

# **Exercise Tracker App**

*Using Mongob Atlas*

Prepared by-

**Jay Jain [32]**

**Satyam Singh [33]**

**Tarunsingh Jodha [37]**

***Tharkur College of Engineering And Technoly***

***Guide:- Rashmi Thakur***

## **Table Of Contents**

### **List Of Figures**

#### **1. Introduction**

- i) *Problem Statement.*

#### **2. Mongoddb Database**

- i) *It should include all settings to be done in mongoddb while creating database with benefits and drawbacks of mongoDB.*

#### **3. Database Design**

- i) *Connection with mongoDB.*

#### **4. Graphical User Interface designing**

- i) *Designing the website and giving it a look.*

#### **5. Connectivity of GUI with Database**

- i) *Connection of user interface with mongodbatlas.*

#### **6. Future Scope**

- i) *What can be done in future.*

## **Introduction:**

Do you often feel overwhelmed by the amount of work you have to do? Do you find yourself missing deadlines? Or do you sometimes just forget to do something important, so that people have to chase you to get work done?

All of these are symptoms of not keeping a proper "To-Do List." These are prioritized lists of all the tasks that you need to carry out. They list everything that you have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom.

By keeping such a list, you make sure that your tasks are written down all in one place so you don't forget anything important. And by prioritizing tasks, you plan the order in which you'll do them, so that you can tell what needs your immediate attention, and what you can leave until later.

To-Do Lists are essential if you're going to beat work overload. When you don't use them effectively, you'll appear unfocused and unreliable to the people around you.

When you do use them effectively, you'll be much better organized

In this tutorial we will be making a simple todo List app which will be used as an exercise tracker where you can log the tasks the you have completed or can also set tasks for future dates using node.js and mongoDB.

Basics :

**Node.js:** Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

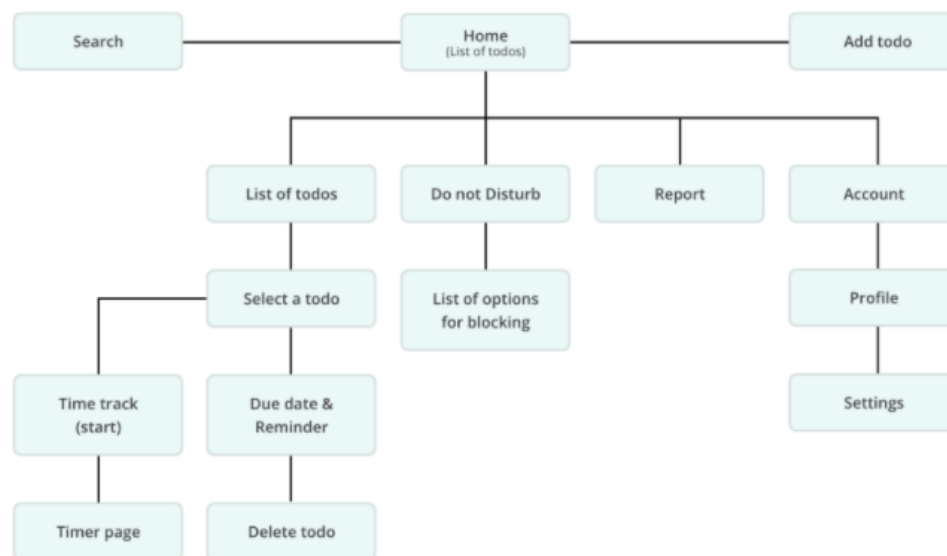
**MongoDB:** MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Although using a paper list is an easy way to get started, software-based approaches can be more efficient in spite of the learning curve. These can remind you of events or tasks that will soon be overdue, they can also be synchronized with your phone or email, and they can be shared with others on your team, if you're collaborating on a project.

There are many time management software programs available. At a simple level, you can use MSWord or MExcel to manage your lists. Some versions of Microsoft Outlook, and other email services such as Gmail™, have task lists as standard features. Remember the Milk is another popular online task management tool that will sync with your smartphone, PDA, or email account. It can even show you where your tasks are on a map. Other similar services include Todoist and Toodledo.

