Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Project Kit

Title of the Project

Web Based School Admin System

Abstract of the project

The Web Based School Administration System is a robust, multi-functional platform that serves as a one-stop solution for managing the operational and academic functions of educational institutions. In today's digital age, schools face the challenge of efficiently handling an ever-growing amount of data related to students, staff, academics, and activities. This system is developed to address these challenges by automating repetitive tasks, improving communication, and offering real-time access to critical information.

The core functionalities of the system include:

1. **Student Management:** This module manages the entire lifecycle of students, from admissions to graduation.
2. **Attendance Tracking:** The system automates attendance recording for students and staff, ensuring accuracy and providing real-time attendance reports.
3. **Grade and Exam Management:** Teachers can use the system to input and track student grades and create performance analytics. The system can integrate exam scheduling, ensuring smooth coordination of test administration and result publication.
4. **Timetable and Schedule Management:** This feature automates the creation of class schedules and timetables, optimizing room usage and teacher allocation. The system can generate alternative schedules in case of emergencies or teacher absences.

5. **Staff and Payroll Management:** The system offers HR functionalities for managing teacher and staff profiles, tracking performance, and automating payroll processes, including leave management and salary calculation.
6. **Fee Management:** The system automates fee collection, tracks pending dues, and generates receipts. It supports multiple payment methods and integrates with financial systems for better financial management.

Keywords

Generic Keywords

Databases, Middleware, Programming

Specific Technology Keywords

C++, Node.js, MongoDB, ReactJS, HTML, CSS, JS

Project Type keywords

Analysis, Design, Implementation, Testing, Graphical User Interface

Functional Components of School Management System

The following is a list of functionality for the school management system project. More functionalities that you find appropriate can be added to this list. In places where the description of functionality is not adequate, you can make assumptions and proceed.

Users of the System:

There are two types of users:

- **Students:** Regular users who can access their personal academic data, attendance, assignments, and other related information.
- **Administrators:** Super users who manage student information, update system data, and handle the backend operations such as course management, faculty allocation, etc.

When the user types in the URL of the school management system website, a Welcome page is displayed with a menu on the left-hand side, a banner at the top, and related links or announcements. Users need to log in to access the system features. The login functionality checks user authenticity from the database.

The menu should contain the following screens:

1. Registration Screen

If the user (student) is not registered, the registration screen should be available. Students can fill in their details such as name, student ID, email, and password to create an account. The system validates the information and stores it in the database.

2. Student Profile

This screen allows students to view and edit their personal details, such as contact information, profile picture, and emergency contacts. It also displays a summary of the student's academic information, including enrolled courses, grades, and attendance records.

3. Course Management

This screen will display the list of available courses and subjects for the academic term. Students can view details like course schedules, credits, and faculty information.

Administrators can add, update, or delete courses. Faculty assignments and changes to course structure are managed by the admin.

4. Timetable Management

This screen displays the daily and weekly timetable for each student based on their enrolled courses.

Administrators can update the timetable in case of class rescheduling, exams, or holidays.

5. Attendance Tracking

This screen allows students to track their daily, weekly, and monthly attendance records. It shows total days present, absent, and any late entries.

Administrators or faculty members can update attendance records for their respective classes**.**

6. Assignment Submissions

Students can view, download, and submit assignments for each course from this screen.

After submitting, they can view feedback and grades provided by the teacher.

Administrators can manage assignment deadlines and ensure faculty are able to upload assignments and provide grades.

7. Exam and Results Management

This screen displays the exam schedule for the students based on their courses.

After exams, students can check their results here, including individual subject marks, total grades, and GPA.

The admin can update exam dates, results, and generate detailed performance reports.

8. Fees Management

Students can view their fee structure, pending dues, and payment history.

The admin manages fee updates, records payments, and handles fee-related notifications to students.

9. Catalog Information (Student Dashboard)

This screen shows a summary of all student-related activities: current courses, exam results, attendance, assignments, and fees due.

A personalized dashboard helps students stay updated on their academic progress.

10. Library Management

This screen shows available books and resources in the school library.

Students can check out books, view return dates, and search for books using keywords.

Administrators can manage inventory, issue/return logs, and overdue fees.

