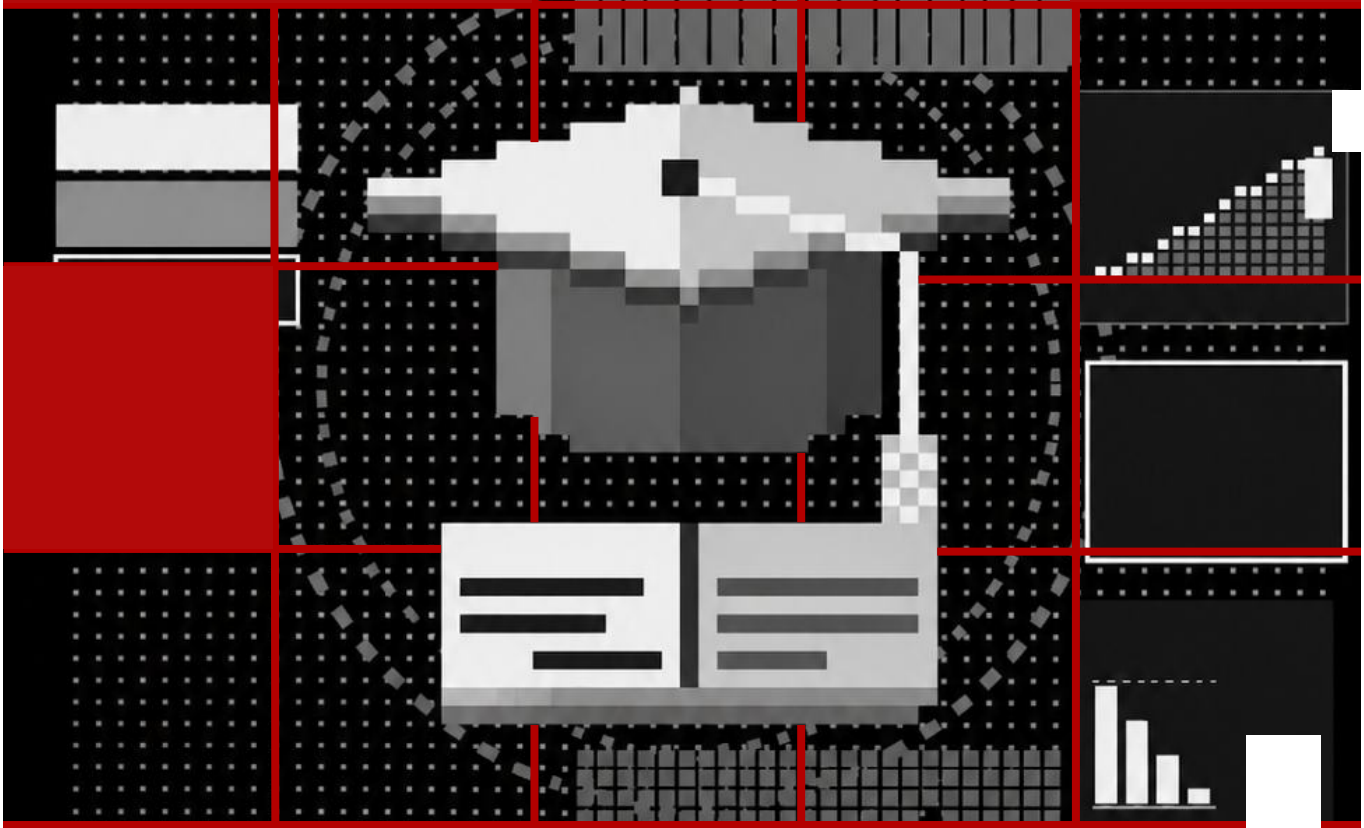
INTER IIT
TECH MEET 15.0



MID PREP

Adobe

Personalised Learning Buddy for Students

ABOUT ADOBE

Adobe builds software that people use to create, communicate and learn. Photoshop, Illustrator and Premiere Pro are standard tools for design and video, and Acrobat and PDF, a format Adobe created, are how much of the world shares documents. Firefly, Adobe's family of generative AI models, now works inside many of these products, and Acrobat AI Assistant helps people read and understand long documents. Adobe Experience Cloud helps organisations personalise what each person sees, and Adobe runs large engineering and research teams in India. This problem statement brings two of these ideas, personalisation and AI that helps people learn, to students at their own institute.

1. PROBLEM STATEMENT

Every college student arrives with a different background and leaves with a different ambition. A chemical engineering student may want a software role, a first-year student may already be planning an MS abroad, and a classmate may be preparing for the civil services or an MBA. Yet the academic journey is largely one-size-fits-all: the same core courses, a long list of electives, minors and open courses, and academic rules that students mostly navigate on their own, through seniors, notice boards and word of mouth.

