**Project Description: Student Management System**

The **Student Management System** is a full-stack web application built with FastAPI for backend services, PostgreSQL with docker for database management, and HTML, CSS, and JavaScript for the frontend. This application provides an intuitive interface to manage student records, offering CRUD (Create, Read, Update, Delete) functionalities with real-time data updates and responsive design. The project is designed for ease of use and scalability, enabling educational institutions or developers to handle student data effectively.

**Technology Stack & Architecture:**

A diagram of a network

Description automatically generated

**Key Features:**

1. **Backend with FastAPI and PostgreSQL**
   * **CRUD API**: A set of RESTful API endpoints implemented with FastAPI allows users to perform CRUD operations on student records.
   * **PostgreSQL Database**: The application uses PostgreSQL as the database, ensuring secure, reliable, and scalable data storage. Database models are designed to efficiently manage and query student data.
   * **CORS Configuration**: Cross-Origin Resource Sharing (CORS) is configured to allow the frontend to communicate with the backend from different domains or ports, providing flexibility for deployment.
2. **Frontend Interface**
   * **HTML and CSS**: The frontend interface is structured with HTML and styled with CSS to ensure a clean, user-friendly design.
   * **JavaScript Operations**:
     + **Student List Display**: The home screen displays all student records dynamically in a table with details such as ID, name, age, and grade.
     + **Add Student**: A form for adding new student data sends the information to the backend API via a POST request.
     + **Update Student**: Users can select a student to update, where the system pre-fills an edit form with the student’s current details. After editing, only updated fields are submitted to the backend in a PUT request.
     + **Delete Student**: Each student record includes a delete option, allowing users to remove records with a DELETE request.
3. **Deployment on Tomcat**
   * The frontend is deployed on an Apache **Tomcat** server, serving static HTML, CSS, and JavaScript files in a production-ready setup.
   * The FastAPI backend deployed on **UVICORN**
   * PostgreSQL database – can be deployed through **Docker**.

**Database Schema (PostgreSQL)**

* **Students Table**: The main table for storing student data, including:
  + **ID** (Primary Key): Unique identifier for each student.
  + **Name**: Name of the student.
  + **Age**: Age of the student.
  + **Grade**: Current grade of the student.

**Project Objectives:**

* **User-Friendly Design**: Provide an easy-to-use interface for managing student data with real-time data visibility.
* **Modular Architecture**: Separating the frontend and backend allows for easy expansion and updates.
* **Scalability**: FastAPI and PostgreSQL together form a highly scalable stack, suitable for handling growing datasets and complex queries.

**Technology Stack:**

* **Backend**: FastAPI, PostgreSQL
* **Frontend**: HTML, CSS, JavaScript
* **Deployment**: Tomcat (for frontend), with backend and database hosted separately

**Sample User Interface:**

**A screenshot of a student management system

Description automatically generated**