Database Design

WeEat

7/21/2024

Kayla Duffy

SWDV 691

Overview

For my web application, I'm using Firebase Firestore to handle all data persistence. Firestore is a NoSQL document database that's perfect for my needs because it offers scalability, real-time updates, and a flexible data model. This flexibility is essential given the dynamic and hierarchical nature of the data structures in my application. Plus, Firestore's seamless integration with Firebase Auth makes user authentication straightforward and secure.

The web application will interact with Firestore through a REST API layer, which is accessed by the frontend portion of the project. Each request that involves data from the database will go through the API, ensuring a clear separation of concerns and adherence to best practices for web application design and development.

Data Specifications

The database will include the following collections:

- Users
- Created Menus
- Saved Menus
- Restaurants
- Reviews *(new for stretch feature*)

Each collection will be structured in JSON format, detailing the data types and relationships where applicable.

Justification

Firestore is an ideal choice for my application because of its flexibility, scalability, and real-time capabilities. The document-based structure makes it easy to store hierarchical data like menus and restaurant details. The integration with Firebase Auth ensures that user authentication and data access are secure and straightforward. Firestore's real-time updates enhance the user experience by providing instant updates to menu items and recommendations.

By designing the data structures carefully and leveraging Firestore's features, I can efficiently manage user data, menus, and recommendations while keeping the codebase clear and maintainable.

Revisions

Incorporating User Reviews (*Stretch*):

- **Purpose:** To allow users to leave feedback and rate restaurants they have visited, enhancing the user experience and providing valuable information to other users.
- **Implementation:** Added a reviews subcollection under both the users and restaurants collections to store user reviews and ratings for specific restaurants.
- **Justification**: Including reviews makes the app more interactive and community-driven. It encourages user engagement and provides additional data points for restaurant evaluations and recommendations.

Storing Restaurant Thumbnails (*MVP*):

- **Purpose:** To visually enhance the user interface by displaying restaurant images alongside their details.
- **Implementation:** Added thumbnailUrl fields in the savedMenus and restaurants collections and photoUrl to createdMenus
- **Justification:** Visual content significantly improves user experience by making the app more appealing and easier to navigate. Thumbnails and photos help users quickly identify restaurants and their menus

Adding Preferred Locations

- **Purpose:** To allow users to save and manage their preferred restaurant locations.
- **Implementation:** Added a preferredLocations subcollection under the users collection, including details like address, coordinates, and photo URL.
- **Justification:** Providing a way for users to save their favorite restaurant locations enhances personalization and convenience. It helps users quickly access and manage places they frequently visit.

Collections

Users Collection

Pupose: The Users collection is essential for managing user-specific information such as profile details, preferences, and allergen information. This collection ensures that each user has a personalized experience based on their stored data. It also is home to two subcollections for the user which contain their saved menus and created menus. *(I will show those separately, as well as a full view of the entire structure).*

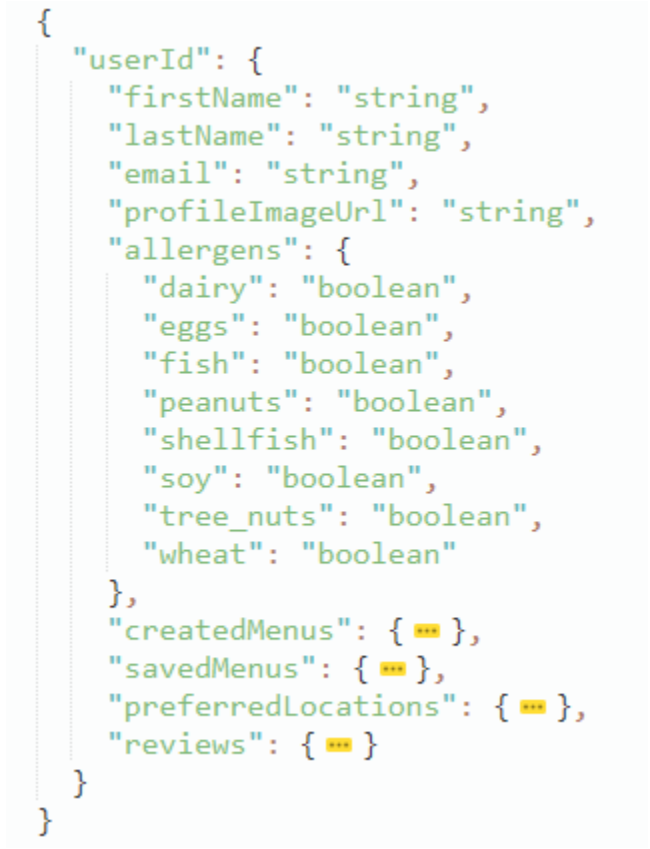
Implementation: When a user registers or logs into the application, their details are stored in the Users collection. Each user document contains personal information, including their allergen preferences. This data is crucial for filtering out menu items containing allergens

