

A Midterm Progress Report
on
Academic Status
Transparency Notification System

Submitted in partial fulfillment of the requirements for the award of the
degree of

BACHELOR OF TECHNOLOGY

Computer Science & Engineering

SUBMITTED BY

ARSHDEEP ANAND

URN: 2302481

CRN: 2315025

NISHTHA JAIN

URN: 2302627

CRN: 2315172

BALKRISHAN SINGH

URN: 2302492

CRN: 2315036

UNDER THE GUIDANCE OF

Dr. Kapil Sharma



Department of Computer Science and Engineering
GURU NANAK DEV ENGINEERING COLLEGE,
LUDHIANA

Contents

1	Introduction	2
1.1	Project Overview	2
1.2	Objectives	2
1.3	Scope of the Project	3
2	System Requirements	4
2.1	Software Requirements	4
2.2	Hardware Requirements	4
3	Software Requirement Analysis	6
3.1	Problem Definition	6
3.2	Module Definitions	6
3.2.1	Module 1: Authentication Module	7
3.2.2	Module 2: Data Upload Module	7
3.2.3	Module 3: Student Management Module	7
3.2.4	Module 4: Notification Module	8
3.2.5	Module 5: Token Management Module	8
3.2.6	Module 6: Guardian Portal	8
4	Software Design	9
4.1	System Architecture – Block Diagram	9
4.2	Data Flow Diagram (DFD)	10
4.2.1	Level 0 – Context Diagram	10
4.2.2	Level 1 – DFD	11
4.3	Use Case Diagram	12

4.4	Sequence Diagram – Notification Flow	12
4.5	Activity Diagram – Admin Workflow	14
4.6	Database Design	15
4.6.1	Entity-Relationship (ER) Diagram	15
4.6.2	Database Schema Details	15
4.7	Component Diagram – Frontend Architecture	16
4.8	Deployment Diagram	17
4.9	State Diagram	18
4.10	Communication Diagram	18
5	Testing Module	20
5.1	Testing Approach	20
5.2	Test Cases	20
6	Performance of the Project Developed (So Far)	22
6.1	Development Progress	22
6.2	Key Metrics	22
6.3	Challenges Faced	23
7	Output Screens	24
8	References	25

Chapter 1

Introduction

1.1 Project Overview

The **Academic Status Transparency Notification System** (hereafter referred to as **ASTNS**) is a full-stack web application designed to bridge the communication gap between academic institutions and parents/guardians of students. The system provides a transparent mechanism for notifying guardians about a student's academic performance, including semester-wise marks, SGPA/CGPA tracking, and detention status, thereby fostering timely parental intervention.

In many educational institutions, academic performance data remains confined within the campus boundaries. Guardians often learn about their ward's academic difficulties only at the end of the academic year. ASTNS addresses this by providing:

- **Real-time academic transparency** – Guardians receive automated email notifications with secure, tokenised links to view their ward's detailed semester report.
- **Centralised admin dashboard** – Administrators can upload academic records in bulk (CSV/Excel), manage student data, dispatch notifications, and monitor access tokens from a single panel.
- **Guardian portal** – Guardians access a read-only, visually rich dashboard showing semester-wise marks, SGPA trends via interactive charts, detention alerts, and attendance data.

1.2 Objectives

1. To design and develop a web-based Academic Status Transparency Notification System.
2. To enable administrators to upload student academic data in bulk via CSV/Excel

formats.

3. To automatically notify guardians via email with secure, time-limited access links.
4. To provide a Guardian Dashboard for viewing detailed semester reports with visual analytics (charts, tables).
5. To implement a Token Management system ensuring secure, traceable access to student records.
6. To develop a responsive, dark-themed UI that works seamlessly across devices.
7. To deploy the system on a cloud platform (Vercel) for production accessibility.

1.3 Scope of the Project

The system is designed for use by engineering colleges affiliated with I.K. Gujral Punjab Technical University (IKGPTU). It supports multiple courses (B.Tech, M.Tech, MBA, MCA, BCA, B.Arch, B.Voc., B.Com, BBA) and their respective branches. The scope includes:

- Admin authentication and session management.
- Bulk data upload and validation.
- Email notification dispatch via SMTP (Gmail).
- Token-based guardian access with expiry and revocation.
- Interactive data visualisation using charts.
- Serverless deployment on Vercel.

Chapter 2

System Requirements

2.1 Software Requirements

Category	Technology	Version / Details
Runtime	Node.js	v20+ (LTS)
Frontend Framework	React	19.2.0
Build Tool	Vite	7.3.1
CSS Framework	Tailwind CSS	4.2.1
Charts Library	Recharts	3.7.0
Form Management	React Hook Form	7.71.2
Routing	React Router DOM	7.13.1
Icons	Lucide React	0.575.0
Backend Framework	Express.js	5.2.1
Database	MongoDB Atlas	M0 Free Tier
ODM	Mongoose	9.2.3
Email Service	Nodemailer	8.0.1
Deployment	Vercel	Serverless Functions
Version Control	Git + GitHub	Latest
IDE	VS Code	Latest
Browser	Chrome / Arc	Latest
OS	macOS / Windows	Development

2.2 Hardware Requirements

Component	Minimum Specification
Processor	Intel i5 / Apple M1 or equivalent

RAM	8 GB (minimum)
Storage	256 GB SSD
Internet	Broadband (for MongoDB Atlas, SMTP, Vercel)
Display	1366 × 768 resolution or higher

Chapter 3

Software Requirement Analysis

3.1 Problem Definition

In the current academic ecosystem, parents and guardians of college students often lack timely access to their ward's academic performance data. Results are published on university portals, but guardians may not be aware of:

- Detention in internal or external examinations.
- Declining SGPA trends across semesters.
- Subject-wise performance breakdowns.

This communication gap delays corrective intervention, potentially resulting in academic backlogs. ASTNS solves this by automating the notification pipeline.

