DESIGN DOCUMENT

Team Number	3
Client Name	Rohit Kandakatla
Team Members	Parshva Bhadra Rishabh Jain Prerak Srivastava Ameya Sharma

You should use any drawing tool for your UML diagrams. Look at the schedule page for links to various UML tools. If your diagrams are too big to cut and paste into this document, provide a reference to the external image file(s) [JPG or PNG] where they can be found or segment your image into legible sections.

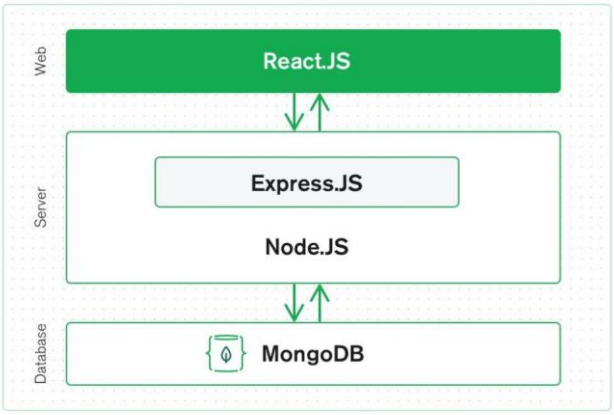
Design Overview

Architectural design

We are using the MERN stack which has a 3 tier architectural design.

The 3 major components of our website are

- Frontend
- Backend
- Database



React.js Front End

ReactJs is a JS framework for creating dynamic client-side applications in HTML. It helps us build up complex interfaces through simple

Components, connect them to data on a backend server, and render them as HTML. **Express.js and Node.js Server Tier**

Express.js is the server-side framework, running inside a Node.js server.. Express.js has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

System interfaces User

Interface

There are 2 kinds of users who will be using the website, namely:

1. Teacher
2. Parent

where the key difference between the teacher view and the parent view is that the teacher will have access and the permissions to add, modify and delete records from the database.

The various web pages that the user will be able to see are:

- **Authentication**: The user will enter their username and password to login to the website.
- **Dashboard**: There will be 2 types of dashboards, depending on the type of user
  - **Teacher Dashboard:** Each teacher will be allocated a particular class, teaching a particular subject to that class. So after logging in, the teacher will be able to see all the marks of all the students of the classes where he/she teaches.

    The user can choose the class whose marks he/she wants to view using a drop-down menu, after which the marks of all the students from that class in the teacher's subject will be neatly displayed in a table. There will be a button on top to add records to this table. Next to each record in the table, there will be buttons which allow the teacher to edit or delete that record.

  - **Parent Dashboard:** The parent dashboard will be able to see all the marks of all the subjects that their child currently has. They will be able to see all the marks neatly displayed in a table, along with the graphs that show the gradual progress of the child in each subject.
- **Profile:** The profile page will show all of the user's details, like their contact number, address etc., and user-specific details, depending on teacher or parent, like class details, or student details.

