MongoDB Realm

July 03, 2023

What is MongoDB Realm

MongoDB is a development platform that supports web,mobile, desktop IoT applications. Realm provides data synchronization, triggers, authentication, serverless functions, charts.

MongoDB Realm

By the end of this tutorial you will be able to:

- Create a Realm application
- Create a MongoDB rule
- Query data using realm sdk web using Nextjs project.

App Explanation

This tutorial will create a NextJS Movie application, in which you can get Movies Information from the MongoDB sample data.

Agenda

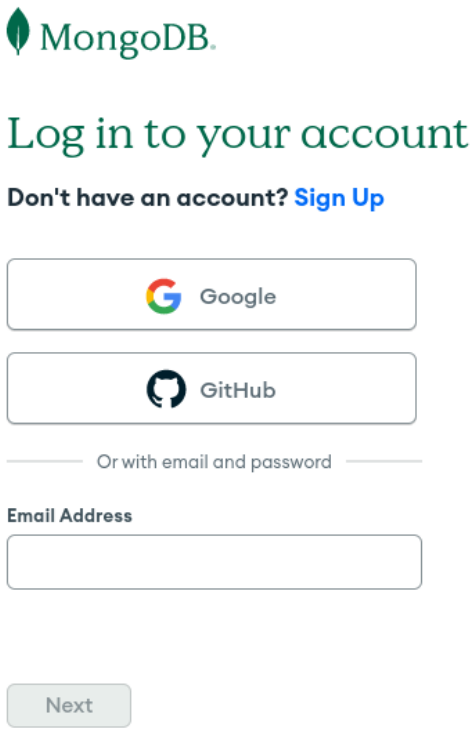
In this forum we are going to discuss the basics of Realm and how we can configure MongoDB atlas in a free server.

1. MongoDB Account Creation
2. App Creation
3. Authentication Set Up
4. Defining Rules
5. React Project

Prerequisites

MongoDB Account Creation

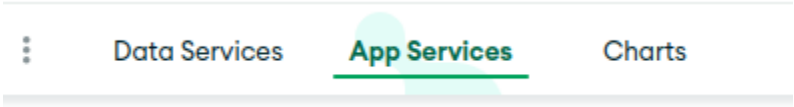
Access MongoDB Atlas and login or create an account



Steps

1. App Services configuration

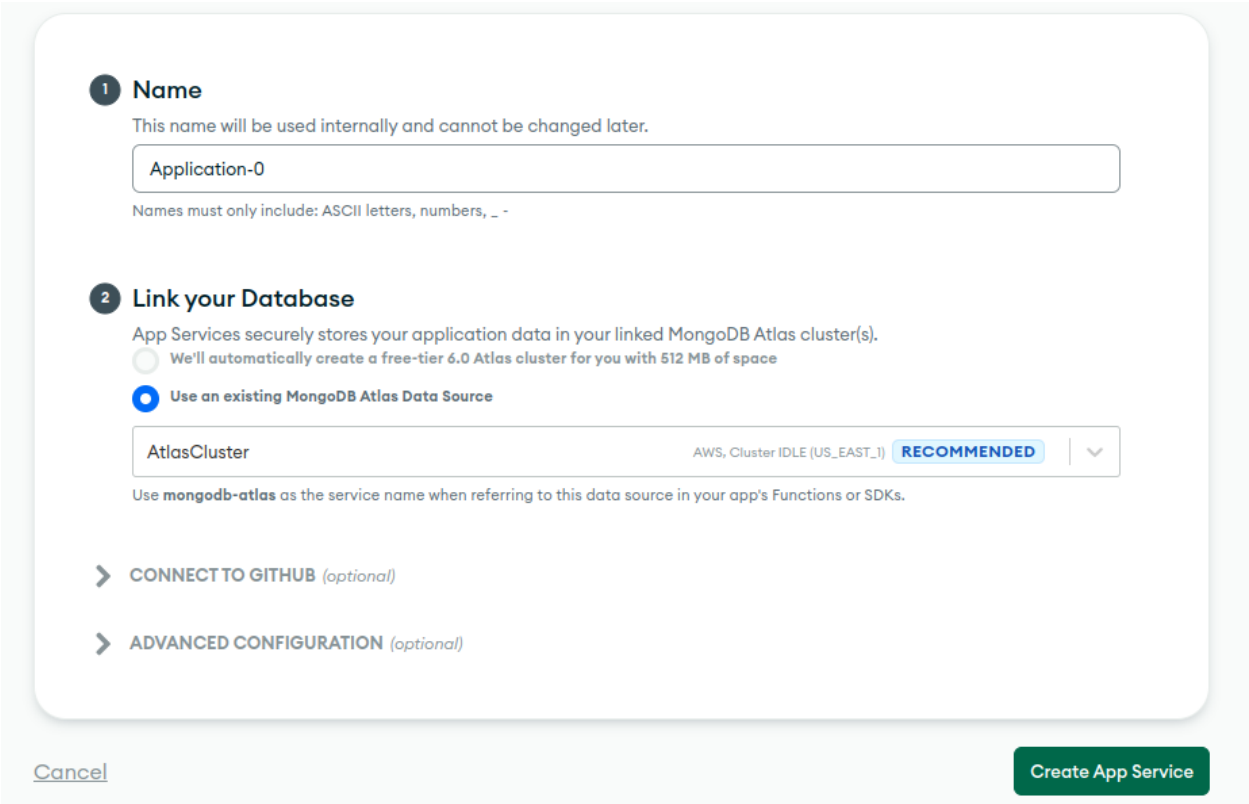
On the home page access the App Services Tab, this tab is going to be the place where all the Realm Configuration is going to take place.