One of the biggest advantages to using a software-based approach to manage your list is that you can update it easily. For example, instead of scratching off tasks and rewriting the list every day,

### Site Map



### Pre-requirements :

VS Code installed

#### i) Problem statement

Social media and other easily accessible online distractions make it hard for us to stay focused on our tasks and make it difficult for us to do our work efficiently.

Also, constantly switching between tasks may give us the false feeling that we are being productive when we are, in fact, not. It's more important for us to prioritize tasks and work on those that are most important, rather than focusing on deleting small items from our exercise tracker just for the sake of appearances.

The goal of this app is to help us become more aware of how we spend time in the process of doing those tasks and how productive that time is. It can help set some constraints on social media to reduce distraction and track the time we spend working on the todo items. When we have a better sense of the estimated time we'll need to spend on our tasks, along with the validated time spent on the items for reference or personal/team reviews, we are able to manage our daily routines more efficiently

Todo lists are a staple in beginning webdev tutorials because they can be very simple. There is, however, a lot of room for improvement and many features that can be added.

Before diving into the code, take a minute to think about how you are going to want to organize your project

## 2. MongoDB Database

# Creating cluster

### 1. First we need to create an account mongodb Atlas

CloudServerTools




## MongoDB Atlas

### Global Cloud Database

Deploy, operate, and scale a MongoDB database in the cloud with just a few clicks. Fully elastic and highly available by default, MongoDB Atlas is the easiest way to try out the latest version of the database, **MongoDB 4.0**.

- Secure from the start
- Fully managed backups
- Comprehensive monitoring and customizable alerts
- Easily migrate existing deployments with minimal downtime
- Cloud-only features, like real-time triggers and global clusters

[Click here to learn more about MongoDB Atlas.](#)

 Google Cloud Platform AWS Azure

### No download necessary

Deploy a free cluster now

✓ 8 characters minimum

✓ One number

✓ One letter


✓ One special character


☐ I agree to the [terms of service](#).


[Get started free](#)


## Then we need to create our cluster


**NORTH AMERICA**

 **N. Virginia** (us-east-1) ★  
[FREE TIER AVAILABLE](#)


 **Ohio** (us-east-2) ★


 **N. California** (us-west-1)


 **Oregon** (us-west-2) ★  
[FREE TIER AVAILABLE](#)


 **Montreal** (ca-central-1)


**EUROPE**


 **Stockholm** (eu-north-1) ★

 **Ireland** (eu-west-1) ★  
[FREE TIER AVAILABLE](#)


 **London** (eu-west-2) ★


 **Paris** (eu-west-3) ★


 **Frankfurt** (eu-central-1) ★  
[FREE TIER AVAILABLE](#)


 **Sao Paulo** (sa-east-1)


**ASIA**


 **Hong Kong** (ap-east-1) ★

 **Tokyo** (ap-northeast-1) ★


 **Seoul** (ap-northeast-2)

 **Singapore** (ap-southeast-1) ★  
[FREE TIER AVAILABLE](#)

 **Mumbai** (ap-south-1)  
[FREE TIER AVAILABLE](#)

 **Sydney** (ap-southeast-2) ★

**SOUTH AMERICA**

 **Sao Paulo** (sa-east-1)


Select **Multi-Region**, **Workload Isolation**, and **Replication Options** (M10+ clusters)  
Increase region availability, configure tagged analytics nodes, and optimize for local service areas. [Read more](#)

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) [Encrypted](#)

Base hourly rate is for a MongoDB replica set with 3 data bearing servers.