The challenge is to build an **AI-powered personalised learning buddy** for your own institute: one that understands an individual student and their goal, identifies their skill gaps, creates a personalised plan from the institute's curriculum and relevant open learning resources within the institute's academic rules, continuously adapts that plan based on progress, and helps the student demonstrate measurable readiness for their goal.

The system should move beyond simply recommending courses and instead create an **adaptive end-to-end learning journey**.

Core Learning Loop

Assess → **Personalise** → **Learn** → **Practise** → **Adapt** → **Demonstrate Mastery**

2. CHALLENGE

Participants should design and build a system that performs the following steps, starting with gathering the data it needs.

2.1 BUILD THE KNOWLEDGE BASE

Teams must collect their institute's curriculum and course data along with relevant open learning resources, and feed it into the platform. This is the foundation the buddy reasons over.

What to collect (indicative only)

- **Course catalogue:** course code, title, department, credits, semester offered, syllabus and learning objectives, prerequisites, skills, course type (core, elective, open / free elective) and mode (offline / online)
- **Academic rules:** credit limits per semester, CGPA or eligibility criteria for minors, prerequisite rules, and rules on earning credit through online courses (e.g. NPTEL / SWAYAM) where the institute allows it
- **Learning resources:** lecture notes, course pages, and openly licensed external content such as NPTEL / SWAYAM, MIT OpenCourseWare and other Open Educational Resources

Teams define their own schema; it should capture what the buddy needs to reason about prerequisites, eligibility and skill coverage. A practical target is to cover one or two programmes in depth (for example 20–40 courses, 100–300 modules or lessons, 20–30 skills, and 100+ exercises and assessment questions) rather than the whole institute superficially.

Data quality is critical: the buddy is only as good as its knowledge base. Data must be accurate, complete for the chosen programmes, consistent across sources, current for the semester, and traceable to its source. Teams should check and document its quality.

Data ethics: use only publicly available institute information, openly licensed content, and data that students or the institute have consented to share. Survey responses should be anonymised, and no real student records (grades, personal details) should be used without permission.

2.2 DIAGNOSE

Understand the student's:

- Existing knowledge and skills (assessed or self-declared)
- Academic background: programme, year, courses completed
- Strengths and weaknesses
- Goals and ambitions (career, higher studies, competitive exams)
- Learning pace and behaviour
- Constraints: available time, remaining credits and elective slots, applicable academic rules

The system should identify gaps between the student's current capabilities and what their goal requires.

Profile data varies in reliability: grades and assessed results are strong evidence, while self-declared interests and skills are weak until backed by evidence. The system must record the source of each item and weigh it accordingly.

2.3 PERSONALISE

Create an individualised plan by selecting:

- Relevant institute courses, electives and minors
- Learning resources (institute and open)
- Exercises and practice activities
- Assessments
- Milestones beyond coursework, such as projects, internships and exam timelines

Every plan must satisfy all constraints (credit limits, prerequisites, semester offerings, eligibility rules, student preferences such as a credit cap or courses not to be taken together, and available time) and be verifiable against them. It must not assign content the student has already mastered. Practice should be active (exercises, projects, case studies) rather than passive content consumption.

2.4 ADAPT AND EVALUATE MASTERY

The learning plan should not remain static. The student should be assessed continuously through quizzes, exercises and project checkpoints, not only one-off tests, and the plan should adapt based on signals such as:

- Assessment scores
- Exercise performance
- Number of attempts
- Time spent
- Topics repeatedly missed
- Course module completion

For example, a student who demonstrates mastery of a topic could skip additional introductory material, while a student struggling with the same topic could receive additional explanations, exercises, prerequisite content or open learning resources.

Plans span multiple semesters, and goals can change. When stated interest and evidence diverge (e.g. a student targeting data science with a weak grade in linear algebra), the system must surface the gap, adjust the plan to close it, and help the student reassess the goal, rather than ignore either signal.

Throughout, the system should determine whether the student has actually acquired the target skill, distinguishing between:

> *"The student completed the course"* **vs.** *"The student demonstrated the required level of proficiency."*

For goal-level outcomes such as placement readiness, mastery may be shown through projects, timed mock tests or mock interviews rather than course grades alone.

2.5 EXPLAIN AND CONVERSE

The system should give the student a clear understanding of:

- What they already know

- What they need to improve
- Why specific content or courses were recommended
- How their learning plan has changed
- Whether they have achieved the target skill
- What they should do next

Interaction is multi-turn. The buddy must:

- Handle follow-ups and side questions fluidly, then return to the plan
- Keep the right context as a session grows: goal, constraints, current plan and recent evidence must never be lost
- Retain long-term memory across sessions (profile, evidence history, plan versions, past decisions), so the student never has to repeat themselves

3. NON-ENGINEERING SCOPE: PRODUCT & DESIGN

Beyond engineering, the challenge is also designed to involve **Product Management and Design**.

