**Proof of Concept (PoC) Document for Yoga pose Guide App**

## **1.Project Writeup**

The Yoga Pose Guide is a web-based application designed to help users learn, track, and improve their yoga practice. It provides an interactive platform for users to explore various yoga poses, log their progress, and create personalized routines. The application offers an extensive library of poses with detailed descriptions, images, and difficulty levels. Additionally, users can set goals, monitor their achievements, and customize their practice routines. With visually appealing pages and an intuitive interface, the app is tailored to support yoga enthusiasts of all levels in their wellness journey.

## **2. Components of the Project**

The project consists of three main components:

**Frontend**

* **Framework**: React.js
  + - **Description**:
      * Provides an intuitive UI for exploring and learning yoga poses.
      * Enables users to log their practice sessions and track their progress.
      * Allows users to create and manage custom yoga routines.
      * Features responsive design with visually appealing pages to enhance user experience
* **Libraries Used**:
  + - * **React Router:** Used for navigation between different pages, such as the Home, Explore, Add Music, Tag Music, and Sort Music pages.
      * **React Hooks (useState, useEffect):** Used for managing the state of the music collection, tags, and sorting functionality.
      * **Basic CSS:** For styling and layout, including page backgrounds, buttons, and interactive elements to ensure a visually appealing and responsive design.
      * **JavaScript Array Methods (filter, sort):** Used for implementing tagging and sorting functionality dynamically.
      * **LocalStorage API:** Used to persist the music collection data, tags, and user preferences even after the page is refreshed.
      * **Material-UI**: Provides UI components and styling for the document interface.
      * **Socket.io**: Enables real-time communication between users, allowing changes

to be propagated instantly across connected users.

**Backend**

* Node.js & Express:
  + - Description:

The backend serves as the intermediary for real-time communication between clients, providing endpoints for managing users, documents, and collaboration sessions.

Socket.io will be integrated to manage real-time document updates and sync changes across all users.

* APIs:
  + - RESTful APIs to manage documents, users, and comments.
    - WebSocket endpoints (via Socket.io) to manage live updates and real-time data transfer between users.
* User Authentication:
  + - * OAuth/Firebase/Auth0: Handle user authentication, user roles, and permissions.
      * Store user session data to allow identification and document access control.

**Database**

* MongoDB/Firebase:

Description:

Store user information, document data (including content, structure, and metadata), and collaboration sessions.

MongoDB is preferred for flexible schema management (documents,user data).

Firebase: Real-time database for managing collaboration data with integrated authentication.

**Hosting Platform**

* Platform: Heroku/Netlify for backend deployment, GitHub Pages/Netlify for frontend deployment.
  + - * + Description:

Static site hosting for the React app on GitHub Pages/Netlify.

Backend hosting on Heroku or a similar platform that supports Node.js and real-time services.

## **3. Frontend Components**

The frontend is structured using React components:

* **Home.jsx**: A welcoming page with an introduction to the app and navigation options to explore poses, track progress, or create routines.
* **ExplorePoses.jsx**: Displays the yoga pose library with search and filter options by difficulty, category, or focus area.
* **TrackProgress.jsx**: Provides an interface for users to log their sessions and view their progress using visual charts or calendars.
* **CustomRoutines.jsx**: Allows users to create, edit, and save personalized yoga routines by selecting poses from the library.
* **PoseDetails.jsx**: Displays detailed information about a selected pose, including instructions, benefits, and tips.
* **UserContext.jsx**: Manages global state for user data, ensuring seamless interaction across components.
* **App.jsx**: Serves as the main routing structure, connecting all components through React Router.

## **4. Backend Components**

* Document Management:
  + - * The backend provides RESTful APIs for CRUD (Create, Read, Update, Delete) operations for documents.
      * Socket.io handles real-time synchronization of document changes.
* User Authentication and Permissions:
  + - * Manages user sign-up, login, and authentication through OAuth/Firebase/Auth0.
      * Role-based access control to manage user permissions (view/edit/comment) for each document.
* Real-Time Collaboration:
  + - * Socket.io facilitates real-time communication between users. It listens for changes made by users and updates the document content across all users instantly.
      * Change tracking: Keeps track of changes made by each user for document

version control.

## **5. Database Components**

* MongoDB/Firebase:
  + - Stores document metadata, including content, document versions, comments,

and collaboration logs.

* + - * Firebase Realtime Database or Firestore can be used for quick syncing between

users in real-time.

## **6. Hosting Platforms**

• Frontend: GitHub Pages/Netlify for static site hosting of the React application.

• Backend: Heroku/Netlify for deploying the backend service that supports real-time

communication and document management.

• Database: MongoDB or Firebase to manage document content and user sessions.

## **7. Flow Diagram of the Project**

1. User lands on the Home page and selects an option: Explore Poses, Track Progress, or Custom Routines.
2. In Explore Poses, they browse or filter yoga poses and view detailed information.
3. In Track Progress, they log completed sessions and view progress over time.
4. In Custom Routines, they create or manage personalized routines.
5. User data, including pose logs and routines, is stored persistently using the backend and database.

