Brad Bosak

Database Design

Type

My application will persist two main objects: the event and to-do item. The event and to-do item are both very simple objects that are well suited to a document-based storage. Since the event and to-do item objects are separate, there are not very many relationships and the way the data is display on the screen, a document-based approach is best.

EVENT Table

```
{
  "id": "string",
  "date": "date",
  "time_st": "string",
  "time_end": "string",
  "loc": "string",
  "menu": {
    "item": "string",
    "assignee": "string"
  },
  "decoration": {
    "item": "string",
    "assignee": "string"
  },
  "guest_list": {
    "first_name": "string",
    "last_name": "string"
  },
  "supply": {
    "item": "string",
    "assignee": "string"
  }
}
```

The EVENT table will have a unique identifier as well as some simple attributes. They are all string datatypes except the date of the event which is a date. There are some nested objects within the document like menu, decoration, guest_list, and supply which all might have multiple items and assignees. This is to simulate how an event might have multiple menu items each with a person assigned to them. It should be relatively easy to loop through all these and display them nicely in the UI.

TODO Table

```
{
  "id": "string",
  "descr": "string"
}
```

The TODO table is extremely simple only needing to store a to-do item's description. Expanding past my minimum viable product, I might add an assignee and some other features to the to-do object which might change the data. This is another reason to choose document-based storage to easily be able to add data without changing multiple tables.