Decision-Making Report

1. Image Storage

1.1. Selection Criteria:

For storing images, I have selected Firebase Storage for its numerous advantages:

- **Firebase Storage:**
  - **Ease of Integration:** Firebase Storage easily integrates with Flutter through the Firebase SDK, allowing quick setup and efficient development. Seamlessly integrates with other Firebase services like Authentication and Firestore, simplifying development and reducing overhead.
  - **Real-time Updates:** Firebase offers real-time synchronization capabilities, making it ideal for applications that require immediate updates in image data.
  - **Pricing:** With a pay-as-you-go pricing model, it is budget-friendly, especially for startups or small applications.
  - **Scalability:** Built on Google Cloud Platform's robust infrastructure, providing excellent scalability and high availability.
  - **Region Availability:** Extensive global coverage with data centers in multiple regions, ensuring low latency for users worldwide.
  - **Security:** Firebase provides robust security features, including Firebase Authentication for user-specific access to images.

1.2. Comparison:

- **Firebase Storage vs. Amazon S3:**
    - **Firebase Storage:**
        - **Pros:** Simplified integration with the Firebase ecosystem, automatic scalability, and real-time updates for users.
        - **Cons:** Can be more expensive than S3 for large-scale storage needs, limited control over specific configurations compared to S3.
    - **Amazon S3:**
        - **Pros:** Highly scalable and cost-effective for large datasets, supports complex configurations, and offers more extensive security features.
        - **Cons:** Requires more setup and configuration time, which can be a barrier for rapid development, more complex to integrate with other services compared to Firebase.

1.3. Implementation Plan:

The plan for storing and retrieving images using Firebase Storage is as follows:

- **Dependencies: Add necessary Firebase dependencies to `pubspec.yaml`.** `firebase_storage`
- **Configuration:** Initialize Firebase in the Flutter app.
- **Integration:** Utilize the official Firebase Storage SDK for Flutter.
- **Data Flow:**
    1. **Upload:**
        - Capture images using the camera or gallery using image picker.
        - The app uses the `FirebaseStorage` SDK to upload the image to a designated storage bucket.
        - Store the download URL of the uploaded image in the relevant entities, such as "recipe" or "profile" in the database.

2. **Retrieval:**
    - Fetch the download URL from the database.
    - Display images in the app using network image widgets with the retrieved URLs.

2. Database

2.1. Selection Criteria:

- **Firebase Firestore:**
    - **Real-time Database:** Ideal for applications requiring real-time updates, such as collaborative features or live feeds (though not strictly necessary for a recipe app).
    - **Flexible Data Modeling:** Supports NoSQL document structure, allowing for flexible and dynamic data organization.
    - **Ease of Use:** Works seamlessly with Firebase Storage to link stored images to corresponding data entities.

2.2. Comparison:

- **Firebase Firestore vs. Realtime Database:**
    - **Firestore:**
        - **Pros:** Scalable, flexible data modeling, offline support, security rules, stores data as collections of documents.
        - **Cons:** Can be more expensive than Realtime Database for large datasets, slightly more complex to query.
    - **Realtime Database:**
        - **Pros:** Simpler data model, better for real-time updates, generally less expensive, stores data as one large JSON tree.
        - **Cons:** Less scalable than Firestore, limited data modeling flexibility.

2.3. Implementation Plan:

- **Firebase Setup:** Enable Firestore from the Firebase project.
- **Data Structure:**
  - Design collections for "recipes" and "users”.
  - Each document in the "recipes" collection will represent a single recipe.
  - Each document will contain fields such as:
    - `recipeId`
    - `name`
    - `imageUrl` (storing the download URL from Firebase Storage)
    - `ingredients` (an array of strings or objects)
    - `instructions` (an array of strings or objects)
    - `Likes` (Object)
    - `Comments` (a list of object)
  - **Integration Steps:**
    - Add Firestore dependencies to Flutter. Such as (`firebase_core, cloud_firestore`)
    - Create CRUD methods to create, read, update and delete data from Firestore.
  - **Data Flow:**
    - When an image is uploaded to Firebase Storage, retrieve the link and store it in the associated recipe or profile entity in Firestore.
    - Fetch and display the data with associated image links in the app, ensuring a dynamic experience as updates occur.