Role: The Users collection is essential for managing user authentication and personalization. It stores user preferences that influence menu recommendations and search results.

Interaction:

- **Profile Page:** Users can view and edit their profile information, including their allergen preferences.
- **Menu Filtering:** The application uses the allergen information to filter out unsuitable menu items when displaying menus.
- **Recommendations:** Personalized recommendations are generated based on the user's preferences and previous interactions.

Revised Document Structure:



Created Menus Collection (subcollection)

Purpose: This collection stores menus created by users, including details about the dishes and their categories

Implementation: Users can create and manage their custom menus. Each created menu document includes a list of dishes with detailed information such as name, description, allergens, and category. When the user creates a menu, if they don't upload an image for the thumbnail, then the Google Places API will fetch that photo from that location's specific Google Places listing.

Role: The Created Menus collection enables users to create and manage their own menus. It allows users to add, update, and delete menu items, which are then displayed in the application.

Interaction:

- **Create Menu:** Users can create a new menu by adding dishes to it.
- **Edit Menu:** Users can update or delete dishes within their created menus.
- **View Menu:** Users can view their custom menus and the details of each dish.

Revised Document Structure:

```
"createdMenus": {
  "menuId": {
    "restaurantName": "string",
    "thumbnailUrl": "string",
    "dishes": [
      {
        "dishId": "string",
        "name": "string",
        "description": "string",
        "category": "string",
        "allergens": [
          "string"
        ]
      }
    ]
  }
},
```

Saved Menus Collection (subcollection)

Purpose: This collection stores menus saved by users, allowing them to keep track of their favorite dishes and restaurants.

Implementation: When a user saves a menu, the menu details are stored in the Saved Menus subcollection. Each saved menu document includes information about the restaurant and the dishes it contains. Each menu is associated with a logo for that restaurant which is stored in the restaurants database.

Role: The Saved Menus collection allows users to save and organize their favorite menu items, providing quick access and the ability to add personal notes.

Interaction:

- **Save Menu:** Users can save menus from restaurants to their profile.
- **Edit Note:** Users can add and update notes on dishes within their saved menus.
- **View Saved Menus:** Users can view their saved menus and notes, making it easy to revisit favorite dishes.

Document Structure:

```
"savedMenus": {
  "savedMenuId (auto-ID)": {
    "restaurantName": "string",
    "dishes": [
      {
        "id": "string (auto-ID)",
        "name": "string",
        "description": "string",
        "allergens": ["string"],
        "note": "string",
        "category": "string"
      }
    ]
  }
}
```

REVISION: Preferred Locations (subcollection) - MVP

Purpose: This collection stores user-preferred restaurant locations.

Implementation: When a user adds a preferred location, the location details are stored in the Preferred Locations subcollection.

Role: The Preferred Locations collection allows users to manage their preferred restaurant locations, which can be used for quick access and recommendations.

Interaction:

- **Add Preferred Location:** Users can add a new preferred location.
- **Edit Preferred Location**: Users can update details of an existing preferred location.
- **View Preferred Locations:** Users can view their list of preferred locations.

Document Structure:

```
"preferredLocations": {
  "locationId": {
    "name": "string",
    "address": "string",
    "coordinates": {
      "latitude": "number",
      "longitude": "number"
    },
    "photoUrl": "string"
  }
},
```

STRETCH: Reviews (subcollection)

Purpose: This collection stores user reviews for restaurants.

Implementation: When a user writes a review, the review details are stored in the Reviews subcollection.

Role: The Reviews collection allows users to write providing feedback and ratings for specific restaurant locations.