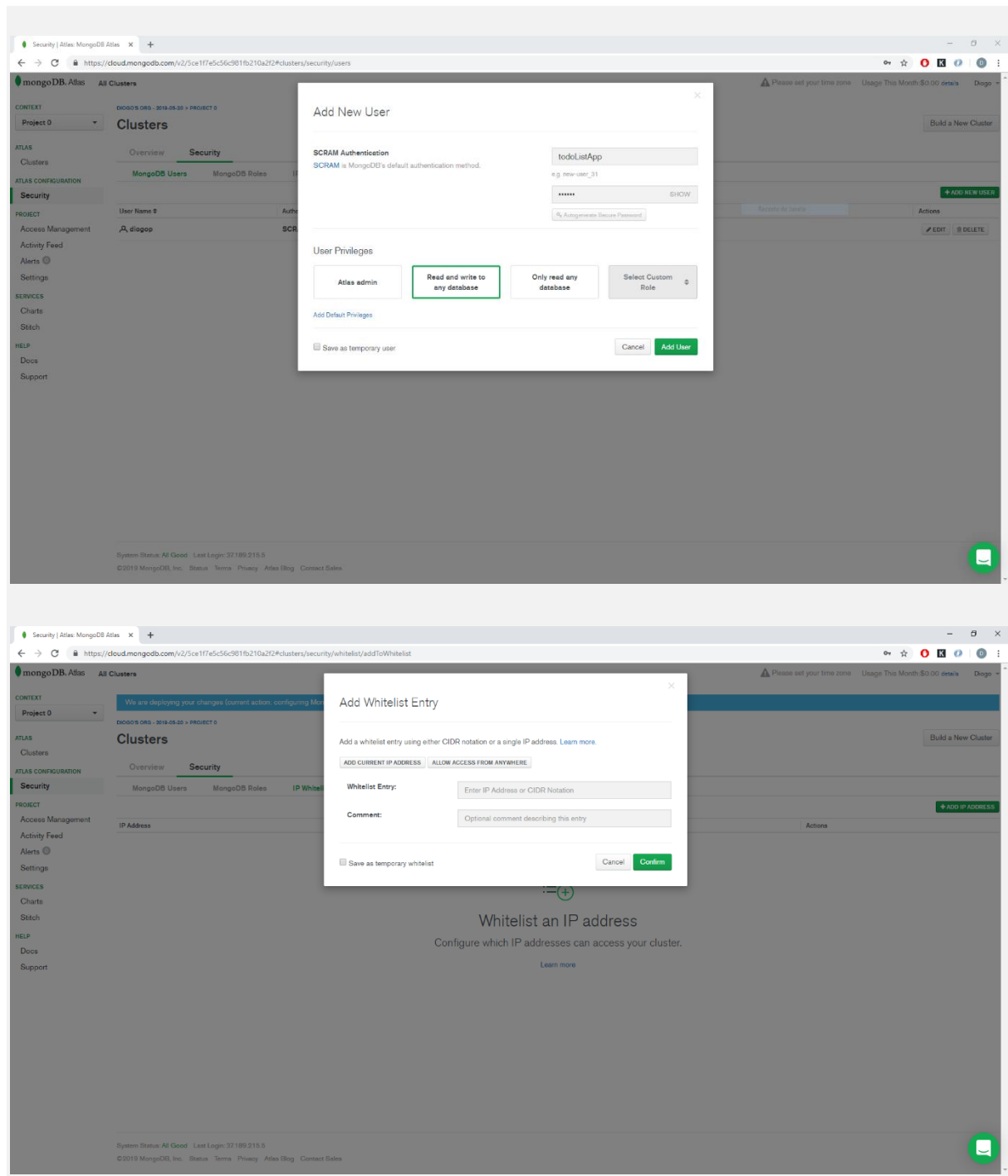
Shared Clusters for development environments and low-traffic applications

Tier	RAM	Storage	vCPU	Base Price
 <b>M0 Sandbox</b>	Shared	512 MB	Shared	<a href="#">Free forever</a>