2.2 App Creation

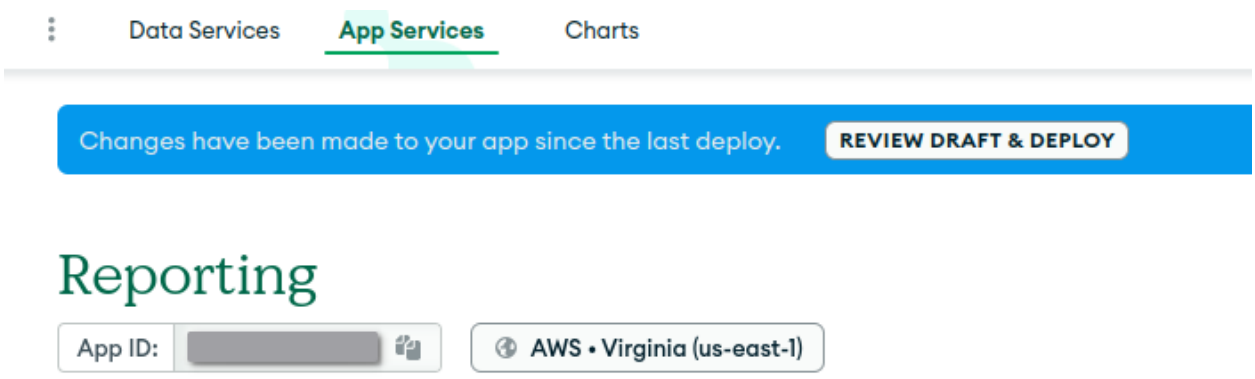
- Click on Create a New App, top right side of the screen

- Next we are going to see a set of options for our application



- Link your Database is the place where we are going to connect our database with the Application, there we can create a new Free Tier Data Base or if that is the case that we have another MongoDB instance we can select it.
- Application Name if you want to customize the name of your app otherwise a Default Name is going to be.
- Finally click on Create App service

After all the steps you should have an *AppId* value which is very important because it's a value that we are going to pass to the sdk in order to connect into a MongoDB instance.



Congratulations if you follow the steps you should have a working Realm MongoDB implementation.

2. Load Sample Data

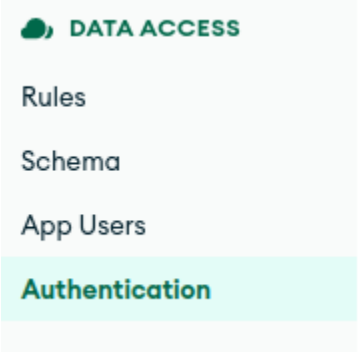
For this tutorial we are going to use the Sample Data that MongoDB provides therefore follow this link to upload the example Data.

3.Authentication setup

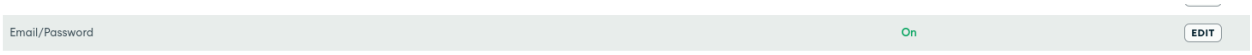
As we discussed Realm is a development platform which provides some features one of the main settings that we can use is authentication.

3.1 Authentication section

On the left section access to the *Authentication* link.



This tab is where we can specify different types of authentication for our application, for this tutorial we are going to use basic authentication such as email-password authentication nevertheless you can use stronger authentication mechanisms. Therefore click on the *Edit* button next to Email Password.

