

# **System Design Document**

*Promedio*

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

# Contents

CRC Cards .....	2
FRONTEND.....	3 - 4
BACKEND.....	5 - 6
System Architecture Diagram .....	7

# CRC Cards

## FRONTEND

<b>Class Name:</b> All Courses Page
<b>Parent Class (If Any):</b> Semesters (?), Navbar
<b>Sub Class (If Any):</b>
<b>Responsibilities / Collaborators:</b> <ul style="list-style-type: none"><li>- Displays all courses available to the user, irrelevant of semester</li><li>- Allows the user to add a new course</li></ul>

<b>Class Name:</b> All Semesters Page
<b>Parent Class (If Any):</b> Home Page, Navbar
<b>Sub Class (If Any):</b> Semester Page
<b>Responsibilities / Collaborators:</b> <ul style="list-style-type: none"><li>- Displays all available semesters to the user</li><li>- Allows user to add/edit semesters</li></ul>

<b>Class Name:</b> App Page
<b>Parent Class (If Any):</b>
<b>Sub Class (If Any):</b> All pages (Home, Login, Signup, Welcome)
<b>Responsibilities / Collaborators:</b> <ul style="list-style-type: none"><li>- Is the main entry point for the router of the app</li><li>- Controls the logged in state (whether the user is logged in)</li><li>- Navigates to home page if logged in, welcome page if not</li></ul>

<b>Class Name:</b> Course Page
<b>Parent Class (If Any):</b> Semester Page
<b>Sub Class (If Any):</b>
<b>Responsibilities / Collaborators:</b> <ul style="list-style-type: none"><li>- Displays a course's information to the user, including the categories attached to it</li><li>- Allows the user to add, edit or delete categories related to the course</li></ul>

- Allows the user to edit or delete the course

**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:** Semester Page

**Parent Class (If Any):** All Semesters Page

**Sub Class (If Any):** Course Page

**Responsibilities / Collaborators:**

- Displays a semester's information to the user, including the courses attached to it
- Allows the user to add a new course
- Allows the user to edit/delete the semester

**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

<b>Class Name:</b> CategoryController
<b>Parent Class (If Any):</b> CourseController
<b>Sub Class (If Any):</b> N/A
<b>Responsibilities:</b> Responsible for creating categories, modifying a category, getting a list of all categories by a course id as well as deleting categories <b>Collaborators:</b> Preyansh Dutta, Gabriel Vainer, Maaz Hashmi

<b>Class Name:</b> CourseController
<b>Parent Class (If Any):</b> SemesterController
<b>Sub Class (If Any):</b> CategoryController
<b>Responsibilities:</b> Responsible for creating courses, modifying a course by id, getting a list of all courses, a specific course by id, as well deleting all courses tied to a semester or a specific course by id. <b>Collaborators:</b> Alejandro Iglesias Llobet, Armando Rojas

<b>Class Name:</b> SemesterController
<b>Parent Class (If Any):</b> UserController
<b>Sub Class (If Any):</b> CourseController
<b>Responsibilities:</b> Responsible for creating semesters, modifying a semester by id, getting a list of all semesters, a specific semester by id, as well deleting all semesters tied to a user or a specific semester by id. <b>Collaborators:</b> Alejandro Iglesias Llobet, Armando Rojas

<b>Class Name:</b> UserController
<b>Parent Class (If Any):</b> N/A
<b>Sub Class (If Any):</b> AuthController, UpdateController
<b>Responsibilities:</b> 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.

