Platform Overview

SANKALP

The Student Platform of BITS Pilani, Hyderabad Campus

Navya Sharma

Chairperson and Developer

sankalpbphc.com

August 27, 2026

This document describes the platform's architecture and capabilities. Implementation details of authentication, authorisation, session handling, deployment and data schemas are deliberately outside its scope.

Contents

1 Introduction

SANKALP's website and student platform serve the student body of BITS Pilani, Hyderabad Campus. The system is live, is in continuous daily use, and has grown into thirteen substantially independent applications operating behind a single front door.

This document sets out what the platform is, how it is organised, and the reasoning behind the handful of architectural decisions that shaped everything else. It is written for two audiences at once: a technically literate reader who wants to understand how a system of this size is structured, and an organisational reader who needs to know what the platform does and why it is built the way it is.

1.1 At a glance

Application code	Approximately 964 PHP files, around 141,000 lines
Independent sub-applications	13, each with its own data and administration
Databases	16, partitioned one per ownership boundary
Platform	PHP 8, MariaDB, Apache
Authentication	Google OAuth 2.0; no password is stored by the platform
Front end	Hand-written CSS and JavaScript; no framework, no build step
Status	Live, in daily use

1.2 A note on the constraint

The platform uses no web framework, no package bundler and no build step. Its server code is procedural PHP behind a single front controller, and its styling and interactivity are written by hand.

This began as a hosting constraint rather than a preference, but it has shaped the project more than any other single fact about it. A framework supplies structure whether or not its users have thought about structure. Without one, every organising decision has to be made deliberately, written down, and then actually adhered to. Much of what follows is a description of the conventions that do the job a framework would otherwise have done — and of the documentation practice that keeps them honest.

2 Architecture

2.1 Portals

The platform presents three distinct portals from one codebase.

The public website
The organisation's outward face: who SANKALP is, its offices, its editorials, and induction information for incoming students. No sign-in required.

The student platform
The signed-in environment, available to members of the institution, which opens out into the thirteen minisites described in Section 3.

The administrative portal
Where the platform and its content are managed, governed by its own independent model of authority.

2.2 Two axes of authority

Administering a website and running an organisation are different problems, and the platform treats them as such. Which portal a person may enter is one question; what they are entitled to do once inside the administrative portal is a separate one, governed independently of the first.

Individual applications then layer their own custodial model on top where their practice demands it. The people who curate a library catalogue are not the people who handle grievance intake, and a single flattened permission scheme would have matched nobody's real working arrangements. The platform permits that divergence deliberately, and each instance of it is documented.

2.3 Isolation

Only the document root is reachable by URL. Application code, third-party dependencies, outbound integrations and user-uploaded files all sit outside the web server's reach entirely.

The same instinct governs the data layer: sixteen databases, partitioned by ownership rather than by convenience, so that each application owns its own records and no application can reach across a boundary into another's.

3 The thirteen applications

The student platform is not one application with thirteen features. It is thirteen independent applications, each with its own data, routing, assets and administration, sharing only the platform's front door and its visual language.

Application	What it does
Academic resources	Course materials, grades, reviews, a collaborative wiki and a doubts board, synchronised against a shared drive held as the source of truth.
Library	A catalogue of 41,938 books with a purpose-built search engine, course-linked shelves and student reviews.
Course Planner	A timetable builder whose generator searches for clash-free schedules by backtracking, rather than leaving a student to discover the conflicts.
Organisation management	The largest application on the platform, with its own kernel and thirty modules: people, offices, groups, tasks, meetings, documents, elections and selections, internal communications, the deliberative chambers, the Election Commission and the Court of Duty.
Buy & Sell	A used-textbook marketplace for the campus.
Sahayta	A grievance and protected-disclosure platform, built as two parallel systems that share only their visual chrome — because the two processes have different confidentiality requirements and should not share machinery.
Pulse	Student-government transparency: representatives, promises and referenda.
FAQ	A searchable campus FAQ with an assistant that answers strictly from a bank of verified articles. See Section 4.

Application	What it does
Fests	Event and fest management, with planning, live scoring and an arena view.
Noticeboard	Campus notices with streams, reactions and scoped moderation.
News	An internal editorial digest with a full editorial workflow and a four-level editor ladder.
Compass	A curated directory of external resources and an administratively curated document library.
Games	An arcade hub, a members-only mock-trial competition, and a social lobby.

3.1 Why they are separate

Splitting the platform this way was the decision that made every subsequent one tractable.

Each application can be extended, redesigned or rewritten on its own schedule without the other twelve needing to know. A change to how the library ranks search results cannot affect grievance intake. A redesign of the fest scoreboard cannot disturb the editorial workflow. Growth adds applications rather than adding weight to an existing one, which is what keeps scale from turning into fragility.

It also matches the organisation. These tools are run by different committees with different needs, different tempos and different notions of who is responsible. An architecture that mirrors that reality requires far less negotiation than one that fights it.

4 Grounding an assistant

The campus FAQ includes an assistant that answers students' questions about the institution. It answers only from the bank of verified articles behind it.

That restriction is *structural*: it is a property of what the system is capable of supplying to the model, not an instruction in a prompt asking the model to restrain itself.

The distinction is the entire point. A prompt that says "use only the provided context" is a request, and a request is something a language model may decline under pressure from an unusual question. An architecture in which no other context exists is a guarantee, and it holds regardless of what is asked. For a service answering students' questions about their own institution — questions whose answers have consequences for deadlines, eligibility and money — only the second kind of assurance is worth deploying.

The principle generalises well beyond this one feature: wherever a language model touches something that matters, prefer a constraint the system enforces over a behaviour the model is asked to exhibit.

5 Documentation as infrastructure

The platform carries roughly twenty thousand lines of prose across about fifty files, kept beside the code it describes rather than in a separate wiki.

Each major component documents itself with the same pair of files. One states the exact, verifiable contract, and is written for whoever will next modify that component. The other explains the same component in prose for the people who use it. An index at the root of the repository maps a task to the single document that answers it.

For a student organisation this is not documentation for its own sake, and it is not ceremony. Committee members change every year, and every year the platform must be handed to people who did not build it. A system of this size that only one person understands is a system with a one-year lifespan. Writing it down, in both registers, is the difference between a platform that is inherited and one that is abandoned and rebuilt from nothing by the next cohort.

This is the decision I would defend most strongly to anyone building something comparable inside a student body, and the one most often skipped.

6 Principles

Three decisions have earned their place, and all three would be made again.

Independence between applications.
Thirteen self-contained systems rather than one large one, so that growth adds surface rather than adding fragility, and so that each committee owns its own tool outright.

Structural constraints over prompted ones.
Wherever a language model touches something consequential, the guarantee should come from what the architecture makes possible, never from what the model has been asked to do.

Documentation written for two audiences.
The engineer who will modify a component and the office-holder who depends on it need the same system explained in genuinely different registers. Serving only one of them is how institutional knowledge is lost.

The platform is live at sankalpbphc.com. Correspondence about its architecture is welcome; operational and security details are not published.