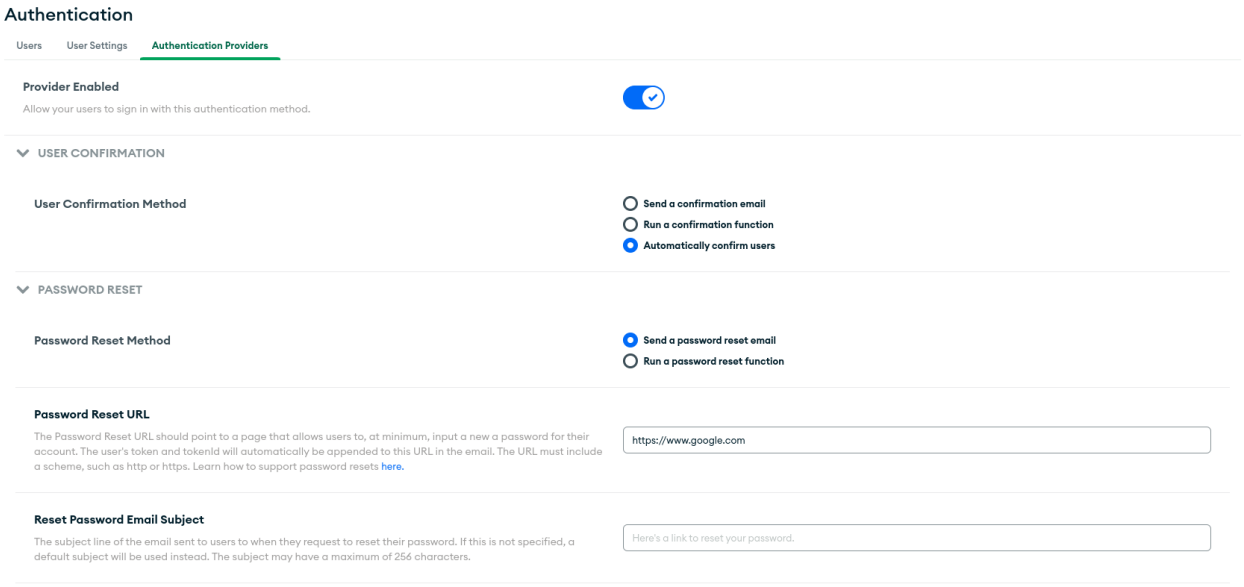


1. First toggle the provider enable option to allow users authentication
2. User Confirmation is a section in which you can specify how you want to manage if a user is automatically confirmed or if you want to do another type of logic, in our case it would be automatically.
3. Password Reset is the section about how to handle when a user wants to reset the information, we have two options: *Send a password reset email* or run a function, for

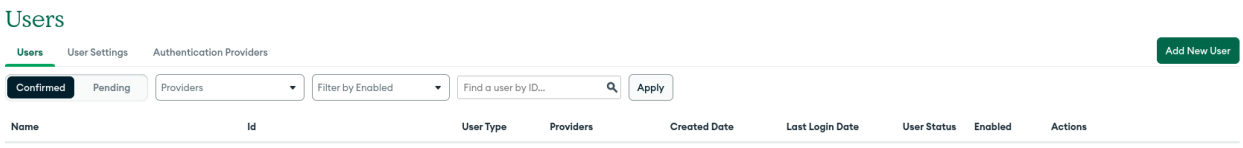
simplicity we are going to use Send a Password reset email and we are going to use *https://www.mongodb.com* as the url. Keep in mind that if you want to use this feature in production you should change the url to a valid endpoint.

