**StudentManager**

**Description :** The Student Manager web application is a comprehensive and user-friendly solution designed to empower individual tutors and coaching academies in efficiently managing their student-related tasks. The application provides a robust platform for organising student information, streamlining fee collection, preparing grade cards, and facilitating effective communication through mass emailing.

**Objective :** Build a complete system for managing student data and launch it as a SaaS product.

**Target Audience :** Individual tutors, coaching academies(any).

**Features**

* **Student Profile Management :** Easily create, update, and organise detailed profiles for each student, including personal information, contact details, and attendance records.
* **Fee collection and tracking :** Seamlessly collect and track fees for individual students, with the ability to generate receipts and maintain a transparent financial record.
* **Grade card preparation :** Generate accurate and professional-grade cards, allowing tutors to communicate student progress effectively.
* **Mass emailing functionality :** Enhance communication with students and parents through a convenient mass emailing feature, ensuring timely updates, announcements, and personalised messages.

**Functional Requirements**

**Authentication :** Authentication is required for logging in admin, students and tutors.

**Grade and fee card export :** A functionality for admin to export grade and fee card of a student in pdf or svg format

**User Interaction :** A functionality for users, tutors and admins to interact with the system.

**Mass Emailing :** A functionality for sending bulk emails in a single click.

**Non-Functional Requirements**

**Availability :** As the system will be used world wide, the system needs to be online 24\*7\*365.

**Performance :** The entire system needs to be performant enough to handle a load of 1000 users at a single time.

**Security :** The system will collect user data and involve monetary affairs so security becomes the top priority.

**Frameworks/ Tech Stack**

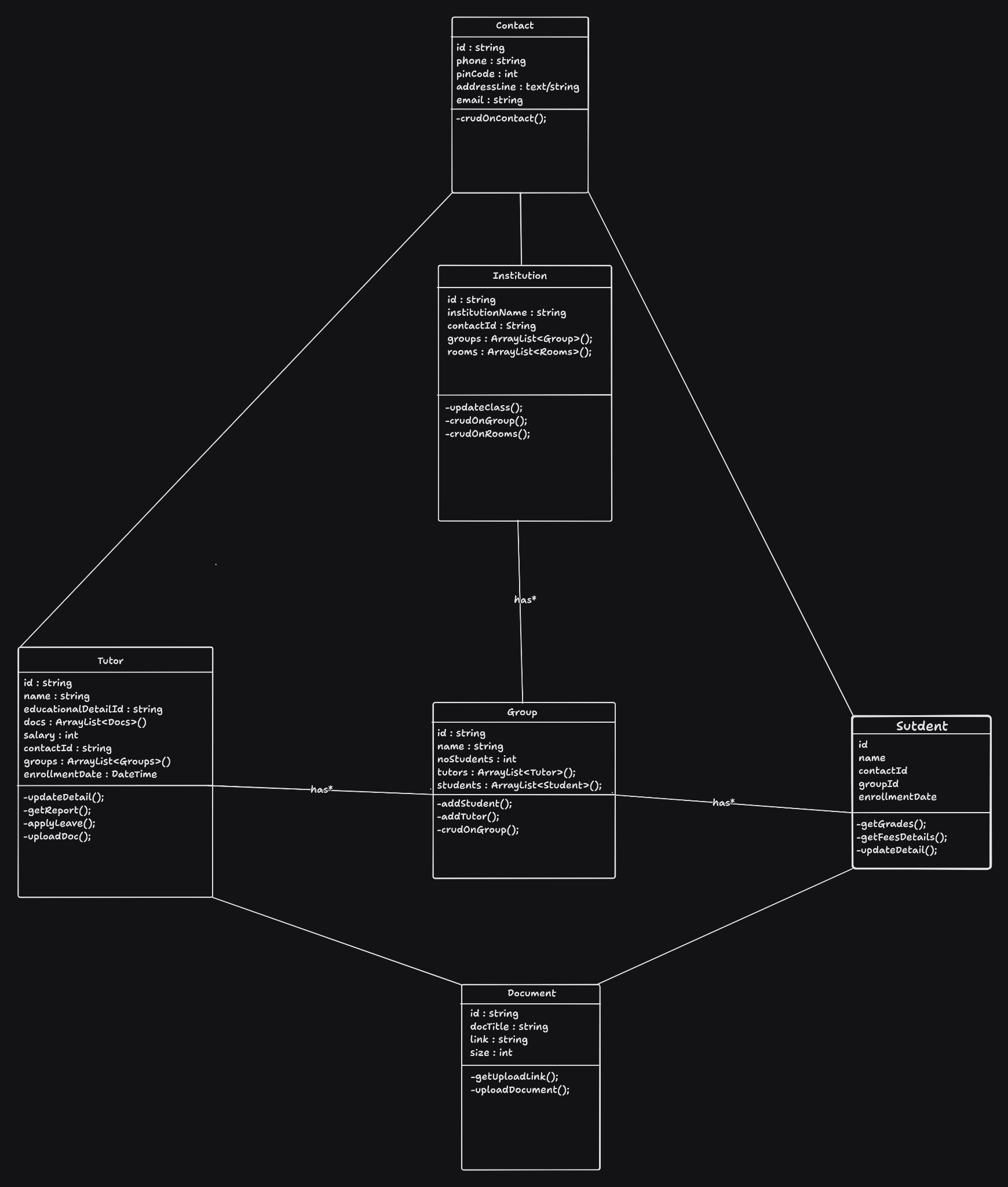
* **Frontend**
  + Nextjs
  + Tailwind CSS
  + Tailwind utilities
* **Backend**
  + ExpressJS
  + Nodejs
  + Mongoose
  + Mongodb(database)