PRODUCT MANAGEMENT

Participants should:

- Survey and interview students at their institute to gather their ambitions, pain points, and how they plan their academics today
- Turn survey findings into personas and a prioritised requirements list with acceptance criteria
- Define learning objectives
- Design the learning journey
- Define the personalisation strategy, including how conflicting signals (e.g. interest vs grades) are weighed
- Define mastery criteria
- Define product success metrics

- Define the evaluation process for the product: how the team will test that the buddy actually works
- Weigh trade-offs between learning effectiveness, time, academic load, and content complexity

Example of product-management output for one persona:

Item	Example
Persona	2nd-year Mechanical Engineering student aiming for a data-analytics role
Survey insight	Non-CS students say they discover useful electives too late, mostly through seniors
Learning objective	Ready for analyst-role placements by the end of third year
Mastery criteria	Scores 70%+ on timed SQL and statistics problem sets; completes one end-to-end analytics project; clears two mock interviews
Success metrics	Share of students who follow their plan for 4+ weeks; improvement in diagnostic scores; student-rated usefulness of recommendations
Evaluation process	Run a set of test student profiles through the buddy and check each plan against a short checklist (valid prerequisites, goal skills covered, no repeated content)
Key trade-off	Extra electives for skill-building versus CGPA and workload in core courses

UX DESIGN

Participants should design:

- Student onboarding
- Diagnostic assessment experience
- Personalised learning-plan interface
- Learning / practice experience
- Progress and mastery visualisation

- Feedback experience
- Explanation of why the system recommends or changes content

The experience should make personalisation useful and understandable rather than simply presenting AI-generated recommendations.

4. STUDENT PROFILES

Teams should **create student profiles** that represent the range of students at their institute, informed by the survey they run (Section 3).

Each profile should describe who the student is, where they stand today, what they want, and what limits their choices. Example:

```
Student ID: S1024
Programme / Year: B.Tech Chemical Engineering, Year 3
CGPA: 7.8
Courses completed: Intro to Programming (B), Maths I–III, ChE core
Goal: Software engineering role after graduation
Interest: machine learning (self-declared)

Diagnostic skills (assessed):
Programming fundamentals 78%
Discrete maths 55%
Data structures & algos 32%
Databases 10%

Constraints:
• 2 open-elective slots remaining
• 6 credits of headroom next semester
• 8 hours/week available outside classes
• Do not keep “Discrete maths” and “Data structures & algos” in the same semester

Preferences: short videos, hands-on coding problems
```

When building the profile set:

- Cover the personas in Appendix A plus any your survey surfaces
- Vary year, branch, CGPA, completed courses, goals, available time and learning preferences
- Include edge cases: backlogs, changing goals, interest that conflicts with evidence, students close to eligibility cut-offs

- Create enough profiles (for example 50–100 or more) that the solution cannot be hand-tuned for a few personas

5. AI / TECHNOLOGY RESOURCES

All LLM and embedding calls must go through the **LiteLLM** library, so the solution can run against any supported model provider without code changes. The model, API key and endpoint must be configurable through environment variables:

```
LLM_MODEL e.g. openai/<model>, azure/<deployment>, gemini/<model>
LLM_API_KEY
LLM_API_BASE optional, e.g. for Azure or self-hosted endpoints
EMBEDDING_MODEL only if embeddings are used
```

Do not hard-code model names, keys or endpoints anywhere in the code or image. If your stack is not Python, run the LiteLLM proxy as a service in your docker-compose project and call it through its OpenAI-compatible API, configured with the same variables.

State the LLM model (and embedding model, if used) your solution was tuned and tested on, in the README and as the default values in .env.example, so the solution can be run under the same conditions it was built for.

Teams are free to decide whether to use traditional ML, recommendation algorithms, knowledge graphs, LLMs, RAG, agentic architectures, or hybrid approaches, but must justify which parts are handled by the LLM and which are computed deterministically (e.g. rule checks, credit totals and scheduling), and how much context is passed to each call.

6. EXPECTED STUDENT DELIVERABLES

Each team should produce:

1. Working Prototype

Demonstrate the complete journey: Assess → Personalise → Learn → Practise → Adapt → Demonstrate Mastery, for at least two or three dissimilar student goals.