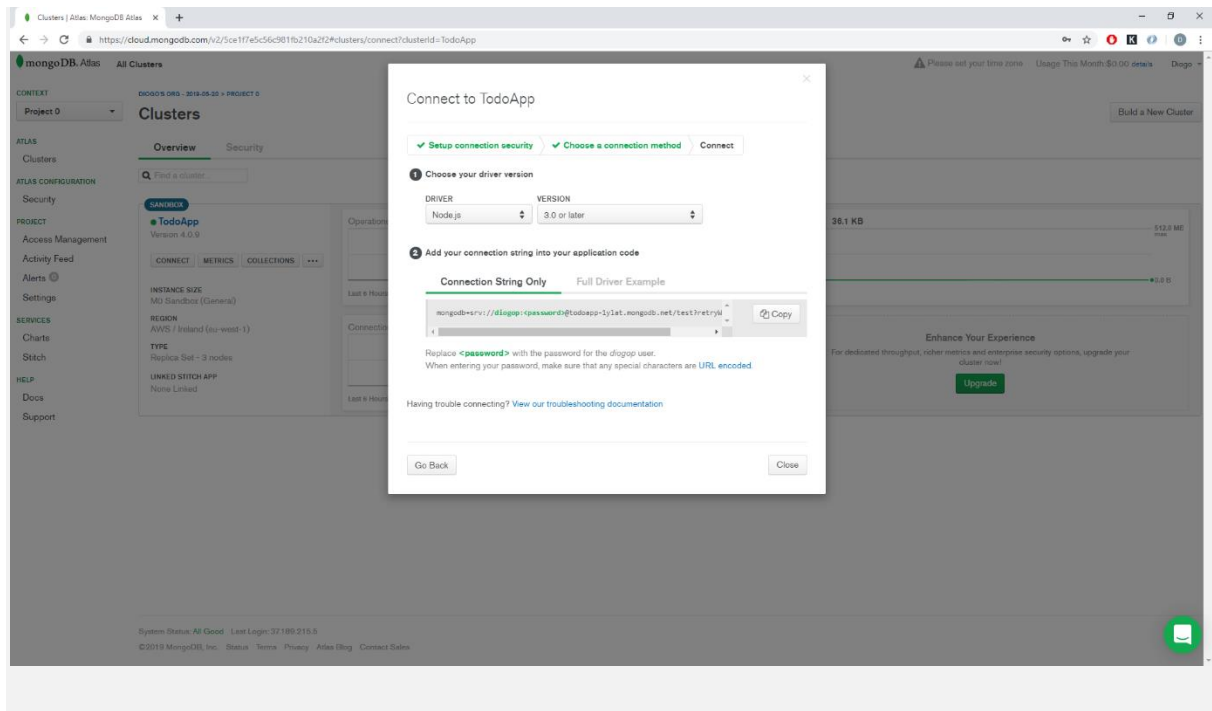
M0 clusters are best for getting started, and are not suitable for production environments.

100 max connections | Low network performance | 100 max databases | 500 max collections

After creating our cluster let's add a new user and then our ip address to the whitelist.



Now we need to get our connection String so we go to the Cluster **overview > connect > connect your application** and then we have our string.



## Db vs sql

MongoDB and SQL databases are two polar opposite sides of the backend world. The former deals with chaotic unstructured data, while the latter works with organized structured data. Both worlds have their own advantages and disadvantages and are meant for different types of use cases. In this article, we will do an in-depth comparison between MongoDB vs SQL databases, (to be precise MySQL database) and will also touch upon the important topic of how we can perform MongoDB analytics similar to the ease with which analytics are done on its SQL counterparts.

As we discussed, we will compare MongoDB with MySQL which is a well-known SQL database and most of our audience will be familiar with it. But it could have been any other SQL database also like Oracle, MS SQL Server, PostgreSQL, etc for our comparison. MySQL analytics is extremely common and is thus a good reference point for looking at doing analytics on Mongo.

schema-less. If you have a flexible schema, this is ideal for a document store like MongoDB. This is difficult to implement in a performant manner in RDBMS . Ease of scale-out. Scale reads by using replica sets. Scale writes by using sharding (auto balancing). Just fire up another machine and away you go. Adding more machines = adding more RAM over which to distribute your working set.



cost. Depends on which RDBMS of course, but MongoDB is free and can run on Linux, ideal for running on cheaper commodity kit.

you can choose what level of consistency you want depending on the value of the data (e.g. faster performance = fire and forget inserts to MongoDB, slower performance = wait til insert has been replicated to multiple nodes before returning)

Cons:

Data size in MongoDB is typically higher due to e.g. each document has field names stored it

less flexibility with querying (e.g. no JOINS)