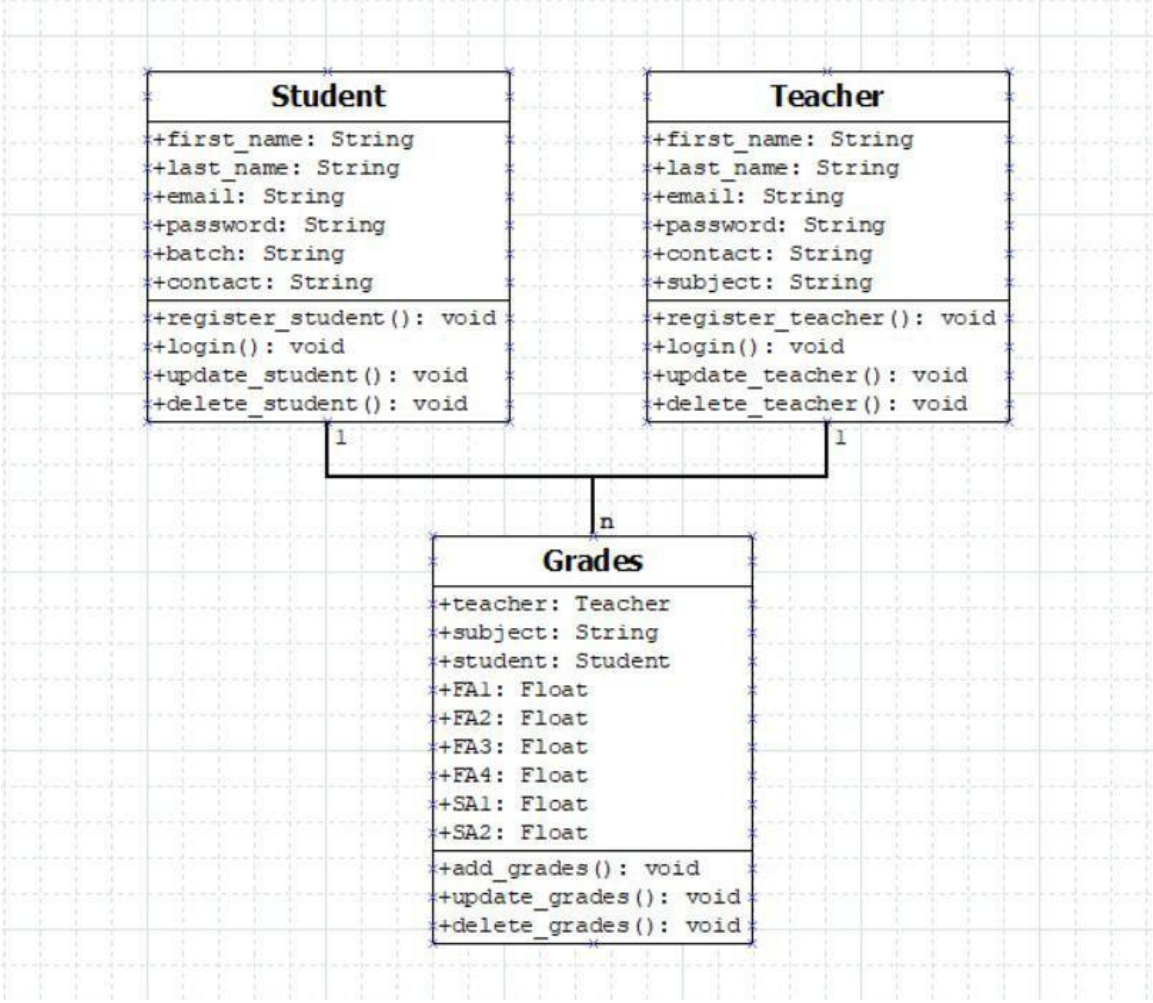
APIs

Our frontend components do make REST API requests to the backend routes. But our product does not have any API which a user can call independently in his/her own program.

Model

There will be 3 models in our database, namely:

1. Student
2. Teacher
3. Grades



Student	**Class state** • The class is responsible for maintaining the credentials (email ID and the password) of all the students/parents. **Class behavior** • The class, in real world, has the following methods: ◦ **register_student():** adds a student to the student table ◦ **login():** authenticates a student's login and password which is already present in the student database. ◦ **update_student():** updates the details of a particular student ◦ **delete_student():** deletes a student record from the students table
Teacher	**Class state** • The class is responsible for maintaining the credentials (email ID and the password) of all the teachers. **Class behavior** • The class, in real world, has the following methods: ◦ **register_teacher():** adds a teacher to the teacher table ◦ **login():** authenticates a teacher's login and password which is already present in the teacher database. ◦ **update_teacher():** updates the details of a particular teacher ◦ **delete_teacher():** deletes a student record from the teachers table
Grades	Class state • This class is responsible for maintaining the marks of all the students in all classes in all exams in all subjects Class behavior • The class, in real world, has the following methods: ◦ **add_grades():** adds grades for a particular student in a particular subject by the teacher. The teacher will only be allowed to add marks for the student who is studying in a class that he/she is allocated to, and in the subject that the teacher is teaching. ◦ **update_grades():** updates the grades of a particular student in a particular class in a particular subject. Authentication for this method is the same as the method above. ◦ **delete_grades():** deletes the grades of a particular student in a particular class in a particular subject. Authentication for this method is the same as the method above.

