Bobby - System architecture

Arun, Ganga

September 2025

1 School Management System Architecture

The School Management System is a Django-based web application designed to manage students, teachers, classes, timetables, announcements, and academic records. The system implements role-based access control, where each participant (Admin, Teacher, Student) interacts with the system through a web interface. All communication occurs over standard HTTP/HTTPS, with JSON used for asynchronous operations such as voting on announcements. The system leverages Django sessions to maintain authentication and authorization state.

The architecture is modular, separating concerns into dedicated Django apps and views. Each module handles specific functionality, such as student registration, timetable management, announcements, and marks management. Data is stored in a relational database (e.g., SQLite, PostgreSQL) using Django's ORM. Security-critical modules, such as CAPTCHA validation and file upload handling, incorporate measures to prevent automated attacks, although some vulnerabilities are intentionally included for learning purposes.

1.1 Roles and Participants

1.1.1 Administrator (Admin)

The Admin is responsible for managing the overall school system. Admins have the highest level of access and can perform the following actions:

- Add, edit, or delete subjects, teachers, classrooms, and timeslots.
- Generate the class timetable based on teacher availability, subjects, and class requirements.
- View all timetable entries with filters for class, subject, teacher, and day.
- Manage student academic records, including marks and grades.
- Monitor announcements, registration data, and system configurations.
- Review system logs and handle error reports.

The Admin interacts with the system via the web interface and is authenticated using the `Admin_login` session decorator. All requests are verified against the session to ensure authorized access.

1.1.2 Teacher

Teachers are responsible for delivering lessons, managing marks, and interacting with student-related data. Their interactions include:

- Viewing their assigned timetable entries for specific days.
- Accessing subjects and class information relevant to their teaching assignments.
- Posting announcements that students can view and vote on.
- Uploading documents or resources, with file type and size validations.
- Recording student marks, assessments, and exam scores in the system.
- Handling errors in form submissions and logging any critical events.

Teachers are authenticated using the `Teacher_login` session decorator. They only see data related to their assignments, ensuring separation of concerns and data privacy.

1.1.3 Student

Students are the end users who consume educational content, track academic performance, and interact with the system. Student actions include:

- Registering via a web form with CAPTCHA validation to prevent automated submissions.
- Viewing their class timetable and filtering entries by day.
- Viewing teacher announcements and voting via upvote/downvote mechanisms.
- Tracking personal information, class level, timetable assignments, and academic records.
- Accessing marks and grades assigned by teachers.
- Receiving notifications on errors or invalid submissions.

Students are authenticated using the `Student_login` session decorator. Access is restricted to data relevant to their class and assignments.

1.2 Workflow and Data Flow

The system operates in the following structured workflows:

- **Registration:** Students and Teachers submit registration forms, validated with CAPTCHA and optional file uploads. Errors during registration are logged and reported. Validated data is stored in the corresponding `StudentReg` or `TeacherReg` tables.
- **Authentication:** Upon login, the system sets session variables to identify the user role (Admin, Teacher, Student) and restrict access to views accordingly. Failed login attempts are logged.
- **Timetable Generation:** Admin adds subjects, teachers, rooms, and timeslots, then generates the timetable using an algorithm that assigns teachers and rooms to periods. Conflicts are detected, reported, and logged.
- **Timetable Viewing:** Students and Teachers fetch timetable entries filtered by their class or assigned schedule. ORM errors or invalid queries are logged.
- **Marks Management:** Teachers enter marks for students for assignments, exams, or assessments. Admins can review or update marks as needed. Students can view their marks and calculate performance summaries. Errors in marks entry are validated and logged.
- **Announcements:** Teachers post announcements. Students view announcements and cast votes, which are tracked in the `AnnouncementVote` table. JSON is used for asynchronous vote updates. Invalid vote submissions are logged.
- **Data Integrity and Security:** The system enforces session-based access control. Modules implement additional validation (CAPTCHA, file type checks). Certain vulnerabilities are intentionally included for educational purposes (CAPTCHA replay, duplicate voting). All exceptions and critical operations are logged for auditing.

1.3 System Communication and Integration

All modules communicate internally through Django's ORM and request-response cycle. Asynchronous operations, such as voting on announcements, use AJAX with JSON payloads. Session management ensures role-specific access control.

- Admin, Teacher, and Student communicate via HTTP requests to Django views.
- JSON is used for client-server interactions in interactive features (e.g., voting, timetable updates, marks submission).

- ORM queries ensure referential integrity across models like `Student`, `Teacher`, `TimetableEntry`, `AnnouncementVote`, and `Marks`.
- Security-sensitive operations, including registration, voting, and marks updates, are logged and validated against session credentials.
- Exception handling is implemented across modules to ensure errors are captured, reported, and logged for auditing purposes.

1.4 Security Considerations

The system-level security considerations include:

- Role-based access control using session decorators (`Admin_login`, `Teacher_login`, `Student_login`).
- Input validation for forms, file uploads, and academic records.
- CAPTCHA validation for automated bot prevention (with intentional replay vulnerability for study purposes).
- Vote integrity concerns, as duplicate votes are currently possible (educational vulnerability).
- Data privacy, ensuring students only access their own marks and timetables.
- Optimized ORM queries to prevent unintentional data leaks.
- Logging of critical operations, errors, and exceptions for auditing and accountability.
- Error handling mechanisms across workflows to provide clear feedback and maintain system stability.