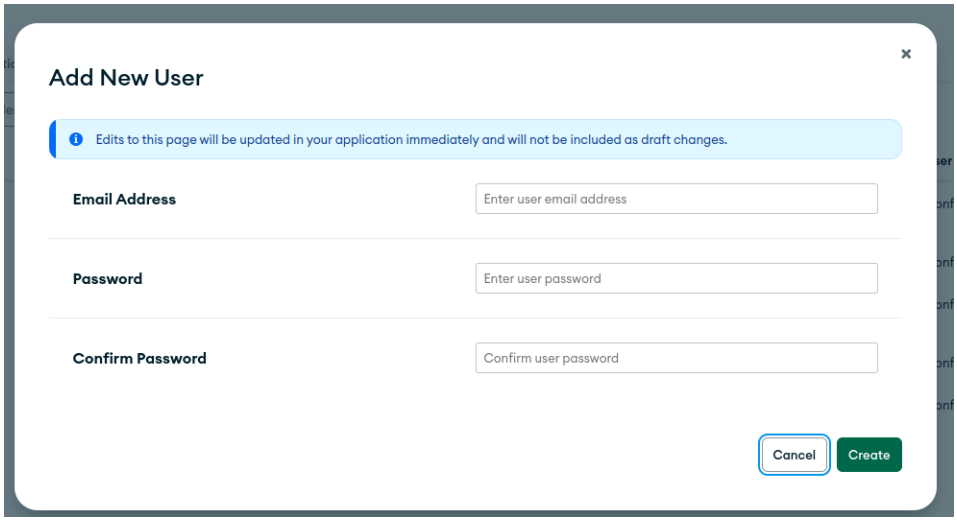
4. Then we need to apply the configuration and click on *Review Draft y Deploy*

Final Configuration



5. Finally go to the *Users* tab click on *Add New User* and complete the information.



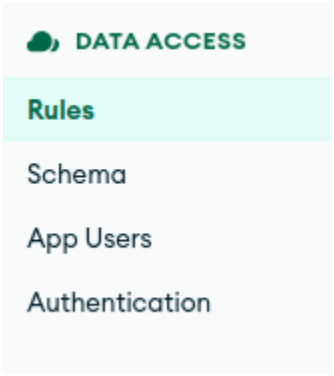

Add New User
Edits to this page will be updated in your application immediately and will not be included as draft changes.
Email Address
Enter user email address
Password
Enter user password
Confirm Password
Confirm user password
Cancel
Create

4. Rules definition

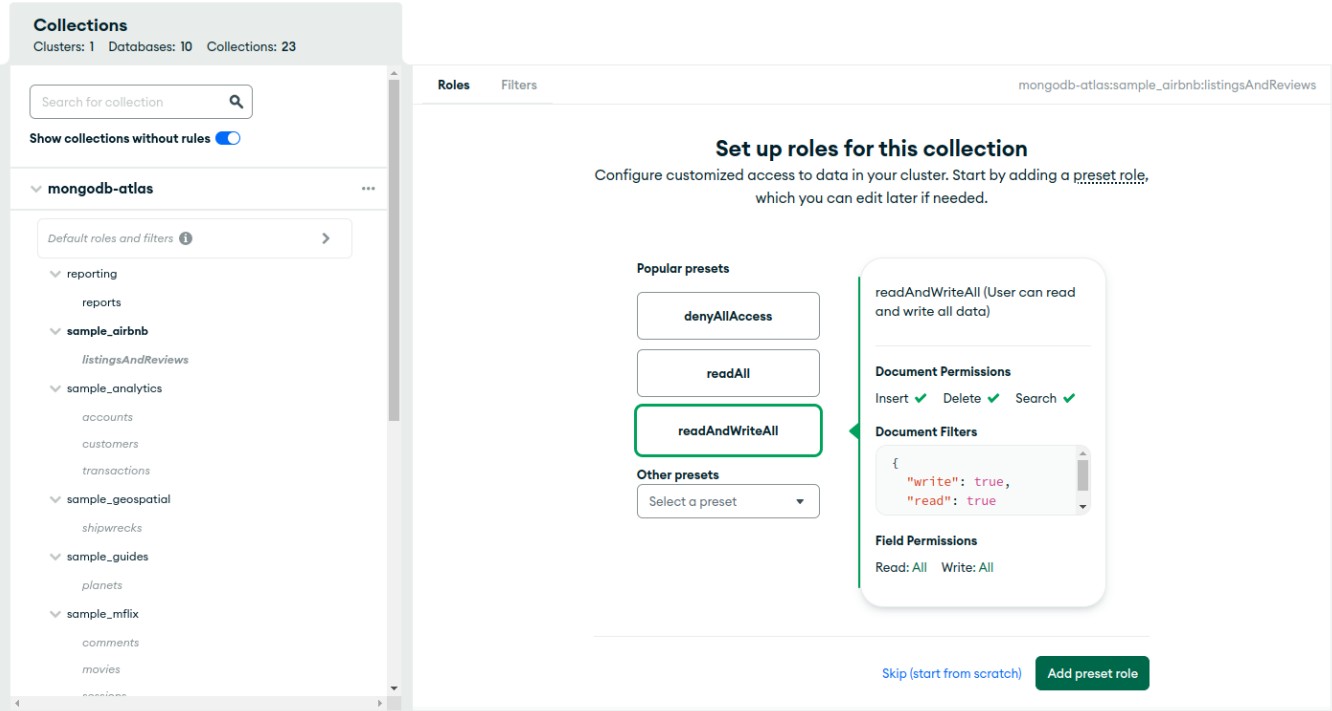
MongoDB has another layer of security which is Rules, with this configuration we are going to specify what actions we can do in our Collections. If we don’t create a rule for our collection we are not going to be able to query, therefore it's a critical step.

