NORTHERN UNIVERSITY
OF BUSINESS & TECHNOLOGY KHULNA

[Project Report]

Project Title :

Student Management System (Zzone Academy)

Submitted by:

Nishan Sami

ID : 11230321272

nishansami32@gmail.com

Course Title: Software Development

Course Code: 2216

Section: 4B

Submitted to:

Anindya Nag

Lecturer,

Northern University of Business and Technology Khulna

Date of Submission: 17.08.2025

Contents :

Table of Figure :

Student Management System

1. Abstract :

Zzone Academy's Student Management System solves the pain of manual record-keeping by offering a responsive front-end that manages students' names, IDs, sections, contacts, courses, and results all in-browser using HTML/CSS/JavaScript and local Storage. It delivers CRUD operations, search, sorting, dashboards, and data persistence without a backend. Key outcomes: smooth user experience, accessible data handling, and basic export/reset functionality for portability.

2. Introduction :

Background

Educational institutions often rely on disjointed methods to track student information. Modern Student Information Systems (SIS) consolidate this data for efficient access and reporting .

Problem Statement

Zzone Academy lacked an intuitive, consolidated interface for managing student profiles, course enrollments, and results without relying on complex server-based systems.

Objectives

- Build a responsive front-end interface using HTML, CSS, JavaScript.
- Enable CRUD operations for student data.
- Implement data persistence using browser local Storage.
- Include search, sort, and filtering features.

- Provide export (JSON) and reset functionality.

Scope

- **Included**: Front-end only; local data; no server/database.
- **Excluded**: Authentication, multi-user support, backend integration.

3. Literature Review :

- Manual systems are error-prone and inefficient; secure web-based solutions at scale.
- Web-based SIS improve accuracy and stakeholder communication, backed by studies implementing UML, testing, and designs in educational contexts.
- The W3C Web Storage API allows persistent client-side storage ideal for lightweight applications with offline capability.
- Open-source SIS platforms serve broader institution needs but come with complexity and infrastructure requirements .

4. Methodology :

Approach

Developed a single-page app using tabs and forms. CRUD actions update the UI and sync student data with localStorage.

Algorithms

- Sorting and filtering student records on the client.
- Search filter by text matching in JS.
- JSON serialization for persistence.

Tools & Technologies

- HTML5, CSS3 (Grid, Flexbox).
- Vanilla JavaScript.
- Web Storage API .
- Browser DevTools for testing.

Experimental Design

Manual testing on mobile and desktop browsers to validate responsiveness, persistence, search/sort behavior, and data export.

5. Figure with Description :

Flowchart Description: Illustrates the user input via form, storage in browser, data retrieval for table render, and export/reset actions. Helps clarify system flow in a visual way.

