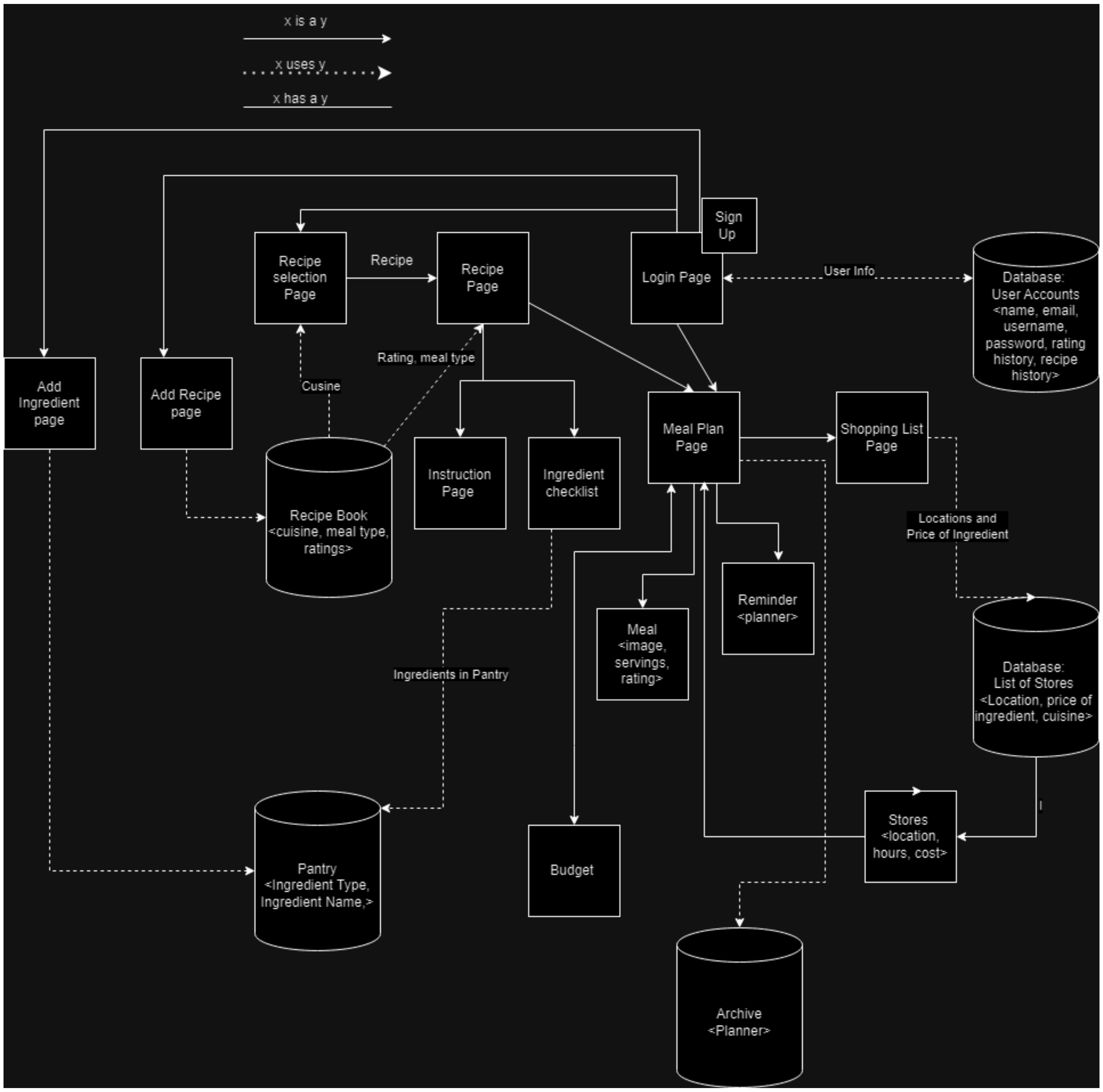
My proposition for Data Management is to utilize NoSQL. Having information stored in Documents in a cloud service such as MongoDB. Less Strain on our local resources and guaranteed data integrity in the event of an emergency on our servers. Given our Software's requirement to be able to manage sessions for creating recipes and constantly updating the state of an accounts budget, ingredients, recipes, and planners. This is to ensure FindMyMeal is accurate in reminding the user of things such as when recipes are or are not craftable, when ingredients are no longer in their pantry and when the budget is low. This complex behavior combined with the fact that it will be replicated as more users join.