4.1 Data Access Rule definition

In the top left section of the page under Data Access click on *Rules*



Next we are going to see all the collections that we have in our database, therefore, we need to click on the collection that we are going to work on, in our case the database *Sample_Mflix* and the collection Name is *Movies* . Then we can select *readAndWriteAll*. Finally click on *Add preset role* and *Review Draft y Deploy*



Keep in mind that since this is a tutorial we are using read and write all. If you use this in production please configure those privileges carefully.

5. Web application creation

For this tutorial we are going to use NextJs however you can use any type of js Framework.

5.1 Prerequisites

- Node install
- NPM install

If NextJS is not installed please use the following link to configure.

5.2 NextJS project creation

In your root folder open a terminal and run the following command, which creates a NextJs project with typescript

```
JavaScript

npx create-react-app "YourProjectName" --template typescript
```

5.3 Install SDK

In order to communicate with Realm we need to install the *realm-web* dependencies, please access the previous project folder and run the following command.

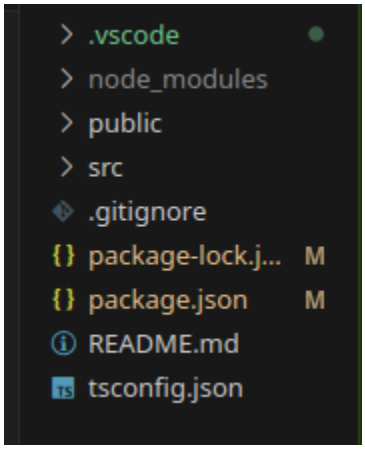
```
JavaScript

cd "YourProjectName"
```

```
npm i realm-web
```

5.4 MongoDB connection class

Right now we have all the resources to be able to connect and process information, therefore we are going to create the project with your favorite editor. You should have the following structure once you open the project.



Under src folder create a new folder called *utils* and create a file called *MongoDBConnection.js* this file is a class that will allow us to connect and run all our queries.

Copy the following code into *MongoDBConnection.js* file and modify *db,Realm Id ,reportsCollectionName* and *username/password* with the correct information.

JavaScript

```
import * as Realm from "realm-web";


export class MongoConnection {

    constructor() {

        this.mongoClient = "mongodb-atlas";

        this.db = "here goes your database name";

        this.RealmId= "here goes your Realm Id";

        this.reportsCollectionName = "here goes your collection name";

        this.app = new Realm.App({ id: this.RealmId });

        this.user = null;

        this.mongo = null;


        this.Login();

    }

    async Login() {


        if (this.user === null) {

            const credentials = Realm.Credentials.emailPassword(

                "user", "password"

            )
```

```
            let user = await this.app.logIn(credentials);
            this.user = user;
        }
    }
async GetValidAccessToken() {
        if (this.user === null) {
            await this.Login();
        } else {

            await this.user.refreshCustomData();
        }

    }
    async GetData() {
        await this.GetValidAccessToken();
        this.mongo = this.app.currentUser.mongoClient(this.mongoClient);
        const collection =
this.mongo.db(this.db).collection(this.reportsCollectionName);
        let result = await collection.find({},{limit:10})
        return result
    }
```

```
	}
```

MongoDBConnection Constructor Explanation

```JavaScript
import * as Realm from "realm-web";

//Constructor Section

export class MongoConnection {

	constructor(){

		this.mongoClient = "mongodb-atlas";

		this.db = "here goes your database name";

		this.RealmId= "here goes your Realm Id";

		this.reportsCollectionName = "here goes your collection name";

		this.app = new Realm.App({ id: this.RealmId });

		 this.user = null;

		  this.mongo = null;

	}

}
```

In the previous code we are setting the variables that we are going to use.

mongoClient: The value refers to the service name for MongoDB, the value it needs to be explicitly be *mongodb-atlas regardless of the exercise*

db: Database name the application will consume

RealmID: We can get this value on the App services main page.

App: Variable that is going to initiate the Realm Process.

