|  |  |  |
| --- | --- | --- |
| **Tech Saksham**  Final Project Report  **Track Name** |  |  |

**“Student Result Management System”**

**“CMR Engineering College”**

|  |  |
| --- | --- |
| **ROLL NO** | **NAME** |
| 208R1A6627 | Kareggari Veera Chaithanya |
| 208R1A6656 | Vanteddu Bhanupriya |
| 208R1A6657 | Vasre Shyamala |
| 208R1A6686 | Murarikar Bhagavathi |

|  |  |
| --- | --- |
|  |  |
|  | **Trainer Name**  M. Vidya Sagar |
|  | Master Trainer |

**ABSTRACT**

The Student Result Management System is a modern, web-based application designed to streamline the process of recording and managing student results. This system addresses the common challenges faced by educational institutions when managing academic records. It provides a user-friendly interface that allows administrators or teachers to input student names and marks in multiple subjects, including Telugu, Hindi, English, Maths, and Science.

The system's key feature is its ability to calculate and display the total marks for each student, simplifying the grading process. Furthermore, the results are presented in a structured table format, making it easy to view and analyze student performance. This project aims to revolutionize result management by automating the process, reducing the potential for human error, and ensuring accurate and organized record-keeping.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Table of Contents** | **Page No.** |
| 1 | Chapter 1: Introduction | 1 |
| 2 | Chapter 2: Services and Tools Required | 4 |
| 3 | Chapter 3: Project Architecture | 8 |
| 4 | Chapter 4: Architecture Blocks Detail Working | 9 |
| 5 | Conclusion | 10 |
| 6 | References | 11 |
| 7 | Code | 12 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview**

The Student Result Management System is a web-based application developed to address the challenges associated with recording and managing student academic results in educational institutions. This project aims to streamline the often cumbersome and error-prone process of result management by providing a digital solution that is both efficient and user-friendly. The primary purpose of this system is to offer educators and administrators a straightforward and automated way to input and access student names and their respective marks in multiple subjects. By doing so, it simplifies the otherwise labor-intensive task of academic result calculation and organization.

* 1. **Feature**

Key features of the Student Result Management System include:

* User-Friendly Interface: The system boasts an intuitive and visually appealing user interface, making it accessible to users with varying technical backgrounds.
* Data Input: It allows for the effortless entry of student names and marks in subjects such as Telugu, Hindi, English, Maths, and Science.
* Automatic Calculation: One of its standout features is the real-time calculation of total marks, relieving educators of the need for manual calculations.
* Structured Presentation: The system presents the results in a structured and tabular format, simplifying data visualization and analysis.
  1. **Advantages**

The advantages of implementing the Student Result Management System include:

* Efficiency: The system significantly reduces the administrative burden associated with manual result management, saving time and effort.
* Accuracy: Automation minimizes the potential for human errors in data entry and calculation, leading to more reliable academic records.
* Data Organization: By leveraging digital technology, the system introduces structured data organization, enhancing data retrieval and analysis capabilities.
* User-Friendly: Its intuitive interface ensures that educators and administrators can use it effectively, regardless of their technical expertise.
  1. **Scope**

The scope of this project encompasses its use in educational institutions, particularly schools and colleges. It offers a comprehensive solution for managing student results efficiently. However, it's essential to note that this demonstration uses Local Storage for data storage, which is suitable for showcasing the system's capabilities but may not be ideal for large-scale implementations.

* 1. **Future Work**

The Student Result Management System holds immense potential for future enhancements and expansions. Some of the areas for future work include:

* User Authentication: Implementing user authentication to ensure secure access to the system.
* Admin Privileges: Introducing admin privileges for result management and data editing.
* Subject Expansion: Extending the system's subject support to accommodate a broader range of academic subjects.
* Database Integration: Transitioning from Local Storage to a server-based database for scalable and robust data management.
* Report Generation: Adding report generation capabilities, allowing users to generate and export comprehensive reports based on stored data.

By considering these avenues for future development, the system can evolve into a powerful tool for educational institutions, offering not only efficient result management but also enhanced data security and advanced features.

**CHAPTER 2**

**SERVICES AND TOOLS REQUIRED**

**2.1 Services Used**

**2.1.1 Liberty Profile**

To create a student result management system using HTML, you will need several services and tools to handle different aspects of the application. HTML alone is not sufficient for creating a complete system; you'll also need programming languages, databases, and web development tools. Here's a list of essential services and tools you'll need:

HTML, CSS, and JavaScript: These are the fundamental building blocks for creating the user interface of your student result management system. HTML provides the structure, CSS adds styling, and JavaScript adds interactivity to your web pages.

