**Documentation for Offline-First Lottie Animation Management System**

**Overview**

The Offline-First Lottie Animation Management System is a web application designed to allow users to upload, search, preview, and download Lottie animations. The application supports robust offline capabilities, ensuring that users can interact with animations and access detailed metadata even without an internet connection.

**Technology Stack**

**Frontend**

* **React**: JavaScript library for building user interfaces.
* **TypeScript**: Superset of JavaScript that adds static typing.
* **GraphQL**: Query language for your API.
* **IndexedDB**: Browser storage for large amounts of structured data.
* **Apollo Client**: State management library for managing GraphQL data.

**Backend**

* **Express**: Web application framework for Node.js.
* **TypeScript**: For type safety and development efficiency.
* **Apollo Server**: For building a GraphQL server.
* **GraphQL**: For defining the schema and resolvers.
* **MongoDB**: NoSQL database for storing animation data.

**Features**

1. **Animation Upload**: Users can upload Lottie animation files.
2. **Animation Retrieval**: Users can search for and retrieve animations.
3. **Search Animation**: Users can search animations based on various criteria.
4. **Download Animations**: Users can download animations to their local system.
5. **Preview Animation**: Users can preview animations within the application.

**Frontend Implementation**

* **React and TypeScript**: Used to build the user interface and ensure type safety.
* **GraphQL with Apollo Client**: Manages data fetching and state management for the app.
* **IndexedDB**: Provides offline capabilities by caching animations locally.
* **Service Workers**: Enhances offline functionality by caching files and API responses.

**Backend Implementation**

* **Express and TypeScript**: Server setup with type safety.
* **Apollo Server**: Manages the GraphQL API.
* **MongoDB**: Stores animation JSON data and user state.

**GraphQL Schema and Resolvers**

* **Animation Type**: Defines the structure of animation data.
* **Queries**:
  + searchAnimations(criteria: SearchCriteria): [Animation]: Searches animations based on criteria.
  + getAnimation(id: ID!): Animation: Retrieves a specific animation by ID.
* **Mutations**:
  + uploadAnimation(file: Upload!): Animation: Handles the upload of Lottie animation files.

**State Management**

* **Apollo Client**: Manages the state of the application, both online and offline.
* **IndexedDB**: Stores cached animations for offline access.

**Offline Capabilities**

* **Service Workers**: Ensure that the application can function offline by caching necessary files and data.
* **IndexedDB**: Stores animations and metadata locally to be accessed offline.

**Performance Optimizations**

* **Code Splitting**: Ensures that the application loads efficiently by splitting the code into manageable chunks.
* **Caching**: Uses IndexedDB and Service Workers to cache animations and other necessary data for offline use.

**Additional Notes**

* **Design Decisions**: Documented in the repository for clarity.
* **API Schema**: Clearly defined in the schema.graphql file.
* **Code Quality**: Emphasis on clean, maintainable code with proper documentation.

**Conclusion**

This documentation outlines the architecture and implementation details of the Offline-First Lottie Animation Management System. The system leverages modern web technologies to provide a robust solution for managing Lottie animations with offline support.