User: MongoDB Actual User

Mongo: MongoDB instance

MongoDBConnection Login Explanation

```
JavaScript

import * as Realm from "realm-web";

//Constructor Section

export class MongoConnection {

       //...Constructor Code

       async Login() {

        if (this.user === null) {

            const credentials = Realm.Credentials.emailPassword(

                "user", "password"

            )

            let user = await this.app.logIn(credentials);
```

```
            this.user = user;
        }
    }
}
```

The previous code is going to use the email and password to login, keep in mind that this is a demo therefore we are using a test user however in real apps user nor the name should be in the code.

MongoDBConnection Token Validation Explanation

As we know, usually when we use some type of authentication we are going to use a token, therefore, it's important to validate that our token is active.

JavaScript

```
async GetValidAccessToken() {
        if (this.user === null) {
            await this.Login();
        } else {
            await this.user.refreshCustomData();
        }
```

```
    }
```

The code above is going to validate if the user is *null* if that is the case we are going to execute the *Login* function otherwise we are going to use the [refreshCustomData] which is going to refresh this data.

MongoDBConnection Find Query

JavaScript

```
async GetData(){
      await this.GetValidAccessToken();
      this.mongo =  this.app.currentUser.mongoClient(this.mongoClient);
      const collection =
this.mongo.db(this.db).collection(this.reportsCollectionName);
      let result = await collection.find({},{limit:10})
      return result
}
```

The above code is going to validate the token and perform a connection to the collection, finally we are going to use a find query without any parameter however you can specify the search with more complexity.

5.5 App.tsx file

Open App.tsx file and replace all the contents

JavaScript

```
import React, { useMemo, useState } from 'react';
import './App.css';
import { MongoConnection } from './utils/MongoDBConnection';
function App() {
  const [results, setResults] = useState<any>([])
  const [isLoading, setIsLoading] = useState(false);
  const mongo = useMemo(() => new MongoConnection(), [])
  const GetInformation = async () => {
    setIsLoading(true);
    const results = await mongo.GetData();
    setResults(results)
    setIsLoading(false)
  }

  const onImageError = (e: any) => {

    e.target.src =
"
///9/f3+/v4AAAD6+vrv7+/29vby8vLs7OwFBQX7+vtpaWn09PSLi4vKysqHh4fm5ubm5ubn
```


