Part A : Design Thinking: Modern School Management System

1) Executive Summary

Designing a School Management System (SMS) from scratch today gives us the chance to move away from outdated, clunky systems that are hard to use and often not mobile-friendly. Instead, we can build a lightweight, secure, and user-friendly platform that meets the real needs of teachers, students, parents, and administrators. The goal is to create a tool that supports everyday school life rather than adding extra work, while also using modern technologies like APIs, cloud storage, and AI-driven insights to make the experience smarter and more efficient.

2) Who Uses It and What They Do

The system is designed with four main user groups in mind, each with their own tailored dashboard and key features.

Teachers would be able to take attendance quickly, see their scheduled classes for the day, and follow up with families when students are absent or falling behind.

Students could view their grades, track their attendance, and see upcoming deadlines or assignments in one central place.

Parents and guardians would have access to a portal where they can check attendance, receive messages from the school, report absences, and even ask questions about school policies.

Administrators would manage enrolments, connect parents to students, oversee key performance indicators (KPIs), send out communications, and ensure data is stored securely.

3) What Would Be Different (Compared to Older Systems)

Unlike legacy systems that are slow and complex, this new SMS would streamline everyday tasks and focus on the user experience. Key differences include:

- Role-based dashboards to reduce clicks and surface the top tasks immediately, saving staff valuable time.
- Offline functionality through a Progressive Web App (PWA), so registers and updates sync silently when the connection returns.
- Built-in policy Q&A that only answers from official school PDFs, with clear citations or an 'I don’t know' response.
- Automation of routine tasks, such as drafting letters after repeated unauthorised absences, which staff can review and approve.
- Easy integration with other systems using REST/GraphQL APIs and webhooks.
- Privacy enforced by design with strict roles, row-level security, and a complete audit trail.

4) What Opportunities Open Up

By combining attendance trends, assignment submissions, and grades, the system can highlight students who might be at risk earlier than traditional methods allow. For example, if a student misses three classes in a row and their grades start to dip, the system could flag this to teachers and suggest an intervention.

Having a single student timeline with attendance, grades, communications, and teacher notes in one place allows staff to make more informed decisions and provide targeted support.

The system also reduces administrative overhead by pre-filling letters and reminders that staff only need to approve before sending. Administrators can also benefit from saved data views, such as 'Year 9 students with attendance below 92%', as well as clear weekly summaries.

5) What the User Interface Looks Like (Screenshots)

Each user role would have a clean, role-based dashboard designed for quick access to their most important tasks.

• Teacher Dashboard : a clear 'Take Register' button, today's classes, and completion bars.

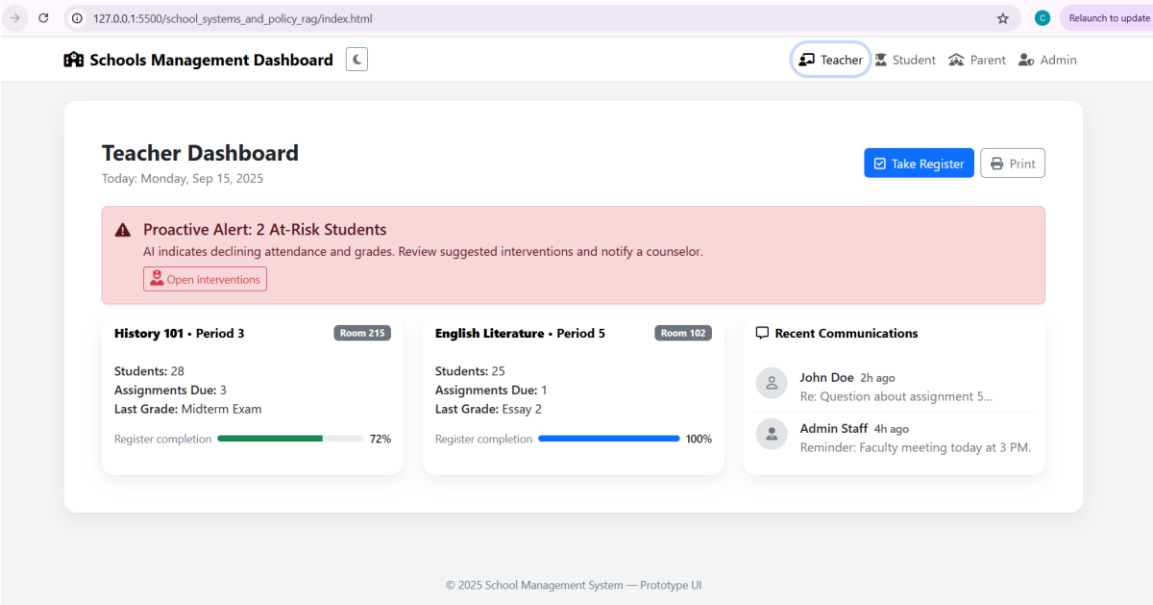
• Student Dashboard : an overview of grades, an attendance ring, and a list of due assignments.

