# **STUDENT HUB**

**INTRODUCTION**

Every academic institute depends on data, but not all of them handles it well. In many schools, student records are scattered across paper files, spreadsheets, or outdated digital systems that lack structure, security, or scalability. This creates confusion, increases administrative workload, and miscommunication especially when it comes to grading. To address these challenges, educational institutions often turn to digital solutions that can streamline student data management.

A student management system is a software-based platform designed to store, process, and manage student information such as academic performance, personal details, and enrollment data. By digitizing this data, the system improves accuracy, accessibility, and organization. However most traditional system fail to account for access control, they treat all users the same, granting unrestricted privileges regardless of roles or responsibility. This chapter seeks to represent information concerning background study, problem statement, objective, and the scope.

Student Hub is a simple, locally-hosted Student management System built using HTML, CSS, JavaScript and node.js, with a backend database. It allows users to register as either **Students** or **Teachers**, offering tailored access to academic records based on their role. Students can **view their grades** and **report inaccuracies**, while teachers can **manage and update grades without modifying student personal information.**

**BACKGROUND STUDY**

Before the emergency of computerized systems, student’s record-keeping was entirely manual. Academic institutions relied on paper files, handwritten grade books, and physical storage cabinets. This traditional method, though effective for small scale operations, became unsustainable as enrollment increased and data complexity grew.

The first steps toward digitizing student information began in the 1960s and 1970s, with university in the United States and Europe experimenting with mainframe-based administrative software. These early systems were custom-built, often expensive, and used only for large institutions.

A major shift occurred in the 1980s and early 1990s with the widespread us of personal computers. Schools began using spreadsheet software such as lotus 1-2-3 and Microsoft excel to manage student data locally. This era marked the beginning of small-scale student information systems (SIS) that were more accessible to schools.

By the late 1990s and early 2000s, dedicated student management systems began to emerge commercially. Products like PowerSchool (launched in 1997), Infinite Campus (in 1993), and open-source tools like Fedena (launched in 2009) introduced web-based access and multi-role support for students, teachers, and administrators.

But all these systems were expensive and mostly available for large institutes leaving the smaller institute behind the evolution of student management systems. Hence, the reason we are building a small scale, less expensive system to manage student data.

**LITERAURE REVIEW**

This chapter offers a critical examination of existing research and systems related to student management platforms. It aims to contextualize the academic information systems by evaluating methodologies, innovations, and limitations of the system.

ALIYU MUSA (2019) conducted research on the development of student grade management system. He designed a basic desktop application for entering and viewing grades.

In 2015 MOHAMMAD A. ALIA and HUSSEIN A. AL BAHADILI highlighted the importance of using role-based access to protect student data by create design and implementation of a student information system. The main focus was to introduce a centralized student information system for university and aim at secure access, data integrity, and multi-user functionality.

In 2017 E. OKONIGENE and E. EHIKIOYA designed and implement an offline-based student management system. The main focus was to create a student management system in areas with poor or no internet connection, emphasizing on offline performance and data security.

In 2013 OMOGBADEGUN Z.O highlights the importance of role-based access control in student management system, and the main focus was on how to separate user’s permissions using role-based access control, preventing unauthorized access to sensitive student data, demonstrating secure login, data hiding, and limited privileges.

**OBJECTIVES**

* To develop a locally hosted student management system.
* To ensure student can securely review their academic records and flag incorrect grades.
* To enable teachers to manage grades while restricting access to sensitive student information.
* To store all data locally independent of internet access
* To reinforce secure and ethical data handling in software systems

**LIMITATION**

* Student cannot interact with each other
* Student cannot edit his/her grade
* Teacher cannot add or edit student personal details
* No SMS or email notification
* Basic password encryption
* No multi-user sessions
* All data must be managed locally

**INOVATION**

Student Hub will introduce new unique features making our web app different from what others have built before. These features will include;

* Role-Based Access Control (RBAC):

Separates permissions between students and teachers for better data security.

* Grade-Reporting Features:

Students can report incorrect grades mimicking real academic systems

* Offline, Local-only Operation:

No hosting or internet required, accessible to institutions with low resources

**Functional and Non-Function Dependencies**

* Functional Dependencies

These are the features the system must perform in other to call this project successful. They include;

* + User registration and login with role selection
  + Role-based access to dash-boards
  + Student access to grades and complaint system
  + Teacher access to grade editor and student list
  + Local data storage in JSON or SQLite file
  + Display and validation of data in the browser
* Non-Functional Dependencies:

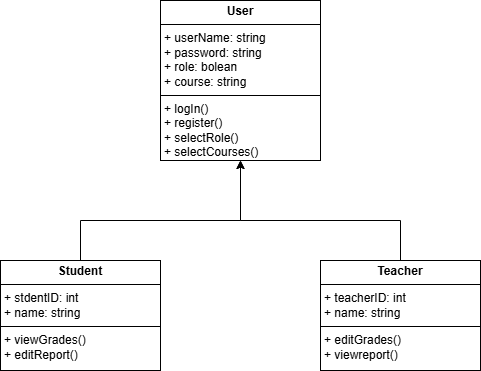
These describe how the system should behave