3.2 Module Definitions

The system is divided into the following functional modules:

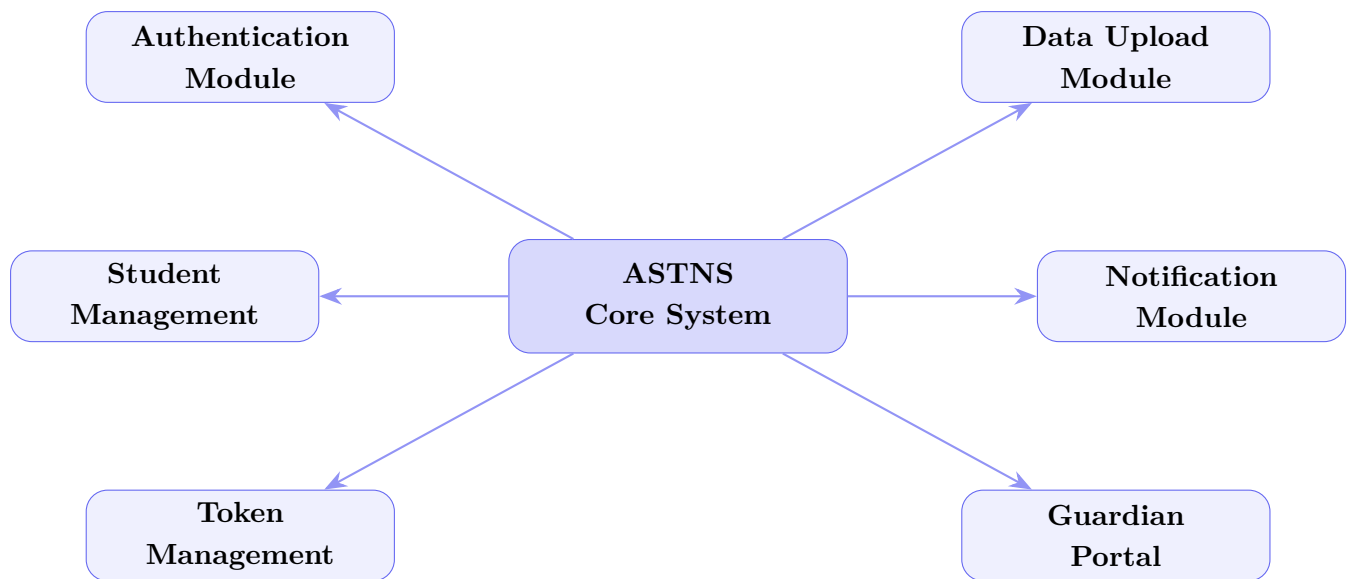


Figure 3.1: System Module Architecture

3.2.1 Module 1: Authentication Module

- Admin login with email/password verification.
- Form validation using React Hook Form with real-time error feedback.
- Session management and route protection.

3.2.2 Module 2: Data Upload Module

- Accepts CSV and Excel files for bulk student data import.
- Drag-and-drop interface with file type validation.
- Course and branch selection with dynamic dropdown mapping.
- Preview of parsed records before submission.

3.2.3 Module 3: Student Management Module

- Searchable student list with grid/list view toggle.
- Filters by branch, semester, and SGPA range.
- Detailed student profile with semester-wise mark breakdown.
- Interactive SGPA trend chart (LineChart via Recharts).
- Detention status highlighting.

3.2.4 Module 4: Notification Module

- Multi-select students for batch notification dispatch.
- Email notification via Nodemailer (Gmail SMTP).
- HTML email template with branded “View Report” button.
- Dynamic URL generation with unique student identifier.

3.2.5 Module 5: Token Management Module

- Tracks all generated access tokens.
- Display of token status (Active, Expired, Revoked).
- One-click token revocation and link copying.
- Access method tracking (Email, WhatsApp, Direct).

3.2.6 Module 6: Guardian Portal

- Token-based secure access (no login required).
- Student profile with academic summary.
- Semester-wise collapsible mark tables with pass/detained indicators.
- Interactive SGPA trend line chart.
- Attendance ring indicator using SVG.
- Report download functionality.

