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	NISHTHA JAIN	BALKRISHAN SINGH
URN: 2302481	URN: 2302627	URN: 2302492
CRN: 2315025	CRN: 2315172	CRN: 2315036

UNDER THE GUIDANCE OF

Dr. Kapil Sharma

Department of Computer Science and Engineering

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA

Contents

1 Introduction **3**

1.1 Project Overview 3

1.2 Objectives 3

1.3 Scope of the Project 4

2 System Requirements **5**

2.1 Software Requirements 5

2.2 Hardware Requirements 6

3 Software Requirement Analysis **7**

3.1 Problem Definition 7

3.2 Module Definitions 7

3.2.1 Module 1: Authentication Module 8

3.2.2 Module 2: Data Upload Module 8

3.2.3 Module 3: Student Management Module 8

3.2.4 Module 4: Notification Module 9

3.2.5 Module 5: Token Management Module 9

3.2.6 Module 6: Guardian Portal 9

4 Software Design **10**

4.1 System Architecture – Block Diagram 10

4.2 Data Flow Diagram (DFD) 11

4.2.1 Level 0 – Context Diagram 11

4.2.2 Level 1 – DFD 12

4.3 Use Case Diagram 13

4.4 Sequence Diagram – Notification Flow 14
4.5 Activity Diagram – Admin Workflow 15
4.6 Database Design 16
4.6.1 Entity-Relationship (ER) Diagram 16
4.6.2 Database Schema Details 16
4.7 Component Diagram – Frontend Architecture 18
4.8 Deployment Diagram 19
4.9 State Diagram 19
4.10 Communication Diagram 20

5 Testing Module **21**
5.1 Proposed Testing Approach 21
5.2 Proposed Test Cases 21

6 Performance of the Project Developed **23**
6.1 Development Progress and Milestones 23
6.2 Remaining Requirements for Finalization 24
6.3 Anticipated Ongoing Challenges 24

7 Output Screens **25**

8 References **31**

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 that enables administrators to upload student academic data in bulk via CSV/Excel formats.

2. To automatically notify guardians via email with secure, time-limited access links, providing a Guardian Dashboard for viewing detailed semester reports with visual analytics.
3. To implement a Token Management system ensuring secure, traceable access to student records.

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
HTTP Client	Axios	1.13.6
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:

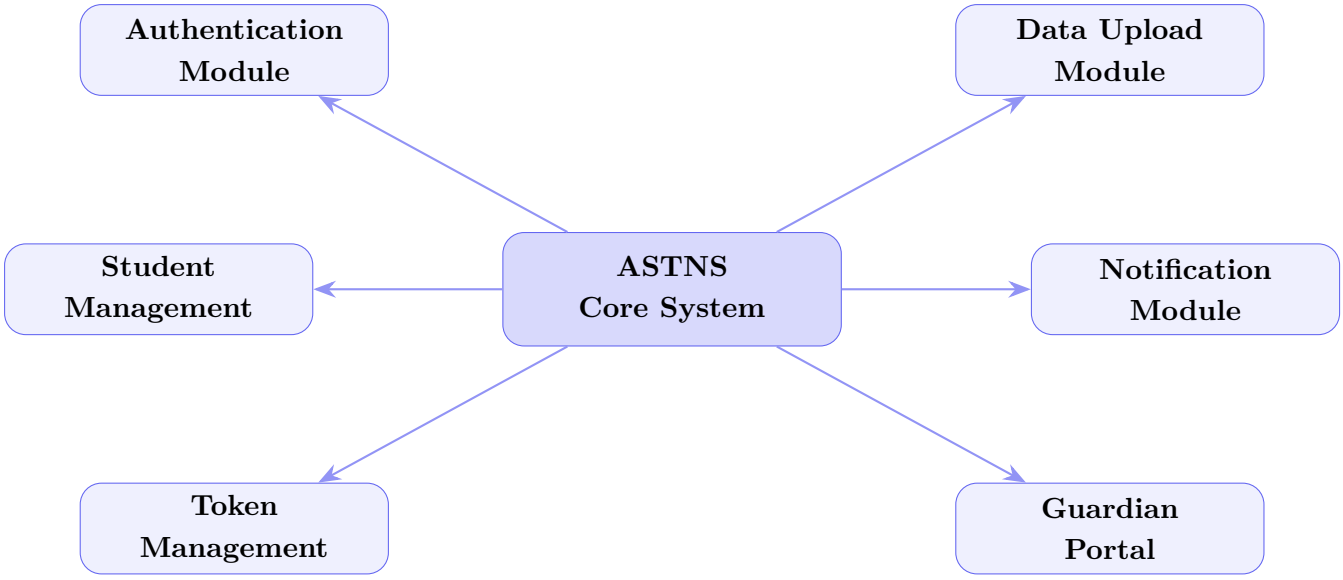

Figure 3.1: System Module Architecture

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

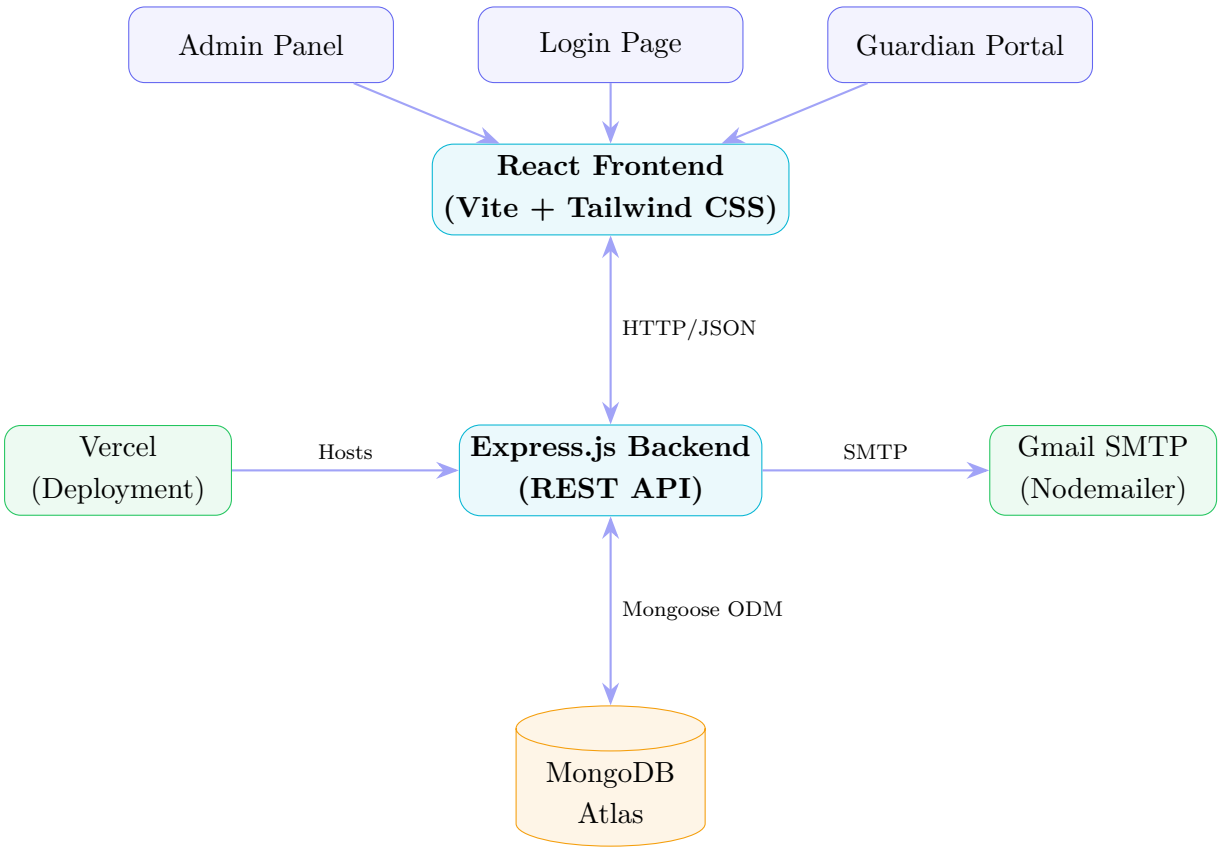

Figure 4.1: System Architecture Block Diagram

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

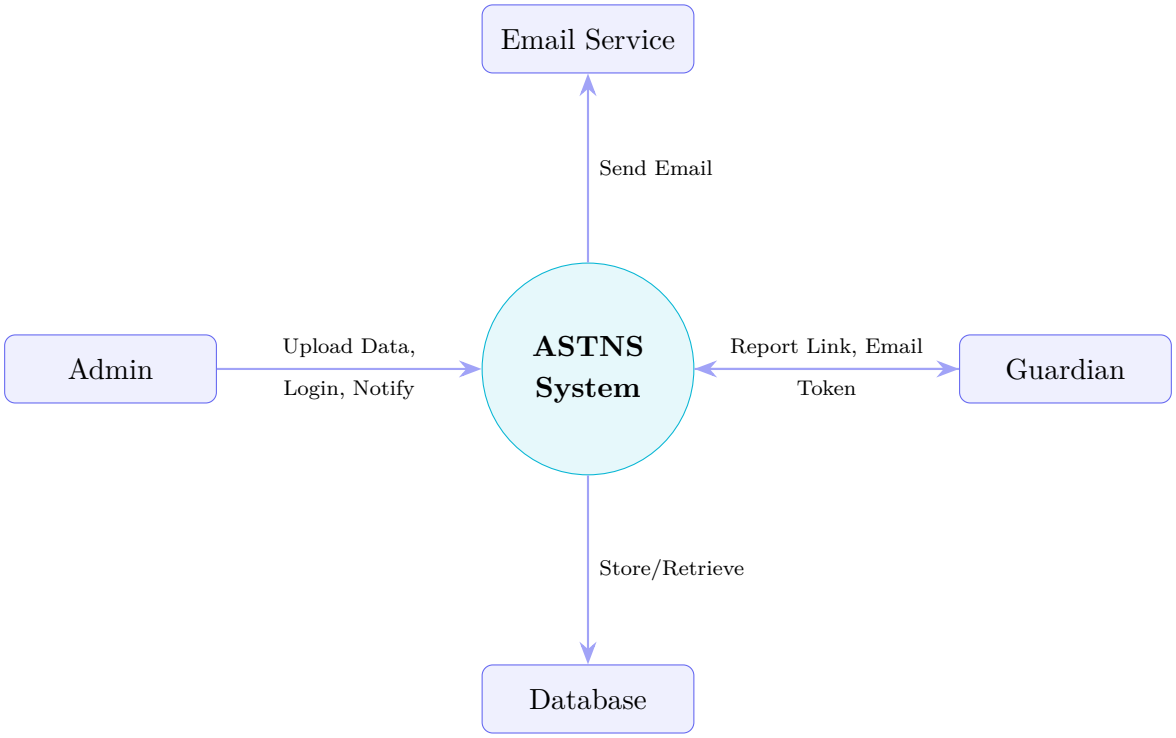

Figure 4.2: Level 0 DFD – Context Diagram

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

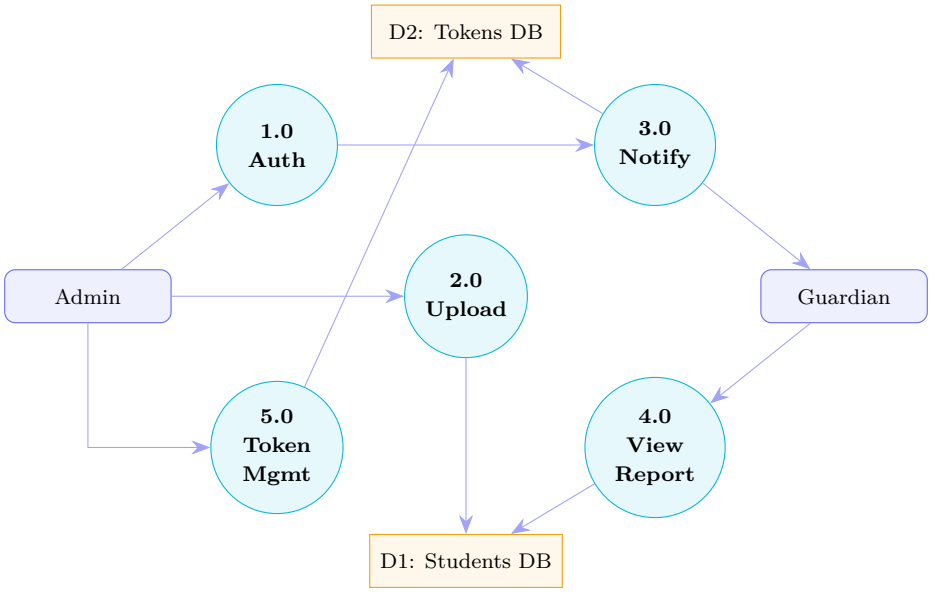

Figure 4.3: Level 1 DFD

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

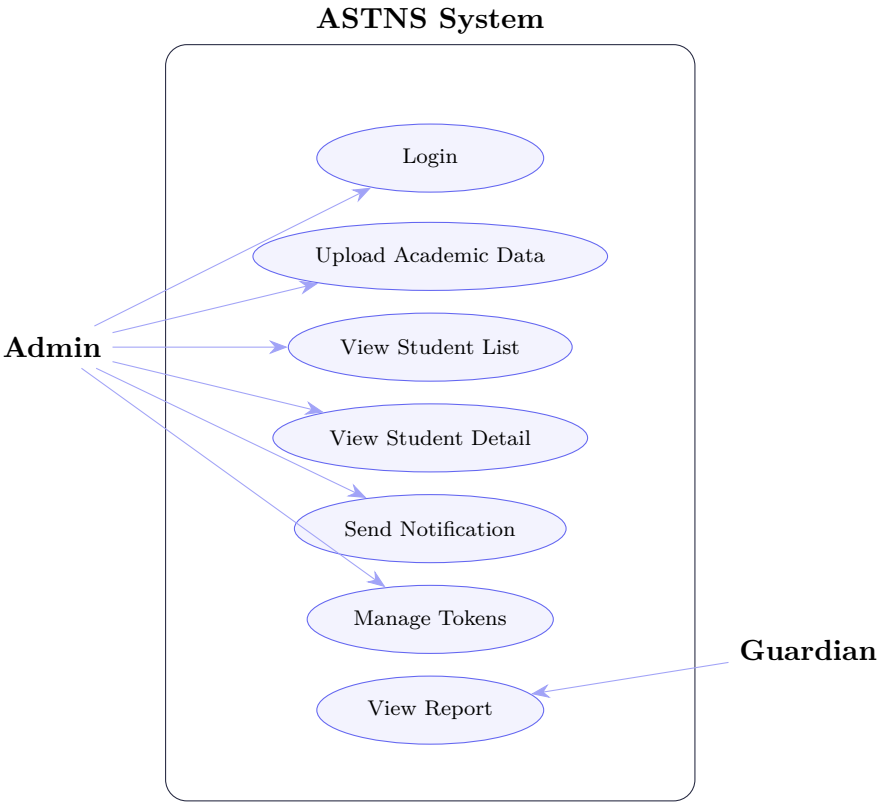

Figure 4.4: Use Case Diagram

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

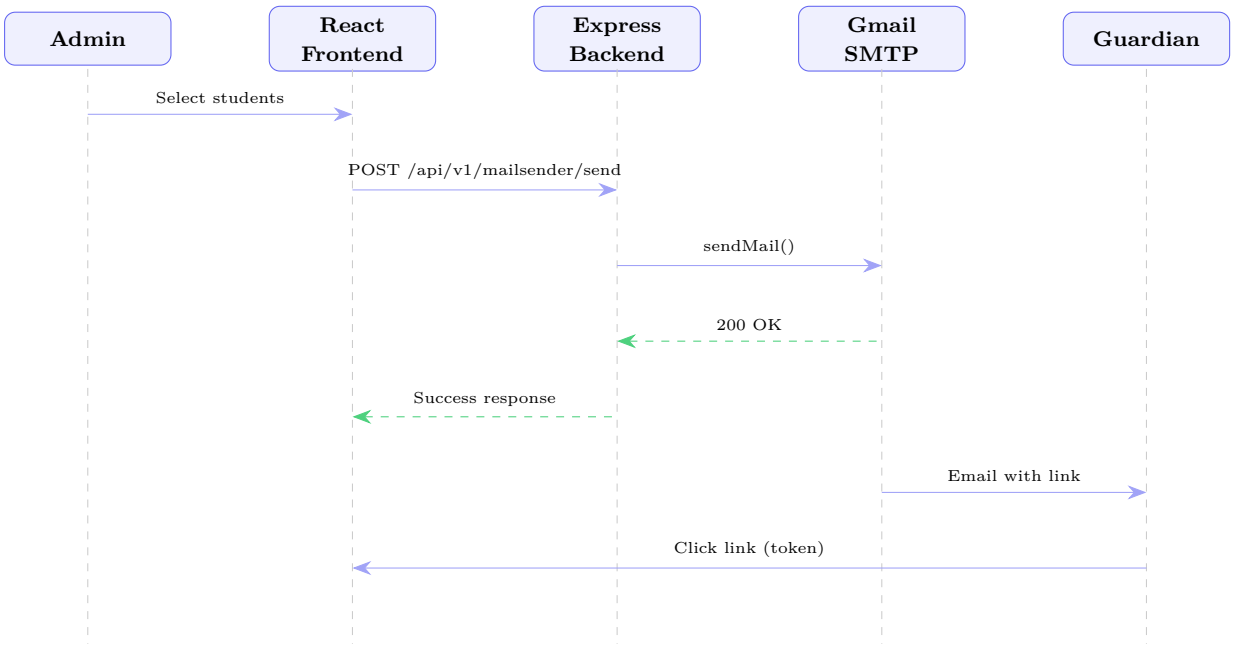

