**Data Analyst**

**1. Data Model**

**This involves defining the structure of your data entities and relationships between them.**

**Schemas to Create:**

1. **Student Profiles:**
   * **Attributes: StudentID, Name, Grade, School, ParentContact, Email, Preferences, etc.**
   * **Example Schema:**

**json**

**Copy code**

**{**

**"StudentID": "String",**

**"Name": "String",**

**"Grade": "Integer",**

**"School": "String",**

**"ParentContact": "String",**

**"Email": "String",**

**"Preferences": ["String"]**

**}**

1. **Chapter Structure:**
   * **Attributes: ChapterID, Subject, Grade, ChapterName, ContentOutline, etc.**
   * **Example Schema:**

**json**

**Copy code**

**{**

**"ChapterID": "String",**

**"Subject": "String",**

**"Grade": "Integer",**

**"ChapterName": "String",**

**"ContentOutline": "String"**

**}**

1. **Exercise Definitions:**
   * **Attributes: ExerciseID, ChapterID, Type, Questions, Answers, etc.**
   * **Example Schema:**

**json**

**Copy code**

**{**

**"ExerciseID": "String",**

**"ChapterID": "String",**

**"Type": "String",**

**"Questions": ["String"],**

**"Answers": ["String"]**

**}**

1. **Question Bank:**
   * **Attributes: QuestionID, QuestionText, Options, Answer, Tags, etc.**
   * **Example Schema:**

**json**

**Copy code**

**{**

**"QuestionID": "String",**

**"QuestionText": "String",**

**"Options": ["String"],**

**"Answer": "String",**

**"Tags": ["String"]**

**}**

1. **Test Results:**
   * **Attributes: TestID, StudentID, Score, Date, DetailedPerformance, etc.**
   * **Example Schema:**

**json**

**Copy code**

**{**

**"TestID": "String",**

**"StudentID": "String",**

**"Score": "Integer",**

**"Date": "Date",**

**"DetailedPerformance": "String"**

**}**

1. **Performance Analytics:**
   * **Attributes: StudentID, TestHistory, WeakTopics, StrengthTopics, ImprovementGraph, etc.**
   * **Example Schema:**

**json**

**Copy code**

{

"StudentID": "String",

"TestHistory": ["String"],

"WeakTopics": ["String"],

"StrengthTopics": ["String"],

"ImprovementGraph": "String"

}

**2. Project Folder Structure**

**pariksha-mitra/**

**├── models/**

**│ ├── student.js**

**│ ├── chapter.js**

**│ ├── exercise.js**

**│ ├── question.js**

**│ ├── testResult.js**

**│ └── performance.js**

**├── routes/**

**│ ├── studentRoutes.js**

**│ ├── questionRoutes.js**

**│ └── analyticsRoutes.js**

**├── app.js**

**├── package.json**

**└── README.md**

**Data Processing**

**Develop algorithms for key functionalities:**

1. **Question Randomization:**
   * **Logic: Randomly shuffle questions while ensuring balanced difficulty levels.**
   * **Example (Python):**

**python**

**Copy code**

**import random**

**def randomize\_questions(questions, difficulty\_levels):**

**random.shuffle(questions)**

**return sorted(questions, key=lambda x: difficulty\_levels[x['QuestionID']])**

**2.Performance Calculation:**

* + **Logic: Calculate scores and analyze strengths/weaknesses.**
  + **Example:**

**python**

**Copy code**

**def calculate\_performance(test\_results):**

**total = sum([res['Score'] for res in test\_results])**

**return {**

**"TotalScore": total,**

**"AverageScore": total / len(test\_results)**

**}**

1. **Analytics Generation:**
   * **Logic: Generate insights like improvement graphs, top subjects, etc.**
   * **Example:**

**python**

**Copy code**

**def generate\_analytics(test\_results):**

**analytics = {**

**"WeakTopics": [],**

**"StrongTopics": [],**

**"ProgressGraph": []**

**}**

**# Process results to fill analytics**

**return analytics**

**4. Deliverables**

**Create the following:**

**1. Data Architecture Document**

* **Document your database schema, relationships, and data flow in a Markdown or PDF file.**

**2. MongoDB/NoSQL Schema Designs**

* **Define and implement schemas in JSON.**

**3. Data Flow Diagrams**

* **Use tools like Lucidchart or Draw.io to design how data moves between different entities.**

**4. Performance Optimization Recommendations**

* **Propose indexing strategies, caching, and optimized query patterns.**

**5. Submission Requirements**

**Ensure the following:**

1. **GitHub Repository:**
   * **Upload all project files, including code, diagrams, and documents.**
   * **Include a README.md with:**
     + **Project overview.**
     + **Setup instructions.**
     + **API details.**
2. **Deployed Application Link:**
   * **If required, deploy a demo version on a platform like Heroku or AWS.**
3. **API Documentation:**
   * **Use Postman or Swagger to document APIs with examples for CRUD operations**

**3.Data Storage Strategy**

1. **Choose database indexing strategies:**
   * **Index fields like StudentID, Grade, and Subject for quick lookups.**
   * **Example:**

**python**

**Copy code**

**db.students.createIndex({"StudentID": 1})**

1. **Ensure data privacy and security:**
   * **Encrypt sensitive data (e.g., passwords, contact info).**
   * **Implement role-based access control (RBAC).**
   * **Use token-based authentication (e.g., JWT) for API access.**

**4.Develop and Test APIs**

1. **Create REST APIs for CRUD operations:**
   * **Endpoints: /students, /chapters, /exercises, /questions, etc.**
   * **Use frameworks like Express.js (Node.js) or Flask (Python).**
2. **Document APIs with:**
   * **Postman: Include request/response samples.**
   * **Swagger: Generate API documentation.**
3. **Test APIs for edge cases and performance.**