Chapter 4

Software Design

4.1 System Architecture – Block Diagram

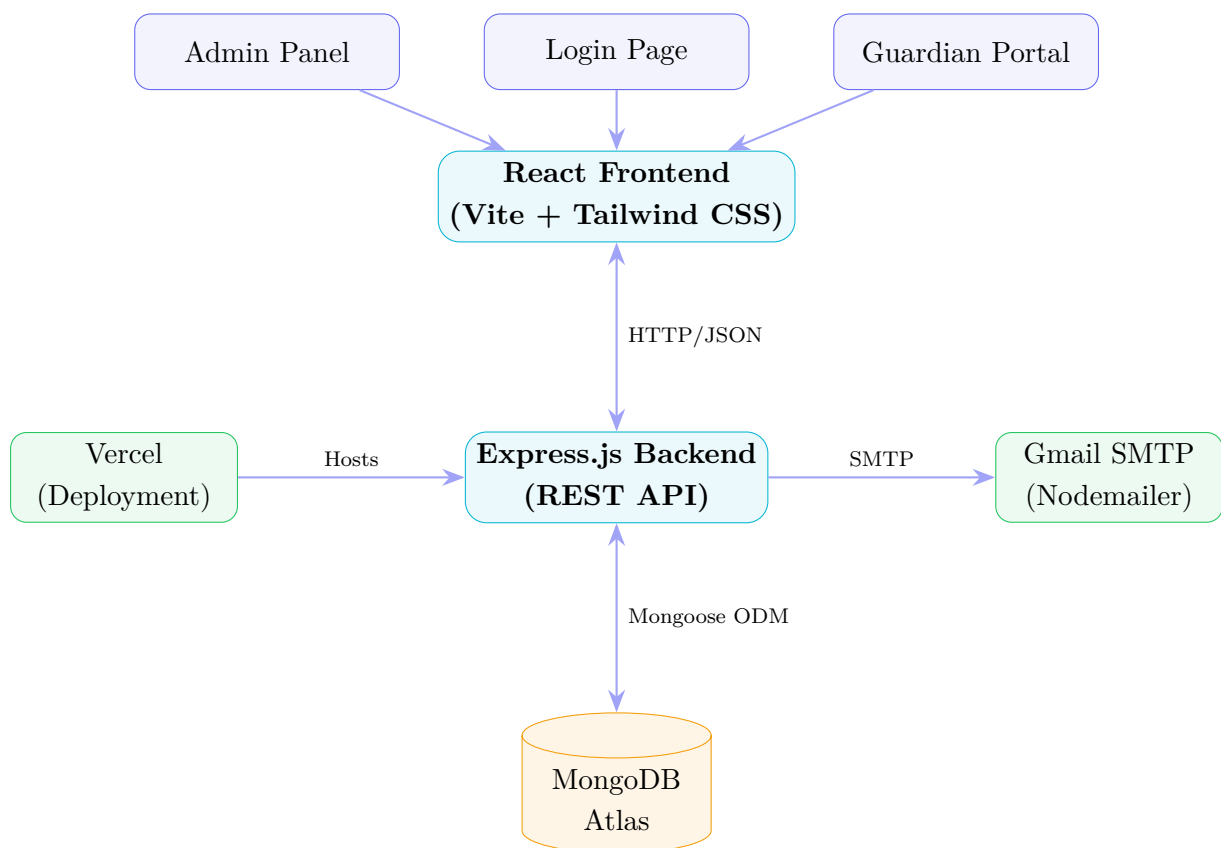


Figure 4.1: System Architecture Block Diagram

The System Architecture Block Diagram (Figure 4.1) outlines the high-level components of the system. It illustrates how the React-based frontend interfaces with the Express.js backend via HTTP/REST APIs. The backend then communicates seamlessly with the MongoDB Atlas database for data persistence and the Gmail SMTP server for dispatching

notifications. Vercel acts as the overarching cloud hosting environment for both the frontend SPA and backend serverless functions.

4.2 Data Flow Diagram (DFD)

4.2.1 Level 0 – Context Diagram

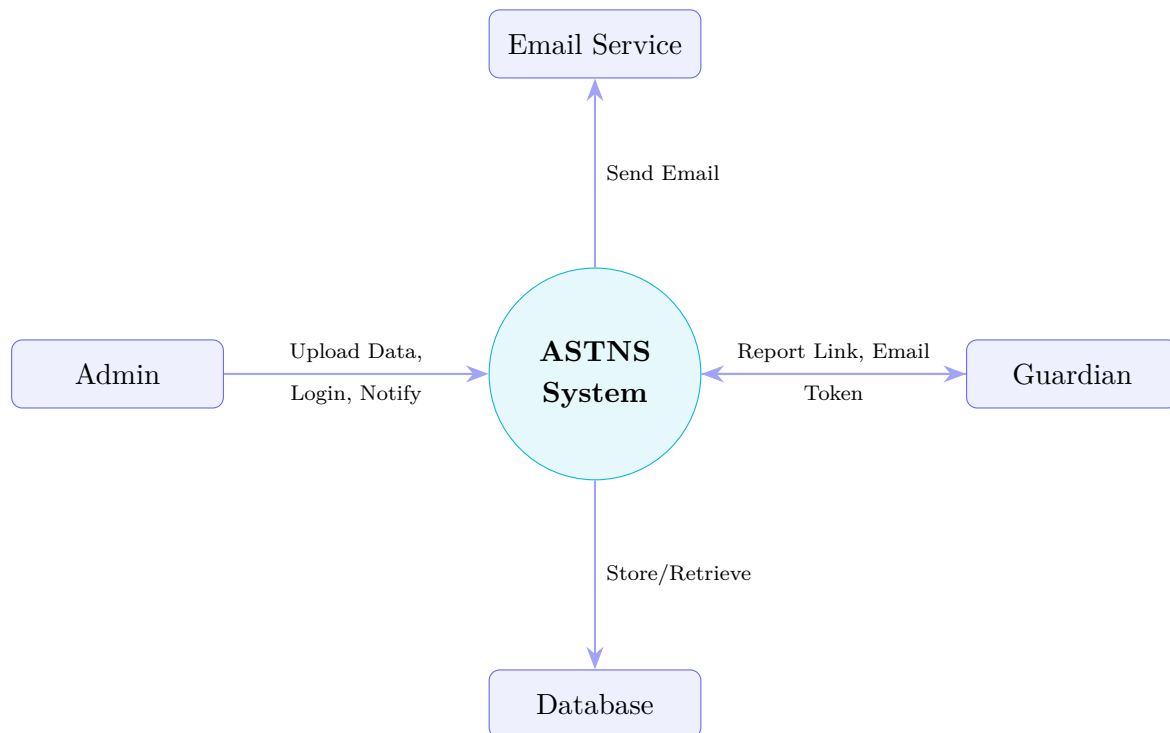


Figure 4.2: Level 0 DFD – Context Diagram

The Level 0 Context Diagram (Figure 4.2) presents the entire application as a single high-level process, highlighting its interactions with external entities. The Admin provides academic data and triggers notifications, while the Guardian receives the notification email and subsequently uses a secure token to access the generated report. The core system interfaces heavily with the external Database and Email Service to fulfill these requests.