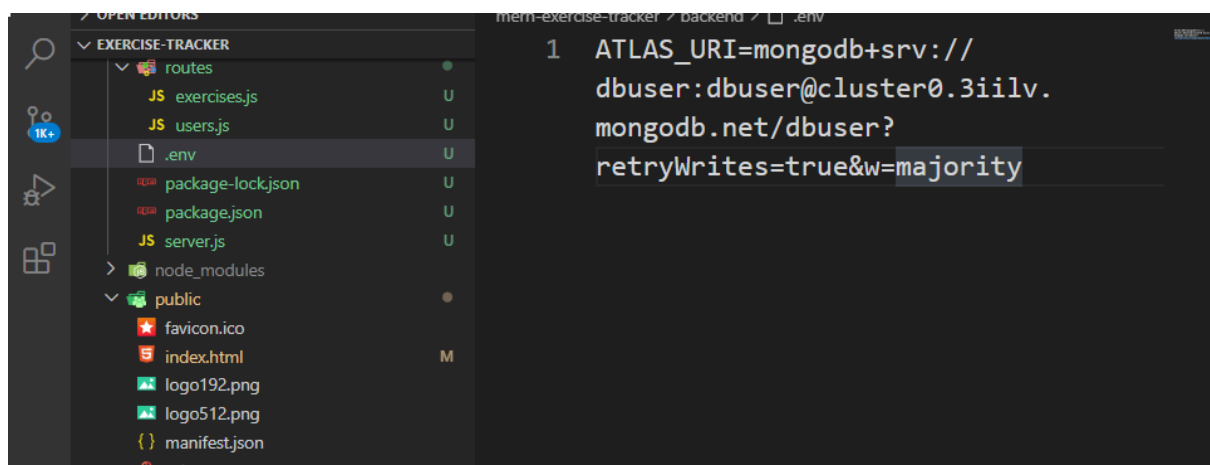
no support for transactions - certain atomic operations are supported, at a single document level

at the moment Map/Reduce (e.g. to do aggregations/data analysis) is OK, but not blisteringly fast. So if that's required, something like Hadoop may need to be added into the mix

less up to date information available/fast evolving product

I recently blogged my thoughts on MongoDB as someone coming from SQL Server background, so you might be interested in that (above are just some of the main points).

If you're looking for a "Is MongoDB better than RDBMS" answer - then IMHO there is no answer. NoSQL technologies like MongoDB provide an alternative, that complements RDBMS technologies. One may be better suited to a particular purpose than the other, so it's all about making a call on what is best for you for a given requirement.



So we took the link from our app cluster and connected the env variable to the cluster

```
mongodb+srv://dbuser:<password>@cluster0.3iilv.mongodb.net/<dbname>?retryWrites=true&w=majority
```

We used react for the interface

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

### **3. Database Design**

#### Constraints

##### Application constraints

- A maximum of 5000 registered user data can be managed by the Bus reservation system.
- A maximum of 200 buses can be added to the reservation system.
- A customer can reserve a maximum of 4 seats per ticket.
- There is no waiting list maintained for each bus. This means that a ticket cannot be booked if all the seats are reserved in a bus.
- Online payment option is not included in this application. Seats in a bus are reserved when a ticket is booked.
- Connecting buses are not shown in the search result. Results will only be shown if there is a single bus from Source to destination.
- Administrator can add a new bus only to the existing route. This means that a new bus is added with the same timings and the route of an existing bus to serve the demand during that period. There can only be a max of 5 extra buses that can be added for any existing bus.
- Advance booking is allowed only for duration of one month.