11. Events and Announcements

This screen lists upcoming school events, holidays, and important announcements. Students can view the calendar and register for school events. Administrators can create, modify, and update events, ensuring timely notifications to all users.

12. Communication and Messaging

Students can use the system's internal messaging feature to communicate with their teachers or school administrators. Administrators can use this feature to send school-wide announcements or communicate individually with students and faculty.

13. Help and Support

Each screen will have a help link that redirects users to the help section, providing answers to common questions and troubleshooting advice. A contact form for support queries can also be included for students to request assistance from the admin.

14. Terms and Conditions

This screen contains a brief text on the website explaining the terms and conditions for using the school management system, with an option to download or print a copy.

15. Contact Information

This screen provides contact information for the school administration, including office addresses, phone numbers, and email addresses for different departments (e.g., Admissions, Accounts, Technical Support).

Additional Functionalities (Optional):

- **Parent Portal:** A separate login for parents to track their child's progress, attendance, and fee payments.
- **Mobile Compatibility:** The system could be optimized for mobile devices so students and staff can access it from smartphones and tablets.

Steps to Start-Off the Project

Platform Choice:

- **Frontend:** The system will be developed using React.js, a popular JavaScript library for building user interfaces.
- **Backend: The system will use Node.js for the backend, leveraging frameworks like Express.js for building RESTful APIs.**

- **Database:** We use NoSQL databases like MongoDB for a flexible, document-based data storage approach..

The following steps will be helpful to start off the project:

1. Get a Firm Grasp on the Technology:

- **Frontend (React.js):**
  - Learn React.js for building the front-end. You will use it to create reusable UI components and manage application state using tools like React Hooks or Redux (for more complex state management).
  - Set up your project using create-react-app to easily configure and start developing.
  - Focus on integrating APIs (provided by Node for backend) with Axios or Fetch API for communication between the frontend and backend.
- **Backend (Python Flask/Django):**
  - **Express.js:** A minimal and flexible Node.js web application framework that provides a robust set of features for building RESTful APIs quickly and efficiently.
  - **Node.js:** A powerful JavaScript runtime that allows you to build scalable backend services using a single language across the stack.
  - Learn how to set up routes (APIs) for different modules like user authentication, student management, attendance, etc., using Express.js.
- **Database:** Use a NoSQL database like MongoDB. Learn how to interact with the database using Mongoose, an elegant MongoDB object modeling tool for Node.js.
- Set up **API authentication** using **JWT (JSON Web Tokens)** or **OAuth** for secure user login and management.

2. Define Users, Roles, and Business Rules

- **Number of Users:** Define primary user roles such as Students, Teachers, Administrators, and Parents (optional).

- **User Profiles:** Decide the features available for each role (e.g., students can view grades, teachers can update attendance, administrators can manage users and system data).

- **Modules:** Course Management, Attendance Tracking, Assignment Submissions, Exam Results, Fees Management, Library System, etc.

- **Business Rules:** Clearly define the rules that the system will enforce:
  - Attendance thresholds for exams.
  - Rules for assignment deadlines and grading.
  - Policies on course registration and fees management.

3. Create a Super User Role

- Implement a Super User (Administrator) role with elevated permissions:
  - The Super User should be able to assign students, teachers, and administrators to their respective roles.
  - Assign rights to different roles. For example:
    - Students can view their profile, assignments, grades, and attendance.
    - Teachers can update attendance, upload assignments, and grade students.
    - Administrators can manage user roles, system settings, and oversee overall operations.
- Build a role-based access control (RBAC) system to manage user permissions across various modules.

4. Make Help Features User-Friendly

- Provide a Help Section for each module of the system, making it easy for users to find information on how to use the system.
  - Include short video tutorials, tooltips, or walkthroughs for key functions such as registering for a course, submitting assignments, and checking attendance.
  - Offer a search feature in the help section to help users find relevant content quickly.
- Include a support form where users can report issues or ask for assistance.

5. User Interface (UI) Consistency and Visual Appeal

- **Consistent Design**: Ensure a uniform design across all pages. In React.js, this can be achieved by:
  - Using reusable components for menus, footers, forms, and buttons.
  - Maintaining consistent color schemes, fonts, and layouts across the entire platform.
- **Visual Appeal:** Add images, icons, and other visual elements to make the system engaging and easy to use.
  - Use modern UI libraries like Material-UI or Ant Design to speed up development and create a polished look.
  - Ensure the interface is responsive, working seamlessly on both desktop and mobile devices.
- Implement a **single-page application (SPA)** architecture using React Router to ensure smooth transitions between different pages without reloading the entire page.

