Student Academic Portal

1. Project Overview

A centralized academic management system designed to streamline the flow of information between the administration and students. The portal serves as a one-stop destination for students to access critical academic data, including examination results, class schedules, and official announcements, while giving administrators a structured interface to disseminate information.

Tech Stack

- **Frontend:** React.js
- **Backend:** Spring Boot (Java)
- **Database:** PostgreSQL
- **Version Control:** Git
- **Authentication:** JWT-based authentication

2. Problem Statement

In many educational institutions, academic information is fragmented across various mediums—physical notice boards, disparate websites, and manual spreadsheet distributions. This leads to information delays, confusion regarding class schedules, and privacy concerns regarding public display of results.

Many traditional systems:

- Lack a unified interface for all academic needs.
- Rely on manual updates for timetables and notices.
- Do not provide secure, private access to individual student grades.
- Lack a structured history of past announcements.

This project addresses these limitations by building a **Student Academic Portal** that digitizes these processes, ensuring secure access to personal records, real-time updates for schedules, and a reliable channel for official communication.

3. Functional Requirements

3.1 User Management

- **Student Login:** Secure access using unique Student IDs and passwords.
- **Admin Login:** Restricted access for faculty/admin to post updates.
- **Role-Based Access:** Strict separation between Student (Read-Only) and Admin (Write/Edit) privileges.
- **Session Management:** Secure logout and token expiration handling.

3.2 Result Management

- **View Results:** Students can view their marks, grades, and GPA for current and past semesters.
- **Data Privacy:** Results are fetched securely and are visible only to the authenticated student.
- **Download:** Option to print or download a provisional marksheet.

3.3 Announcement System (Admin-Controlled)

- **Post Notices:** Admins can create, edit, and delete announcements.
- **Categorization:** Notices tagged by type (General, Exam, Holiday, Urgent).
- **Real-time View:** Students see the latest announcements immediately upon login.

3.4 Timetable Management

- **Weekly Schedule:** Structured view of classes, subjects, and room numbers.
- **Dynamic Updates:** Reflects changes in the schedule (e.g., rescheduled classes) immediately.
- **Filtering:** View timetable by day or subject.

4. Non-Functional Requirements

4.1 Security

- **Authentication:** JWT (JSON Web Tokens) for stateless, secure API communication.
- **Authorization:** Middleware to ensure students cannot post notices or change grades.
- **Data Protection:** Password hashing (BCrypt) and secure database connections.

4.2 UI/UX (Structured UI)

- **Component-Based Design:** React components for a modular and consistent look and feel.
- **Responsiveness:** Accessible on mobile devices, tablets, and desktops.
- **Intuitive Navigation:** distinct sections for Results, Notices, and Timetable.

4.3 Maintainability (Git-based Versioning)

- **Version Control:** Entire codebase managed via Git with proper commit history.
- **Branching Strategy:** Feature branches for distinct modules (e.g., feature/results, feature/auth).
- **Code Structure:** Clean separation of concerns in Spring Boot (Controller-Service-Repository).

4.4 Reliability & Performance

- **Data Integrity:** PostgreSQL ensures ACID compliance for critical grade data.
- **Scalability:** Spring Boot architecture supports high concurrent user loads during result declaration.

5. High-Level Design (HLD)

5.1 System Architecture Overview

The system utilizes a standard **Three-Tier Architecture**:

- **Presentation Layer:** React Frontend handles the UI and user interactions.
- **Application Layer:** Spring Boot Backend contains the business logic and REST APIs.
- **Data Layer:** PostgreSQL stores all student data, results, and content.

5.2 Data Flow at High Level

1. Student logs in via React interface.
2. Frontend sends credentials to Spring Boot Auth Service.
3. Upon validation, a JWT is returned to the client.
4. Student requests specific data (e.g., "View Timetable").
5. Spring Boot fetches data from PostgreSQL and returns JSON.
6. React renders the data into a structured table/card view.

