High Level System Design

Architecture:

This application will use a client-server architecture where the client- side will be implemented using **React** and the server-side will be implemented using a server-side programming language like **NodeJS**. The website will use a database **MongoDB** to store the user data. We will use a 3-tier application architecture that consists of a user interface, Business logic layer and Data layer. The data layer stores information, the business logic layer handles logic, and the user interface consists of the front end.

User Interface:

This layer is distributed to a computing device using a web browser or a web-based application and is constructed with ReactJS and CSS. Application program interface (API) calls are the primary communication between the user interface and other layers of the application like business logic layer.

Business logic layer:

This layer is implemented using NodeJS and contains the business logic that supports the application's core functions. This choice is because it allows the app to be used by many users and provides flexibility for future development.

Data layer:

The data layer, sometimes called database layer, data access layer or back-end, is where the information processed by the application is stored and managed. We will be using the NoSQL Database (MongoDB) for this layer.

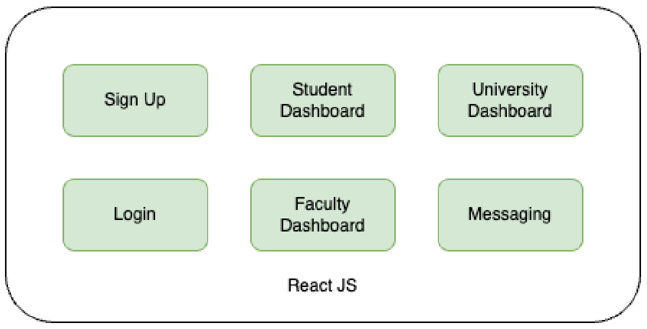
This architecture consists of a **presentation tier, an application tier, and a data storage tier**.

- **Presentation Tier:** The presentation tier is responsible for handling user interface and user input. This tier includes the following subsystems:
    - **User Interface:** This subsystem includes the web pages and forms that allow users to interact with the website.
    - **Client-side Logic:** This subsystem includes the JavaScript and other client-side code that handles user input validation and other user interactions.

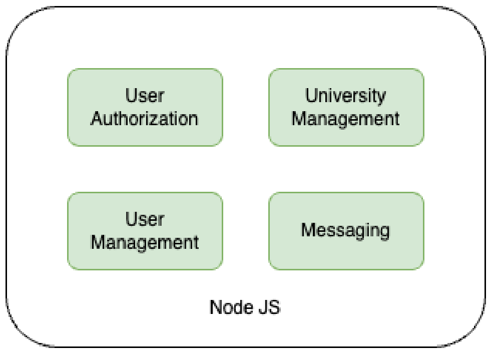
- **Application Tier:** The application tier is responsible for handling the business logic and processing user input. This tier includes the following subsystems:
    - **Controller:** This subsystem is responsible for managing the flow of data between the user interface and the data storage tier.
    - **Business Logic:** This subsystem is responsible for implementing the business logic of the website, such as user registration, post creation, and ranking evaluation.

- **Data Storage Tier:** The data storage tier is responsible for storing and managing data. This tier includes the following subsystems:
    - **Data Access Layer:** This subsystem is responsible for accessing and retrieving data from the database.
    - **Database:** This subsystem is responsible for storing all of the website's data, such as user information, course data, submission data, grade data.

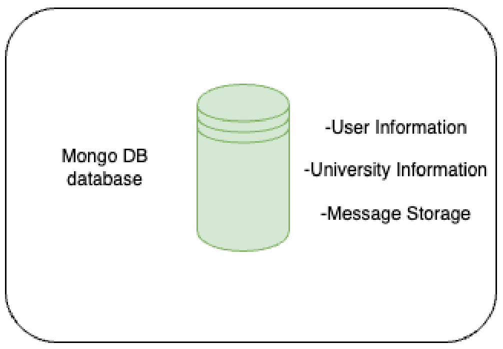
➔ Overall, this three-tier architecture allows for a clear separation of concerns between the presentation, application, and data storage tiers. The use of subsystems within each tier further helps to organize and modularize the codebase, making it easier to maintain and update in the future.



**Presentation Tier
(User Interface)**



**Application Tier
(Business Logic Layer)**



**Data Storage Tier
(Data Layer)**

Subsystems:

1. **User Management System:** This subsystem handles user registration, authentication, and authorization. It also manages user profiles.

2. **University Management System:** This subsystem manages the creation, management of Universities within the software. It allows admins and faculty to create and manage universities and its data. It also allows students to view resources of their university.

3. **Analytics and Reporting System:** This subsystem provides data analytics and reporting tools for professors and students as well as admin. It tracks and reports on faculties and user information.

4. **Communication system:** This subsystem allows faculties and students to communicate via chat system.