Submit the prototype as a docker-compose project that meets these requirements:

- **Single-command start:** docker compose up starts the complete system on linux/amd64 within a few minutes, with no GPU and a total image size under 5 GB
- **No secrets:** model, API key and endpoint read from the environment variables in Section 5, listed in a .env.example file; no API keys or other secrets in the image
- **Seed data and reset:** seed data as plain files (CSV / JSON) in a known folder, loaded at startup, plus a reset command that restores the original state
- **Data control:** an admin page or documented API to view and edit student profiles, scores and course data, plus a bundled database viewer (e.g. Adminer); any edit must trigger re-processing so the buddy reflects it immediately
- **No other network calls:** no outbound calls except to the configured model provider

Include a README with run steps and a short guide to where each kind of data lives, and a 3–5 minute demo video.

2. Data & Knowledge Base

Share the data collected (institute and open sources, with sources cited), the schema, how its accuracy and completeness were checked, the student profiles, and a summary of the student survey.

3. Product Experience

Show the student-facing experience, including onboarding, personalised plan, learning / practice, progress, feedback, and mastery.

4. Technical Solution

Explain the architecture and core models: how the student profile is represented and updated from evidence (learning), how plans are generated and validated (planning), and how each request in a conversation is handled (solving). Cover memory design, context management, the LLM vs deterministic split, mastery evaluation and data flow.

5. Product Rationale

Explain target users, problem being solved, survey findings, design decisions, success metrics, evaluation process, and key trade-offs.

7. EVALUATION FRAMEWORK

The solution is evaluated as a **complete product**, not just on model accuracy.

Dimension	Weight	What is evaluated
Data & Knowledge Base	15%	Is the collected data accurate, complete for the chosen programmes, consistent, current and traceable to its source? Is the schema fit for reasoning about prerequisites, eligibility and skill coverage?
Personalisation	15%	Does the plan reflect the student's skills, gaps, goal and preferences, weighing evidence by its reliability? Is it valid against all constraints and realistic for the semester calendar and weekly time? Does it recommend only courses that exist in the knowledge base, citing course codes?
Adaptation	15%	Does the system appropriately change the plan over multiple semesters based on behaviour, goal changes, interest–evidence mismatch and student overrides (e.g. rejecting a recommended course), and explain the trade-offs? Does it say clearly when a request is not possible and offer the best valid option? Does it move the student toward the goal efficiently, without redundant content?

Dimension	Weight	What is evaluated
Mastery Evaluation	15%	Can the system determine whether the target skill was acquired? Is the diagnostic short and adaptive, are generated questions and grading reliable, and is mastery based on tested evidence?
User Experience / Design	15%	Is the experience intuitive, engaging, and understandable? Does the buddy handle multi-turn conversations, remember the student across sessions, answer side questions and return to the plan, and ask clarifying questions when a goal is vague?
Product Thinking	15%	Is there a clear user problem, journey, and success measurement?
Engineering	10%	Quality, latency, token cost, robustness (graceful handling of LLM timeouts and rate limits), consistent results on re-runs, long-term memory and context management, a sound LLM vs deterministic split, safe handling of off-topic requests and prompt injection, and scalability.

Demonstrating adaptation

Teams will not have months of real student activity, so they should create student personas and simulate their journeys. The Adapt feature can take simulated assessment scores and activity signals as input and show how the plan changes in response, for example the same persona's plan after a strong versus a weak mid-semester quiz.

APPENDIX A. ILLUSTRATIVE STUDENT PERSONAS

The personas below show the range of students the buddy should serve. Teams should use them as a starting point and add personas that emerge from their own survey of students at their institute.

Student	Goal	What the buddy should do
2nd-year B.Tech, Chemical Engineering	Software / IT role after graduation	Map open electives and a CS minor (if eligible) to software skills; sequence programming → DSA → databases and operating systems; fill gaps with open resources (e.g. NPTEL, MIT OpenCourseWare, coding-practice platforms); add projects and placement preparation; stay within credit limits.
B.Tech or dual-degree student, any branch	Higher studies: MS, PhD or research career (India or abroad)	Protect CGPA in core courses; match the student to faculty whose research fits their interests; plan research-oriented or postgraduate-level electives and a UG research project, thesis or research internship; build research skills (reading papers, technical writing) with open courses; plan timelines for GRE / TOEFL or GATE, etc.
B.Tech student, any branch	UPSC civil services (IAS)	Choose humanities and social-science electives that support a likely optional subject; build a weekly preparation plan alongside academics using open resources (e.g. NCERT textbooks, NPTEL humanities courses); protect CGPA and graduation requirements.
Student interested in a startup	Build and launch a venture	Recommend entrepreneurship, product-design and business-basics courses and open resources; point the student to the incubation cell and relevant clubs, which sit outside what the buddy tracks.