Proposed Updated Architecture Diagram:


x is a y
x uses y
x has a y
Sign Up
Recipe selection Page
Recipe
Recipe Page
Login Page
User Info
Database: User Accounts <name, email, username, password, rating history, recipe history>
Rating, meal type
Cusine
Add Ingredient page
Add Recipe page
Instruction Page
Ingredient checklist
Meal Plan Page
Shopping List Page
Recipe Book <cuisine, meal type, ratings>
Locations and Price of Ingredient
Reminder <planner>
Meal <image, servings, rating>
Ingredients in Pantry
Database: List of Stores <Location, price of ingredient, cuisine>
Stores <location, hours, cost>
Pantry <Ingredient Type, Ingredient Name,>
Budget
Archive <Planner>

Data management for FindMyMeal would need the following databases. A database for Ingredients (“Pantry”). A database for Recipes (“Recipe Book”). A database for User Information. A database for Planners. A database for Stores.

Sample code to connect to MongoDB

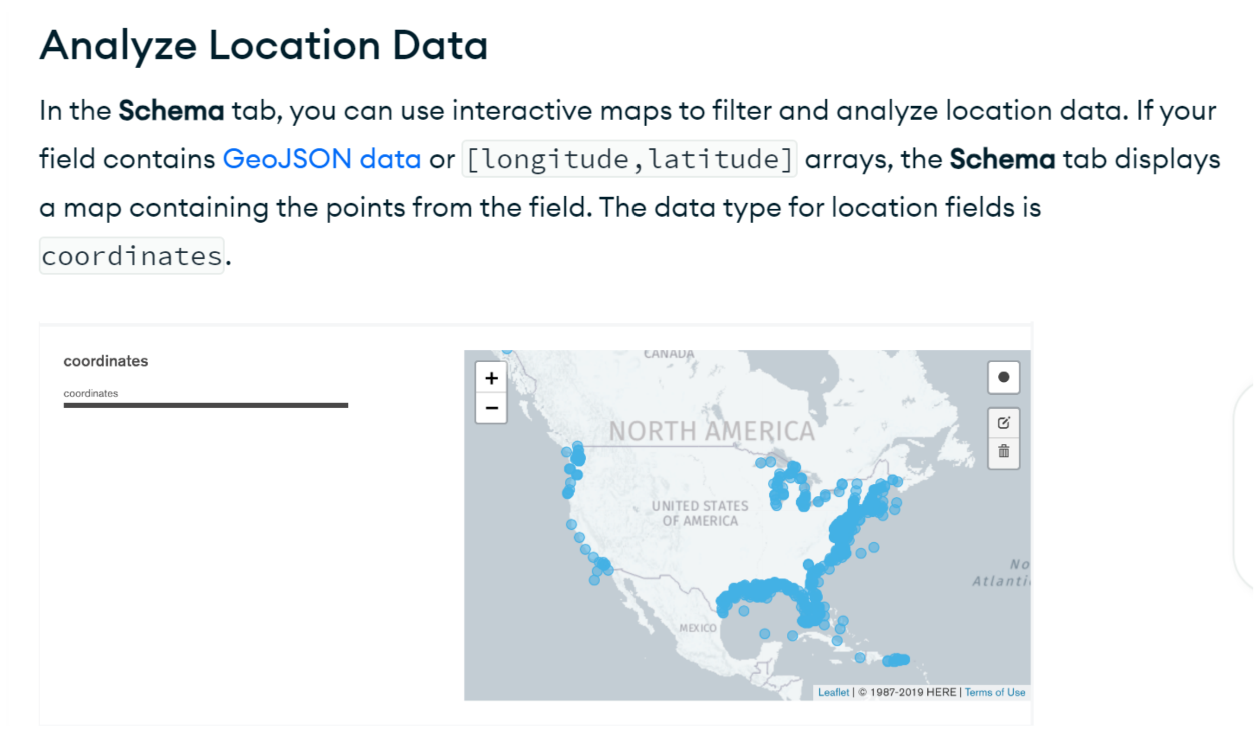
```
const { MongoClient } = require('mongodb')

let dbConnection

module.exports = {
  connectToDb: (cb) => {
    MongoClient.connect('mongodb://localhost:27017/bookstore')
      .then(client => {
        dbConnection = client.db()
        return cb()
      })
      .catch(err => {
        console.log(err)
        return cb(err)
      })
  },
  getDb: () => dbConnection
}
```