V1dV4eHi/v7+ysrLFxcWAgIBycnLW1tZnZ2eioqJiYmKbm5umpqZVVVUXFxcdHR01NTVNTU0/Pz8mJi
Y5OTkqKioeHh5EREQQEBBSUlLiJkNpAAAJPElEQVR4nO2bC3uaShOAZ5YBcVUCiBe8oc3lJG36///eN
7NcRA3knO8c2trO+6QRECK8HWdnlwVAURRFURRFURRFURRFURRFURRFURRFURRFURRFURRFURRF
URRFURSlE4Om/Gk2GLfBlIsfH4TU9Y7SJiBC/mnkslhWR2QBG8k38P6oej/DgJUXltVsQSebtwF
2RChLlxfV+wkGprOHfA/nrMA+wc/m+2IdTQ18GKEG1ot8kZBFtduFS66Qe553gNIgxy9aM5p99RxvD0
mZki8PQwjl3TWgyu3EcGaFpeeNvUktl9MDRV/Y3Hg85u2ed8SbzEABHERupHmhB8mu8dcLuUgwF7Nl5
MpLYTg+W6HLcYyFp3I/w3CbdXKeGrkWEuf2dbac7w9ucWWw7ZZXMmdd5fYgjRUU7vvfilx7EqHpSIoF
u/7CGh9DOMuVMPafPJcyVG43iAFF5de/kUuwFrc5ECdadAnZ81JJzSW8jWAhblVuLxy30yfPe9lcNGg
HXvsq/QqWa6XlGntvAdWhy5ppzpsWj5oWeuEKVmJwnbrIrbbFEqoFBFLd8g6udfPW9iwX43fPewo053
YjnVpnjjNA3pILR7GWSFIQgEaSdR/qaoGFB9ICZqA5txvOnEgSpU8+PDRyWdVGtkFr1OaZN3yv0wJXa
qn37qVALnI7h3X+bNxAzXf5xkNbLgQvvLKBRhpCyju9+9UaQcSrJ4ulXFC5H4FcF8zYExcCF3Jdt/bY
liapw4vl+y+VQsgt4DjhZk3ldsNZgWsu75s0XA+uWhCM9CBkzKDlzJVmTiMrNRPed8URrHJ7CCiQsZl
EumRtuWLcy1rOMJNyYek0kjmWuxqjcnsgV8CupEd2IXfu5LZ2xK3ndpTqgiSu30IZC1O5XcgQ4oqV7d
gxL7ZzblmJtauApJFrAqkc1jJwpnI7YaFShb2PMDAfyY3P0kwtV4YXpAWccaFhVW4nluuwF1eFyY2wa
7ljJ7feF2u5FMCSF56nAcgApMrtAMml2Qe2fCNXsgU3cy25LufOwQbhX7yw5VZNyrKmE6F2LyGSnsCj
RHAgcme1XF5eVtVCLQ1NJHIjnBppAY/1LUtbypUbm+q3DdFXicGQ8cNwRDK2sCPfj6emHAaPzncrq+I
sNih1xGEUh7EcF7rxneWIl3T+QhvOmNUNnJpxtbYACquq9jw67ooz35UX7WMaQkM9H/anIXLHJeV9yP
H4vRz9zgF8GQRL4Tx+65IGd+TKbFz+X5QHlsd6PmHvx/1hlKMu7cgd15ELeJJOMeBZ7rMbyaGySLsI9
xJfxx3bUDBLZw1p+syinopZms9ZajkIVs5ikvYqeHftGGGUu90fSmSA3dvw+iLonk/2B3Ljomi6v6y0
qrycXJkt5u5EtO9QVoxdnVv+veHP+W6QuzdozrgR20k5+wYpkDu7ryzXouViDWR142bdlMixXEvgWa7
S5rr0P8uVzvDR3VnnPEAyZSSVtcxVs21Ubhe9cl3ojr0998Q443Kp8C7jOyr3/wMrueQ6DuWNHDb3dF
wu00dZfI/N1XS7Rq42ZZ8gk26k+4uVXKSjhGtdpLFCunbYyCVQub2Ucg/VIAHnBWvmVd9CatkvnHCv4
1PklreDUDsQvSDMN4tNgabqC1BAJllUkfs4C9DeCJTcvMgXm0TnPt9AFt1EsGa6s3upmznjCtcgKtLZ
fhtAOfPm49xqLOi4whUBSuNVjxVeFAFQbamUk5QJ0CkXdczmhup7TtfOTLMuz/NwJ0Jmmcv9MmoF9gX
mtuf2p4Ow+f78nPt4VWK15bpOHJEfy+g4rA6jj91ufsgJ/0JcVPt4hdtInh/Gy1cM5LYCyZNS6CbgEi
cMkvLKWHJ9CJinrjg7hYFxsyLJUvn0iUzvZzzo+qDzB/5UGf81ppr92QmxE0LwKN6cVtbEm122xunss
AoCeQBtvjzlI8TjKQ3Cw/Pc4DR9KSg7bDKII6I4CqPicEQIN6fM6/2g+7TbG5jud+/1Wo//Rjymv7Jw
MQ+/ZcnkBLtVONuzePK9WbyawKKI58+0X2x52/KURKckOWXRAnD9sPWW8WRJr1Gce/C52/7Q/knuTR+
dp1pdUN/1cuROdodTRNE0LtJ5AbjdbU9+mJy4+QpGT5wePHyCACbZciZ1QziBwxYh2mxzoHW63SCsZ/
M9WPhbcnu/Sd2X+LPcfnZF/ZeL6IVxTAZz6T0UawriSfaWLxb5lAyNXjlnfIk33I4dVyyXd4938BITJ
pMkB9aaPQBFRbFy+eVfncpP0/vvDPZeDYoTMnNu61fFMkezPmx3/JErKWt9kftIbxy5p0QaNDThAQ6Z
gWyTLIBW6TbnyE2X/NZ/IrfzNO9SLhkPrLW4/ZYsv+9o87A/THCxT3J5agdHb9x38CBPk+IA610izwE
eaPs94rQbv2TRa5otkNYzeFknE5V7DeGKf1sMok0xXYVxtEwmFBSLFQVSjM05567QrDZHXjuuOJinc4
JtnieGsjyVaoEwjIw/W4TL4c7yTuW6tIsSwTLldvuc+PnRjR1a1zEDucHjOrbu+mTnarCXygpXsFUH2

```
Q54kncq19ZO5MGTbHfYn7daR+n/orBrWlJ5W/ao/w3Fvcqtz156uK5vFtyUb7Xcujw5Lw7o8+r07lmu
nD/3c4k7ZvZarjGmXSNdiv4x3Ltcdw3oxr5+QX4Dub8uKndAVO6AqNwBUbkDonIHROUOiModEJU7HL9
DD+2XReUOiModkgHtqlyVOyAqd0BU7oCo3AFRuQOicgdE5Q6Iyh0QlTsgKndAVO6AqNwBUbkDonIHRO
UOiModEJU7IHcvt5xIbolkoi5jgyCgK3gnQnJ7tbBkh5xXjncrt7ZS6QoumTaU6/U+l8pltZrnP9BZ3
p1c24Rq5ZQVjgT/zPW629K8IZy1u/i1wwi+T7mtYK0khpf4n8KGz1Gtckvabl3M+jd2G4F9ouvoVbkt
2imhygiXctsGu2PYHaxyr2gn3Do5TOu8+xmtlo6atk5zbkMpgpshbGXe24Khm1YpVlZnKrfBVo9Jnuu
Gqny9LGKbb/v1LrZ8o/pTLQY41/uU656CvDR3YfFabe3uUqTK/TvYHn7YSXzAbyP3w40qd1h++BO/F5
/9m8m9calyf09U7oCo3AH5Z3L/B+pspjvZ0ZcbAAAAAElFTkSuQmCC"

  }

  return (

    <div className="App">

      <button onClick={GetInformation} className='GetDataBtn'>Get Data</button>

      {

        isLoading ? <div className="spinner-container container">

          <h1>Loading</h1>

          <div className="loading-spinner ">

          </div>

        </div> : null

      }

      <div className="container blogs-page">

        <div className="blog-section">

          {

            results.map((result: any) => (
```

```
            <div className="blogs" key={result._id.toString()}>
              <h3>{result.title}</h3>
              {
                result.poster !== "" ?
                  <img
                    src={result.poster}
                    alt=""
                    onError={onImageError}
                  />
                  : null
              }
            </div>
          ))
        }
      </div>
    </div>
  </div>
  );
}
```

```
export default App;
```