6. Low-Level Design (LLD)

6.1 Backend Layer Design

- **Controller Layer:** Handles HTTP requests (/api/results, /api/notices).
- **Service Layer:** Contains business logic (e.g., GPA calculation, formatting dates).
- **Repository Layer:** Uses Spring Data JPA to interact with PostgreSQL.

6.2 Database Design (Detailed)

Users Table

- user_id (PK)

- username (Student ID)
- password_hash
- role (STUDENT, ADMIN)
- full_name

Results Table

- result_id (PK)
- student_id (FK)
- subject_code
- subject_name
- marks_obtained
- semester

Announcements Table

- notice_id (PK)
- title
- content
- posted_date
- posted_by (Admin ID)

Timetable Table

- slot_id (PK)
- day_of_week
- start_time
- end_time
- subject_name
- room_number

7. API Design

Authentication APIs

- POST /api/auth/login - Authenticate user and return JWT.

Result APIs

- GET /api/results/my-results - Fetch logged-in student's grades.
- GET /api/results/history - Fetch past semesters.

Announcement APIs

- GET /api/announcements - Public feed of notices.
- POST /api/announcements - Create new notice (Admin only).
- DELETE /api/announcements/{id} - Remove notice (Admin only).

Timetable APIs

- GET /api/timetable - Fetch weekly schedule.

8. Module Division

Frontend Modules (React)

1. **Auth Module:** Login page with validation.
2. **Dashboard Module:** Landing page showing latest 3 notices and quick links.
3. **Result Module:** Table view for marks with semester filtering.
4. **Timetable Module:** Grid view for weekly classes.
5. **Admin Panel:** Form to post/delete announcements (conditionally rendered).

Backend Modules (Spring Boot)

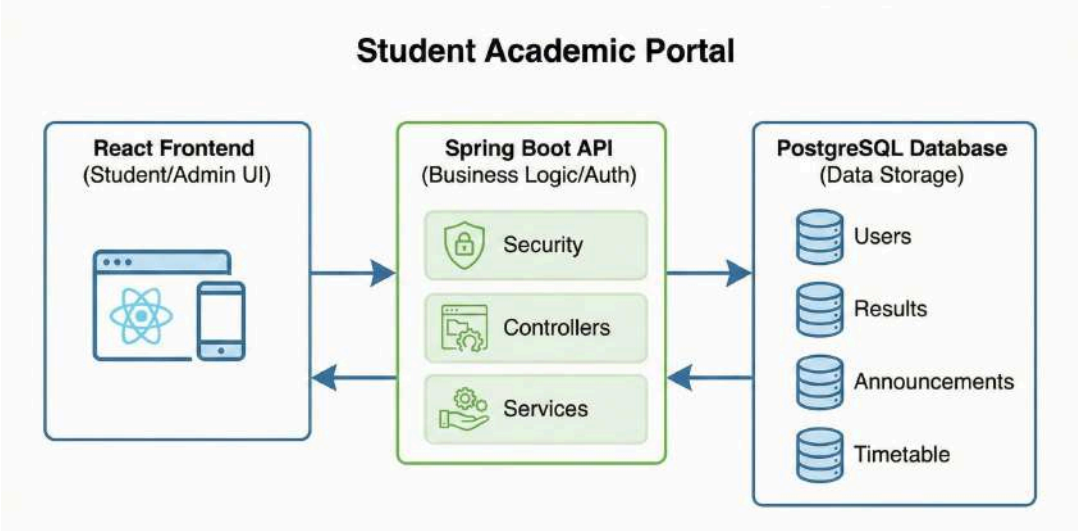
1. **Security Module:** Spring Security config, JWT Filter.
2. **User Service:** User authentication and profile data.
3. **Academic Service:** Logic for fetching and processing results/timetables.
4. **Notice Board Service:** CRUD operations for announcements.