• Parent Portal : attendance summaries, recent messages, and a simple absence reporting form.

• Administrator Dashboard : headline KPIs such as total students, attendance rates, and recent alerts.

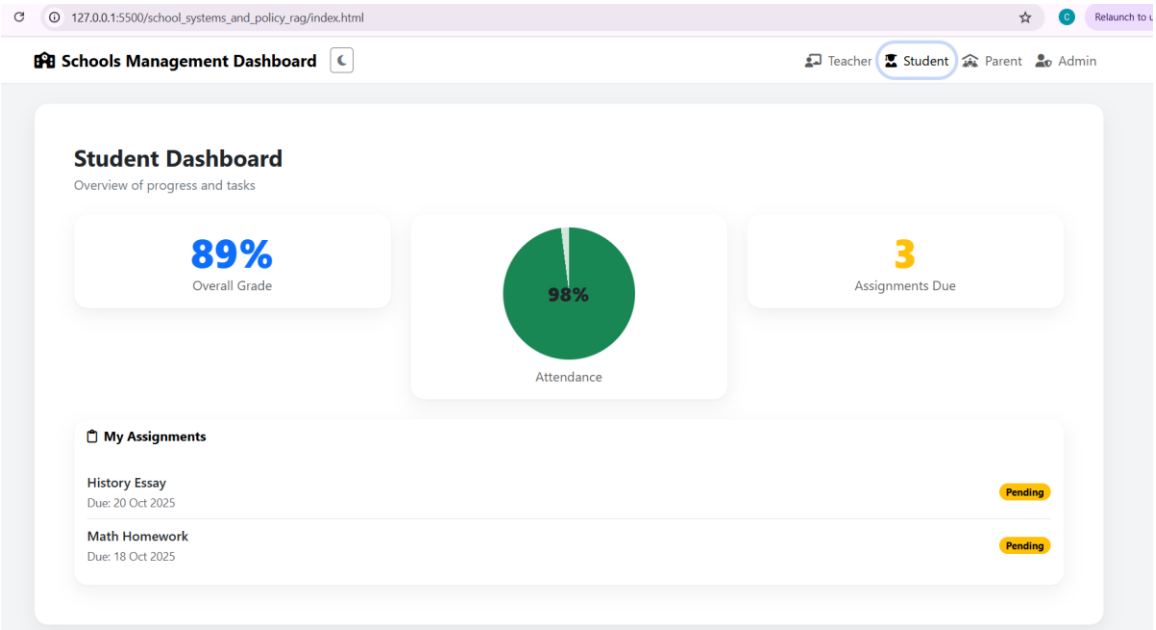
Below are screenshots of a prototype user interface:

Teacher Dashboard (Prototype) :



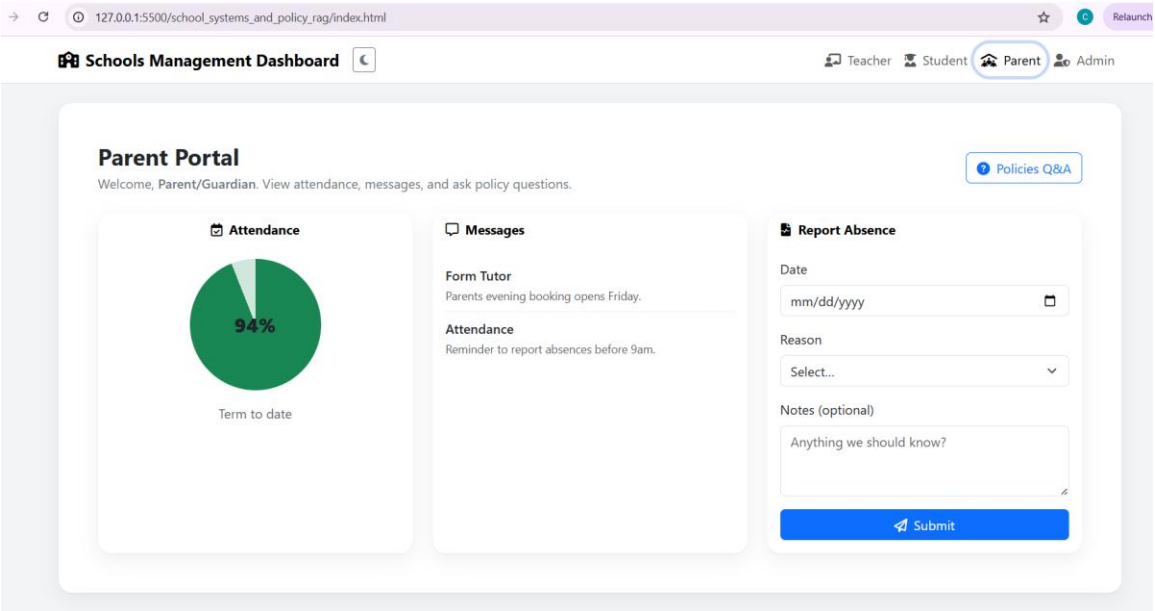
What to notice: a clear 'Take Register' button, today's classes, and completion bars.

