System Design Document

Promedio

Designed by: Tegh, Preyansh, Gabriel, Alejandro, Maaz, Armando

Contents

CRC Cards

FRONTEND

Class Name: App Page
Parent Class (If Any):
Sub Class (If Any): All pages (Home, Login, Signup, Welcome)
Responsibilities / Collaborators: - Is the main entry point for the router of the app - Controls the logged in state (whether the user in logged in) - Navigates to home page if logged in, welcome page if not

Class Name: Home page
Parent Class (If Any): App Page
Sub Class (If Any):
Responsibilities / Collaborators: - Main entry point for logged in users - All main features will go here or be routed from here

Class Name: Login Page
Parent Class (If Any): App Page
Sub Class (If Any):
Responsibilities / Collaborators: - This is the page where a user can input their email and password into the form and login - Makes a request to the backend - Controls form input - Tells its parent component if a login was successful or not, and the user is navigated based on that.

Class Name: Signup Page
Parent Class (If Any): App Page
Sub Class (If Any):

Responsibilities / Collaborators: - A user can input their name, gpa, email, password and create an account - Makes a request to the backend - Controls form input for these fields - Tells parent component if the signup was successful or not, and the user is navigated based on that.

Class Name: Welcome
Parent Class (If Any): App page
Sub Class (If Any):
Responsibilities / Collaborators: - This is the main landing page for our application when a user is logged out - It contains imagery, quick links to login or signup, and eventually relevant copy. - It is meant to be a brief showcase of our app.

BACKEND

Class Name: UserController
Parent Class (If Any): N/A
Sub Class (If Any): CalculatorController, AuthController, UpdateController
Responsibilities: Main controller for our application that controls other controllers such as CalculatorController, AuthController, UpdateController. Will determine what type of response to send back to a user upon request. **Collaborators:** Preyansh Dutta, Gabriel Vainer

Class Name: CalculatorController
Parent Class (If Any): UserController
Sub Class (If Any): N/A
Responsibilities: Responsible for calculating and sending back the responses of the key computations such as cGPA, major GPA, grade required for X grade, and semester GPA. **Collaborators:** Preyansh Dutta, Gabriel Vainer

Class Name: AuthController
Parent Class (If Any): UserController
Sub Class (If Any): N/A
Responsibilities: Responsible for encrypting the password and controlling the login and register for the user. Generates a JWT token to make sure the server knows the user is logged in and expires in a reasonable amount of time. Sends the correct responses i.e. errors and success codes to frontend. **Collaborators:** Preyansh Dutta, Gabriel Vainer

Class Name: Semester Model
Parent Class (If Any): N/A
Sub Class (If Any): N/A
Responsibilities: Defines the Models & Schemas for Courses and Semesters. Incorporates some request handling for GET & POST, to be separated and extended into a new controller class in the future (i.e. SemesterController which may also include PUT, PATCH & DELETE request handling).

Collaborators: Alejandro Iglesias Llobet

Class Name: User Model

Parent Class (If Any): N/A

Sub Class (If Any): N/A

Responsibilities: Defines the Model & Schema for Users. Handling already incorporated in User Controller (open for further extension).
Collaborators: Alejandro Iglesias Llobet

Endpoints

- **/api/login (POST)**
  - This is used in the Promedio/frontend/src/routes/Login.js to send a post request to mongoDB in order to compare the credentials with the users in the database to check if the user exists. If a user exists, the user is logged in and a JWT is created.
  - Request Body: { email: string, password: string }
  - Response: { name: string, email: string, gpa: string, token: string }

- **/api/register (POST)**
  - This is used in the Promedio/frontend/src/routes/Signup.js to send a post request to mongoDB in order to create a new user with the following information: Name, Email, Password.
  - Request Body: { name: string, gpa: string, email: string, password: string }
  - Response: { name: string, email: string, gpa: string, token: string }

- **/api/update (POST)**
  - This is used in the Promedio/frontend/src/routes/User.js to edit the user details. The fields Name, and Email can be updated through a post request.
  - Request Body: { name: string, gpa: string, email: string, password: string }
  - Response: { name: string, email: string, gpa: string }

- **/api/user/:userId/semester/**
  - **GET**
    - This is used in the Promedio/frontend/src/routes/Home.js to display all semesters mapped to an individual user (via “userId”).
  - **POST**
    - This is used in the Promedio/frontend/src/routes/Home.js to create a new semester (i.e. a new document within the “Semester” Model/“semesters” collection) mapped uniquely to a user (via “userId”). The semester name is passed via the request body.