Figure 4.5: Sequence Diagram – Notification Flow

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

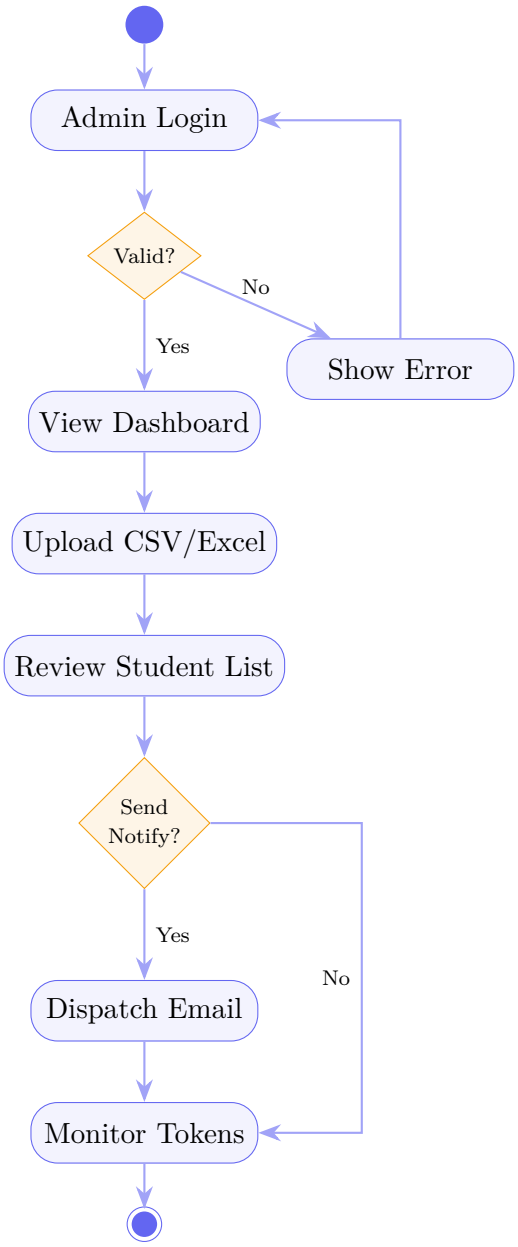

Figure 4.6: Activity Diagram – Admin Workflow

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

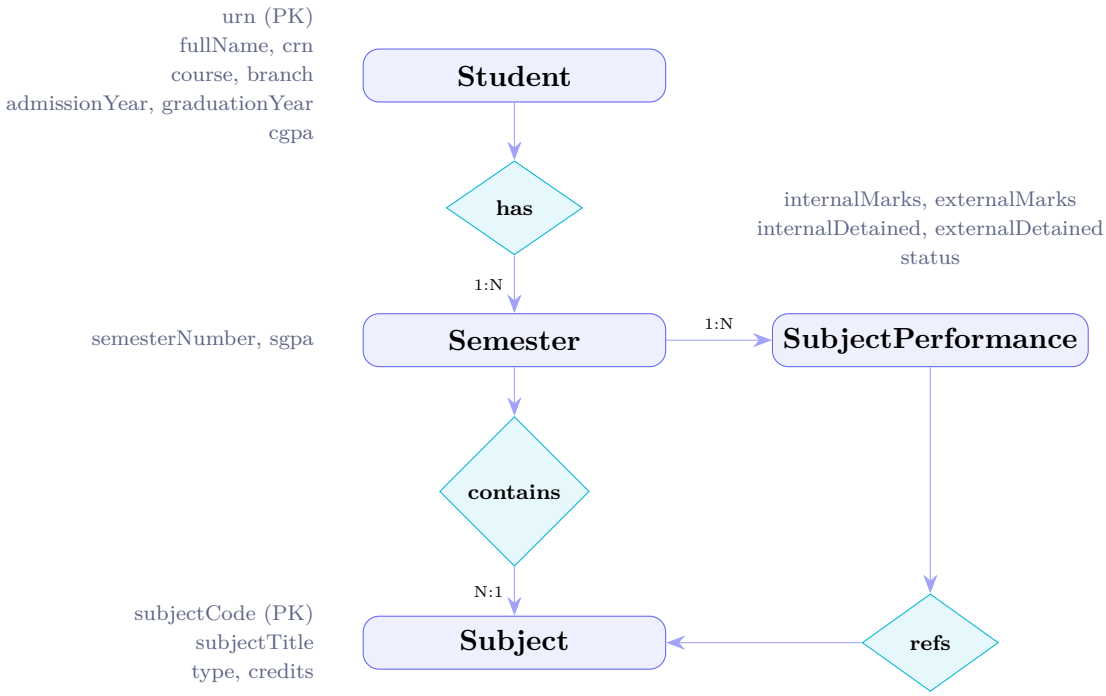

Figure 4.7: Entity-Relationship Diagram

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
guardianEmail	String	Parent/Guardian email for notifications
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	$\geq$ maxInternal + maxExternal
minInternalPassMarks	Number	Minimum to pass internally
minExternalPassMarks	Number	Minimum to pass externally
minTotalPassMarks	Number	$\leq$ maxTotalMarks