Student Dashboard (Prototype):



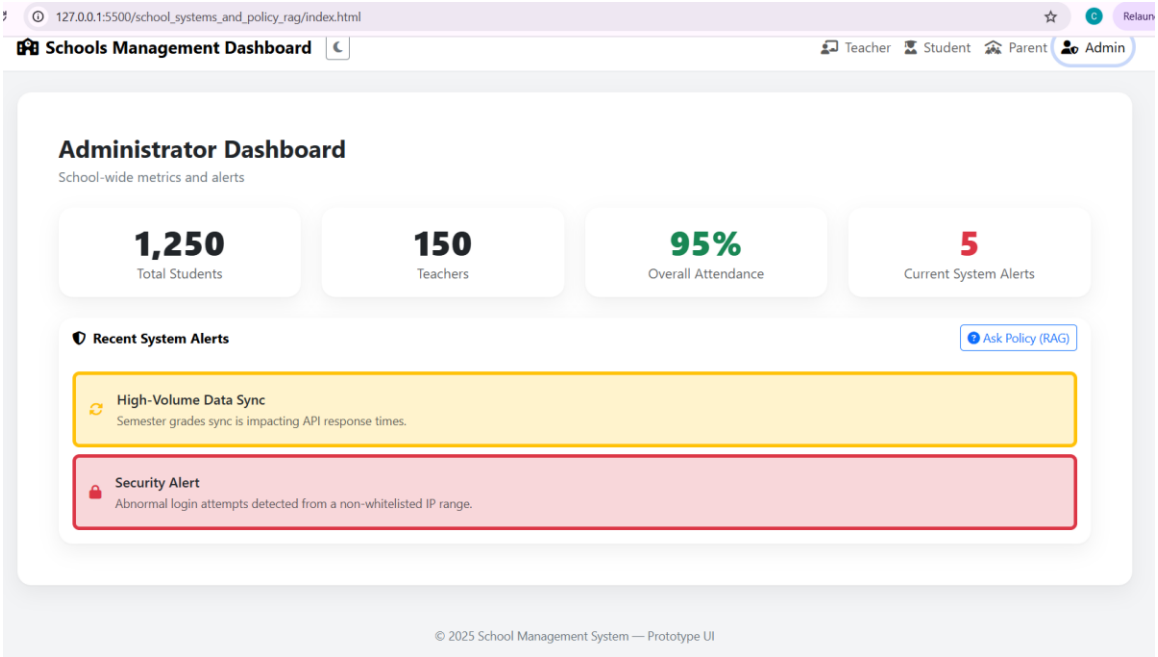
What to notice: overall grade, attendance ring, and due work list.

Parent Portal (Prototype) :



What to notice: attendance summary, messages, and a simple 'Report absence' form.

Administrator Dashboard (Prototype) :



What to notice: headline KPIs and recent system alerts.

6) Enrol a Student and Link a Parent (Admin Flow)

The enrolment process is designed to be straightforward but secure. Administrators create a student record with details such as name, date of birth, year/tutor group, and a school-issued student ID. Parents or guardians are either selected from existing records or created anew, with duplicate checks based on email address. Parents are then linked to students using a link table that records relationship details, permissions, and primary guardian status. Parents receive a secure invitation to set up their login, and every action is fully audited.

7) What Parents Can See and Do

Parents would be able to see their child's attendance summary and recent messages. They could easily report absences using a simple online form, and would receive confirmation emails for their submissions. Parents could also ask policy-related

questions, and the system would provide answers with clear citations to the source document. If the answer is not known, the system would respond with 'I don’t know.

8) Emailing (Notices and Confirmations)

Emails such as notices and confirmations would be sent using a single provider (SMTP/SendGrid in the MVP). The system would store a message log for transparency, with details such as recipient, subject, status, and timestamp. Templates would allow variables such as guardian name and student name to be inserted automatically. Bulk sending would be rate-limited, and administrators could view the message history for each student.