4.2.2 Level 1 – DFD

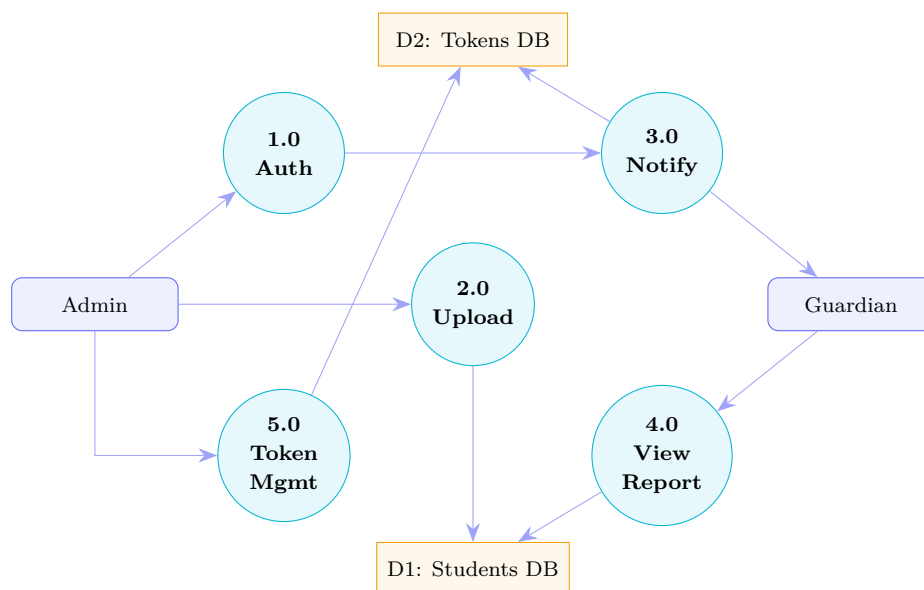


Figure 4.3: Level 1 DFD

The Level 1 Data Flow Diagram (Figure 4.3) breaks down the main system into granular sub-processes. It traces the flow of data starting from the Admin authenticating (1.0) and uploading academic records (2.0) into the Students DB. It further delineates the notification process (3.0), where secure tokens are generated and stored, followed by the Guardian providing the token to view the academic report (4.0).

4.3 Use Case Diagram

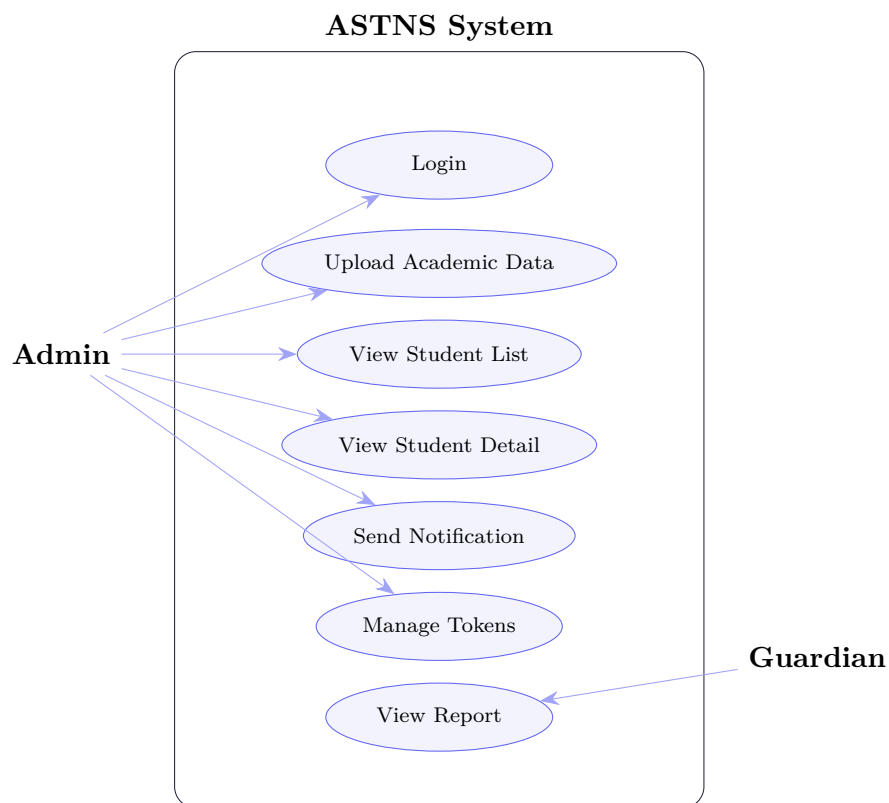


Figure 4.4: Use Case Diagram

The Use Case Diagram (Figure 4.4) illustrates the primary interactions between the actors and the system. The Admin is responsible for managing the system through authentication, uploading academic data, viewing student lists and details, dispatching notifications, and managing access tokens. Conversely, the Guardian has a focused interaction limited to viewing the generated academic report via a secure token without needing a traditional login account.