Token Collection:

Field	**Type**	**Description**
student	ObjectId	Reference to the Student document
token	String	Unique cryptographically secure token string
semester	Number	The academic semester for this report
sentVia	String	Delivery method (Email, SMS, Both)
expiresAt	Date	Timestamp after which the token is invalid
isRevoked	Boolean	Manual override to invalidate token access

4.7 Component Diagram – Frontend Architecture

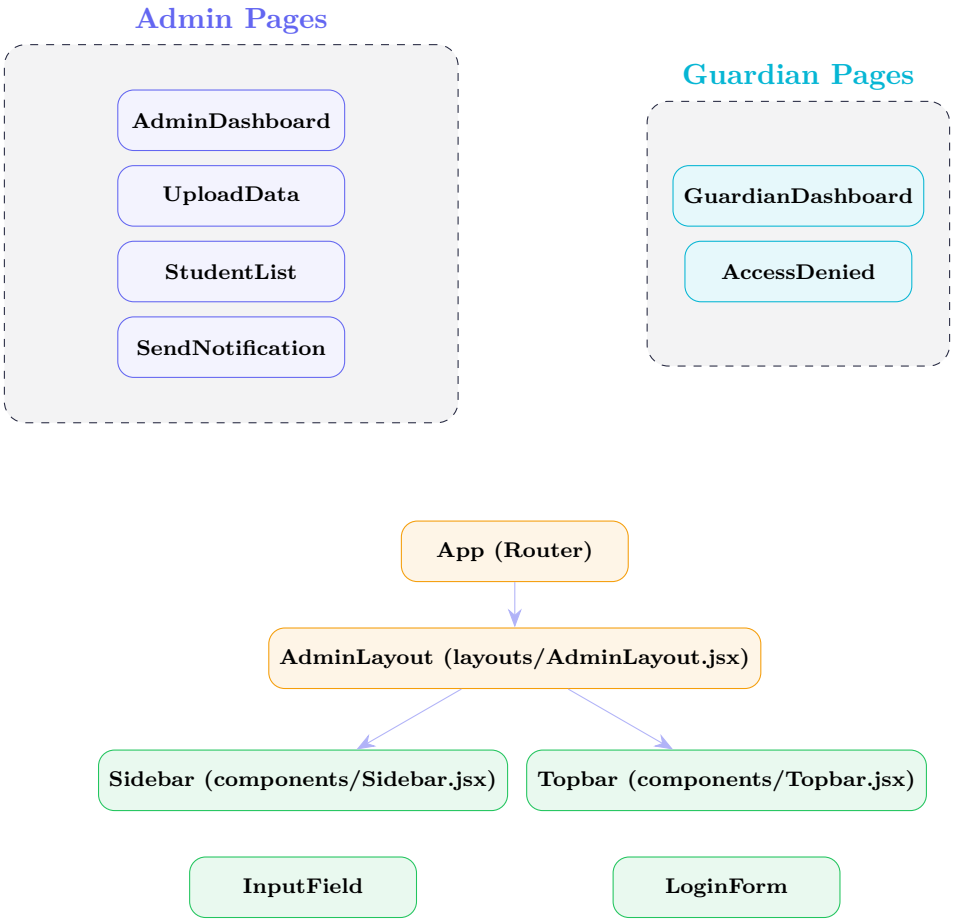

Figure 4.8: Frontend Component Architecture

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

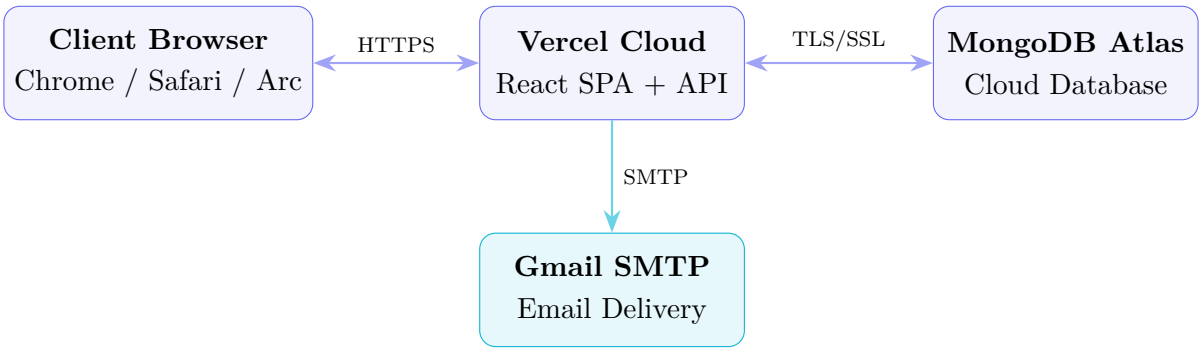

Figure 4.9: Deployment Diagram

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

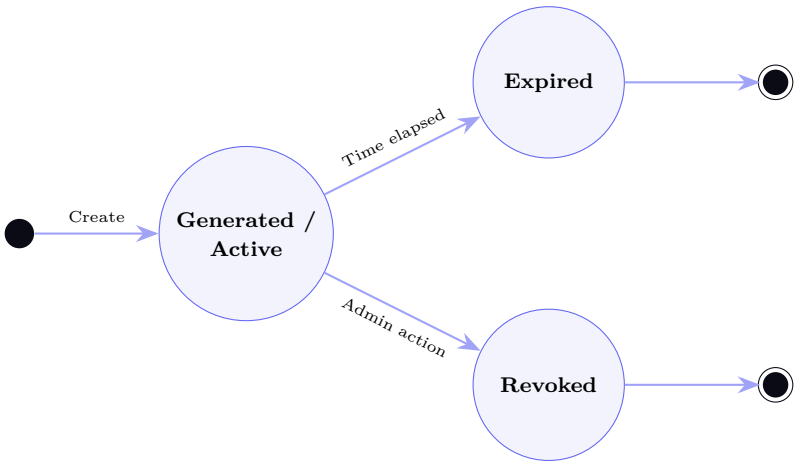

Figure 4.10: State Machine Diagram – Token Lifecycle

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

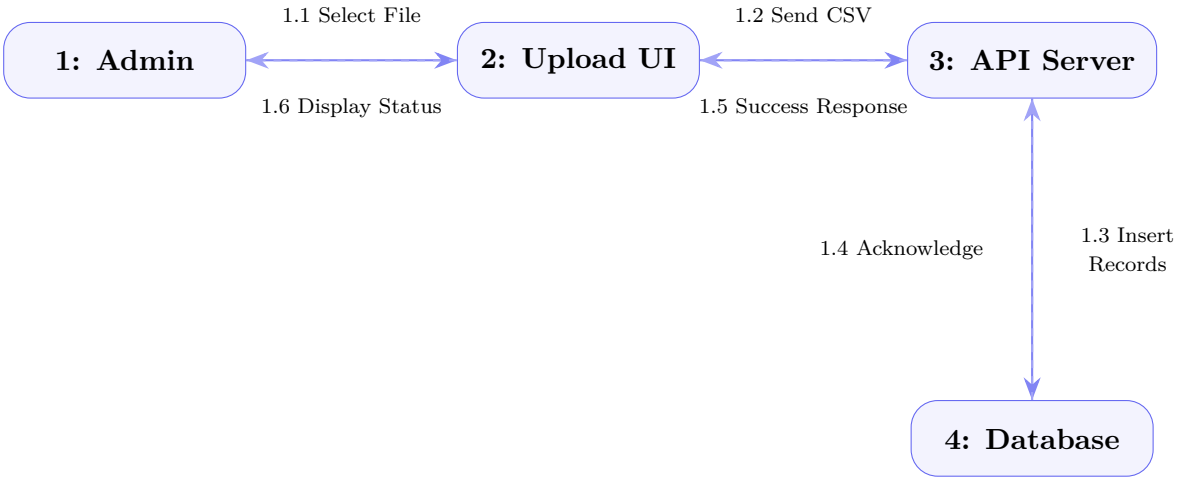

Figure 4.11: Communication Diagram – Bulk Data Upload

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 Proposed Testing Approach

As the system is currently under development, this section outlines the proposed testing techniques that will be employed to ensure the robustness, reliability, and functionality of the Academic Status Transparency Notification System (ASTNS). The system will undergo testing using a combination of manual functional testing, automated unit testing, API integration testing, and cross-browser validation.