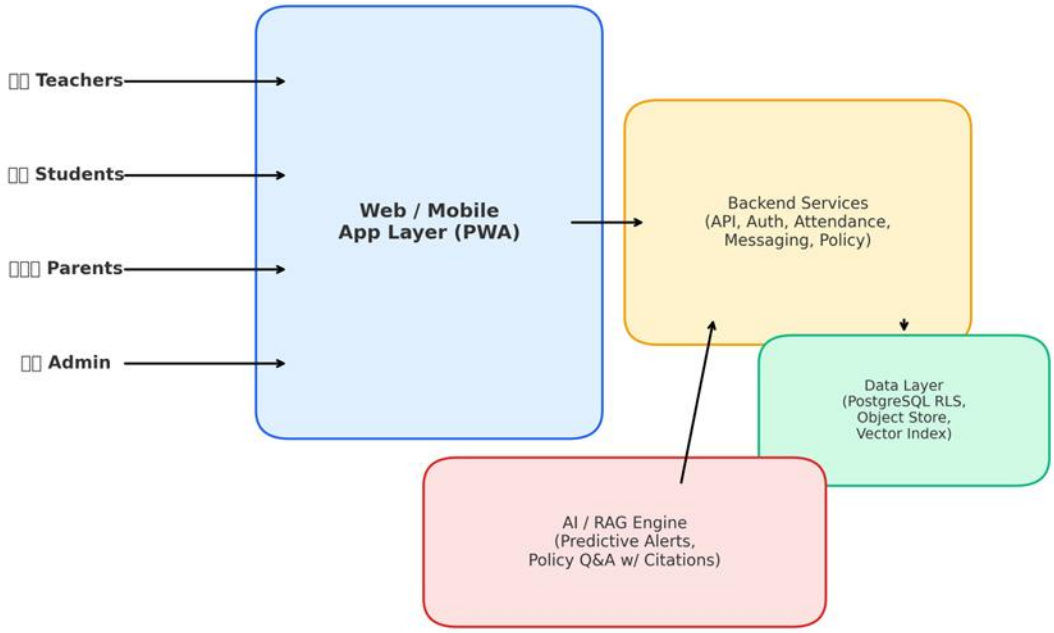
9) Offline: How It Works in Practice

The system would run as a Progressive Web App (PWA), meaning it could be installed on any device and continue to work without an internet connection. Teachers could still take registers offline, with data stored locally and synced automatically once the connection is restored. The interface would clearly show which records are queued and which have synced. If conflicting edits occur, the system would provide options such as 'Use server', 'Keep mine', or 'Merge'. For policy Q&A, a small offline index could provide limited answers, defaulting to 'I don’t know' when unsure.

10) How We Would Architect It (Simple View + Diagram)

The architecture is designed around modularity, security, and scalability. The front end would be a responsive web app that works seamlessly across desktop, tablet, and mobile devices, with registers available offline. An API gateway would handle authentication, rate limits, and routing. Core services would manage attendance, classes, assessments, communications, and files, while safeguarding data would have stricter controls. Key actions would be stored as events for audit and reporting. Data would be stored in PostgreSQL with row-level security, supported by a search index for documents and object storage for files. Finally, an AI-powered Policy Q&A service would retrieve passages from PDFs and return short answers with citations.

System Architecture Diagram :



11) Security and Privacy (Plain Rules)

The system would be secure by design, following clear, easy-to-understand rules:

- Parents can only see their own child(ren).
- Teachers only have access to their assigned classes.
- Administrators have school-wide access.
- PostgreSQL Row-Level Security (RLS) enforces permissions on the server.
- Safeguarding data is stored separately with stricter controls.
- All data is encrypted in transit and at rest, and every sensitive change is logged.
- GDPR compliance is supported with subject access requests, erasure processes, and a DPIA before launch.

12) MVP and How We Will Measure Success

The Minimum Viable Product (MVP) would include the teacher register (with offline support), teacher and admin dashboards, the parent absence reporting form, policy Q&A with citations, and basic email functionality.

Success would be measured by the end of the first term through practical outcomes:

- Teachers can submit registers in under two minutes.
- Most absences are reported online instead of by phone.

• Policy answers are delivered in under two seconds, with clear citations or an 'I don’t know' response.
• A complete audit trail is available, and staff usage of the system is consistently high.

Conclusion

In conclusion, designing a modern School Management System from scratch gives us the opportunity to move beyond outdated, clunky tools and create something genuinely supportive of teaching and learning. By prioritising user-friendly design, offline capability, strong privacy rules, and AI-powered insights, this system would reduce administrative workload and help schools intervene earlier when students need support. Ultimately, the goal is not just to manage information but to build stronger connections between teachers, students, and families.