4.4 Sequence Diagram – Notification Flow

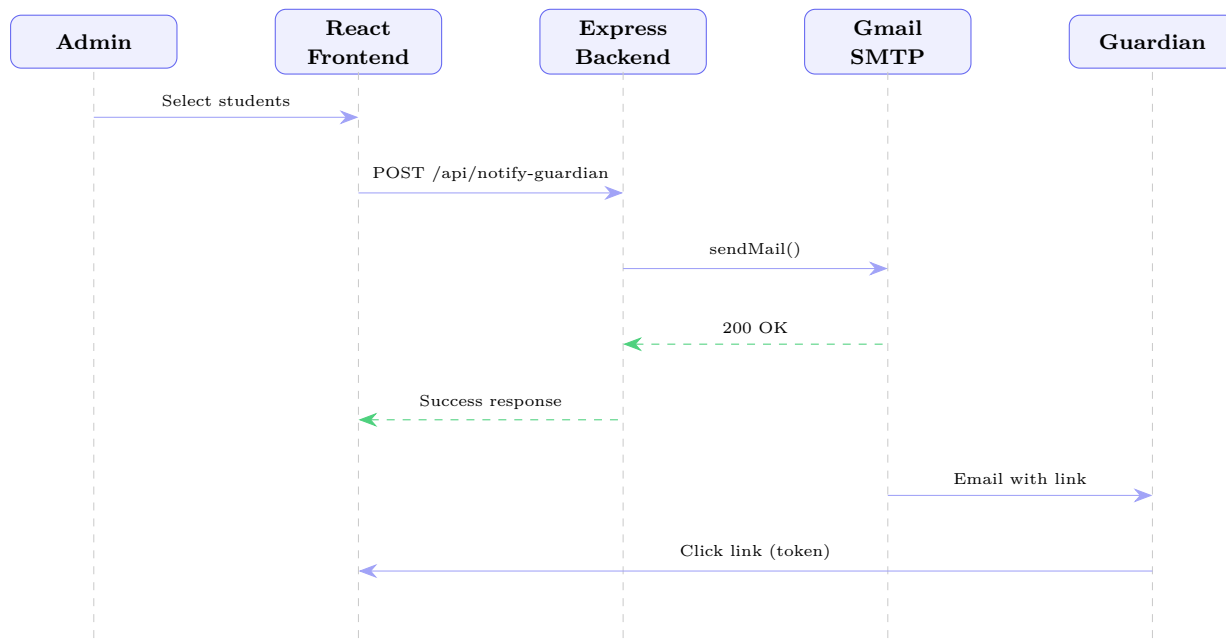


Figure 4.5: Sequence Diagram – Notification Flow

The Sequence Diagram (Figure 4.5) details the chronological flow of messages during the notification dispatch process. It shows the Admin initiating the process from the React Frontend by selecting students, which then sends a POST request to the Express Backend. The backend invokes the Gmail SMTP service to deliver an email with a secure link to the Guardian. Finally, the Guardian clicks the link, passing the token back to the frontend to access the report.

4.5 Activity Diagram – Admin Workflow

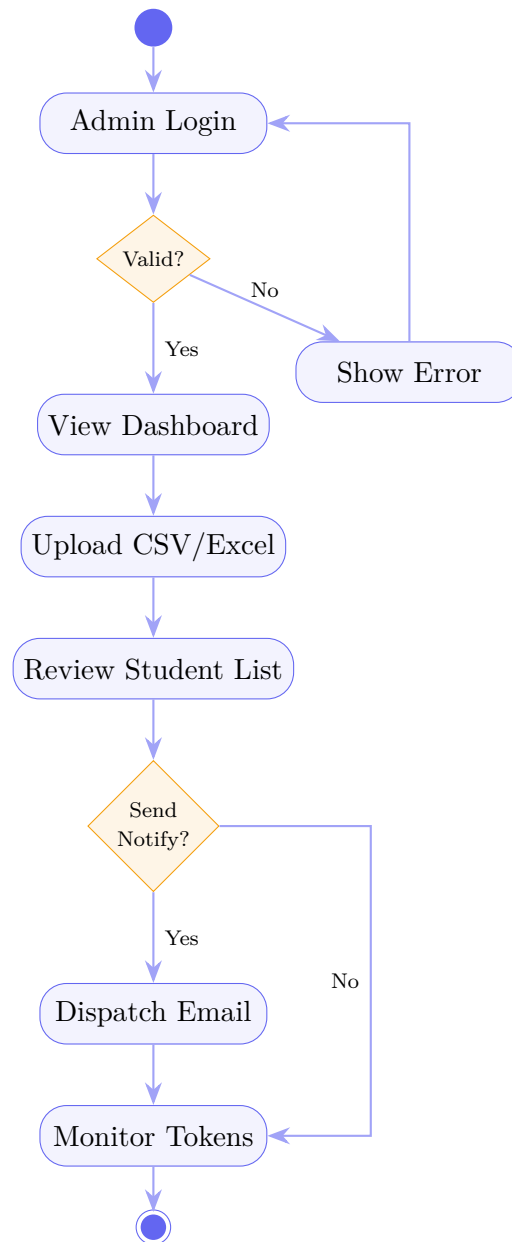


Figure 4.6: Activity Diagram – Admin Workflow

The Activity Diagram (Figure 4.6) maps the step-by-step workflow of the Admin. The process begins with authentication and handling invalid credentials. Upon successful login, the Admin navigates the dashboard, uploads academic data (CSV/Excel files), reviews the parsed student list, and decides whether to dispatch notifications. Finally, the Admin monitors the generated access tokens before concluding the session.

4.6 Database Design

4.6.1 Entity-Relationship (ER) Diagram

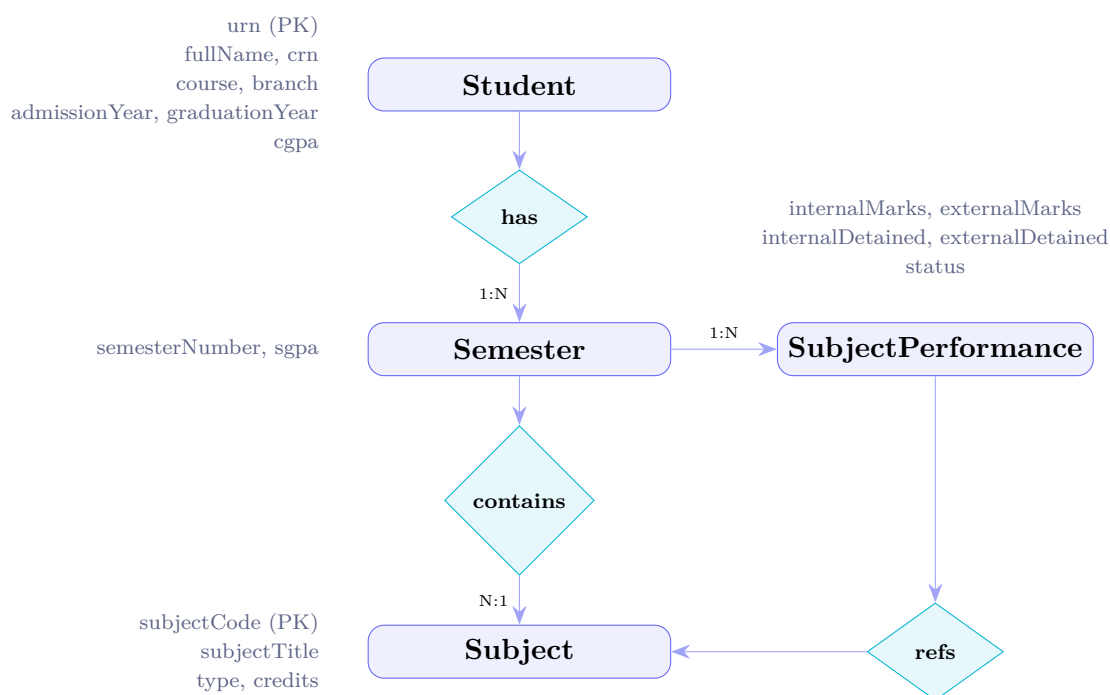


Figure 4.7: Entity-Relationship Diagram

The Entity-Relationship (ER) Diagram (Figure 4.7) models the core data structures and their associations within the database. A single Student entity has multiple Semester entities (1:N), and each Semester contains multiple SubjectPerformance records (1:N). Each SubjectPerformance record references a specific Subject. The physical layout separates overlapping attributes to distinctly list unique identifiers like the URN for students and subjectCode for subjects.