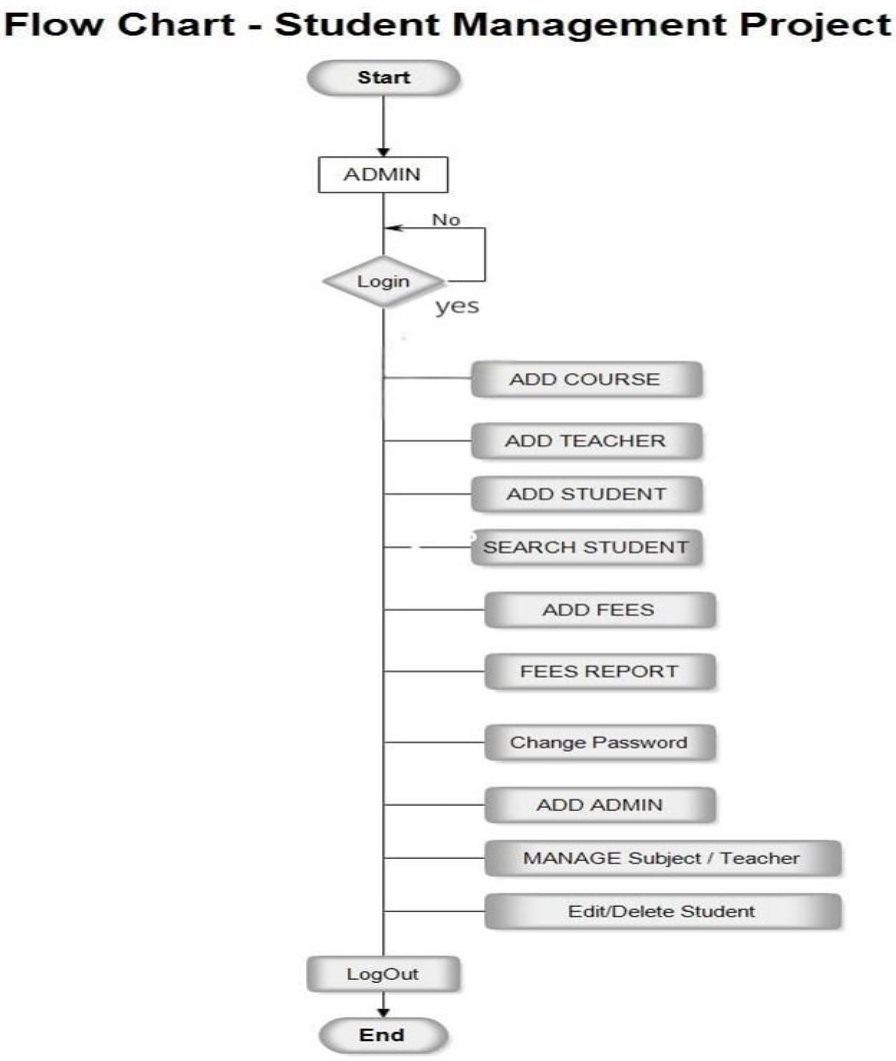

*Figure :1 :*Flowchart Of Student Management System

Description :

- It starts with an **Admin** user who must **login**.
- If login fails ("No"), the flow cycles back to the login step; if successful ("Yes"), the admin proceeds.
- After login, the admin can perform a series of actions: **Add Course**, **Add Teacher**, **Add Student**, **Search Student**, **Add Fees**, **Generate Fees Report**, **Change Password**, **Add Admin**, **Manage Subject/Teacher**, or **Edit/Delete Student**.

- Finally, once completed, the admin selects **Log Out**, which leads to the process **End**.
- It’s a linear, user-driven flow showing how the admin interacts with system modules after authentication

6. Implementation

System Architecture

- **Model:** JS array of student objects.
- **View:** HTML forms and table UI.
- **Controller:** JS event handlers (submit, click, search, sort).

Components

- **Tabs**: Dashboard, Students, Courses, Services, Results, Contact.
- **Dashboard**: KPIs (count, average), recent results, quick actions.
- **Students**: Form and table with search/sort/edit/delete features.
- **Courses/Services**: Static listings from JS arrays.
- **Results**: Filterable result bars and average display.
- **Contact**: Dummy form for interface completeness.

Code Excerpts :

```
const STORAGE_KEY = 'zzone_students_v1';
function loadStudents() {
  return JSON.parse(localStorage.getItem(STORAGE_KEY)) || [];
}
function saveStudents(list) {
  localStorage.setItem(STORAGE_KEY, JSON.stringify(list));
}
```

*Figure :2 :*Storage Key

```
$('#student-form').addEventListener('submit', e => {
  e.preventDefault();
  upsertStudent();
});
```

*Figure :3 :*Student Form

Description :

- const STORAGE_KEY = 'zzone_students_v1'; defines where student data is stored.
- loadStudents() retrieves and parses that data from localStorage, defaulting to an empty list if none exists.
- saveStudents(list) converts the student array to JSON and saves it under the key.

- On form submission, the submit event is intercepted to prevent a page reload and instead call upsertStudent(), which updates the list and re-renders the UI.

7. Results and Analysis :

Results

- Full CRUD: add, edit, delete.
- Live search and sortable table.
- Data persists across sessions.
- Dashboard KPIs and visual bars update dynamically.
- JSON export and reset restore seed data.

Analysis

The system meets objectives and is fast, accessible, and reliable especially for demo or lightweight use. The local Storage-based persistence works without complexity.

