NextGen Scholars

Software Architecture:

- **Monolithic**: the monolithic architecture is ideal for the NextGen Scholars website as it is a client-side user interface that primarily promotes a single program, the 4-year plan creation algorithm. Additionally, the integration of a relational database management system within this architecture ensures strong data consistency, integrity, and easy handling of relationships between users, courses, and plans, all within a cohesive and unified system.

  Benefits:

  1. Feasible development plan for a small group, with defined roles such as frontend and backend developers, a monolithic architecture is beneficial because all components are developed together.
  2. Simple to test. This allows for easier integration and unit testing, ensuring the system works end-to-end.
  3. Simple to deploy, deployment is straightforward and reduces the complexity of managing multiple services or deployment pipelines.
  4. Simple to scale horizontally. As the NextGen Scholars platform grows and reaches more students, especially those from underserved backgrounds, the platform needs to scale to accommodate more users efficiently. A monolithic setup ensures this is possible without overcomplicating the architecture too early in the project's life cycle.

- **Decomposition**:

  Presentation Layer:

  - Built using HTML, CSS, JavaScript, and React for a dynamic, responsive, and engaging user interface.
  - REACT components are responsible for rendering the user interface dynamically based on the data received from the server and presenting it to the user in a user-friendly format.
  - REACT Router ensures smooth navigation between different pages like personalized profiles, academic resources, and 4 year plans.

    Explanation: The presentation layer is the user interface developed using REACT, This is where users interact with the system, input data, view their profiles, and interact with their 4 year plan creation algorithm. The use of REACT ensures the user interface is dynamic, responsive, and engaging, creating a seamless experience for students navigating between different sections of the platform.

  Application Layer:

  - React (front-end routing)
  - Node.js
  - Express.js

    Explanation: The application layer handles the flow of data between the presentation layer and the backend. Using Express.js as the core framework in the back-end allows us to handle routing and HTTP requests in a streamlined manner. React handles the front-end navigation and routing for a seamless user experience.

Business Logic Layer:

- Node.js
- JavaScript Logic

  Explanation: The business logic layer contains the core functionality of the platform. This is where we define the rules and processes for generating personalized content for students. Using Node.js for asynchronous operations allows us to manage multiple user requests and handle data efficiently.

Data Layer:

- MongoDB Database

  Explanation: This layer is responsible for handling all the data storage and retrieval tasks. The database stores user profiles, academic information, and plan data. The data is accessed by the controller when necessary, either to update the database or to retrieve data.

- **Model View Controller:** Allows the user to seamlessly interact with the NextGen Scholars website via a series of HTTP requests that sends commands to the controller. The controller then gathers information from the model (database) and sends it to the view so that it is visible to the user. For instance, if the user wishes to see information about AP courses tailored to his/her profile, they first signal the controller via the GUI. The controller will query data specific to the request from the MongoDB database and send that information back to the view (GUI). The user will then be able to see the information.
- **External environments**
  - State/County college resource websites
  - College Board API
  - Users (students, guardians, teachers)

System Context Model