`_id : 5fbc22b6f9e93847807b36a4`

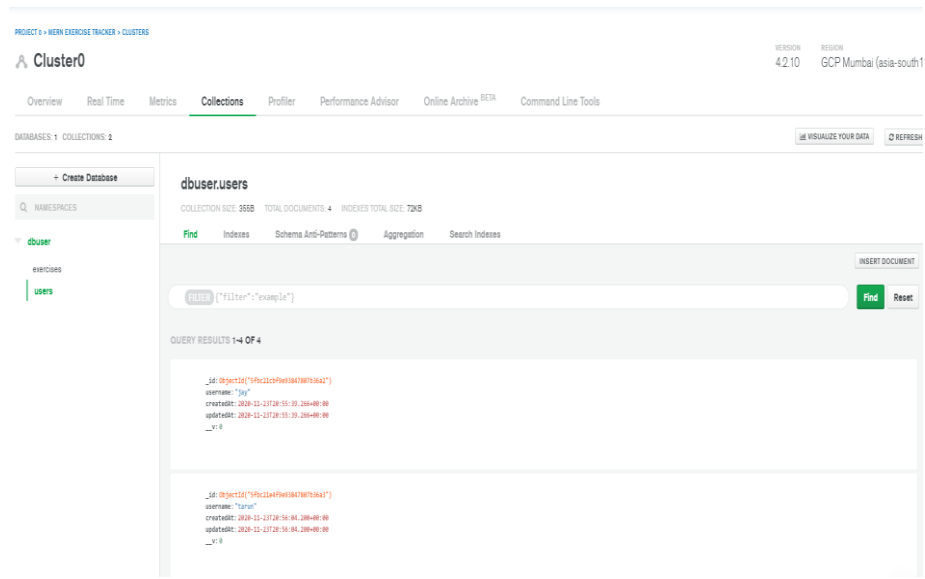
`username : " jay "`

`description : "run"`

`duration : 9`

`date : 2020-11-23T20:55:39.266+00:00 createdAt : 2020-11-23T20:59:34.569+00:00`

`updatedAt : 2020-11-23T20:59:34.569+00:00 __v : 0`



Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. ... This theoretical representation of the data is called an ontology.

The design process consists of the following steps:

Determine the purpose of your database. ...

Find and organize the information required. ...

Divide the information into tables. ...

Turn information items into columns. ...

Specify primary keys. ...

Set up the table relationships. ...

Refine your design. ...

Apply the normalization rules.

```
const app = express();  
const port = process.env.PORT || 5000;
```

Open the new Folder at VS Code

Now execute the following command at the VS Code terminal.

```
npm init
```

Installing Packages

1. First let's execute the following code to install Express, mongoose, ejs and dotenv.

```
npm install --save express mongoose ejs dotenv cors
```

2. Then we will install nodemon

```
npm install --save-dev nodemon
```

## 4.Graphical User Interface designing

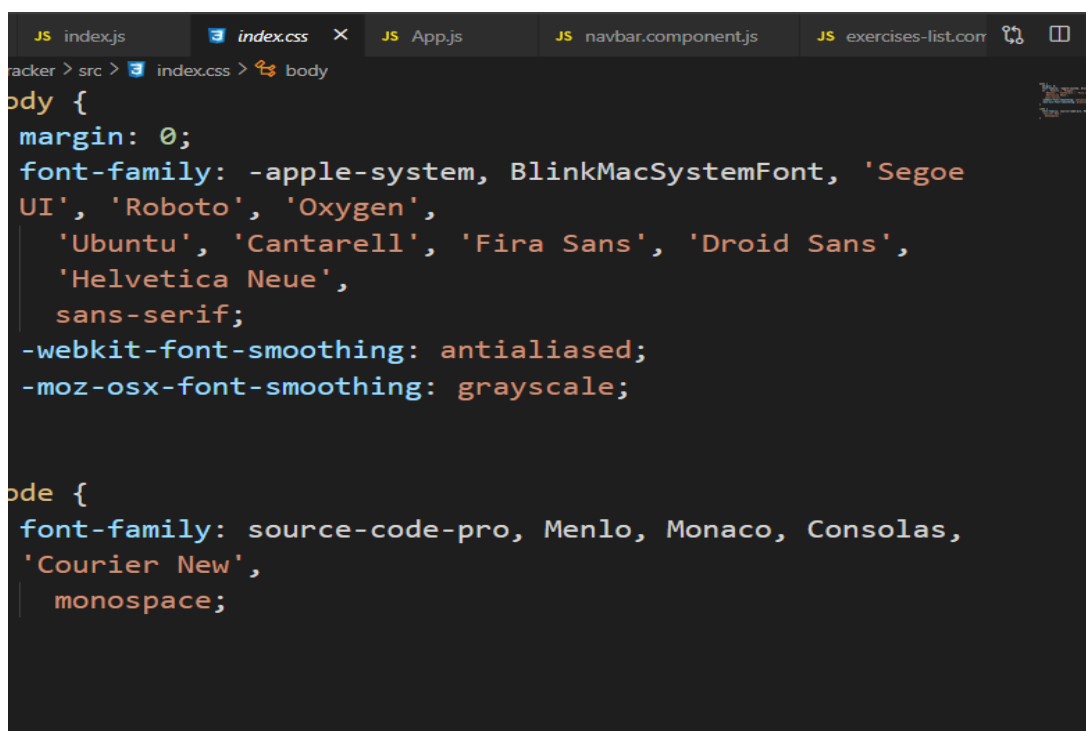
### Gui design

A graphical user interface builder (or GUI builder), also known as GUI designer, is a software development tool that simplifies the creation of GUIs by allowing the designer to arrange graphical control elements (often called widgets) using a drag-and-drop WYSIWYG editor. Without a GUI builder, a GUI must be built by manually specifying each widget's parameters in source-code, with no visual feedback until the program is run.