<b>Collaborators:</b> Preyansh Dutta, Gabriel Vainer
--

<b>Class Name:</b> AuthController
-----------------------------------

<b>Parent Class (If Any):</b> UserController
--

<b>Sub Class (If Any):</b> N/A
--------------------------------

<b>Responsibilities:</b> 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.
--

<b>Collaborators:</b> Preyansh Dutta, Gabriel Vainer
--

<b>Class Name:</b> Category Model
-----------------------------------

<b>Parent Class (If Any):</b> Course Model
--

<b>Sub Class (If Any):</b> N/A
--------------------------------

<b>Responsibilities:</b> Defines the Model & Schema for Academic Categories. Categories are given a name, a consistent weight, number of assessments as well as a list of grades and are tied to a specific course by a courseId.
---

<b>Collaborators:</b> Alejandro Iglesias Llobet, Armando Rojas
--

<b>Class Name:</b> Course Model
---------------------------------

<b>Parent Class (If Any):</b> Semester Model
--

<b>Sub Class (If Any):</b> Category Model
---

<b>Responsibilities:</b> Defines the Model & Schema for Courses. Courses are given a name and tied to a specific semester.
--

<b>Collaborators:</b> Alejandro Iglesias Llobet, Armando Rojas
--

<b>Class Name:</b> Semester Model
-----------------------------------

<b>Parent Class (If Any):</b> User Model
--

<b>Sub Class (If Any):</b> Course Model
---

<b>Responsibilities:</b> 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

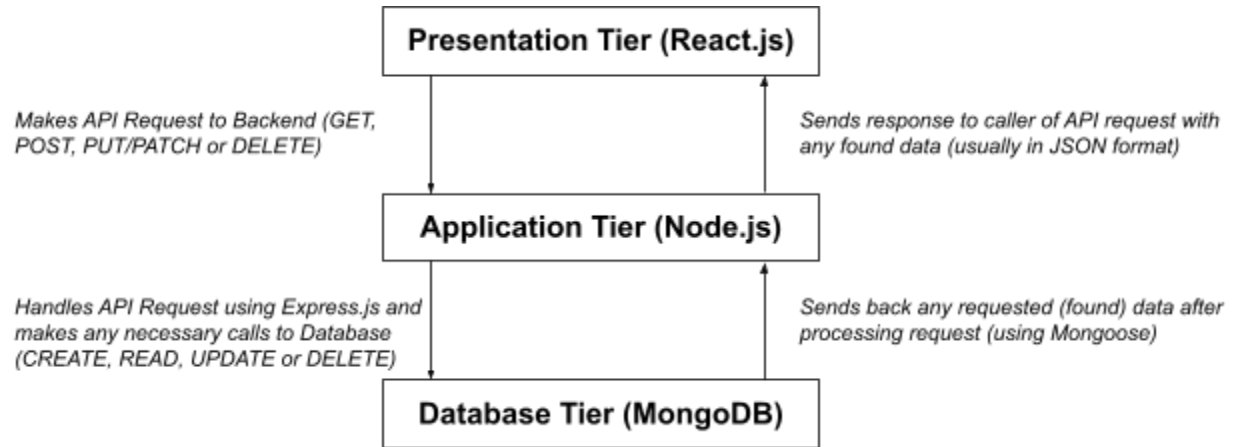
- **/api/login**
  - **POST**
    - This is used in the Promedio/frontend/src/routes/Login.js to send a post request to MongoDB in order to compare the credentials with the users in the database to check if the user exists. If a user exists, the user is logged in and a JWT is created.
    - Request Body: { 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/semester/**
  - **GET**
    - This is used in the Promedio/frontend/src/routes/Home.js to display all semesters mapped to an individual user.
    - Response: { [ { name: string, userId: Mongoose.Schema.Types.ObjectId } ] }
  - **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 . The semester name is passed via the request body.
    - Request Body: { semesterName: string }
    - Response: { message: “Semester added successfully.”, semester: { name: string, userId: Mongoose.Schema.Types.ObjectId } }
  - **DELETE**
    - This is an extra function to use for debugging in case there’s a need to delete all semesters related to a user.
    - Response: { message: “Successfully deleted (x) semesters and all related courses.” }