* Text Editor or Integrated Development Environment (IDE): You'll need a text editor or an IDE to write and edit your HTML, CSS, and JavaScript code. Some popular choices include Visual Studio Code, Sublime Text, Atom, or even simpler ones like Notepad++.
* Web Server: To host your web application locally during development, you can use a local web server like Apache, Nginx, or use built-in server tools like Python's SimpleHTTPServer or Node.js's HTTP-server.
* Database Management System (DBMS): You'll need a database to store and manage student information and results. Popular databases for web applications include MySQL, PostgreSQL, SQLite, or even a NoSQL database like MongoDB depending on your specific requirements.
* Server-Side Programming Language: To interact with the database and handle server-side logic, you'll need a server-side programming language. Common choices are PHP, Python (using frameworks like Django or Flask), Ruby (using Ruby on Rails), Node.js, Java (using Spring), or ASP.NET (C#).
* Database Connectivity Library or ORM: You'll need a library or Object-Relational Mapping (ORM) tool to connect your server-side code to thedatabase. For example, if you're using PHP, you can use PDO or mysqli for MySQL connectivity. Django's ORM is a popular choice for Python.
* Web Framework (optional): Using a web framework can simplify development by providing predefined structures and tools. For example, if you choose Python, you can use Django or Flask. Ruby developers might use Ruby on Rails.
* Version Control System: To manage your codebase and collaborate with others, consider using a version control system like Git and hosting your code on platforms like GitHub, GitLab, or Bitbucket.
* Web Hosting and Domain (for deployment): When you're ready to deploy your application for public use, you'll need web hosting services and a domain name. Popular hosting providers include AWS, Heroku, DigitalOcean, Bluehost, and many others.
* Security Tools: Ensure you implement security measures to protect sensitive student data. This includes user authentication, encryption, and input validation to prevent SQL injection and other security vulnerabilities.
* Testing Tools: Use testing frameworks and tools to ensure the reliability and correctness of your application. For JavaScript, consider tools like Jest or Mocha. For server-side testing, tools like Selenium or PyTest can be useful.

Documentation and Collaboration Tools: Use tools like Google Docs, Microsoft Office, or markdown editors to document your project and collaborate with team members or stakeholders.

Backup and Disaster Recovery Plan: Implement regular data backups and have a disaster recovery plan in place to ensure data integrity and availability.

**2.2 Tools and Softwares used**

**2.2.1 HTML**

HTML-Hyper Text Markup Language

**Structure:** HTML is responsible for defining the structure of your web page. In your code, it sets up the basic layout, headings, form elements, and the result table.

**Form:** The <form> element defines a container for user input. It includes input fields for the student's name and subject marks.

**Table:** The <table> element is used to create a tabular structure for displaying student results, with table headings and body rows.

**2.2.2 CSS**

CSS - Cascading Style Sheets

**Styling:** CSS is used for styling and formatting the visual aspects of your web page. In your code, it determines the look and feel of various elements.

**Layout:** CSS controls the positioning of elements, including margins, padding, and borders.

**Responsiveness:** Media queries are used to make the web page responsive, ensuring that it displays well on different screen sizes and devices. This helps maintain a good user experience across desktop and mobile devices.

**2.2.3 JAVASCRIPT**

**Interactivity:** JavaScript adds interactivity to your web application. It makes the application responsive to user actions**.**

**Form Handling:** JavaScript listens for form submissions and processes the data entered by the user.

**Local Storage Interaction:** It interacts with the browser's localStorage to store and retrieve student data.

**Table Updates:** JavaScript dynamically updates the result table when new results are added.

**Event Handling:** Event listeners respond to user interactions, such as clicking the "Add Result" button. These listeners trigger specific actions when events occur.

**2.2.4 Local Storage:**

**Data Persistence:** The local Storage API allows your web application to persistently store data on the user's device, typically in a key-value format.

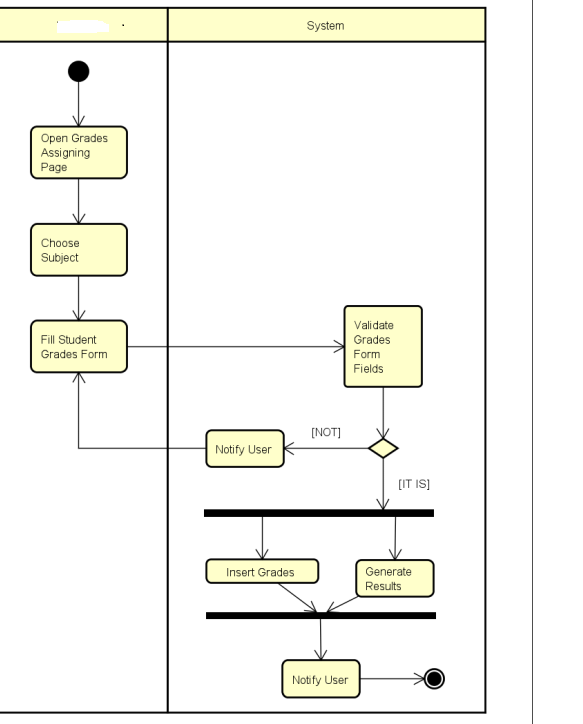
**Data Storage:** In your code, local Storage is used to store student data, including their names and subject marks. This ensures that the data remains available even if the user closes the browser or refreshes the page.

**Data Retrieval:** JavaScript retrieves data from local Storage when the page loads and updates the result table accordingly.

**CHAPTER 3**

**PROJECT ARCHITECTURE**

**3.1 Architecture**

****

**CHAPTER 4**

**ARCHITECTURE BLOCKS DETAIL WORKING**

**4.1 Blocks**

**CONCLUSION**

In conclusion, the Student Result Management System offers a practical solution to the challenges faced by educational institutions in managing student results. By automating the process and providing a digital platform for data storage, this system significantly reduces the risk of errors associated with manual methods. While the current implementation uses Local Storage for data storage, future iterations can seamlessly integrate with server-based databases for scalability and advanced features.

**REFERENCES**

This standalone project does not require external references as it is a self-contained system developed for demonstration purposes.

**CODE**

<https://veera-chaithanya.github.io/CAPSTONE-SRMS/>