perplexity

# ThinkRivals this is a good name will finalize this now, now give the detailed plan for both backend+frontend along with folder structures and all to create our web app, tell about all the routes and also

Great choice with "ThinkRivals"! Here's a detailed project plan including backend and frontend structure, folder organization, and essential routes for your AI quiz platform with the unique features you want to implement.

### ThinkRivals Project Plan

### 1. Tech stack

- **Backend:** Node.js, Express.js, MongoDB (Mongoose), WebSocket (e.g., Socket.io) for real-time features
- **Frontend:** React.js (or Next.js for SSR), Tailwind CSS for styling, context/hooks for state management
- **AI Integration:** OpenAI GPT API for question generation, explanations, adaptive logic
- **Authentication:** JWT-based user auth
- **Real-time:** Socket.io for multiplayer quizzes
- **Storage:** MongoDB for quizzes, user progress, reports

### 2. Backend Folder Structure

```
thinkrivals-backend/
│
├── config/              # DB connection, environment variables config
│   └── db.js
│
├── controllers/         # Request handlers for routes
│   ├── authController.js
│   ├── quizController.js
│   ├── userController.js
│   └── realtimeController.js
│
├── middleware/          # Middleware for auth, error handling, validation
```

```
│   ├── authMiddleware.js
│   ├── errorHandler.js
│   └── uploadMiddleware.js
│
├── models/             # MongoDB models (User, Quiz, Question, etc.)
│   ├── User.js
│   ├── Quiz.js
│   ├── Question.js
│   └── ChatSession.js
│
├── routes/             # Express routes
│   ├── authRoutes.js
│   ├── quizRoutes.js
│   ├── userRoutes.js
│   └── realtimeRoutes.js
│
├── services/           # Business logic & AI integrations here
│   ├── aiService.js
│   ├── pdfParserService.js
│   └── adaptiveLogicService.js
│
├── utils/              # Utility functions (token generation, validation)
│   └── helpers.js
│
├── sockets/            # Socket.io config & event handlers
│   └── quizSocket.js
│
├── app.js              # Express app setup
├── server.js           # Server & socket.io startup
└── .env                # Environment variables
```

### 3. Frontend Folder Structure (React / Next.js)

```
thinkrivals-frontend/
│
├── public/             # Static files (images, icons)
│
├── src/
│   ├── api/            # API calls to backend end points
│   │   ├── authApi.js
│   │   ├── quizApi.js
│   │   └── realtimeApi.js
│   │
│   ├── components/     # Reusable UI components (Buttons, Inputs, etc.)
│   │
│   ├── features/       # Features grouped (Quiz, User, Auth, Real-time)
│   │   ├── quiz/
│   │   │   ├── QuizMain.jsx
│   │   │   ├── QuizQuestion.jsx
│   │   │   └── ExplanationPanel.jsx
│   │   ├── auth/
│   │   ├── profile/
│   │   └── realTime/
│   │       └── MultiplayerQuiz.jsx
```

```
│   │
│   ├── hooks/          # Custom React hooks (useAuth, useQuiz)
│   │
│   ├── context/        # React Contexts for state management
│   │
│   ├── pages/          # Route pages (if Next.js)
│   │   ├── index.jsx
│   │   ├── login.jsx
│   │   ├── dashboard.jsx
│   │   └── multiplayer.jsx
│   │
│   ├── styles/         # Tailwind and custom CSS
│   │   ├── globals.css
│   │   └── tailwind.config.js
│   │
│   └── utils/          # Frontend utilities
│
├── package.json
└── tailwind.config.js
```

## 4. Backend API Routes

| Route | Method | Description | Auth Required |
|---|---|---|---|
| /api/auth/register | POST | Register user | No |
| /api/auth/login | POST | Login and receive JWT token | No |
| /api/auth/logout | POST | Logout user | Yes |
| /api/users/me | GET | Get current user profile | Yes |
| /api/quizzes/upload | POST | Upload PDF or content to generate quiz | Yes |
| /api/quizzes/generate | POST | Generate questions from uploaded content | Yes |
| /api/quizzes/:id | GET | Get quiz details/questions | Yes |
| /api/quizzes/:id/submit | POST | Submit quiz answers, receive report | Yes |
| /api/explanations/:qid | GET | Get explanation for question by id | Yes |
| /api/realtime/join | POST | Join a multiplayer quiz room | Yes |
| /api/realtime/start | POST | Start multiplayer quiz | Yes |
| /api/realtime/events | WS | WebSocket for real-time quiz events | Yes |

## 5. Key Features Workflow and Components

- **Adaptive Learning Path:** Backend tracks user progress, identifies weak topics, feeds quiz generation API with adaptive topics, frontend adjusts quiz based on feedback.
- **Multi-Modal Content Understanding:** PDF Parser service extracts text, images, and tables; AI service generates varied question types from extracted data.

- **ELI5 Explanation Mode:** Explanation API endpoint returns user-friendly, beginner mode or advanced explanation based on user preference.
- **Real-Time Collaborative Quizzes:** Socket.io backend manages room join/start; frontend syncs players' answers, scores, and final results in realtime.

Would you like me to help generate starter code files, design the database schema, or create detailed user stories based on this plan?