- **/api/semester/:semesterName**
  - **GET**
    - This is used in the Promedio/frontend/src/routes/Home.js to list the details of a specific semester.
    - Request Parameters: { semesterName: string }
    - Request Body: { name: string, userId: Mongoose.Schema.Types.ObjectId }
  - **PATCH**
    - This is used in the Promedio/frontend/src/routes/Home.js to update an existing semester (i.e. a current document with id “semesterId” 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.
    - Request Parameters: { semesterId: string }
    - Request Body: { semesterName: string }
    - Request Body: { message: “Semester updated successfully.” }
  - **DELETE**
    - This is used in the Promedio/frontend/src/routes/Home.js to delete an existing semester (i.e. a current document with id “semesterId” within the “Semester” Model/“semesters” collection). Nothing happens otherwise within the database if no such semester exists.
    - Request Parameters: { semesterId: string }
    - Request Response: { message: “Semester deleted successfully.” }
- **/api/course**
  - **GET**
    - This is used in the Promedio/frontend/src/routes/Home.js to display all courses for a specific user.
    - Request Response: [ { name: string, semesterId: Mongoose.Schema.Types.ObjectId } ]
  - **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).
    - Request Body: { semester: string, name: string, description: string, markGoal: string }
    - Request Response: { message: “Course added successfully.”, course: { name: string, semesterId: Mongoose.Schema.Types.ObjectId, description: string, markGoal: string } }
  - **DELETE**
    - This is an extra function to use for debugging in case there’s a need to delete all courses attached to an user.
    - Request Response: [ { message: “Successfully deleted [x] courses.” } ]

- **/api/course/:courseId**
  - **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.
    - Request Parameters: { courseId: string }
    - Request Response: { name: string, semesterId: Mongoose.Schema.Types.ObjectId, description: string, markGoal: string }
  - **PATCH**
    - This is used in the Promedio/frontend/src/routes/Home.js to update an existing course (i.e. a current document with id “courseId” within the “Course” Model/“courses” collection) with a new name, description or mark goal. Nothing happens otherwise within the database if no such course exists.
    - Request Parameters: { courseId: string }
    - Request Body: { name: string, description: string, markGoal: string }
    - Request Response: { message: “Course updated successfully.” }
  - **DELETE**
    - This is used in the Promedio/frontend/src/routes/Home.js to delete an existing course (i.e. a current document with id “courseId” within the “Course” Model/“courses” collection). Nothing happens otherwise within the database if no such course exists.
    - Request Parameters: { courseId: string }
    - Request Response: { message: “Course deleted successfully.” }
- **/api/addCategory**
  - **POST**
    - This is used in Promedio/frontend/src/components/EditCategoryForm.js to add a new category to a specific course. A category handles part of the cGPA calculation for that specific course.
    - Request Body: { name: string, weight: number, numAssessments: number, marks: Array<int> }
    - Request Response: { message: “Category added successfully.” }
- **/api/editCategory**
  - **POST**
    - This is used in Promedio/frontend/src/components/EditCategoryForm.js to edit an existing category in relation to a specific course. A category handles part of the cGPA calculation for that specific course. A user can change the name, course, weight, number of assessments and marks of the category.
    - Request Body: { id: string, courseId: string, name: string, weight: number, numAssessments: number, marks: Array<int> }
    - Request Response: { message: “Category edited successfully.” }
- **/api/deleteCategory**
  - **POST**

- This is used in Promedio/frontend/src/components/Course.js to delete an existing category in relation to a specific course. A category handles part of the cGPA calculation for that specific course.
- Request Body: { id: string }
- Request Response: { message: "Category deleted successfully." }
- **/api/categories/:courseId**
  - **POST**
    - This is used in Promedio/frontend/src/components/Course.js to get all the categories in relation to a course. A category handles part of the cGPA calculation for that specific course. By getting all of them, we can use their averages to calculate the overall cGPA for the course.
    - Request parameters: { courseId: string }
    - Request Response: [{ id: string, courseId: string, name: string, weight: number, numAssessments: number, marks: Array<int> }]

# System Architecture Diagram



**Collaborators:** Alejandro Iglesias Llobet