World News Org.

World New Org is something I randomly thought about while trying to imagine the most intuitive way to build a REST API.

Assume you are backend developers for World News Org contracted to rebuild the backend for their articles so that their frontend Software Engineers could use that API to build a flashy Web Application that article Authors could use to post articles and their 300 million readers get the latest updates on news such as the Melting Ice in NorthPole under Climate, Tesla Stocks under Business and,Ten Reasons Why Giannis Antetokounmpo is better than Lebron James under science fiction.

From this brief description, you have a simple problem that could potentially get complicated and require a lot of time rephrasing into non-abstract requirements. However, due to resource constraints, I will oversimplify the problem while dropping hints on areas for improvement.

Database Design

The overall goal is to have articles. Therefore the most basic interpretation of this is to create a Schema called articles. The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). (source: Wikipedia).

However, what are the attributes of an Article? An article will have a Title, Body, Image, Author, Category, Date, Video(optional), etc. These are some easily observable attributes I could pick from the WSJ articles on my phone.

The simple Schema of an article will be:

Article = {
Id: Auto-generated by MongoDB
Title: String - Required
Body: String -Required
Image: URL - Optional
Video: URL - Optional
Authors: ???? Required
Category: ??? Required
}

We now have a minor problem. We can't just write the name of the Author and leave it to that. What if the user wants the author's email. What if we want to query our database for all the articles by a certain author and we have two authors with the same exact name. What if there is more than one author? Ask these same questions replacing the author with Category.

Case 1 - Author.
We create a Schema to define what an author looks like.
Author = {
Id: EmployeeID
Name: String (required)
Description: String
Email: String
Avatar: String
}
However, for every article displayed on any part of the front-end the complete author information is displayed as shown above. Therefore, adding a reference to the author in the article schema is not adequate because it compels the front-end engineers to making two different API requests every single time they render an article. The solution is, exploit the full flexibility of using a NoSQL database such as MongoDB by embedding this document(Author) inside the main document(Article).

```
Article =  {
        Id:  Auto-generated by MongoDB
        Title: String - Required
        Body:  String -Required
        Image: URL - Optional
        Video: URL - Optional
        Authors: [
                 {
                        Id: EmployeeID
                        Name: String (required)
                        Description: String
                        Email: String
                        Avatar: URL
                },
                ...
                {
                        Id: EmployeeID
                        Name: String (required)
                        Description: String
                        Email: String
                        Avatar: URL
                }
        ]
        Category:  ??? Required
}
```

Thus Authors is an array of all the authors who participated in writing the article.

Case 2.
Categories. You only need to indicate the category at the top of each article and display no further information such as the description of the category

or even how many followers the category has and even the image and criteria used to pick articles for the category.

Therefore if this shown below is a category Schema:

```
Category: {
        ref: Name of the category(Politics)
        Description: String
        poster: URL
        Criteria: String
}
```

The final article schema is:

```
Article =  {
        Id:  Auto-generated by MongoDB
        Title: String - Required
        Body:  String -Required
        Image: URL - Optional
        Video: URL - Optional
        Authors: [
                {
                        Id: EmployeeID
                        Name: String (required)
                        Description: String
                        Email: String
                        Avatar: URL
                },
                ...
                {
                        Id: EmployeeID
                        Name: String (required)
                        Description: String
                Email: String
                Avatar: URL
            }
        ]
        Category: {
            Name: categoryname
        }

}
```

Download the starter project, unzip it then open it in your code editor of choice.
Run npm install. Follow the steps from the MongoDB lab to create a mongodb Atlas project, copy the connection link and paste it in the line where we have mongoose.connect. Start the app by typing npm start.