.

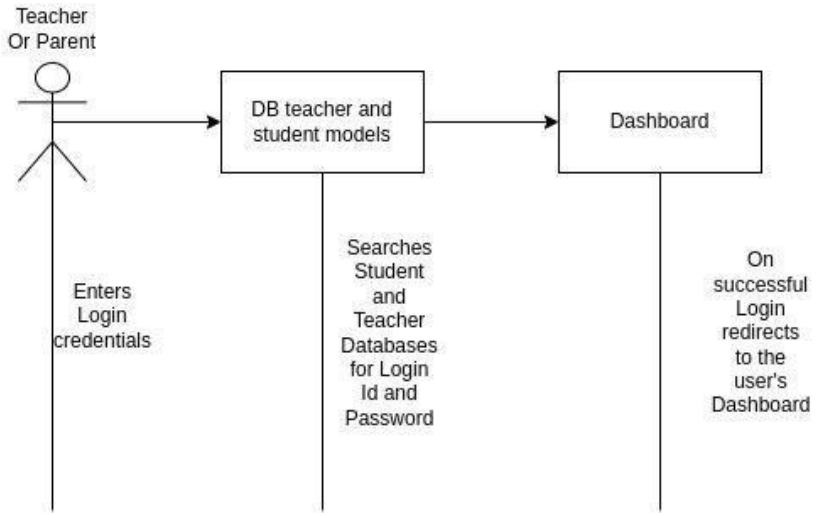
Sequence Diagram(s)

UML sequence diagram for the agreed significant use cases (about 4 major use cases).

*While you may reference an external file here, most instructors **strongly** prefer embedded images, or, failing that, external files in generic formats such as JPEG, PNG or PDF.*

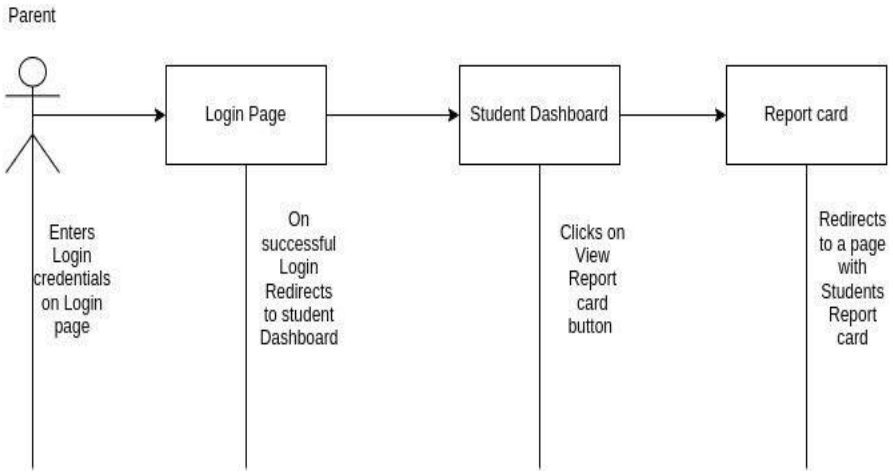
USE CASE 1: Teacher/ Parent logs in, gets redirected to a particular dashboard depending on the user