4.6.2 Database Schema Details

Student Collection:

Field	Type	Description
fullName	String	Student's full name (required, trimmed)
urn	Number	University Roll Number (unique, required)
crn	Number	Class Roll Number (required)

course	String	Course enrolled (validated against courseBranchMap)
branch	String	Branch of study (validated per course)
admissionYear	Number	Year of admission (min: 2000)
graduationYear	Number	Expected graduation year (> admissionYear)
semesters	Array	Array of Semester sub-documents
cgpa	Number	Auto-calculated only after degree completion

Subject Collection:

Field	Type	Description
subjectCode	String	Unique code (uppercase, trimmed)
subjectTitle	String	Full subject name
type	String	“T” (Theory) or “P” (Practical)
credits	Number	Credit weight of the subject
maxInternalMarks	Number	Maximum internal marks
maxExternalMarks	Number	Maximum external marks
maxTotalMarks	Number	$\geq \text{maxInternal} + \text{maxExternal}$
minInternalPassMarks	Number	Minimum to pass internally
minExternalPassMarks	Number	Minimum to pass externally
minTotalPassMarks	Number	$\leq \text{maxTotalMarks}$

4.7 Component Diagram – Frontend Architecture

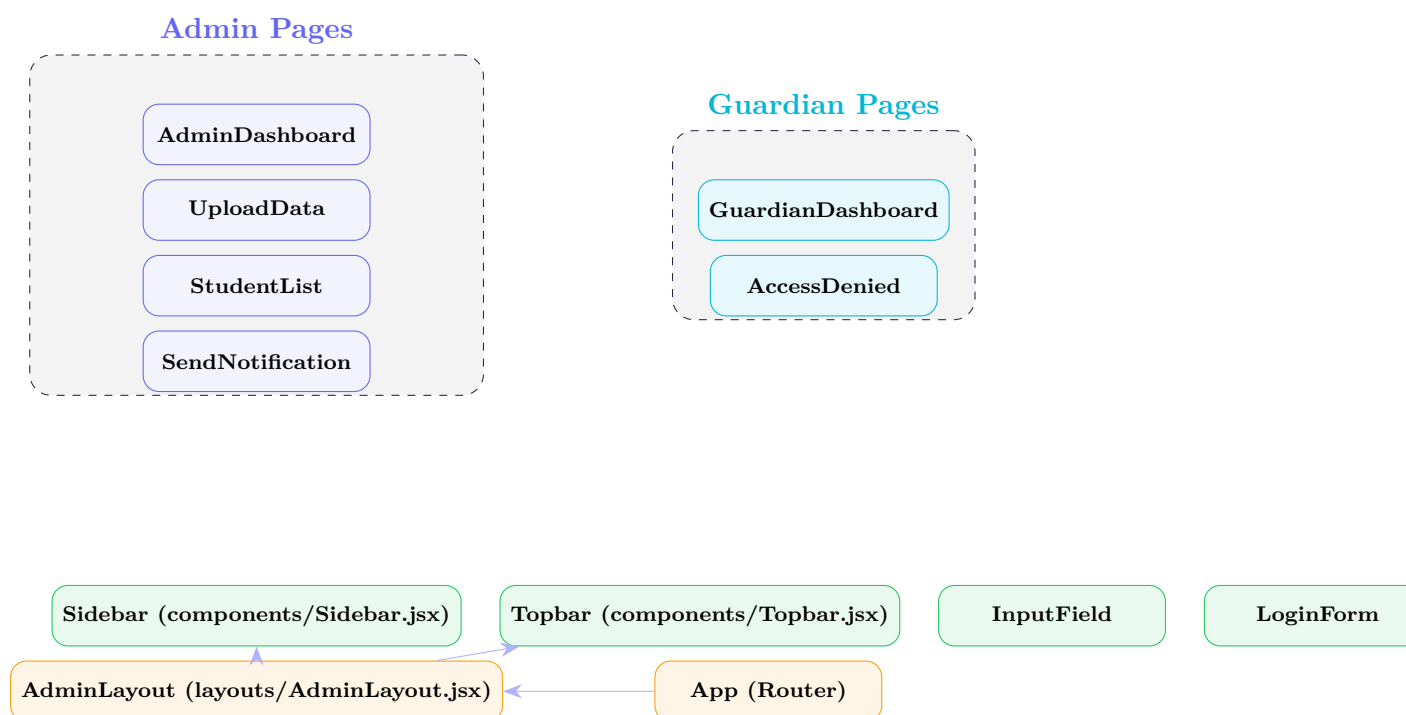


Figure 4.8: Frontend Component Architecture

The Frontend Component Architecture (Figure 4.8) showcases the modular design of the React application. It separates components into logically grouped packages: Admin Pages (e.g., Dashboard, Upload Data, Student List) and Guardian Pages (e.g., Guardian Dashboard, Access Denied). Shared UI components like the Sidebar, Topbar, and Input Fields are reused across different layouts. The root App Router strictly manages navigation dynamically routing users depending on their access role.