The proposed testing strategy will cover:

1. **Unit-Level Validation**: Validating individual components such as Mongoose schema validators (e.g., marks ranges, course-branch mapping, CGPA calculation constraints).
2. **API Integration Testing**: Testing the REST API endpoints using tools like Postman or Thunder Client to ensure correct request handling and response payloads.
3. **UI/UX Functional Testing**: Performing manual browser testing on Google Chrome, Apple Safari, and Arc for cross-browser parity and user experience.
4. **Cross-Device Verification**: Verifying the responsive layout on desktop, tablet, and mobile viewports.
5. **Email Delivery Testing**: Simulating notification dispatch to verify Gmail SMTP delivery capabilities, HTML template rendering, and dynamic link mapping.

5.2 Proposed Test Cases

The following is a curated list of relevant test cases on which the system will be systematically tested during the verification phase.

ID	Test Case	Steps	Expected Outcome
TC1	Admin Login – Valid	Enter assigned admin credentials, click Submit	Successful redirection to Admin Dashboard
TC2	Admin Login – Invalid	Enter wrong credentials, click Submit	Appropriate error message displayed
TC3	Login Form Validation	Attempt to submit an empty or incomplete login form	Required field validation errors shown
TC4	Email Regex Validation	Enter malformed or non-Gmail email address	Appropriate pattern constraint error shown
TC5	Bulk CSV Upload	Upload a correctly formatted CSV/Excel data file	Academic records parsed and previewed correctly
TC6	Invalid File Upload	Attempt to upload unsupported file type (e.g., .txt)	File rejection with corresponding error alert
TC7	Student Search Filtering	Input part of a student name in the search bar	List filtered down to match search string dynamically
TC8	View Student Profile	Click on a specific student record card	Detailed profile page with accurate academic charts loads
TC9	Batch Notification Dispatch	Select multiple students and trigger Send Notification	Successful API execution with UI confirmation message
TC10	Guardian Token Validation	Access Guardian Portal via uniquely generated token link	Student semester report accurately displayed for authorized token
TC11	Invalid Token Handling	Attempt accessing portal with expired or altered token	Access Denied interface shown instead of reports
TC12	Token Revocation Process	Admin clicks Revoke for an active token in dashboard	Token status instantly updates to Revoked across system
TC13	SGPA Chart Rendering	Open student detail page containing multiple semesters	Multi-node line chart dynamically renders correct SGPA trend
TC14	Responsive Layout Check	Resize browser window to mobile width (< 768px)	Layout reflows correctly, sidebar collapses into hamburger menu
TC15	Email Template Rendering	Guardian receives dispatched notification email	Branded HTML email loads properly with clickable CTA button

Chapter 6

Performance of the Project Developed

6.1 Development Progress and Milestones

As of the midterm evaluation, the core foundational architecture of the Academic Status Transparency Notification System (ASTNS) has been established. Our primary focus thus far has been to secure the backend infrastructure and prepare the frontend layout necessary for data ingestion and user interaction.

The following modules have been substantially developed and are currently undergoing internal testing:

- **Authentication Module**: Admin login interface with robust **Axios**-based session management and route protection is functional.
- **Database Schema**: Mongoose models for Students, Semesters, Subjects, Performance, and **Secure Tokens** have been deployed to MongoDB Atlas.
- **Frontend Layouts**: The overarching React application architecture is fully integrated using Tailwind CSS, featuring a responsive Sidebar with logout confirmation.
- **Automated Email Notification Pipeline**: Integration with Gmail SMTP via Nodemailer for batch notification dispatch is operational.
- **Secure Token Generation System**: Cryptographic token generation and lifecycle management (expiry/revocation) are fully implemented.

6.2 Remaining Requirements for Finalization

To transition the project from its current midterm state to a fully finalized application, several critical features and integrations must be completed:

1. **Robust Bulk Data Processing Integration**: Finalising the UI-to-Backend mapping for all student fields (including guardian emails) across complex multi-semester datasets.
2. **Data Visualization Polishing**: Enhancing Recharts integration to provide even more granular academic insights on the Guardian Dashboard.
3. **Comprehensive Role-Based Access Control Audit**: Performing a final security audit on server-side validations to ensure absolute data isolation.
4. **Production Deployment**: Finalising Vercel production environment configurations and custom domain mapping.

6.3 Anticipated Ongoing Challenges

During the subsequent development phases, We anticipate the following technical challenges that will require mitigation:

- **Performance at Scale**: Ensuring the database querying and frontend rendering remain performant when simultaneously loading hundreds of detailed student profiles.
- **Email Delivery Reliability**: Guaranteeing that automated midterm and end-semester notifications bypass strict parental spam filters and bounce-backs are handled gracefully.
- **Asynchronous Error Handling**: Providing clear, actionable error messages to the administrator if a batch upload fails partially (e.g., if 5 out of 100 rows in a CSV are corrupted) rather than failing the entire process silently.

Chapter 7

Output Screens

1. **Login Page** – Dark-themed admin login with email/password fields, React Hook Form validation, gradient logo, and grid background.

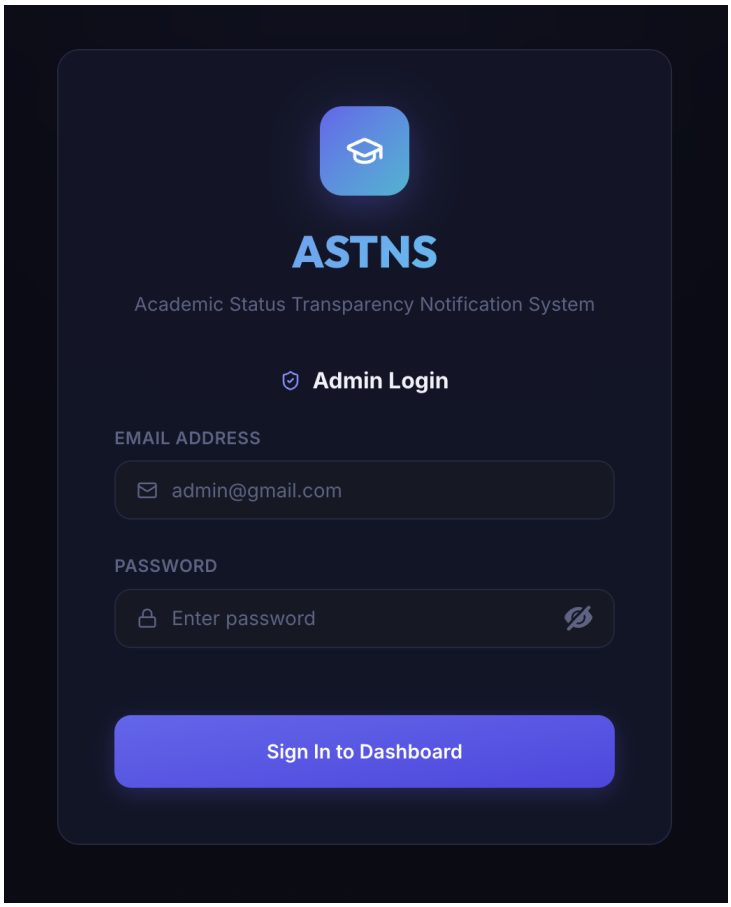

Figure 7.1: Login Page

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.

