Database Design: Meals Weekly

My choice of database technology to use for my web application is MongoDB. Originally, I was envisioning using a relational database, but I realized that it would be overcomplicated. To recap, my app, Meals Weekly, is a meal planning app that also includes shopping list functionality. The data needed for the project is user-provided meal items for each day of the week, as well as a shopping list to go with each week. Users can add new week plans as they choose to. Because of this, a relational database would require a sort of layered design, containing multiple many-to-many relationships (e.g. many meals to may days). Using a document-based structure ends up making a lot more sense, as each week can simply have its own document, split up into days. Meals won’t need to be separate tables, since they’ll simply fall under one data point for a specific day. Flexibility is also very important if I want to add the capability of adding extra meals or entire weeks. It will simply be added as another document data entry. The only potentially complicated component is the relationship between ingredients and meals, though for the MVP, I think the shopping list and meal planner can be separate entities. Users can input a meal as a tag for a specific ingredient, rather than it being done automatically, a feature that could come in later releases,

The structure of each MongoDB JSON document will be similar to as follows:

{

\_id: ObjectID(“1234567890”),

week: “Week 1”,

day: “Sunday”,

breakfast: [“Scrambled eggs”, “Waffles”],

lunch: [“Turkey sandwich”],

dinner: [“Grilled chicken breast”, “Jasmine rice”]

}

In practice, I will likely group multiple days under one week, instead of having separate sets for each day. In any case, the document structure will remain largely the same. Users will input meal data in a form, providing information similar to what’s in the example (like “Scrambled eggs”). Once submitted, the main UI for the calendar view would pull the breakfast, lunch, and dinner items from these documents in order to display to the user. Each day will have the same structure as the above example, regardless of the week it falls under. This will be best for the user personas because it’s a relatively simple data structure while still being flexible enough for all potential user needs in the MVP.

The second type of document needed is for shopping or ingredients lists:

{

\_id: ObjectID(“1234567890”),

week: “Week 1”,

ingredients: {

{item: “Eggs”, qty: 2, tags: {“Sunday Breakfast”}},

{item: “Flour”, qty: 1, tags: {“Sunday Breakfast”}},

{item: “Deli Turkey”, qty: 1, tags: {“Sunday Lunch”}}

}

The document structure for the shopping list is also relatively simple. Aside from the ID, it includes the week that the list is for and an embedded list of ingredients and their attributes. Users will input each of these separate attributes in a form. Though this structure is technically unbounded and there can be unlimited items in the list, I don’t think it will end up being an issue. Allowance for infinite pieces of data could be a problem if there would realistically be an excessively large amount, but users wouldn’t realistically put in more than a few dozen pieces of data.