Requirements

Hardware requirements

Number	Description	Alternatives (If available)
1	PC with at least 5 GB hard-disk and 4 GB RAM	Not-Applicable
2	Processor: Dual Core 2.0 GHz or higher	Quad-Core for faster performance
3	Internet connectivity for accessing the system	Not-Applicable
4	Monitor with 1024x768 resolution	Higher resolution monitors
5	Backup storage (external HDD or cloud backup)	Cloud storage services (e.g., Google Drive, OneDrive)

Software requirements

Number	Description	Alternatives (If available)
1	**Operating System:** Windows 10 / macOS / Linux	Not Applicable
2	**Frontend Development:** React.js	Angular or Vue.js
3	**Backend Development:** Node.js	Not Applicable
4	**Database:** MongoDB	MySQL
5	**IDE for Backend:** Visual Studio Code	Sublime Text or Atom
6	**IDE for Frontend**: Visual Studio Code / WebStorm	Sublime Text or Atom
7	**API Development and Testing:** Postman	Insomnia
8	**Version Control**: Git and GitHub	Bitbucket or GitLab
9	**Web Browser:** Chrome / Firefox for testing the frontend	Safari, Edge
10	**Libraries and Frameworks:** Material-UI/Ant Design for UI	Bootstrap
11	**Deployment Environment:** Heroku / AWS / Digital Ocean for hosting the system	Microsoft Azure or Google Cloud Platform (GCP)

Manpower requirements

2 to 3 students can complete this in 4 – 6 months if they work fulltime on it.

Milestones and Timelines

Number	Milestone Name	Milestone Description	Timeline Week no. from the start of the project	Remarks
1	Requirements Specification	Complete specification of the School Management System (with appropriate assumptions). Write a document detailing the same and prepare a presentation.	2-3	Attempt to add additional relevant functionalities such as parent communication modules, or enhanced fee management features
2	Technology familiarization	Understanding the technologies used for the project—React.js (frontend) and Node.js (backend), and databases like MongoDB.	4-5	Focus on applying knowledge to the project. The presentation should cover the tech stack with a practical project implementation perspective.
3	Database	Create a database	5-7	Finalize the database design

	creation	schema for at least 100 users (students, teachers, admins, etc.) and 20 different courses or classes.		and populate it with sample data to allow smooth development and testing phases.
4	High-level and Detailed Design	Outline all possible user scenarios (e.g., logging in, viewing assignments, marking attendance) and create flowcharts or pseudocode for handling them.	7-9	Ensure all scenarios map to the requirements specified in milestone 1. For each requirement, there should be a corresponding design and scenario.
5	Implementation of the front-end of the system	Develop the frontend components: Login screen, dashboard for different users (students, teachers, admins), and individual option screens (e.g., attendance).	10-12	Start working on the test plan for the entire system during this phase, which can be updated with new scenarios as development progresses.
6	Integrating the front-end with the database	Integrate the frontend with the backend, allowing data to flow between React.js and the Node.js backend. This includes database interactions (CRUD operations).	12-13	The system should be fully functional and ready for integration testing at this point.

7	Integration Testing	Test the system thoroughly using the test cases developed. Validate all user scenarios (login, user registration, attendance tracking, grades viewing, etc.).	14-15	Allocate two additional weeks to handle any issues that arise during testing and fix bugs.
8	Final Review	Resolve all issues identified during integration testing. Ensure that all requirements from milestone 1 are fully met or provide reasons for unfulfilled requirements.	16-18	The system should be reviewed in detail to confirm that all functionalities work as expected. Conduct a final project demo for stakeholders or reviewers.

Guidelines and References

- https://docs.oracle.com/cd/E17952_01/ (SQL Documentation)
- https://git-scm.com/book/en/v2 (Git Documentation)
- https://www.uml-diagrams.org/ (UML Design Guidelines)
- https://www.geeksforgeeks.org/software-requirement-specification-srs-format/ (Software Requirements Specification (SRS) Format)