**Deployment**

* **Frontend**
  + Frontend is deployed using free deployment service provided by vercel with no extra configurations
* **Backend**
  + Backend will be deployed on AWS and will use apache web server for handling the incoming traffic. The deployment will use https for security.

**System Design**

The low level system design and flow is described in the below pasted image. This image describes the relationship between the different entities and how they are connected to each other.



**Database Model**

**StudentModel :**

| **Column Name** | **Data Type** |
| --- | --- |
| **id** | **integer** |
| **name** | **string** |
| **contactId** | **Ref (Contact Table)** |
| **groupId** | **Ref (Group Table)** |
| **enrollmentDate** | **timestamp** |

**TutorModel :**

| **Column Name** | **Data Type** |
| --- | --- |
| **id** | **Integer (primaryKey)** |
| **name** | **string** |
| **educationalDetails** | **string** |
| **docs** | **List<Docs>(Ref : Docs Table)** |
| **Salary** | **integer** |
| **contactId** | **Ref (Contact Table)** |
| **groups** | **List<Groups>(Ref Group Table)** |
| **enrollmentDate** | **TimeStamp** |

**InstitutionModel:**

| **Column Name** | **Data Type** |
| --- | --- |
| **id** | **Integer (PrimaryKey)** |
| **name** | **string** |
| **contactId** | **Ref ( Contact Table)** |
| **groups** | **Ref (Group Table)** |

**DocumentModel :**

| **Column Name** | **Data Type** |
| --- | --- |
| **id** | **Integer (PrimaryKey)** |
| **title** | **string** |
| **link** | **string** |
| **size** | **integer** |

**API Design**

**Student :**

* get /student/<id> : fetch detail of a student
* put /sutdent/<id> : update info of a student
* post /student/ : insert new student info
* delete /student/<id> : delete a specific student

**Institution :**

* get /institute/<id> : get details of a particular institution
* post /institute : insert a new record in the institution table
* put /institute/<id> : update details of a particular institute in the table
* delete /institute/<id> : delete a particular institute

**Contact :**

* get /contact/<stu\_id> : get contact detail of a particular student
* get /contact/<tutor\_id> : get contact detail of a particular tutor
* get /contact/<ins\_id> : get contact detail of a particular institute
* put /contact/<id>?t=<type> : update detail of student/tutor/ins/ with id = <id>
* post /contact/ : insert a new record in the contact table
* delete /contact/<id> : delete a particular contact from the table

**Document Model :**

* get /docs/<id> : gets the detail about a particular document
* put /docs/<id> : updates detail about a particular document
* post /docs/<id> : inserts a new record in the docs table
* delete /docs/<id> : deletes a particular doc in the table

**Authentication**

The Student Manager web application prioritises security and user authentication to ensure the integrity of student data. The chosen authentication mechanism is JSON Web Token (JWT), a stateless and secure method that enhances the overall protection of user accounts and sensitive information.

**Scalability**

The Student Manager web application is designed with scalability in mind to accommodate the dynamic needs of individual tutors and coaching academies. By adopting both vertical and horizontal scalability principles, the system aims to seamlessly handle increased user registrations, growing student databases, and higher transaction volumes.