Snippet

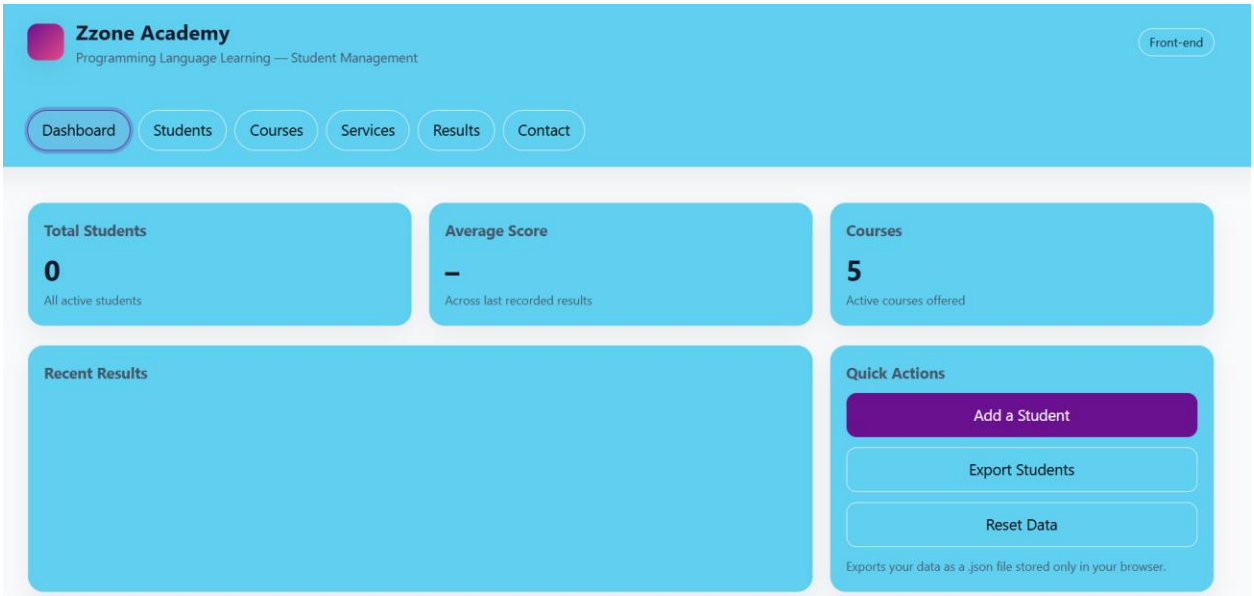

Figure :4 : Web page of Zzone Academy

Description :

- At the top is the **Zzone Academy header**, showing the program focus and a responsive navigation bar with tabs like Dashboard, Students, Courses, etc.
- Below, three clean **info cards** display key KPIs: **Total Students (0)**, **Average Score (–)**, and **Courses (5)** a quick overview at a glance.
- The **Recent Results** section sits blank right now, ready to show student performance once data is added.
- On the right, the **Quick Actions** panel offers clear calls to action: **"Add a Student," "Export Students,"** and **"Reset Data."**
- Overall, the layout is minimalist, modern, and functional each element serves a clear purpose without clutter.

8. Discussion

Interpretation

This front-end-only design works well as a prototype or teaching tool. It demonstrates how much mileage you get from HTML/CSS/JS without backend complexity.

Challenges

- Ensuring smooth responsive layout across devices.
- Handling sort/filter logic manually.
- Designing a user-friendly UI with vanilla JS.

Limitations

- No multi-user or server-side data.
- Data limited to local browser; shared use requires manual export/import.
- No advanced analytics or real-time collaboration.

9. Conclusion

Summary

We built a functional, responsive Student Management System prototype for Zzone Academy. It supports student record management, visual dashboards, and local persistence entirely client-side.

Contributions

- Clean front-end architecture with ARIA tabs and responsive design.
- No dependency on frameworks or servers.

- Working local data persistence and export/reset workflows.

Future Work

- Add backend (Node, PHP, Firebase, etc.) for shared data and authentication.
- Introduce analytics, attendance, reporting features.
- Create mobile app or PWA version.

10. References

1. Falebita, O. S. (2022). Secure Web-Based Student Information Management System using Laravel.
2. Pasaribu, J. S., & Argadikusuma, I. S. (2024). Design and Testing of a Web-Based Student Information Management System.
3. W3C. Web Storage API: LocalStorage and SessionStorage.
4. Huang, H., & Li, B. (2024). Student Management with programmable device integration.
5. Wikipedia. openSIS.
6. Wikipedia. SchoolTool SIS.