Figure 7.2: Admin Dashboard

3. **Upload Data** – Course/branch/semester selection dropdowns, drag-and-drop CSV/Excel upload zone, parsed data preview table.

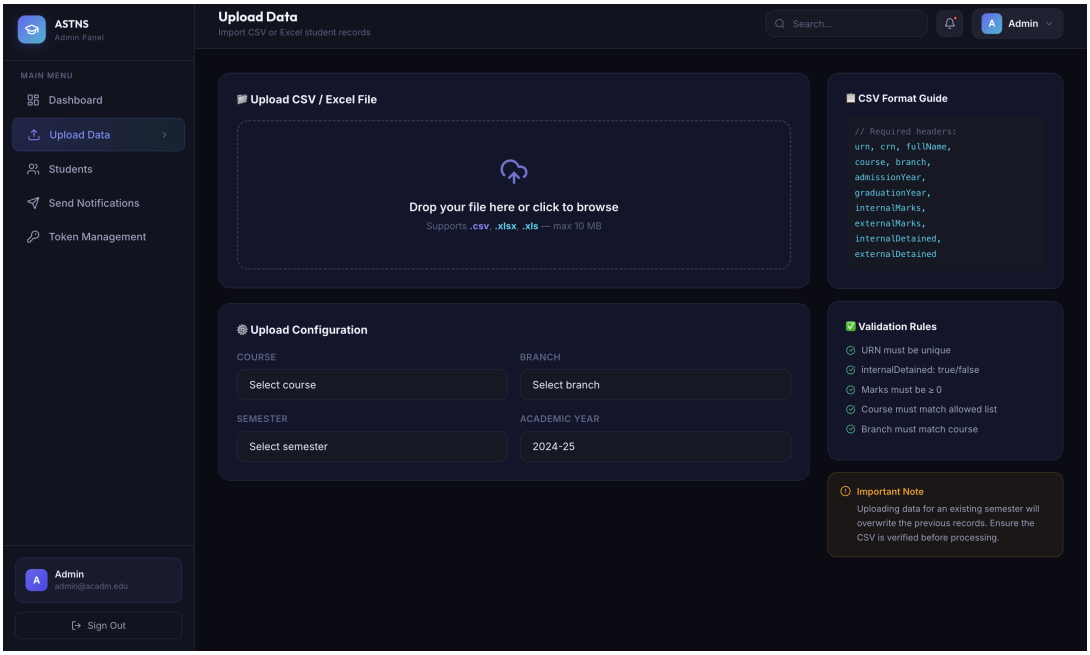

Figure 7.3: Upload Data

4. **Student List** – Searchable grid/list view with SGPA colour coding, branch filter, click-to-navigate to detail.

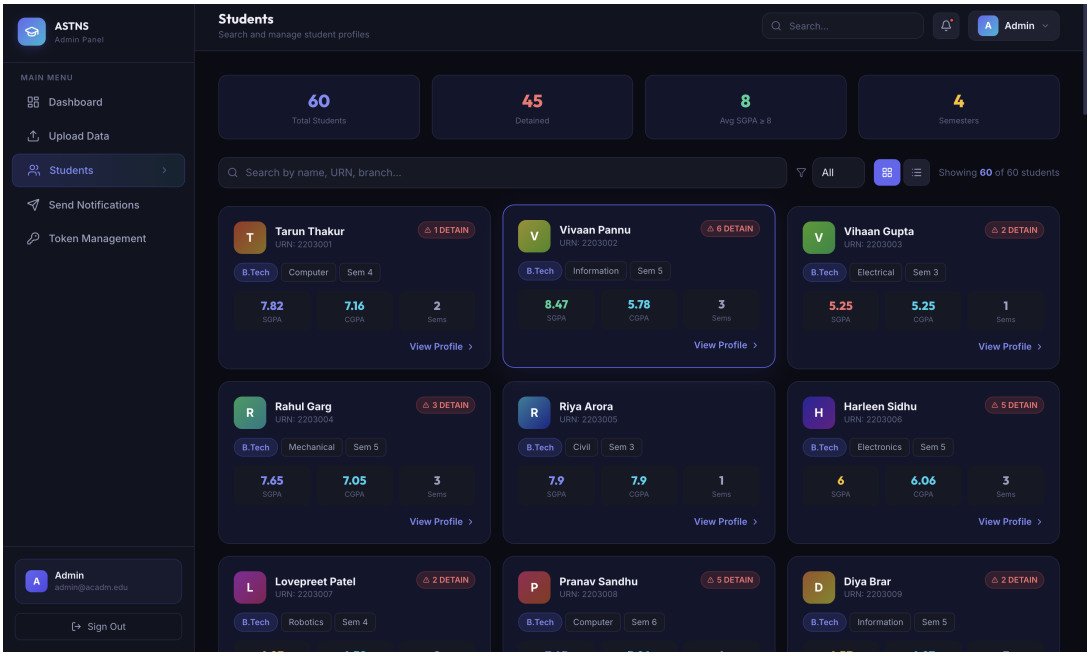

Figure 7.4: Student List

5. **Student Detail** – Student profile header, SGPA trend line chart, semester-wise collapsible tables showing internal/external marks and detention flags.

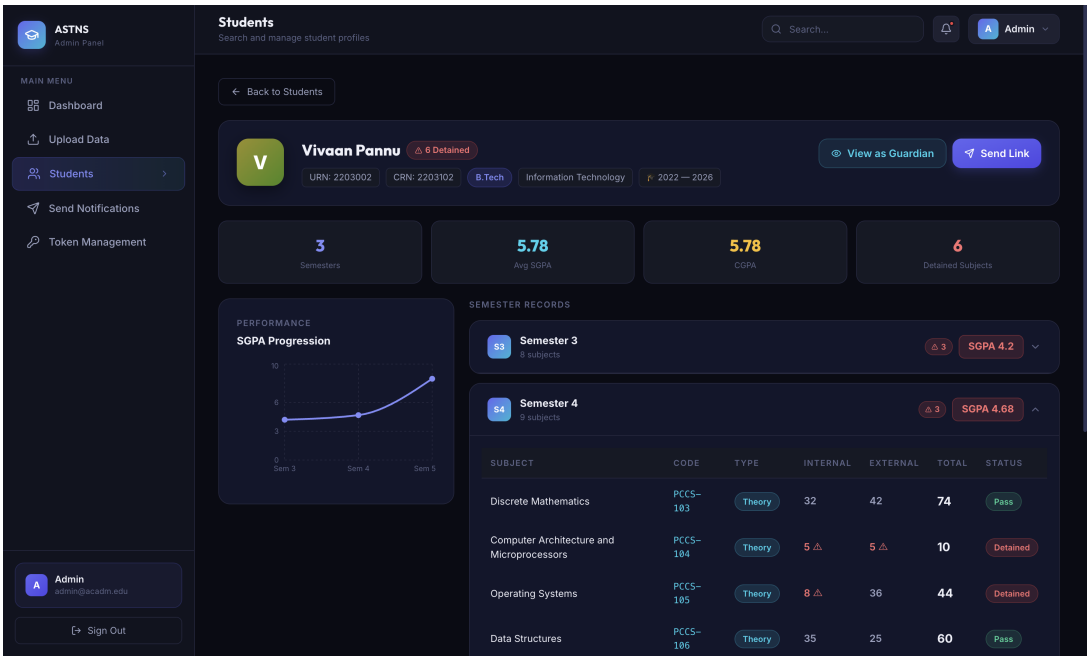

Figure 7.5: Student Detail

6. **Send Notification** – Multi-select student table, dispatch button with success/failure feedback.