Interaction:

- **Add Review:** Users can write a new review for a restaurant location
- **Add Rating:** Users can leave a rating for a restaurant
- **View Reviews**: Users can view all reviews for a specific restaurant location

Document Structure:

```
"reviews": {
  "reviewId": {
    "restaurantName": "string",
    "address": "string",
    "rating": "number",
    "review": "string",
    "userId": "string"
  }
}
```

Full Users Collection Document Structure


```
{
  "userId": {
    "firstName": "string",
    "lastName": "string",
    "email": "string",
    "profileImageUrl": "string",
    "allergens": {
      "dairy": "boolean",
      "eggs": "boolean",
      "fish": "boolean",
      "peanuts": "boolean",
      "shellfish": "boolean",
      "soy": "boolean",
      "tree_nuts": "boolean",
      "wheat": "boolean"
    },
    "createdMenus": {
      "menuId": {
        "restaurantName": "string",
        "thumbnailUrl": "string",
        "dishes": [
          {
            "dishId": "string",
            "name": "string",
            "description": "string",
            "category": "string",
            "allergens": [
              "string"
            ]
          }
        ]
      }
    },
    "savedMenus": {
      "menuId": {
        "restaurantName": "string",
        "dishes": [
          {
            "dishId": "string",
            "name": "string",
            "description": "string",
            "category": "string",
            "allergens": [
              "string"
            ]
          }
        ]
      }
    },
    "preferredLocations": {
      "locationId": {
        "name": "string",
        "address": "string",
        "coordinates": {
          "latitude": "number",
          "longitude": "number"
        },
        "photoUrl": "string"
      }
    },
    "reviews": {
      "reviewId": {
        "restaurantName": "string",
        "address": "string",
        "rating": "number",
        "review": "string",
        "userId": "string"
      }
    }
  }
}
```

Restaurants Collection

Purpose: This collection stores information about restaurants, including their menus.

Implementation: Each restaurant document includes the restaurant's name and a subcollection of menu categories. Each category document contains the dishes offered by the restaurant.

Role: The Restaurants collection provides the core data for displaying restaurant menus and details within the application. It is used for search results and menu displays.

Interaction:

- **View Restaurant Menus:** Users can browse through the restaurants and view their menus.
- **Search:** Users can search for restaurants based on location, radius, and keywords.
- **Recommendations:** The application provides restaurant recommendations based on user preferences and saved/created menus.

Document Structure:

Data Relationships and Justifications

User Data

User Authentication and Preferences: Storing user data, including allergen preferences, in the Users collection allows for personalized experiences and secure authentication through Firebase Auth.

- **Purpose and Interaction:**
    - **Personalized Experiences**: Storing allergen preferences allows the application to filter menu items and recommendations based on user-specific dietary restrictions, enhancing the overall user experience.
    - **Secure Authentication**: Using Firebase Auth, user authentication is secure and streamlined, ensuring that only authenticated users can access and modify their data. This integration also simplifies the management of user sessions and access control.

Menu Data

Menu Management: Separating created and saved menus into distinct collections ensures clear organization and easy management of user-generated content.

- **Purpose and Interaction:**
    - **User-Created Menus:** Users can create and customize menus by adding dishes, which are stored in the Created Menus subcollection. This functionality supports personalized menu creation and management, allowing users to curate their dining experiences.
    - **Saved Menus:** Users can save menus from various restaurants to easily access their favorite dishes. The Saved Menus subcollection ensures that users can revisit and manage their preferred menus efficiently.
    - **Note Editing:** Users can add or update notes on dishes within their saved menus, enhancing their ability to track preferences and experiences.

Restaurant Menu Data

Restaurant Information: Centralizing restaurant data in the Restaurants collection facilitates efficient querying and integration with external APIs (e.g., Google Places API). This approach allows the application to offer comprehensive and up-to-date restaurant information, enhancing the user experience with accurate and relevant data.

- Purpose and Interaction:
    - **Efficient Querying:** By centralizing restaurant data, the application can efficiently query restaurant menus and details. This structure supports quick data retrieval and enhances the performance of the application.
    - **Integration with External APIs**: The centralized data model facilitates seamless integration with external APIs like the Google Places API. This integration allows the application to provide users with comprehensive and accurate restaurant information, including location, contact details, and menu items.
    - **Real-Time Updates:** Firestore's real-time capabilities ensure that any changes in restaurant data are immediately reflected in the application. This feature is crucial for maintaining up-to-date menus and recommendations for users.