**Towson University Academic Pathway (TAP) System Requirements**

The Towson Academic Pathway (TAP) is a web application designed to help students efficiently plan and schedule their courses. TAP integrates a backend with the frontend through API calls to provide personalized academic planning based on individual student needs.

### **Frontend:**

The TAP frontend will be built using **React.js** and will feature the following components:

* **HomeComponent:** Homepage to display a little bit about the team, product, and mission statement
* **FormComponent**: Gathers user input regarding preferences, academic progress, and Towson University policies. This data will be processed using a script that leverages OpenAI to generate a personalized four-year academic plan. User inputs will be stored in the database and referenced in the SettingsComponent.
* **SettingsComponent**: Manages user preferences, academic interests, and personal information. Users can modify their preferences and update their academic schedule as needed.
* **CatalogComponent**: Displays all available courses with filtering options to help users browse and select courses.
* **DegreeCompletionPlannerComponent**: Provides a detailed academic plan for four years (8 semesters by default). Users can add or remove semesters and courses to customize their plan, with dynamic updates based on prerequisite requirements, course selection, and interests.
* **EnrollmentPlannerComponent**: Focuses on the upcoming semester’s schedule.
* **AboutComponent**: Introduces the development team behind TAP.
* **FAQComponent**: Provides a list of frequently asked questions and allows users to submit email inquiries which will be accomplished through **Nodemailer**.
* **AdminComponent**: Offers administrative functions for managing student data and key interfaces, such as handling user-submitted FAQs.
* **AuthenticationComponent\***: Handles user authentication, including login and logout functionality. This may require additional supporting components.

All components are styled using **Tailwind CSS**

**HTTP requests** between the frontend and backend will be handled using **Axios**.

### **Backend:**

The backend will be developed using **Express.js**, responsible for handling user requests and managing database interactions. The database will be hosted using either **MongoDB** or **Firebase**, depending on the final decision. It will contain the following clusters:

* **Student Data Cluster**: Stores student-related data, including preferences, academic progress, and schedules.
* **Course Data Cluster**: Holds course data from Towson University.. Will accomplish this by creating a web scraping program to scrape data from TU website.
* **FAQ Cluster**: Store user-submitted questions which will be displayed in the AdminComponent for review and response.

### **APIs:**

### Integrating a **REST API** between the frontend and backend for seamless communication and data exchange.

### **OpenAI API**: Used to generate personalized academic recommendations based on user input.

**Interactions**

### **1. User Authentication**

#### *User Interactions:*

* Users will login and log out using the **AuthenticationComponent**, providing their credentials.

#### *System Interactions:*

* The frontend sends the login request to the backend, where user credentials are validated against stored data. The backend then provides access tokens if authentication is successful or returns an error if the login fails.

### **2. User Preferences and Academic Planning**

#### *User Interactions:*

* Through the **FormComponent**, users will input their personal information, course preferences, academic progress, goals, etc.
* Users can adjust or modify their personal information or preferences through the **SettingsComponent** with the option to update to their academic plan.

#### *System Interactions:*

* Upon form submission, the frontend sends the input data to the backend databases. The data then communicates with an **OpenAI API** to generate a personalized four-year academic plan based on the user’s preferences and university policies.
* The backend stores the user inputs and generated academic plan in the **Student Data Cluster** and returns the results to the frontend. These results are then displayed in the **DegreeCompletionPlannerComponent** for the user to view or modify.
* When preferences are updated in the **SettingsComponent**, the changes are sent to the backend database, which updates the stored data and recalculates the academic plan if requested.

### **3. Course Selection & Scheduling**

#### *User Interactions:*

* Users will browse and select courses using the **CatalogComponent**, which includes filters to help narrow down available courses. Selected courses are added to their degree completion plan or upcoming semester schedule.
* In the **DegreeCompletionPlannerComponent**, users can add or remove semesters and courses to customize their plan. These changes can also be made to reflect prerequisite requirements.
* The **EnrollmentPlannerComponent** allows users to focus on scheduling courses for the upcoming semester. Users can add or drop courses here based on their immediate needs, such as course availability and time slots.

#### *System Interactions:*

* The **CatalogComponent** sends a request to the backend to retrieve available courses from the **Course Data Cluster**, populated from TU courses. Course data is displayed on the frontend, and when a course is selected, this information is sent to the backend to update the user’s academic plan.
* Changes made in the **DegreeCompletionPlannerComponent** are communicated to the backend databases, where course selections are validated.
* Changes made in the **EnrollmentPlannerComponent** are sent to the backend, which updates the current semester schedule in the databases.

### **4. Admin and FAQ Management**

#### *User Interactions:*

* Users can browse common inquiries in the **FAQComponent**. If they have additional questions, they can submit them via an email form.
* Admins will manage the application through the **AdminComponent**, where they can handle user-submitted FAQs, student data, and course management tasks.