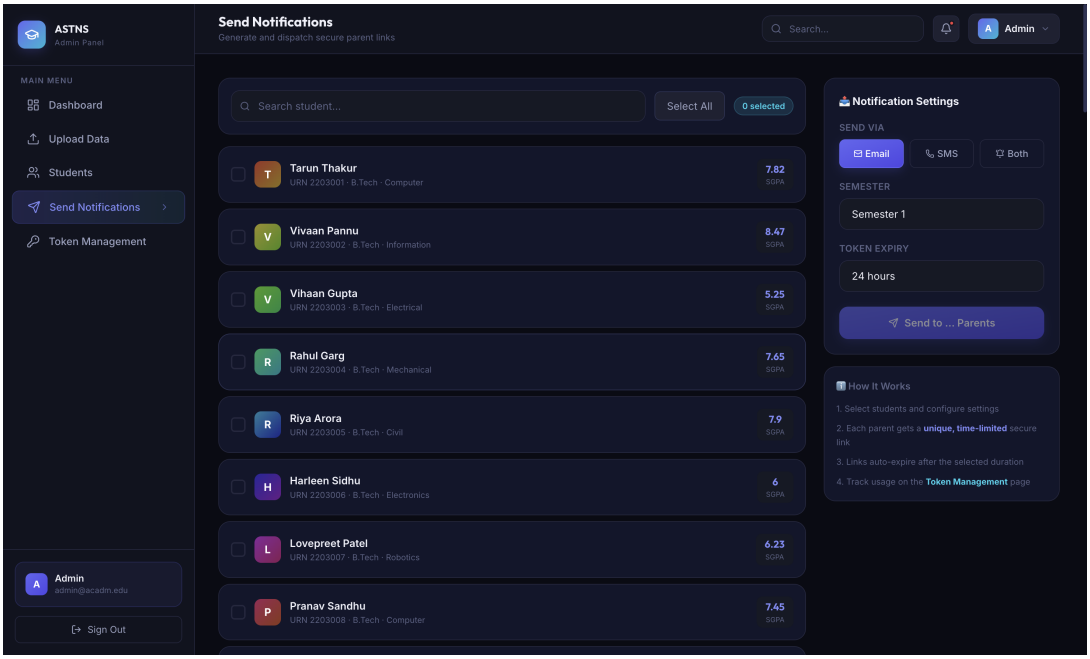

Figure 7.6: Send Notification

7. **Token Management** – Token table with status badges (Active/Expired/Revoked), access method tags, revoke and copy-link actions.

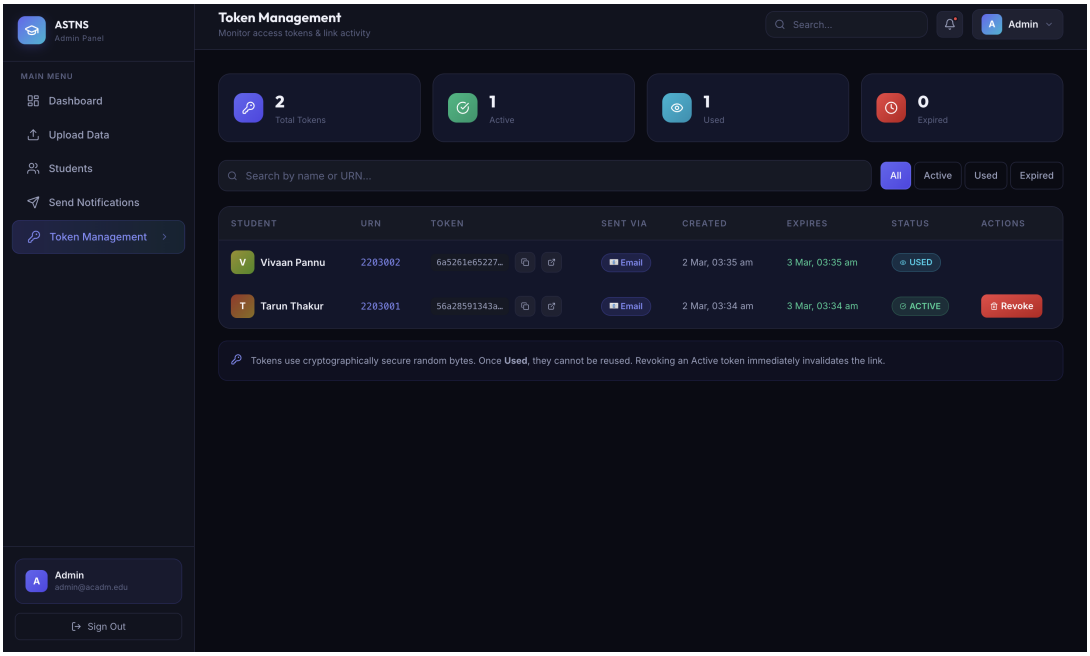

Figure 7.7: Token Management

8. **Guardian Dashboard** – Token-authenticated view with student summary, SGPA chart, semester breakdown tables, attendance ring, and download report button.

