**Class Diagram:**

Reason how class diagram will be made:

In MERN stack projects, while you may not use traditional object-oriented classes as defined in languages like Java or C++, you still define **data models**, **functional components**, and **service layers** that can be represented similarly to classes. Each of these components has attributes (data) and methods (functions), which are essentially what class diagrams are meant to capture.

**For Example:**

* **MongoDB Models (Schemas)**: These are essentially the equivalent of classes in an object-oriented paradigm.
  + You define entities like **User**, **Interview**, or **MCQ** that have attributes (e.g., name, score) and methods (e.g., validation, data fetching).
* **React Components**: In React, functional components are reusable pieces of the user interface. Though they don't define traditional classes, they encapsulate behavior and state, which can be represented as a class.
* **Node.js/Express.js Services**: Backend services can be represented by service classes that define logic for handling requests and manipulating data.

User, MCQ Mongodb Model

Score Calculation,(js service)

Resume Optimization (External services)