Avontece Cabrera

SWDV 691

3/28/2021

Database Re-Design

Technology Stack: JavaScript , React & Redux (make request), Node (parse request), Express (generate database request), MongoDB Atlas (store data), and Heroku (deploy).

The purpose of this application is to create an easy to use and simple blog application titled “VonnyBlogs”. The problem I am looking to solve is helping to shine a light on the underrepresented groups in my local community. This application will allow the user to browse posts, and if they have any post suggestions or would like for the administrator to post their blog or story, they can fill out a contact form for submission. The application will start on the main landing page, which will consist of the weekly highlighted posts. The stretch feature will allow a user to like a post. The administrator will be able to log in , post, update, or delete blog posts. I will be using the document-based database MongoDB since it is a NoSQL database that is flexible to use in order to do the above. I also chose MongoDB because I would not have to worry about switching between JS and SQL while using JavaScript.

JSON object example:

fields: unique ID number, title (string value), message/ content (string value), creator (string value) , tags [string value], selectedFile (string value), createdAt (date type).

```
{
   "_id": ObjectId("xxxxxxxxxxxx"),
   "title": "Happy Quarantine",
   "creator": " Jane Doe”
   "message": [{
      "base": "http://VonnyBlogs.com
      "type": "text/html",
      "value": ”xxxxxxx ",
      "language": "en"
   }],
   "tags": "tag:XXXXX",
   “selectedFile”: “XXXXXX”,
   "CreatedAt":  ("2021-09-17 ")
}
```

Mongoose schema code:

Posts- (updated for like stretch feature)

```
const postSchema = mongoose.Schema({
   title: String,
   message: String,
   name: String,
   creator: String,
   tags: [String],
   selectedFile: String,
   likes: { type: [String], default: [] },
   createdAt: {
      type: Date,
      default: new Date(),},
})
```

Added User schema for the MVP for the user logins

```
const userSchema = mongoose.Schema({
  name: { type: String, required:  true },
  email: { type: String, required: true },
  password: { type: String, required: true },
  id: { type: String },
});
```

Added Contact schema for the MVP for the contact form

```
const contactSchema = mongoose.Schema({
          name: { type: String, required: true },
          email: { type: String, required: true },
          message: { type: String },
          createdAt: {type: Date, default: new Date(),
   },
})
```