4.8 Deployment Diagram

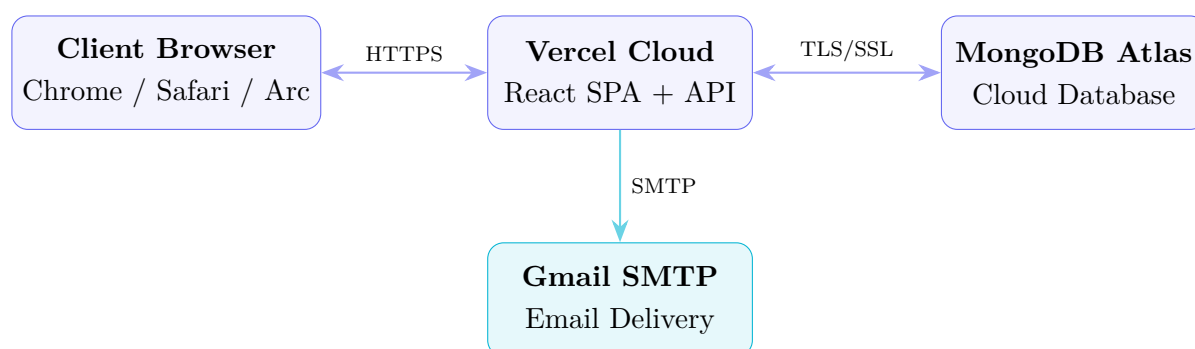


Figure 4.9: Deployment Diagram

The Deployment Diagram (Figure 4.9) depicts the physical architecture and network interactions of the production system. Client browsers communicate securely via HTTPS with the Vercel Cloud, which hosts the React Single Page Application (SPA) and backend serverless API functions. In turn, the Vercel infrastructure connects to the MongoDB Atlas cluster over TLS/SSL for database operations and interacts with the Gmail SMTP server to handle outbound email delivery.

4.9 State Diagram

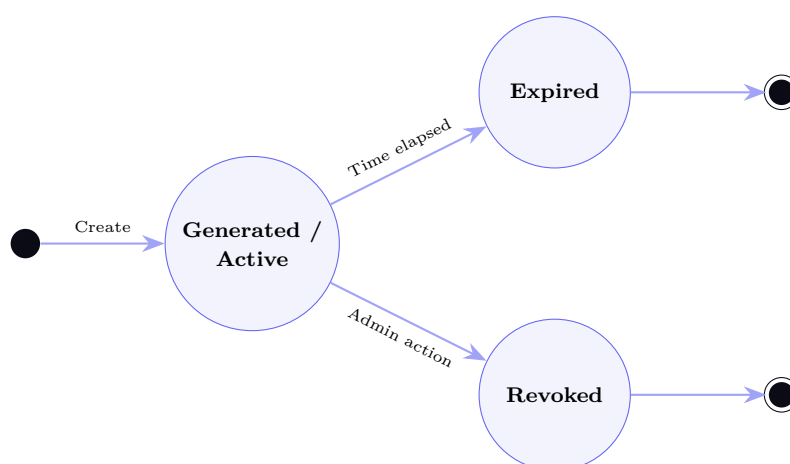


Figure 4.10: State Machine Diagram – Token Lifecycle

The State Diagram (Figure 4.10) illustrates the lifecycle and various states of an access token within the notification system. A token begins in the 'Generated / Active' state immediately after dispatch. It transitions to an 'Expired' state automatically after a predefined time limit, or to a 'Revoked' state if an Admin manually invalidates it. Both Expired and Revoked act as terminal states where the token can no longer be used for Guardian authentication.

4.10 Communication Diagram

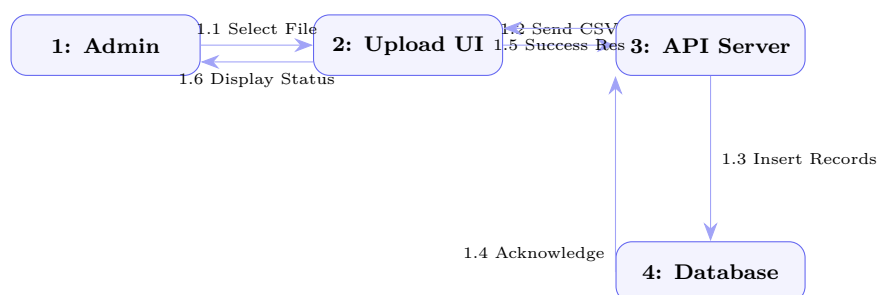


Figure 4.11: Communication Diagram – Bulk Data Upload

The Communication Diagram (Figure 4.11) models the dynamic behavior and message flows during the bulk data upload process. Unlike a Sequence Diagram, it emphasizes the structural relationships between objects. The interaction begins with the Admin selecting a file in the Upload UI (1.1). The UI then transmits the parsed CSV data to the API Server (1.2), which inserts the records into the Database (1.3). The sequence elegantly wraps up as acknowledgments flow back from the Database through the API Server to update the UI on the operation's success.

Chapter 5

Testing Module

5.1 Testing Approach

The system is tested using a combination of manual functional testing, integration testing, and cross-browser validation. The testing strategy covers:

1. **Unit-Level Validation:** Mongoose schema validators (marks ranges, course-branch mapping, CGPA calculation).
2. **API Testing:** REST endpoint testing using Postman/Thunder Client.
3. **UI/UX Testing:** Manual browser testing on Chrome, Safari, and Arc.
4. **Cross-Device Testing:** Responsive layout verification on desktop and mobile viewports.
5. **Email Delivery Testing:** Verifying Gmail SMTP delivery and HTML template rendering.