9. Use Case Flow (Example)

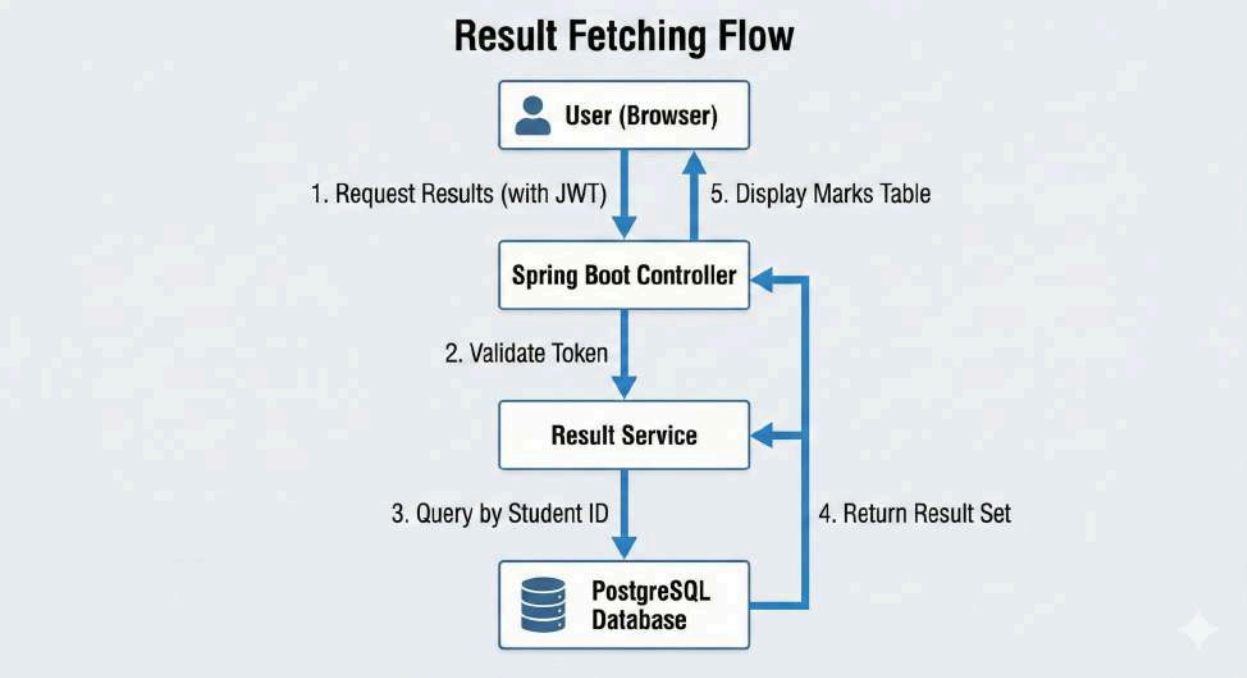
1. **Login:** Student enters ID/Password -> System verifies against Users table -> Returns Token.
2. **Check Result:** User clicks "Results" -> React calls GET /api/results/my-results with Token -> Backend queries Results table where student_id matches token -> Data displayed in tabular format.
3. **View Notice:** User sees "Exam Postponed" on Dashboard -> Click for details -> React displays full content from Announcements table.

10. Architecture Diagrams

10.1 High-Level Architecture



10.2 Result Fetching Flow



11. Evaluation Criteria Mapping

- **Structured UI:** Implemented via React's component hierarchy (Header, Sidebar, Content Area).
- **Admin-Controlled Notices:** Secured via Spring Security (Role-based endpoints) and stored in PostgreSQL.
- **Git-based Versioning:** Project structure maintained in a Git repository with clear commit history (e.g., "Added Auth Module", "Fixed Timetable Bug").

12. Future Scope

- **Push Notifications:** Integration with Firebase/Email for instant notice alerts.
- **Attendance Tracking:** Visual graphs showing student attendance percentages.
- **Fee Payment Gateway:** Integration for paying semester fees online.
- **Parent Portal:** Separate login for parents to view ward's progress.

13. Conclusion

The **Student Academic Portal** effectively solves the problem of decentralized academic information. By leveraging **React** for a responsive UI, **Spring Boot** for robust backend logic, and **PostgreSQL** for reliable data storage, the system ensures that students have timely, secure, and organized access to their academic life.