* + Fast and responsive in modern browsers
  + Simple and intuitive interface
  + Reliable data access with minimal crashes
  + Consistent design across all screens

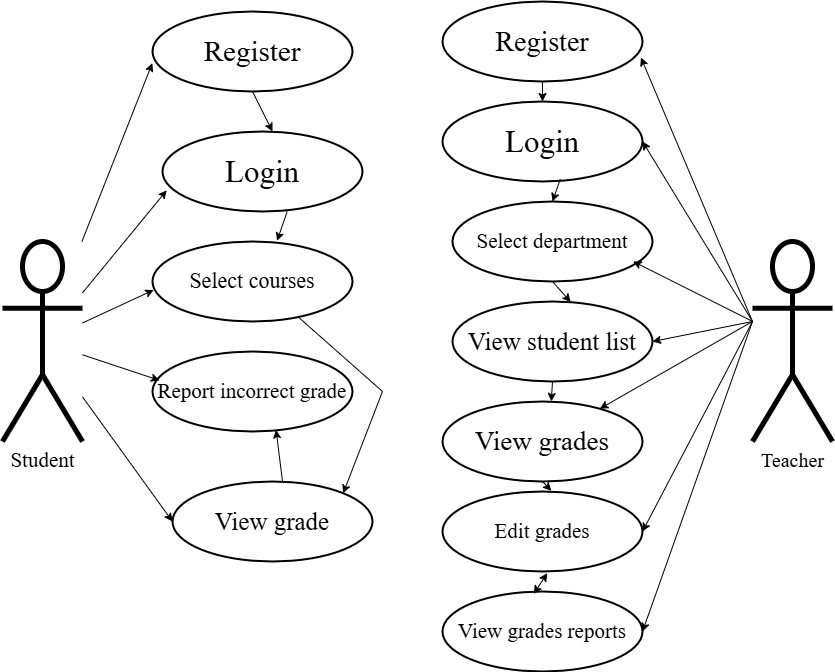
**UML Diagrams**

The following diagrams were drawn using ***Draw.io***

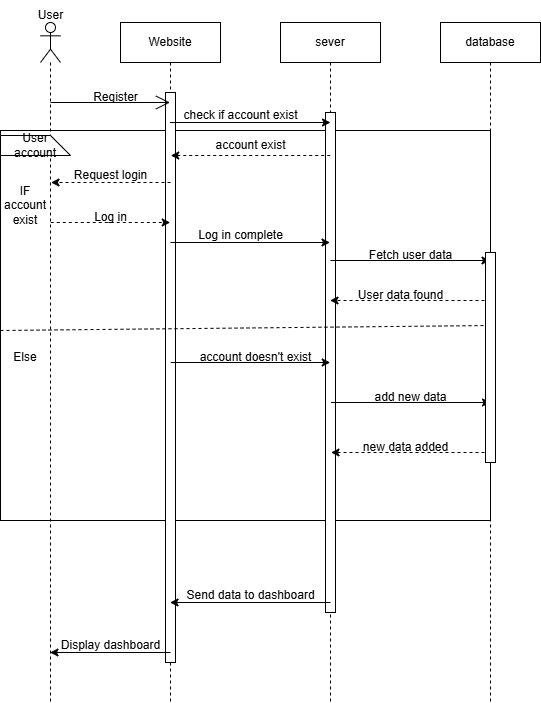
* Class Diagram;



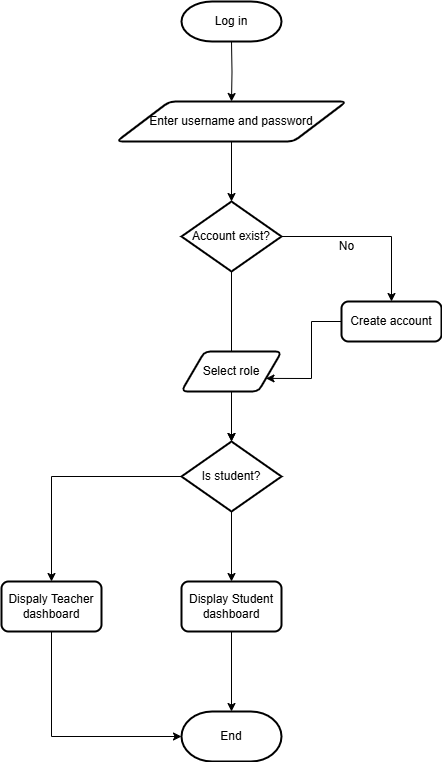
* Use-Case Diagram;



* Sequence Diagram;



* Flow Chart;



* Architectural Diagram;