DESIGN DOCUMENTATION

Tech Stack

- **Frontend**: React.js, Tailwind CSS (or other CSS framework)
- **Backend**: Node.js, Express.js
- **Database**: MongoDB (with Mongoose , Offline)

System Overview

The system provides a quiz platform for students and teachers. Teachers can create quizzes dynamically. Students log in, attempt quizzes, and receive detailed reports with hints,and can practice the questions that we have put on the website.

User Roles

1. Student

- Login/Signup
- View subjects/quizzes based on class, Solve Questions for practice.
- Attempt quizzes
- View reports of past attempts

2. Teacher

- Login/Signup
- Create dynamic quizzes (questions + hints)

- View student performance reports

Data Models

School

```
schoolId: string (pk)
schoolName: string
location: string
```

Student

```
studentId: string (pk)
name: string
phone: string
schoolId: string (ref: School)
password: string
class: string
quizAttempted: string[] (array of quizIds)
```

Teacher

```
teacherId: string (pk)
name: string
phone: string
schoolId: string (ref: School)
password: string
quizzesCreated: string[] (quizId)
```

Question

```
questionId: string (pk)
subject: string
class: string
options: string[]
correctAnswer: string
hintsId: string (ref: Hints)
```

Hints

```
hintId: string (pk)
text: string
visualLinks: string[]
```

Quiz

```
quizId: string (pk)
teacherId: string (ref: Teacher)
questions: string[] (ref: Question)
attemptedBy: string[] (ref: Student)
correct: number
incorrect: number
unattempted: number
```

Report

```
studentId: string (ref: Student)
quizId: string (ref: Quiz)
correct: number
incorrect: number
unattempted: number
quesitonSolved: string[]
```

App Flow

Student Flow:

1. Home → Student Login → Class Selection
2. Class selection → Choose Subject → List of Topics
3. Topics → Questions with Options + Hints
4. Attempt Quiz → Store result in `report` & `student.quizAttempted`

5. Generate report (with hints & stats)

Teacher Flow:

1. Home → Teacher Login
2. Create Quiz → Add Questions → Add Hints (Optional)
3. Submit Quiz → Save in DB → Associated with teacher
4. Monitor reports of student attempts

Authentication & Security(for now we are not including)

- JWT stored in HttpOnly cookies
- Routes protected via middleware:
    - `/api/teacher/*` → Teacher only
    - `/api/student/*` → Student only

API Design

Auth APIs

- `POST /api/auth/login` (student/teacher)
- `POST /api/auth/signup`
- `GET /api/auth/logout`

Student APIs

- `GET /api/quiz/:class` → Available quizzes by class
- `POST /api/quiz/attempt/:quizId` → Submit answers
- `GET /api/report/:studentId` → Past attempts

Teacher APIs

- `POST /api/quiz/create`
- `POST /api/question/create`
- `GET /api/report/:quizId` → Get students' reports

Features

- **Dynamic Quiz Activation**
- **Group Study Mode**
- **Detailed Report Generation**
- **Multimedia Hints (Text, image, Video(links))**
- **Topic-wise Question Bank**
- **Teacher-Centric Quiz Management**
- **School-Based Access Control**

Folder Structure (React vite + Express)

```
/client
  /src
    /components
```

```
    /pages
    /services
    App.tsx
    index.tsx

/server
  /controllers
  /routes
  /models
  /middleware
  server.js
```