User interfaces are commonly programmed using an event-driven architecture, so GUI builders also simplify creating event-driven code. This supporting code connects widgets with the outgoing and incoming events that trigger the functions providing the application logic.

Some graphical user interface builders, such as e.g. Glade Interface Designer, automatically generate all the source code for a graphical control element. Others, like Interface Builder, generate serialized object instances that are then loaded by the application.

We made use of css html and js to design our site

A screenshot of a code editor with a dark theme. The editor has several tabs at the top: 'JS index.js', 'index.css' (active), 'JS App.js', 'JS navbar.component.js', and 'JS exercises-list.com'. The active tab 'index.css' shows CSS code. The code is as follows:

```
body {  
  margin: 0;  
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',  
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans',  
    'Helvetica Neue',  
    sans-serif;  
  -webkit-font-smoothing: antialiased;  
  -moz-osx-font-smoothing: grayscale;  
  
code {  
  font-family: source-code-pro, Menlo, Monaco, Consolas,  
    'Courier New',  
    monospace;  
}
```

After the execution of react files we created .js files

And connected them through Axios

Using

```
npm install axios
```

The screenshot shows the 'Create New Exercise Log' form in the ExcerTracker application. The form has a dark header with the title 'ExcerTracker' and navigation links 'Exercises', 'Create Exercise Log', and 'Create User'. The form fields are: 'Username' (a dropdown menu with 'Satyam' selected), 'Description' (a text input with 'Walk'), 'Duration (in minutes):' (a text input with '40'), and 'Date:' (a date picker showing '11/24/2020'). Below the date input is a calendar for November 2020, with the 24th highlighted.

```
import React, { Component } from 'react';  
import axios from 'axios';  
import DatePicker from 'react-datepicker';  
import "react-datepicker/dist/react-datepicker.css";
```

Datepicker is installed so that we can access the dates

```
npm install datepicker
```

The screenshot shows the 'Create New User' form in the ExcerTracker application. The form has a dark header with the title 'ExcerTracker' and navigation links 'Exercises', 'Create Exercise Log', and 'Create User'. The form has a single text input field for 'Username:' and a blue 'Create User' button below it.

Whenever a new user is put up the entry is shown in the list tab

```

const Exercise = props => (
  <tr>
    <td>{props.exercise.username}</td>
    <td>{props.exercise.description}</td>
    <td>{props.exercise.duration}</td>
    <td>{props.exercise.date.substring(0,10)}</td>
  >
    <td>
      <Link to={"/edit/"+props.exercise._id}>edit
</Link> | <a href="#" onClick={() => { props.deleteExercise(props.exercise._id) }}>delete</a>
    </td>
  </tr>
)

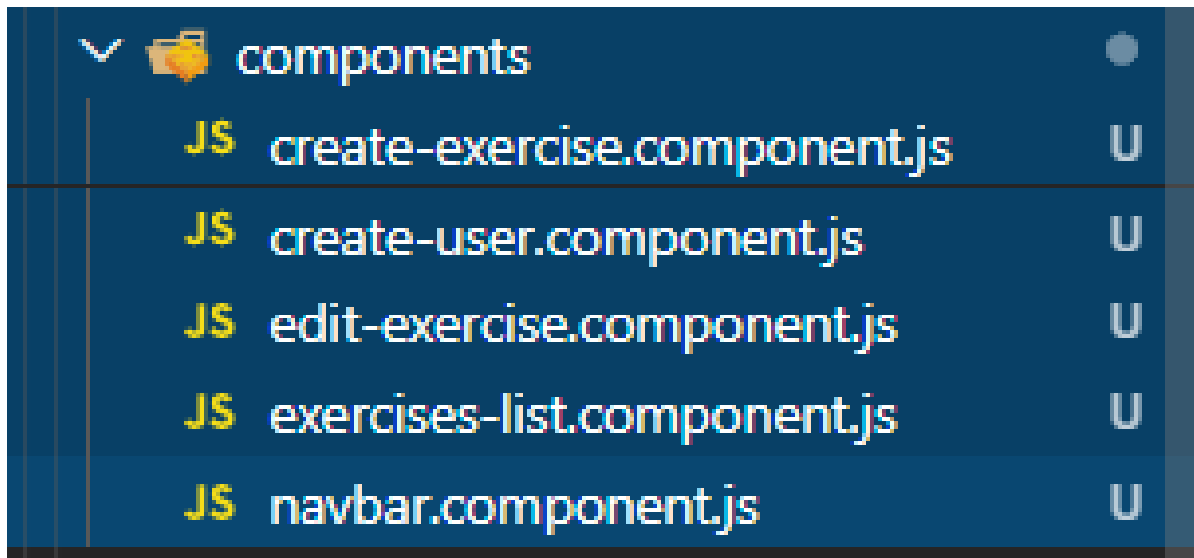
```