Use case 1 **(link to the below diagram)**



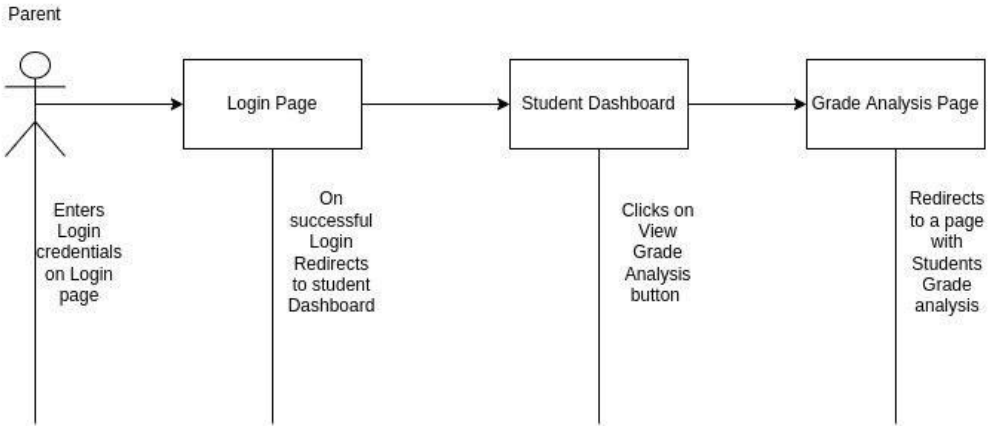
USE CASE 2 : Parent logs in, navigates to the report card section, views their child's report card.

Use case 2 **(link to the below diagram)**



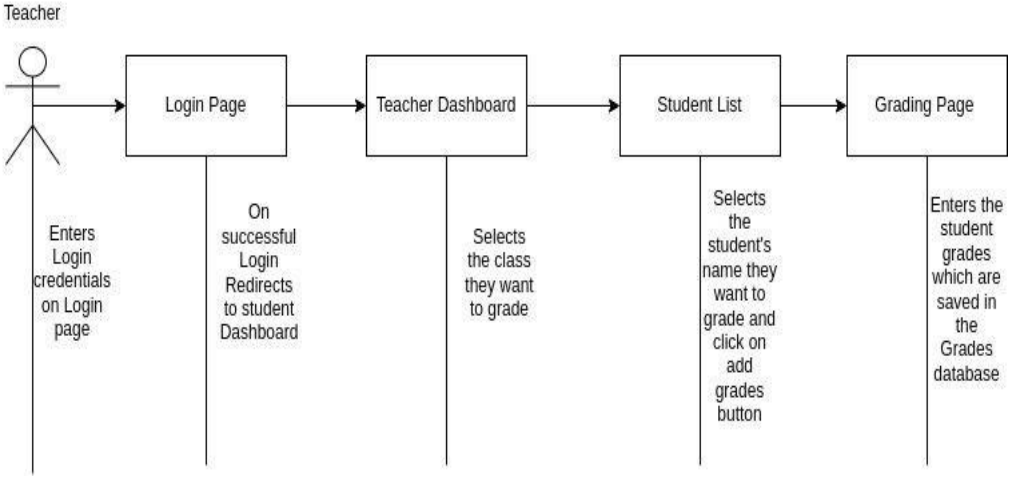
USE CASE 3 : Parents logs in, navigates to grade analysis page, and views the respective child's grade analysis.