To ensure security of User’s data. Store hashed passwords on MongoDB through Mongoose. Hashing passwords can be achieved using bcrypt.js.

```
const mongoose = require("mongoose")
const bcrypt = require("bcryptjs")

const UserSchema = new mongoose.Schema({
  username: String,
  password: String
})

UserSchema.pre("save", function (next) {
  const user = this
  if (this.isModified("password") || this.isNew) {
    bcrypt.genSalt(10, function (saltError, salt) {
      if (saltError) {
        return next(saltError)
      } else {
        bcrypt.hash(user.password, salt, function(hashError, hash) {
          if (hashError) {
            return next(hashError)
          }

          user.password = hash
          next()
        })
      }
    })
  } else {
    return next()
  }
})

module.exports = mongoose.model("User", UserSchema)

const UserModel = require("./models/user.js")

module.exports = {
  createANewUser: function(username, password, callback) {
    const newUserDbDocument = new UserModel({
      username: username,
      password: password
    })

    newUserDbDocument.save(function(error) {
      if (error) {
        callback({error: true})
      } else {
        callback({success: true})
      }
    })
```

```
 }
}
```

For the Location feature, MongoDB can accept location in an array of latitude and longitude in one field. Latitude and Longitude can be received based on the user's entry of ZipCode, City and State, or Full Address. Visualization is achieved in the schema tab that uses interactive maps to analyze location data.



Analyze Location Data

In the **Schema** tab, you can use interactive maps to filter and analyze location data. If your field contains GeoJSON data or `[longitude,latitude]` arrays, the **Schema** tab displays a map containing the points from the field. The data type for location fields is `coordinates`.

```
var mongoose = require("mongoose");
var Schema = mongoose.Schema;
var MessageSchema = new Schema(
{
 username: String,
 text: String,
 location: {
  type: { type: String },
  coordinates: []
 },
{
 timestamps: true
}
);
MessageSchema.index({ location: "2dsphere" });
var Message = mongoose.model("Message", MessageSchema);
```

```
module.exports = Message;
```

A document for a FindMyMeal UserAccount would look like this:

```
{
  "_id": ObjectId(“),
  "first_name": "Ron",
  "last_name": "Swandaughter",
  "user_name": "UserName",
  "password": "password",
  “Email”: “email@email.com”,
  “Recipe Book”: (see below),
  "Pantry": (see below),
  "location": [ -86.536632, 39.170344 ],
  “Budget”: Budget,
  “Planner Archive”: Planner Archive
}

"Recipe Book": [
        {
        “Recipe Name”: recipename,
        “Recipe Serving”: 3,
        “Ingredients”: [
                {
                “Ingredient Name”: ingredientname,
                “Ingredient Type”: Ingredient Type,
                “Ingredient Count For Recipe”: Ingredient Count for recipe
                },
                {
                “Ingredient Name”: ingredientname,
                “Ingredient Type”: Ingredient Type,
                “Ingredient Count For Recipe”: Ingredient Count for recipe
                },
        …
        ]
        }
   ]

“Pantry”: [
        {
        “Ingredient Name”: ingredientname,
        “Ingredient Type”: Ingredient Type,
        “Ingredient Count in Pantry”: Ingredient Count in Pantry
        },
        {
        “Ingredient Name”: ingredientname,
        “Ingredient Type”: Ingredient Type,
        “Ingredient Count in Pantry”: Ingredient Count in Pantry
        },
        {
        “Ingredient Name”: ingredientname,
        “Ingredient Type”: Ingredient Type,
        “Ingredient Count in Pantry”: Ingredient Count in Pantry
        },
        …
]

"Planner Archive": [
        {
        “Planner Name”: Planner Name,
        “Planner Entry”: [
        {
                “Day”: Monday
                “Meals”: [Recipe1, Recipe5, Recipe2]
        },
        {
                “Day”: Tuesday
                “Meals”: [Recipe1, Recipe5, Recipe2]
        },
        {
                “Day”: Wednesday
                “Meals”: [Recipe1, Recipe5, Recipe2]
        },
        {
                “Day”: Thursday
                “Meals”: [Recipe1, Recipe5, Recipe2]
        },
        {
                “Day”: Friday
                “Meals”: [Recipe1, Recipe5, Recipe2]
        },
        ]
        },
{
        “Planner Name”: Planner Name,
        “Planner Entry”: [
        {
                “Day”: Monday
                “Meals”: [Recipe6, Recipe3, Recipe7]
        },
        {
                “Day”: Tuesday
                “Meals”: [Recipe6, Recipe3, Recipe7]
        },
        {
                “Day”: Wednesday
                “Meals”: [Recipe6, Recipe3, Recipe7]
        },
        {
                “Day”: Thursday
                “Meals”: [Recipe6, Recipe3, Recipe7]
        },
        {
                “Day”: Friday
                “Meals”: [Recipe6, Recipe3, Recipe7]
        },
        ]
        },
]
```