- **/api/user/:userId/semester/:semesterName**
  - **GET**
    - This is used in the Promedio/frontend/src/routes/Home.js to list the details of a specific semester (currently identified with “semesterName” → plan is to either change this to be semesterId or to enforce unique semester names in schema).
  - **PATCH**
    - This is used in the Promedio/frontend/src/routes/Home.js to update an existing semester (i.e. a current document with name “semesterName” within the “Semester” Model/“semesters” collection) with a new name (as it is currently the

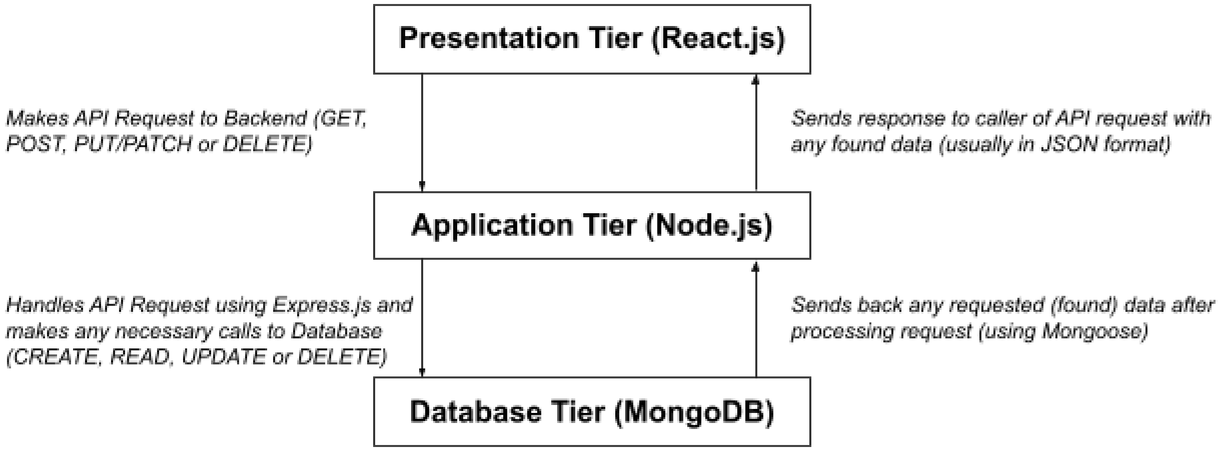
        only updateable field within the semester schema). Nothing happens otherwise within the database if no such semester exists.

    - **DELETE**
        - This is used in the Promedio/frontend/src/routes/Home.js to delete an existing semester (i.e. a current document with name "semesterName" within the "Semester" Model/"semesters" collection) that is mapped to "userId". Nothing happens otherwise within the database if no such semester exists.

- **/api/user/:userId/semester/:semesterName/course**
    - **GET**
        - This is used in the Promedio/frontend/src/routes/Home.js to display all courses listed within a specific semester (currently "semesterName" → plan is to either change this to be semesterId or to enforce unique semester names in schema).
    - **POST**
        - This is used in the Promedio/frontend/src/routes/Home.js to create a new course (i.e. a new document within the "Course" Model/"courses" collection) mapped uniquely to a semester (identified via "semesterName" parameter → plan is to either change this to be semesterId or to enforce unique semester names in schema).
    - **DELETE**
        - This is used in the Promedio/frontend/src/routes/Home.js to delete all courses mapped to a specific semester (identified via the "semesterName" parameter → plan is to either change this to be semesterId or to enforce unique semester names in schema).

- **/api/user/:userId/semester/:semesterName/course/:courseName**
    - **GET**
        - This is used in the Promedio/frontend/src/routes/Home.js to list the details of a specific course. A user should also be able to play with the course details for that requested course (i.e. accessing the CSCA08 course in the Fall 2020 semester to set a mark goal) after fetching.
    - **PATCH**
        - This is used in the Promedio/frontend/src/routes/Home.js to update an existing course (i.e. a current document with name "courseName" within the "Course" Model/"courses" collection) with a new name (as it is currently the only updateable field within the semester schema). Nothing happens otherwise within the database if no such course exists.
    - **DELETE**
        - This is used in the Promedio/frontend/src/routes/Home.js to delete an existing course (i.e. a current document with name "courseName" within the "Course" Model/"courses" collection) that is mapped to "userId". Nothing happens otherwise within the database if no such course exists.

System Architecture Diagram


Presentation Tier (React.js)
Makes API Request to Backend (GET, POST, PUT/PATCH or DELETE)
Sends response to caller of API request with any found data (usually in JSON format)
Application Tier (Node.js)
Handles API Request using Express.js and makes any necessary calls to Database (CREATE, READ, UPDATE or DELETE)
Sends back any requested (found) data after processing request (using Mongoose)
Database Tier (MongoDB)

Collaborators: Alejandro Iglesias Llobet