use case 3 **(link to the below diagram)**



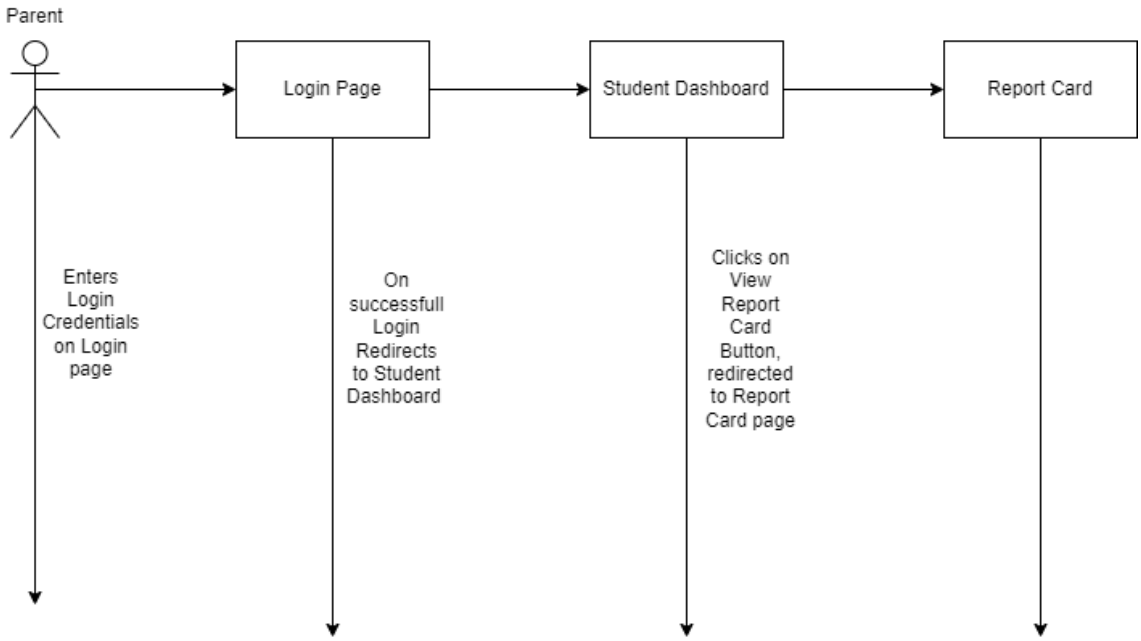
USE CASE 4: Teacher logs in, selects the class she wants to grade, and enters the grade for the particular subject that he/she teaches for the respective students.

Use case 4 **(link to the below diagram)**



USE CASE 5: Parent Logs in, clicks on view report card button

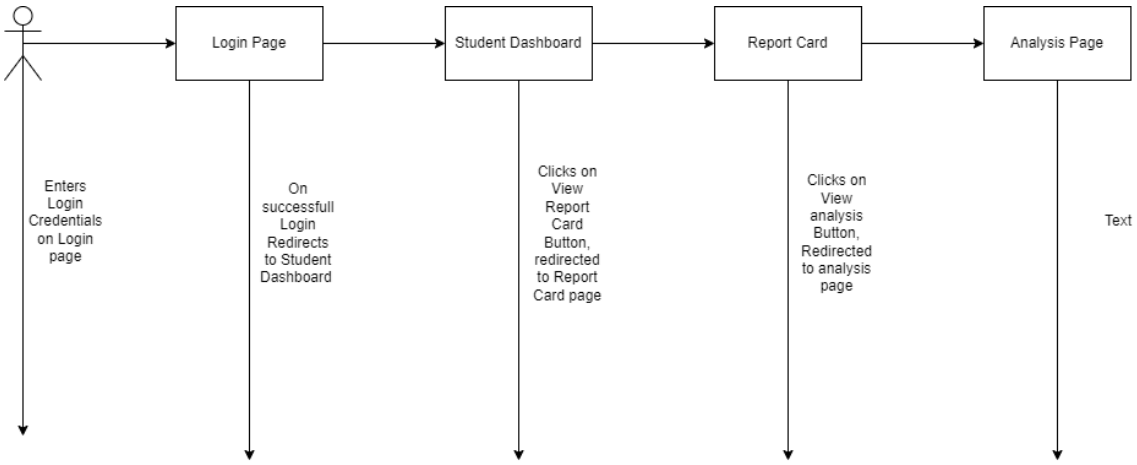
Link: https://drive.google.com/file/d/1M-C3HsqpDOKrtJ7n2SIhwcCtm8SUWS3V/view?usp=sharing



USE CASE 6:

Parent Logs in, clicks on view report card, then clicks on analysis button

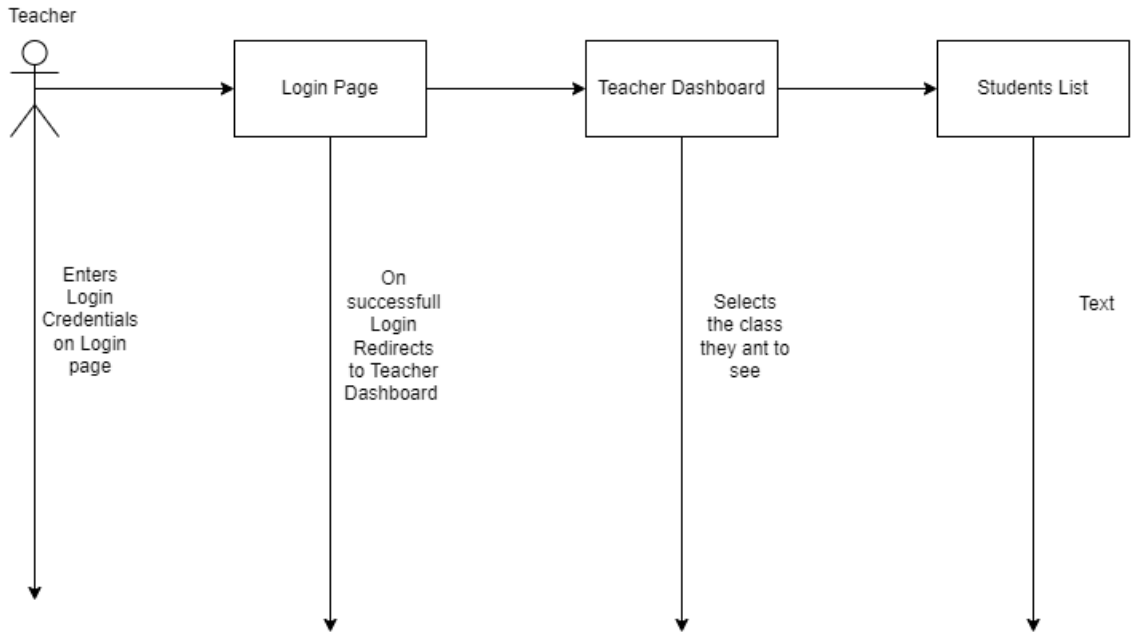
Link: https://drive.google.com/file/d/1M-C3HsqpDOKrtJ7n2SIhwcCtm8SUWS3V/view?usp=sharing



Use Case 7:

Teacher logs in, selects class and subject and the student whose data they want to view

Link:
https://drive.google.com/file/d/16i8ZYdBuQjMe62a6BJHEw8cqwJ9ty6lu/view?usp=sharing



Design Rationale

Design Rationale 1:

The original model structure for the website consisted of another model called “Users” which was eventually omitted in our first draft of the model structure.

The contents of the “Users” model would be:

- User email-ID
- User password
- User type (Teacher or student)

Choice 1: Keeping the “Users” model in the model structure

General Advantages: This model would be used for general authentication during login and registration. The main advantage of using the Users model is that the login and registration API endpoints will only have to access one model rather than using both Students and Teachers models to carry out authentication.

General Disadvantages: Overall, there would be a repetition of data in the model structure, namely in the Users, Students and Teachers model, which for large sets of data could prove to be problematic.

Choice 2: Removing the “Users” model in the model structure

General Advantages: There is less repetition of data, which is useful for us when dealing with large sets of data

General Disadvantages: As the API endpoint has to check through 2 tables for authentication, the readability of code is slightly reduced.

We chose Choice 2 for the following reasons:

- In our website, we will be having multiple teachers, teaching a particular subject to multiple classes, with multiple students in each class. This implies that the amount of data that we will be dealing with will be large, and we decided that repetition of data should be reduced as much as possible to save on memory for future features and maintain usability and efficiency of the website.