Steps Involved in Making the "notNTA" Project

The following document outlines the steps involved in the development of the "notNTA" project, which gamifies competitive exams like JEE and NEET. The project is developed in phases, with each phase building upon the previous to create a complete, engaging, and functional product. The steps include both frontend and backend development, real-time communication implementation, and game logic integration.

# Step 1: Planning and Design

Before starting the development process, it is essential to plan the overall structure of the game, define its features, and create a detailed design of the user interface and experience (UI/UX). This step involves:

* - Identifying the core functionalities (e.g., team-based quiz, question format, scoring system).
* - Defining the user interface for team selection, question display, and scoreboard.
* - Designing the backend architecture to handle user connections, game data, and real-time communication.

# Step 2: Frontend Development

In this step, the frontend of the application is developed, providing the interface through which users will interact with the game. Key tasks include:

* - Using technologies like React, Vue.js, or Angular to build responsive and dynamic UI components.
* - Implementing the question display, answer selection, and timers for each round.
* - Creating the team selection screen, and the user lobby where participants wait for the game to begin.

# Step 3: Backend Development

The backend is responsible for handling the logic of the game and storing data. Key tasks include:

* - Setting up a Node.js server using Express.js for managing API routes and handling game logic.
* - Integrating a database (MongoDB or PostgreSQL) to store user information, questions, scores, and game states.
* - Implementing the core game logic such as score calculation, team turns, and question retrieval from the database.

# Step 4: Real-Time Communication with Socket.IO

To ensure a smooth gaming experience, real-time communication is implemented using Socket.IO. This step involves:

* - Setting up WebSocket connections between the server and clients for real-time data exchange.
* - Implementing event listeners for user actions (e.g., answering questions) and broadcasting updates (e.g., score updates, next turn).
* - Managing game states on the server-side to ensure all players are in sync and game flow remains consistent.

# Step 5: Testing and Debugging

Once the frontend and backend are connected and real-time functionality is implemented, thorough testing is performed to ensure that the game works as expected. Tasks include:

* - Testing for correctness in game logic, such as scoring, timer functionality, and round progression.
* - Verifying that real-time communication is smooth and consistent between all connected users.
* - Debugging issues related to incorrect answers, disconnections, or any other game-breaking bugs.

# Step 6: Adding Game Features and Enhancements

After testing, additional features are implemented in later phases to enhance the gameplay experience. These enhancements include:

* - Adding animations, sound effects, and transitions to create a fully immersive gaming experience.
* - Implementing customizable game settings, including number of rounds, time per question, and question difficulty.