Bytexl’s guided project

Final Project report

|  |  |
| --- | --- |
| Name of the educator | JUHI DARJI |
| Project title | INSTITUTE MANAGEMENT APP |
| Tools / platforms used | HTML5 - For structuring the web pages.  CSS3 - For styling and layout design (Responsive design with media queries).  JavaScript - For adding interactivity, event handling, and dynamic content.  LocalStorage - For saving data locally in the browser (such as users, students, questions).  Git & GitHub - For version control and project hosting.  Nimbus - As the primary IDE for development.  Google Chrome (DevTools) - For debugging and inspecting code in the browser.  Bootstrap (Optional) - For responsive UI design (if used, though it's not visible in the code directly). |

About the project: The Institute Management System is a web-based application designed to streamline and automate administrative tasks within educational institutions. It provides a centralized platform for managing students, employees, assessments, and other educational activities. The system allows users to log in as Admin, Employee, or Student, each with specific roles and privileges. The admin can manage student details, create and assign tasks, and generate reports. Employees can submit marks, update curricula, and track student progress, while students can view assessments, reports, and reading materials.

This system enhances the workflow of educational institutions by automating manual tasks, reducing administrative overhead, and improving accessibility. The web interface ensures that the platform is easy to use and accessible across different devices, offering a seamless experience for all users involved in the educational process.

### System requirements: both software and hardware:Hardware Requirements

* **Processor**: Intel i3 or higher
* **RAM**: 4GB or higher
* **Hard Disk**: 500MB free space for application files
* **Display**: 1280 x 720 resolution
* **Network**: Internet connection for web access

**Software Requirements**

* **Operating System**: Windows 10 or higher, Linux, or macOS
* **Web Browser**: Chrome, Firefox, or Edge (latest versions)
* **Development Tools**: Visual Studio Code, Node.js, Git (for version control)
* **Database**: Local storage (JSON or simple file-based) or a lightweight database system like SQLite for persistent data storage

Functional requirements: User Authentication: Users (Admin, Employee, Student) must be able to log in using a unique username and password.

Role-Based Access: Access levels are granted based on the role of the user (Admin, Employee, Student).

Student Management: Admin can add, update, and delete student details.

Task Management: Admin and employees can assign tasks to students or themselves.

Assessment Management: Admin can create assessments, and students can view their assessments and grades.

Curriculum Management: Employees can update curriculum details.

Report Generation: Admin and employees can generate and view student reports.

Data Storage: All information should be stored in a secure, retrievable format (local storage, JSON).

User interface requirements if any: Responsive Design: The user interface should adapt to different screen sizes (desktop, tablet, mobile).

Login Screen: For authentication (username and password fields).

Dashboard: Role-based dashboards for Admin, Employee, and Student with their respective tasks.

Forms: For adding/editing student, employee, and task data.

Error Messages: Informational popups for invalid inputs or system errors.

Reports & Data Views: Display reports in readable formats (tables, lists).

### Inputs and Outputs:

### Inputs:

* Username and password for authentication
* Student details: Name, ID, etc.
* Task and assignment details: Name, description, due dates
* Assessment answers or marks for students
* Curriculum updates

**Outputs:**

* Login success or failure message
* Confirmation messages for actions (e.g., "Student added successfully")
* Lists of students, tasks, assessments, and reports
* Generated reports of student progress or marks

List of subsystems:

Authentication System: Handles user login and role-based access control.

Student Management: Subsystem for adding, updating, and deleting student records.

Task Management: Subsystem for creating and assigning tasks to users.

Assessment Management: Manages the creation of assessments and student performance data.

Curriculum Management: Allows employees to update the curriculum details.

Reporting System: Generates reports based on student data, tasks, and assessments.

Other Applications relevant to your project: Mention in what different contexts this project can be used with other flavours.

This project can be adapted for various educational management systems, such as:

School Management Systems: Managing primary/secondary school students and teachers.

College/University Management: Administering courses, professors, students, and grades.

Employee Training Systems: Assigning tasks and assessments to corporate employees.

E-learning Platforms: Online courses with assessment and progress tracking.

Different versions could include features like:

Integration with online course providers (e.g., Moodle, Coursera).

Mobile app version for accessibility on-the-go.

Cloud storage integration for scaling the data storage needs.

**Designing of Test cases:**

Test Cases List:

Test Case ID: TC01

Test Scenario: User Login

Test Steps: Enter valid credentials (username and password), click Login button.

Expected Result: User is logged in and redirected to the correct dashboard.

Test Status: Pass

Test Case ID: TC02

Test Scenario: Invalid Login

Test Steps: Enter incorrect username or password.

Expected Result: Display "Invalid username or password" error.

Test Status: Pass

Test Case ID: TC03

Test Scenario: Add New Student (Admin)

Test Steps: Enter valid student details, click Add button.

Expected Result: Student added successfully, confirmation message displayed.

Test Status: Pass

Test Case ID: TC04

Test Scenario: View Student Tasks (Employee)

Test Steps: Employee logs in, navigate to Tasks section.

Expected Result: Assigned tasks for the employee are displayed.

Test Status: Pass

Future Work:

Scalability: Implement a database system (e.g., MySQL, PostgreSQL) to handle large amounts of data.

Mobile Version: Develop a mobile application for easier access and management.

Advanced Reporting: Enhance the reporting system with customizable filters and export options.

Multi-language Support: Add support for multiple languages to cater to a wider audience.

Cloud Integration: Store and manage data using cloud services like AWS or Google Cloud.

References: ((negative points if missing or inadequate)

W3Schools – HTML, CSS, and JavaScript tutorials.

MDN Web Docs – Official Mozilla documentation for JavaScript and Web APIs.

JavaScript: The Good Parts by Douglas Crockford – For understanding JavaScript’s core concepts.

**Negative Points**:

The system does not currently scale well for large institutions without a backend database.

There are no real-time notifications for task assignments or assessment results.

**Reflection of the Project Creation**

Technical Challenges:

Role-based Access Control: Implementing a smooth and secure login system with differentiated access for Admin, Employees, and Students.

Data Persistence: Managing student and task data without a backend system led to using local storage, which limits scalability and performance.

UI Responsiveness: Ensuring that the interface looks good on both mobile and desktop devices presented design challenges.

Existing Software Engineering Knowledge:

Problem-Solving: Solved issues by applying modular design principles and breaking down tasks into smaller, manageable chunks.

Web Development Fundamentals: Used core knowledge of HTML, CSS, and JavaScript to create the application interface and logic.

Benefits from the Project:

Gained hands-on experience in developing a web-based management system.

Improved my understanding of role-based access and how to handle data persistence without relying on a backend.

Additional Knowledge:

Learning about database management systems (DBMS) could have helped scale the project.

Familiarity with backend frameworks (Node.js, Express) would have enabled me to build a more robust and scalable solution.