Document-Level Permissions in Appwrite

What are Document-Level Permissions?

Document-level permissions allow you to define access control for individual documents within a collection. This enables fine-grained control over who can read, update, or delete specific documents in a database.

Why Use Document-Level Permissions?

Document-level permissions are essential for scenarios where:

- **Data is user-specific**: Each user needs isolated access to their data, such as user profiles, saved items, or private content.
- **Granular control is required**: Permissions need to be restricted to specific users or groups for each document.
- **Enhanced security**: Protect sensitive data by isolating document access per user or group.

Broad-Level Collection Permissions vs. Document-Level Permissions

Aspect	Broad-Level Permissions	Document-Level Permissions
Definition	Access control is applied to the entire collection.	Access control is applied individually to each document.
Use Case	Public data or shared data with similar permissions for all users.	Private or user-specific data requiring granular access control.
Customization	Limited to collection-wide rules.	Fully customizable for each document.
Security	Broad and generalized access.	Granular and secure, isolating access per user or group.
Example	A collection of public recipes where all users have read access.	User profiles where only the document owner can read or update.

When to Enable Document-Level Permissions

You should enable document-level permissions if:

1. **User-Specific Data Exists**:
    - Each user needs access to their data, such as user profiles, saved recipes, or private notes.
2. **Sensitive or Private Data**:
    - Data must be isolated per user to maintain security and privacy.
3. **Personalized Features**:
    - Features like "Save Favorite Recipes" or "Generate Shopping List" require user-specific storage and access.

Example: MealMind Use Case

- **Users Collection (`usersCredentials`)**:
  - Contains user profiles (username, avatar, bio).
  - Only the profile owner should be able to view or edit their profile.
- **Saved Recipes Collection (`savedRecipes`)**:
  - Stores recipes favorited by users.
  - Each user should access only their saved recipes.
- **Shopping Lists Collection (`shoppingLists`)**:
  - Contains shopping lists created by users.
  - Each user must manage their own shopping list.

What Happens When You Enable Document-Level Permissions?

1. Each document can have **custom permissions**.
2. Users can only access documents they are authorized to interact with.
3. You gain complete control over read, update, delete, and write operations per document.

Key Benefits

- **Granular Access Control**: Isolate access per user or group.
- **Improved Security**: Restrict unauthorized access.
- **Flexibility**: Define permissions per document for custom use cases.

Example Permissions Configuration

Users Collection (`usersCredentials`):

Each user can only access their profile:

```
{
  "read": ["user:<USER_ID>"],
  "update": ["user:<USER_ID>"],
  "delete": ["user:<USER_ID>"]
}
```

Saved Recipes Collection (`savedRecipes`):

Each user can only manage their own saved recipes:

```
{
  "read": ["user:<USER_ID>"],
  "update": ["user:<USER_ID>"],
  "delete": ["user:<USER_ID>"]
}
```

How to Apply Document-Level Permissions Programmatically

Use the **Appwrite SDK** to define permissions when creating or updating a document.

Example: Create a Document with Permissions

```
const savedRecipe = await databases.createDocument(
    'your-database-id',
    'savedRecipes',
    'unique()',
    { recipeName: "Pasta Carbonara", userId: "user:12345" }, // Document data
    [
        Permission.read('user:12345'),    // Only this user can read
        Permission.update('user:12345'), // Only this user can update
        Permission.delete('user:12345')  // Only this user can delete
    ]
);
```

Summary

- **Enable Document-Level Permissions** for collections that store user-specific data or require granular control.
- Use **broad-level permissions** only for public or shared data that doesn't require isolation.
- MealMind, for example, needs document-level permissions to ensure users can only access and manage their profiles, saved recipes, and shopping lists.

By properly configuring these permissions, you can enhance the security and usability of your app while providing a personalized experience for your users.