**Architecture Document**

1. Application
   1. Microservices
   2. Event-Driven
   3. Serverless
2. Database
   1. ER Diagram
   2. Schema Design
3. Data Exchange Contract
   1. Frequency of data exchanges
   2. Data Sets
   3. Mode of Exchanges (API, File, Queue etc.,)

**1. Application Architecture**

**1.1 Microservices**

* **Scope:  
  Although MindMaster is primarily an Android app, it could interact with external microservices if it includes features such as task synchronization with cloud storage or user authentication.**
* **Potential Services:**
  + **Authentication Service: OAuth or Firebase Authentication**
  + **Task Management Service: Backend API to store tasks remotely (if integrated with a cloud platform like Firebase or a custom server)**

**1.2 Event-Driven Architecture**

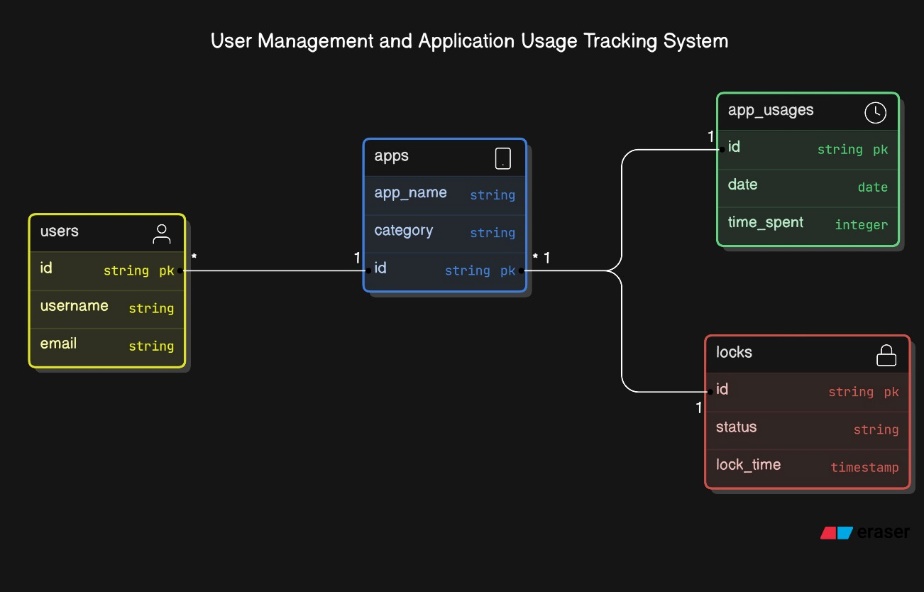
* **Event Sources:**
  + **User events (clicks, notifications, task completion) trigger actions locally.**
  + **Push notifications via Firebase Cloud Messaging (FCM) may be used to notify users of task deadlines.**
* **Event Consumers:**
  + **Background services in the app listen for events (e.g., daily reminders).**

**1.3 Serverless Components**

* **Usage (if applicable):**
  + **Cloud Functions: Potentially used for lightweight backend logic, such as sending notifications or syncing data.**
  + **Firebase Firestore: If the app syncs data to the cloud, this could be used for serverless database management.**

**2. Database Architecture**

**2.1 ER Diagram**

****

**If the app uses an SQLite database for local data storage, a simple Entity-Relationship (ER) diagram might include the following entities:**

* **User (if accounts are needed)**
* **Task (Task ID, Description, Status, Deadline)**
* **Category (Optional grouping of tasks)**

**Sample relationships:**

* **User → Manages multiple Tasks**
* **Task → Belongs to a Category**

**2.2 Schema Design**

**Here’s a basic schema design for local storage using SQLite:**

* **Table: Task**
  + **task\_id (Primary Key)**
  + **title (Text)**
  + **description (Text)**
  + **status (Text) – e.g., Pending, Completed**
  + **deadline (Datetime)**
* **Table: Category**
  + **category\_id (Primary Key)**
  + **name (Text)**
  + **description (Text)**
* **Table: User (if required)**
  + **user\_id (Primary Key)**
  + **username (Text)**
  + **email (Text)**

**2.3 Data Exchange Contract**

**If the app exchanges data with a backend server or external API:**

* **Input/Output Format: JSON or XML**
* **Example JSON Contract (for a task object):**

**json**

**Copy code**

**{**

**"task\_id": 1,**

**"title": "Finish project",**

**"description": "Complete the final report for the project",**

**"status": "Pending",**

**"deadline": "2024-10-30T18:00:00Z"**

**}**

**3. Data Exchange Architecture**

**3.1 Frequency of Data Exchanges**

* **Task Syncing: Data could be exchanged daily or in real-time, depending on the task synchronization setup (if cloud storage is used).**
* **Push Notifications: Sent as events occur, such as reminders or task status changes.**

**3.2 Data Sets**

* **User Information (optional, if accounts are used)**
* **Tasks Data (titles, descriptions, deadlines, and statuses)**
* **Categories (if tasks are grouped)**

**3.3 Mode of Exchanges**

* **API:**
  + **The app may connect to a backend API for syncing tasks or user data.**
  + **Example: REST API endpoint /tasks for CRUD operations.**
* **File:**
  + **Users could export tasks to CSV or JSON for backup purposes.**
* **Queue:**
  + **If the app supports push notifications or reminders, a message queue (like Firebase Messaging) could manage notification delivery.**

**Conclusion**

**This document outlines the high-level architecture of the MindMaster app. While primarily an Android productivity app, it offers opportunities for integration with external services through microservices or serverless functions. The database design supports both local and remote synchronization of tasks, and data exchange is managed via APIs and events for a seamless user experience.**