#### *System Interactions:*

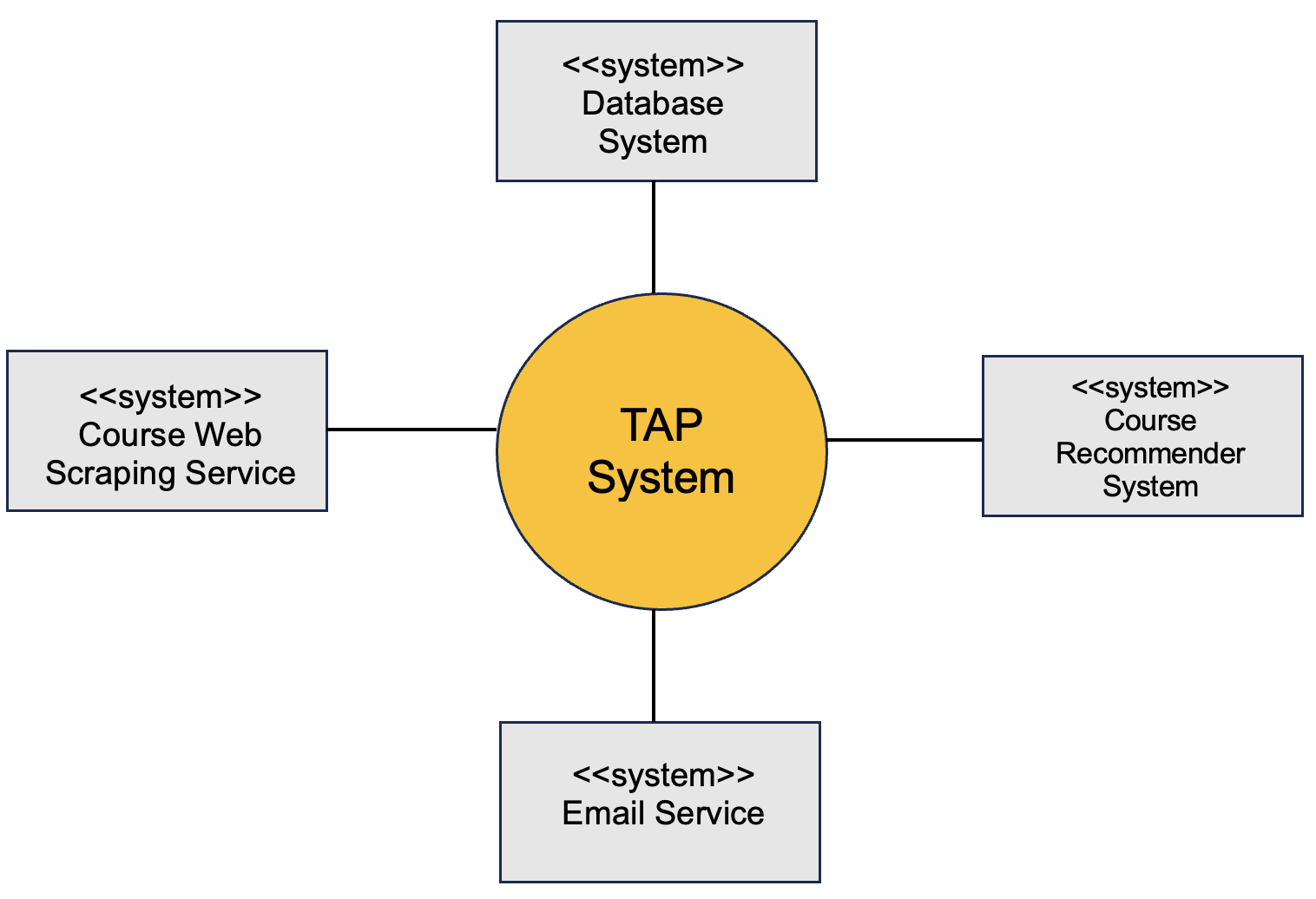
* The FAQ submissions are sent to the backend, where they are stored in the **FAQ Cluster**. Admins, using the **AdminComponent**, retrieve these questions from the backend and manage responses.
* The backend ensures admin privileges for accessing sensitive data related to student interactions, preferences, and course scheduling. Admin updates are saved in the appropriate clusters, such as student or course data, and reflected in the respective user components.

### **5. Data Management and API Interactions**

#### *System Interactions:*

* The backend, built with **Express.js**, serves as the mediator between the frontend and the database. It handles requests from the frontend (e.g., updating course schedules, retrieving course data) and processes them before updating the **Student Data Cluster** and **Course Data Cluster**.
* **API Integration**: The backend interacts with the **OpenAI API** to generate academic plans based on user inputs.

**Context Model**

****