Overview:

We are going to use MongoDB as our database where we will have one database with multiple collections representing different types of data that our application requires. Each collection will have multiple documents as outlined in this document.

Collections:

- users: Holds user related information
- opportunities: Holds volunteer opportunity information
- review: Holds reviews for organization or users

Collection Representation:
Required fields represent fields that can not be null or undefined in the database. Optional fields represent fields that can be null or undefined in the database.

“users” collection document structure:

_id, email and phone numbers are unique identifiers.

```
{
"_id": ObjectId, # required, automatically generated by MongoDB.
"email": string, # required, email of the user.
"password": string, # required, hashed password of the user.
"userType": string, # required, “Volunteer” or “Organization”.
"phoneNumber": string, # optional, phone number of the user.
"fullName": string, # required, name of the user.
"noOfTimesVolunteered": number, # optional, number of times volunteered.
"createdAt": Date, # required, date the document was created.
}
```

Example:

```
{
"_id": ObjectId(“65ea3a9ca520d21a8d9439c1”),
"email": “example-user@gmail.com”,
"password": “hashed-password”,
"userType": “Volunteer”,
"phoneNumber": “204-913-2942”,
"fullName": “First Last”,
"noOfTimesVolunteered": 0
"createdAt": 2023-04-09T00:22:30.151+00:00
}
```

"opportunities" collection document structure:

_id is a unique identifier for this document.

```
{
"_id": ObjectId, # required, automatically generated by MongoDB.
"organization": {
"id": ObjectId, # required, id of the organization.
"name": string # required, name of the organization.
}, # required, organization that created the opportunity.
"name": string, # required, name of the opportunity.
"time": Date, # required, time the opportunity occurs.
"description": string, # required, description of the opportunity.
"volunteersNeeded": number, # required, amount of volunteers needed.
"duration": number, # required, estimated duration (hours) of the
opportunity.
"location": string, # required, address of the opportunity.
"signedUpUsers": [{
"fullName": string, # required, full name of the user volunteering.
"email": string, # required, email of the user volunteering.
"phoneNumber": string, # required, phone number of the user volunteering.
}] # optional, array of users volunteering.
"status": string, # required, “Pending”, “Accepted” or “Declined”
}
```

Example:

```
{
"_id": ObjectId(“65ea3a9ca520d21a8d9439c1”),
"organization": {
"id": ObjectId(“65ea3a9ca520d21a8d9439b1”),
"name": “Organization 1”,
},
"name": “Homeless Shelter”,
"time": 2023-04-09T00:22:30.151+00:00,
"description": “Help distribute food”,
"volunteersNeeded": 1,
"duration": 2,
"location": “421 University Dr”,
"signedUpUsers": [{
"fullName": “First Last”,
"email": “example-user@gmail.com”,
```

```
"phoneNumber": “204-913-2942”
}],
"Status": “Accepted”,
}
```

“review” collection document structure:

_id is a unique identifier for this document.

```
{
"_id": ObjectId, # required, automatically generated by MongoDB.
"revieweeId": ObjectId, # required, id of the reviewee.
"reviewerId": ObjectId, # required, id of the reviewer.
"description": string, # required, report details.
"rating": number, # required, rating of the reviewee out of 5.
"revieweeType": string, # required, “Volunteer” or “Organization”.
}
```

Example:

```
{
"_id": ObjectId(“65ea3a9ca520d21a8d9439c1”),
"revieweeId": ObjectId(“65ea3a9ca520d21a8d9439b1”),
"reviewerId": ObjectId(“65ea3a9ca520d21a8d9439a1”),
"description": “Did not show up”,
"rating": 1
"revieweeType": “Volunteer”
}
```

Instructions:
These are the recommended steps to follow in order to set up the application and run the seeding script. If you prefer manual steps follow the instructions in the README.

Create a .env file in the root directory if it does not exist and define the MongoDB connection string:

MONGO_CONNECTION_STRING=mongodb://localhost:27017/matchAid

- Replace your-database with the name of your MongoDB database.

Build and run the Docker containers using Docker Compose:

docker-compose up --build

- This command will start the MongoDB container, the app container, and the seeding script container that runs the script to seed the database with sample data.

Access the app:

Once Docker Compose has finished setting up the containers, you can access the app by visiting http://localhost:3000 in your web browser.