Result MS

****

****

Project Name: Result Management System

Version 1.0

Date: [16/05/2025]

Prepared by:

Md. Kamrul Hasan

Roll : 79

Section : A

Lab Group : Q

Reviewed by:

Mr. Md. Monir Hossain

Assistant Professor

Dhaka City College

****

****

Requirements:

Result Processing System: This system keeps the assignment, class test, midterm and final examination marks of a specific student with different session, dept, semester etc, and finally it will generate the student GPA/CGPA. This system should facilitate that student can view his/her individual result of a specific semester.

Software Requirements Specification (SRS)

Project Title: Result Processing System

Platform: Web Application (React.js with Vite)

Technologies: Tailwind CSS, MUI, easy-peasy, react-hook-form, yup

---

1. Introduction

1.1 Purpose

The Result Processing System is a web-based application designed to help academic institutions manage student performance data. It stores and processes assignment, class test, midterm, and final examination marks for students, categorized by session, department, and semester. The system calculates GPA and CGPA for students and enables them to view their individual results for specific semesters.

1.2 Intended Audience

- **Students**: To view their academic results.
- **Teachers/Admins**: To input and manage student marks.
- **Developers**: To understand the technical requirements.

1.3 Scope

The system will:

- Keep records of various assessment types (assignment, class test, midterm, final).
- Organize data by student, session, department, and semester.
- Compute GPA/CGPA.
- Allow students to view their results.
- Feature a modern UI and smooth user experience using Vite, Tailwind, and MUI.

1.4 Definitions and Acronyms

- **GPA**: Grade Point Average
- **CGPA**: Cumulative Grade Point Average
- **MUI**: Material-UI (React component library)
- **Vite**: A build tool for fast React development
- **easy-peasy**: A state management library for React
- **Yup**: A JavaScript schema builder for value parsing and validation
- **react-hook-form**: A form validation library for React

---

2. Functional Requirements

2.1 User Management

- Add/edit/delete student details (name, roll, session, department, semester).

- Assign a unique identifier to each student.

2.2 Marks Entry

- Allow authorized users to input marks for:
  - Assignment
  - Class Test
  - Midterm
  - Final Exam
- Marks should be numeric and validated using `yup` (e.g., between 0 and 100).
- Each entry must be linked to a subject and semester.

2.3 Result Calculation

- Calculate GPA based on a weighted average of:
  - Assignment (e.g., 10%)
  - Class Test (e.g., 10%)
  - Midterm (e.g., 30%)
  - Final Exam (e.g., 50%)
- Calculate CGPA across semesters.

2.4 Result Viewing

- Allow students to:
  - Select semester
  - View their results by subject
  - See GPA for the semester and CGPA

2.5 Filtering & Sorting

- Admins can filter student records by:
  - Session
  - Department
  - Semester

---

3. Non-Functional Requirements

3.1 Usability

- Intuitive and responsive user interface
- Clear navigation between sections

3.2 Performance

- Fast data loading and transitions using Vite

- Optimized React rendering and lightweight components

3.3 Compatibility

- Cross-browser support (Chrome, Firefox, Edge)
- Mobile-friendly layout using Tailwind CSS

3.4 Security

- Data validation using `yup` on all input forms
- Optional: Basic role-based access control for students vs. admins

---

4. System Design Overview (Frontend Only)

4.1 Page Components

- **Login Page (optional)**
- **Dashboard**
  - Navigation for students/admins
- **Student Management**
  - Form to add/edit/delete student data
- **Marks Entry**
  - Form for entering subject-wise marks
- **Result Calculation Logic (internal)**
  - GPA/CGPA calculator component
- **Result Viewer**
  - View results semester-wise
  - Visual summary (tables, optional charts)

4.2 State Management

- Use `easy-peasy` for managing:
  - Student data
  - Subject marks
  - Session/department/semester filters
  - GPA/CGPA calculations

4.3🗂️ Component Tree (High-Leve)

```
App
├── 📁 Layout
```

```
│   ├── Navbar
│   ├── Sidebar (optional for admin/student navigation)
│   └── Footer
│
├── 📁 Pages
│   ├── Home
│   ├── StudentDashboard
│   ├── AdminDashboard
│   ├── StudentResultView
│   ├── MarksEntry
│   ├── StudentForm (Add/Edit)
│   └── NotFound
│
├── 📁 Components
│   ├── StudentListTable
│   ├── ResultTable
│   ├── GPAResultCard
│   ├── CGPAResultCard
│   ├── FilterBar (Session / Dept / Semester filters)
│   ├── InputField (Reusable Form Input with react-hook-form)
│   ├── Loader
│   └── Notification (Optional Toasts)
│
├── 📁 Store (easy-peasy models)
│   ├── studentModel.js
│   ├── marksModel.js
│   └── resultModel.js
│
├── 📁 Utils
│   ├── calculateGPA.js
│   ├── calculateCGPA.js
│   └── formValidationSchemas.js
│
├── 📁 Hooks
│   └── useStudentData.js (optional custom hook)
│
└── 📁 Routes
    └── AppRouter.jsx
```

🔍 Component Breakdown

🔷 App

- Root of the application
- Wraps everything with `easy-peasy` store provider and routing

🔷 Layout

- Shared layout with navbar/sidebar
- Could use MUI's `AppBar` and `Drawer`

🔷 Pages

Each page handles a specific feature:

- `Home`: Intro or login selector
- `StudentDashboard`: Simple view with student's results
- `AdminDashboard`: Controls to manage students and marks
- `StudentResultView`: View GPA/CGPA for a student
- `MarksEntry`: Form for entering marks
- `StudentForm`: Add or edit student info

🔷 Components

- Reusable UI elements like tables, filters, and result cards
- `InputField` can simplify working with `react-hook-form`

🔷 Store

- Models for students, marks, and calculated results
- Holds global app state using easy-peasy

🔷 Utils

- Logic for calculating GPA and CGPA
- Form validation schemas using yup

🔷 Routes/AppRouter.jsx

- Manages navigation between routes using `react-router-dom`

---

5. Tools and Technologies

Functionality	Technology
Frontend Framework	React.js with Vite
UI Components	MUI, Tailwind CSS
State Management	easy-peasy
Form Handling	react-hook-form
Validation	yup

---

6. Assumptions and Dependencies

- Data is entered manually through UI (no external API or backend used).
- GPA/CGPA calculation logic will be predefined and fixed unless modified.
- LocalStorage can be used for temporary data storage (if needed).

- Internet access is required to load external styles and libraries.

7. Future Enhancements

- Backend integration (Node.js, Firebase, etc.)
- User authentication (student login/admin dashboard)
- Export results to PDF or Excel
- Graphical analysis of performance
- Notifications on GPA changes or semester result publication