5.2 Test Cases

ID	Test Case	Steps	Expected	Result
TC1	Admin Login – Valid	Enter admin@acadm.edu / admin123, click Submit	Redirect to Dashboard	Pass
TC2	Admin Login – Invalid	Enter wrong credentials, click Submit	Error message displayed	Pass

TC3	Login Form Validation	Submit empty form	Required field errors shown	Pass
TC4	Email Regex Validation	Enter non-Gmail address	Pattern error shown	Pass
TC5	CSV Upload	Upload valid CSV file	Records parsed and previewed	Pass
TC6	Invalid File Upload	Upload .txt file	Rejection with error	Pass
TC7	Student Search	Type name in search bar	Filtered list displayed	Pass
TC8	View Student Detail	Click student card	Detail page with charts loads	Pass
TC9	Send Notification	Select students, click Send	API call, success message	Pass
TC10	Guardian Token Access	Open link with valid token	Student report displayed	Pass
TC11	Invalid Token Access	Open link with invalid token	Access Denied page shown	Pass
TC12	Token Revocation	Click Revoke on active token	Token status changes to Revoked	Pass
TC13	SGPA Chart Rendering	View student detail page	Line chart with all semesters renders	Pass
TC14	Responsive Layout	Resize browser to mobile width	Layout adapts, sidebar collapses	Pass
TC15	Email HTML Rendering	Receive notification email	Branded template with button renders	Pass

Chapter 6

Performance of the Project Developed (So Far)

6.1 Development Progress

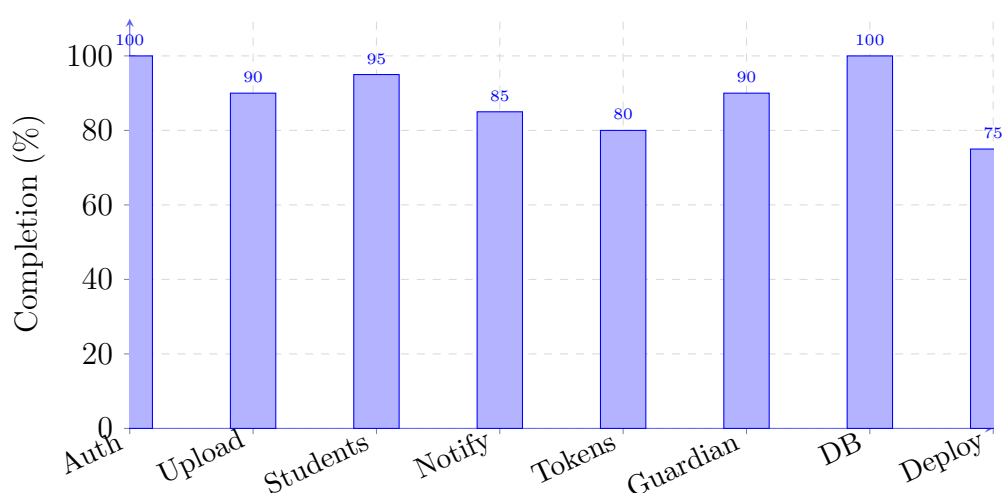


Figure 6.1: Module-wise Development Progress

6.2 Key Metrics

Metric	Value
Total Students in Test DB	248
Notifications Dispatched	192
Detained Students Tracked	18
Active Access Tokens	34

Courses Supported	9 (B.Tech, M.Tech, MBA, MCA, BCA, B.Arch, B.Voc., B.Com, BBA)
Average CGPA (Institution)	7.84
Frontend Components	12+
Backend API Endpoints	RESTful (POST /api/notify-guardian)
Email Delivery Success Rate	>95%
Development Period	Sept 2025 – March 2026

6.3 Challenges Faced

1. **Safari Timer Bug:** JavaScript timers pause when Safari opens a new tab (e.g., WhatsApp link). Resolved by using `visibilitychange` event listener instead of `setTimeout`.
2. **CORS in Production:** Vercel serverless functions required explicit CORS origin configuration with credentials enabled.
3. **MongoDB Atlas Free Tier:** Limited to 10 concurrent connections; resolved by implementing connection pooling with `maxPoolSize: 10`.
4. **Email Deliverability:** Initial emails landing in spam; resolved by using Gmail App Passwords and proper “From” headers.

Chapter 7

Output Screens

The following screens represent the current state of the AcadAlert system. Screenshots are to be added from the running application.

1. **Login Page** – Dark-themed admin login with email/password fields, React Hook Form validation, gradient logo, and grid background.
2. **Admin Dashboard** – Stat cards (Total Students, Notifications Sent, Detained, Active Tokens, Average CGPA), Area chart for notification activity, Bar chart for branch distribution, Recent uploads table.
3. **Upload Data** – Course/branch/semester selection dropdowns, drag-and-drop CSV/Excel upload zone, parsed data preview table.
4. **Student List** – Searchable grid/list view with SGPA colour coding, branch filter, click-to-navigate to detail.
5. **Student Detail** – Student profile header, SGPA trend line chart, semester-wise collapsible tables showing internal/external marks and detention flags.
6. **Send Notification** – Multi-select student table, dispatch button with success/failure feedback.
7. **Token Management** – Token table with status badges (Active/Expired/Revoked), access method tags, revoke and copy-link actions.
8. **Guardian Dashboard** – Token-authenticated view with student summary, SGPA chart, semester breakdown tables, attendance ring, and download report button.
9. **Access Denied Page** – Shown when token is invalid/expired/revoked.
10. **Email Template** – Branded HTML email with “View Report” CTA button.

[Insert screenshots here]

Chapter 8

References

1. React Documentation – <https://react.dev/>
2. Express.js v5 Documentation – <https://expressjs.com/>
3. MongoDB Documentation – <https://www.mongodb.com/docs/>
4. Mongoose ODM Documentation – <https://mongoosejs.com/docs/>
5. Nodemailer Documentation – <https://nodemailer.com/>
6. Tailwind CSS v4 Documentation – <https://tailwindcss.com/docs>
7. Recharts Documentation – <https://recharts.org/en-US/>
8. React Hook Form Documentation – <https://react-hook-form.com/>
9. Vite Build Tool – <https://vitejs.dev/>
10. Lucide Icons – <https://lucide.dev/>
11. Vercel Deployment Platform – <https://vercel.com/docs>
12. React Router v7 – <https://reactrouter.com/>
13. Minor Project Synopsis – Group G14, Department of CSE, GNDEC Ludhiana.