## Student CRUD App Project Documentation

### Project Overview

This is a full-stack Student CRUD (Create, Read, Update, Delete) application.

**Frontend:** React.js

**Backend:** Node.js with Express.js

**Database:** MongoDB

The app allows users to add, edit, delete, and view student information including subjects and marks, and calculates total marks and percentage.

### Folder Structure

project/  
 ├── frontend/  
 │ ├── src/  
 │ │ ├── components/  
 │ │ │ ├── StudentForm.js  
 │ │ │ └── StudentList.js  
 │ │ ├── App.js  
 │ │ └── App.css  
 │ └── package.json  
 └── backend/  
 ├── models/  
 │ └── Student.js  
 ├── routes/  
 │ └── students.js  
 ├── server.js  
 └── package.json

### Backend Setup

1. Open terminal and navigate to the backend folder:

cd project/backend

1. Install dependencies:

npm install express mongoose cors

1. Start MongoDB service (if not running) and connect to database studentdb.
2. Run the backend server:

node server.js

* Server runs on port 5003
* Endpoints:
  + GET /students - fetch all students
  + POST /students - add new student
  + PUT /students/:id - update student by id
  + DELETE /students/:id - delete student by id

### Frontend Setup

1. Open a new terminal and navigate to the frontend folder:

cd project/frontend

1. Install dependencies:

npm install axios

1. Start the frontend app:

npm start

* App runs on port 3000 (or another if 3000 is in use)
* Frontend communicates with backend via Axios HTTP requests.

### Working Process

1. **Frontend Components:**
   * StudentForm.js : Handles adding and editing student data.
   * StudentList.js : Displays all students in a table and handles edit/delete actions.
2. **State Management:**
   * useState is used to manage form inputs and student list.
   * useEffect is used to fetch data from backend on initial load.
3. **API Communication:**
   * axios.get fetches students.
   * axios.post adds a new student.
   * axios.put updates an existing student.
   * axios.delete removes a student.
4. **Styling:**
   * App.css contains styles for form, buttons, tables, and responsive design.
5. **Flow:**
   * User enters student info in the form.
   * On submit, frontend sends request to backend.
   * Backend stores data in MongoDB and sends response.
   * Frontend updates student list table dynamically.

### How to Use

1. Start MongoDB.
2. Run backend: node server.js
3. Run frontend: npm start
4. Open browser: http://localhost:3000
5. Add, edit, or delete students using the UI.

### Notes

* Ensure backend is running before frontend.
* Axios error handling is implemented to show alerts on API failures.
* Form allows multiple subjects using dynamic input fields.
* Total marks and percentage are automatically calculated in backend.

### Project By,

Harshni A.

**End of Documentation**