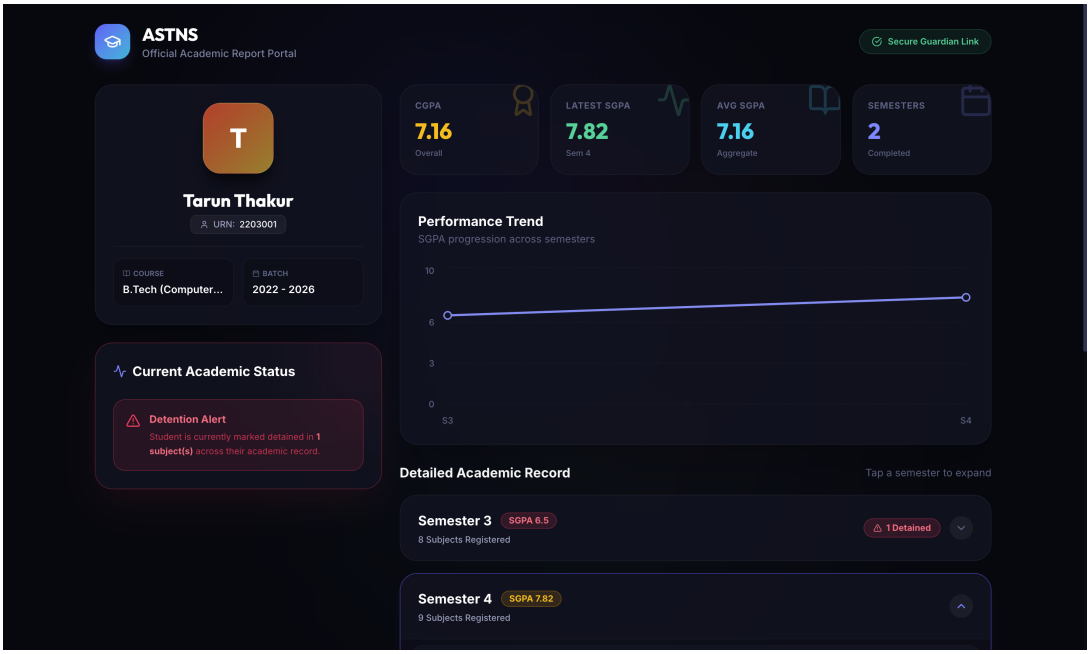

Figure 7.8: Guardian Dashboard

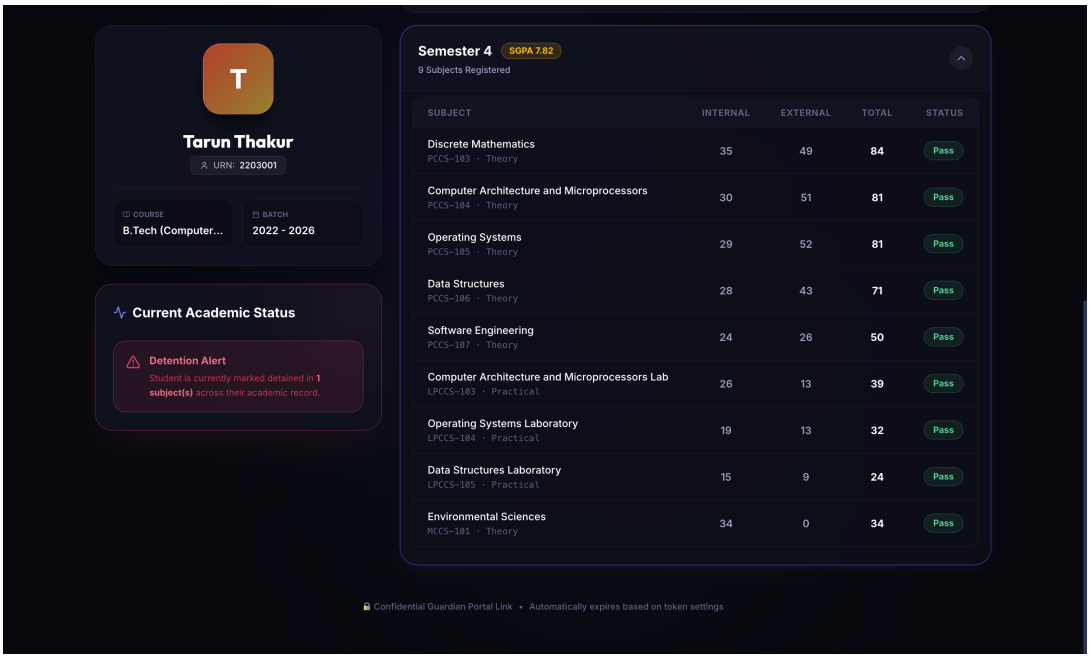

Figure 7.9: Guardian Dashboard

9. **Access Denied Page** – Shown when token is invalid/expired/revoked.

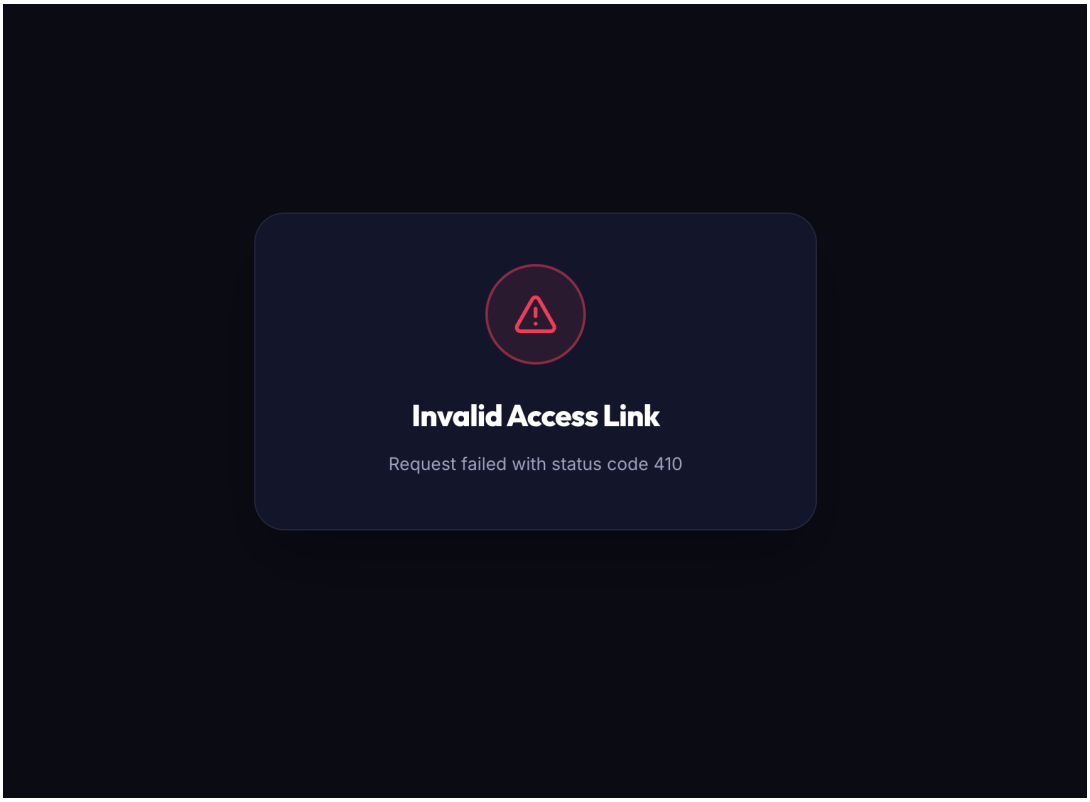

Figure 7.10: Access Denied Page

10. **Email Template** – Branded HTML email with "View Report" CTA button.

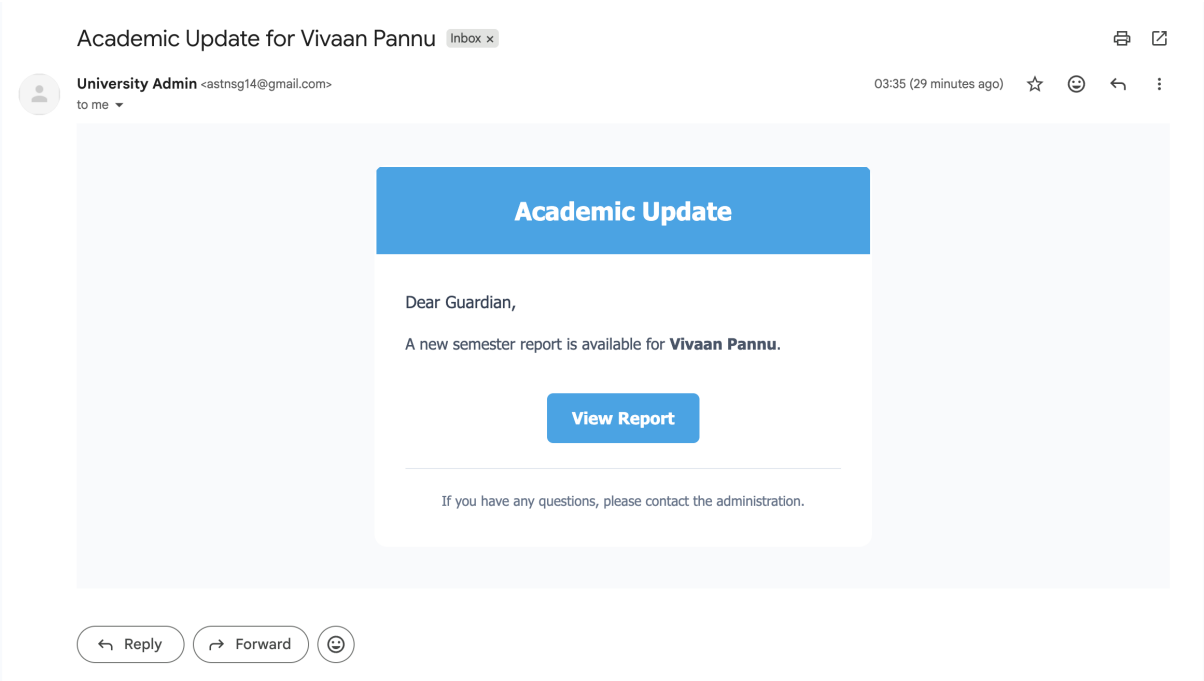

Figure 7.11: Email Template

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.