5.6 Modify App.css

Open App.css file and replace all the contents

JavaScript

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
```

```
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
```

```
  }
  to {
    transform: rotate(360deg);
  }
}

body {
  box-sizing: border-box;
  color: #220f5f;
  font-family: 'Poppins', sans-serif;
}

.container {
  width: 100%;
  max-width: 1000px;
  padding: 0 15px;
  margin: 0 auto;
  margin-top: 70px;
}
```

```css
.blogs-page {
}

.page-title {
}

.blog-section {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 20px;
}

.blogs {
  border: 1px solid gainsboro;
  border-radius: 4px;
}

.blogs img {
  width: 100%;
}
```

```
.blogs h3 {
  padding: 5px 8px;
  cursor: pointer;
  font-size: 16px;
  font-weight: 600;
}

@keyframes spinner {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}
.loading-spinner {
  width: 50px;
  height: 50px;
  border: 10px solid #f3f3f3; /* Light grey */
```

```
  border-top: 10px solid #383636; /* Black */
  border-radius: 50%;
  animation: spinner 1.5s linear infinite;
  margin: 0 auto;
}


.GetDataBtn {
	background-color:#44c767;
	border-radius:28px;
	border:1px solid #18ab29;
	display:inline-block;
	cursor:pointer;
	color:#ffffff;
	font-family:Arial;
	font-size:17px;
	padding:16px 31px;
	text-decoration:none;
	text-shadow:0px 1px 0px #2f6627;
  margin-top: 5px;
}
```

```
.GetDataBtn:hover {
	background-color:#5cbf2a;
}
.GetDataBtn:active {
	position:relative;
	top:1px;
}

html, body {
  width: 100%;
  height:100%;
}

body {
    background: linear-gradient(-45deg, #5FAD41, #CDCACC, #07A0C3, #086788);
    background-size: 400% 400%;
    animation: gradient 15s ease infinite;
}

@keyframes gradient {
```

```
    0% {
        background-position: 0% 50%;
    }
    50% {
        background-position: 100% 50%;
    }
    100% {
        background-position: 0% 50%;
    }
}
```

5.7 Run the Application

Run the following command on the root folder.

JavaScript

```
npm start
```

Get Data

Traffic in Souls	Civilization	The Saphead	The Four Horsemen of the Apocalypse
	404 Page not found		404 Page not found

Foolish Wives	Robin Hood	He Who Gets Slapped	The Navigator
404 Page not found