ExcerTracker Exercises Create Exercise Log Create User

#### Logged Exercises

Username	Description	Duration	Date	Actions
jay	run	9	2020-11-23	<a href="#">edit</a>   <a href="#">delete</a>
jay	bike ride	30	2020-11-27	<a href="#">edit</a>   <a href="#">delete</a>
jay	run	90	2020-11-25	<a href="#">edit</a>   <a href="#">delete</a>
tarun	swim	30	2020-11-24	<a href="#">edit</a>   <a href="#">delete</a>
Satyam	walk	60	2020-11-27	<a href="#">edit</a>   <a href="#">delete</a>





▼	📁	components	●
	JS	create-exercise.component.js	U
	JS	create-user.component.js	U
	JS	edit-exercise.component.js	U
	JS	exercises-list.component.js	U
	JS	navbar.component.js	U

These Files were used to make our react page working

### **5.Connectivity of GUI with Database**

MongoDB is a NoSQL Database, we are going to be using this for creating and querying the database.

There are more NoSQL Databases like Redis, Amazon DynamoDB, Azure Cosmos DB, Couchbase, and MongoDB.

MongoDB is quite popular among them and it is used in all types of stacks like MERN and MEAN stacks

```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');

require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

const uri = process.env.ATLAS_URI;
mongoose.connect(uri, { useNewUrlParser: true, useCreateIndex: true });

const connection = mongoose.connection;
connection.once('open', () => {
  console.log("MongoDB database connection established successfully");
})
```

```
const exercisesRouter = require('./routes/exercises');
const usersRouter = require('./routes/users');

app.use('/exercises', exercisesRouter);
app.use('/users', usersRouter);

app.listen(port, () => {
  console.log(`Server is running on port: ${port}`);
});
```

GET ▼ http://localhost:5000/exercises/ Send 200 OK 20.2 ms 422 B 11 Hours Ago ▼

**JSON** ▼ Auth ▼ Query Header 1 Docs

```
1 {
2   "username": " jay ",
3   "description": "bike ride",
4   "duration": 30,
5   "date": "2020-11-27T20:55:39.266Z"
6 }
```

**Preview** ▼ Header 8 Cookie Timeline

```
1 [
2   {
3     "_id": "5fbc22b6f9e93847807b36a4",
4     "username": " jay ",
5     "description": "run",
6     "duration": 9,
7     "date": "2020-11-23T20:55:39.266Z",
8     "createdAt": "2020-11-23T20:59:34.569Z",
9     "updatedAt": "2020-11-23T20:59:34.569Z",
10    "__v": 0
11  },
12  {
13    "_id": "5fbc22d4f9e93847807b36a5",
14    "username": " jay ",
15    "description": "bike ride",
16    "duration": 30,
17    "date": "2020-11-27T20:55:39.266Z",
18    "createdAt": "2020-11-23T21:00:04.996Z",
19    "updatedAt": "2020-11-23T21:00:04.996Z",
20    "__v": 0
21  }
22 ]
```

Beautify JSON \$.store.books[\*].author ?

## 6.Conclusion

Whether you need a to-do list or not is of personal preference. If you, however, keep a to-do list, it's imperative that you get the inversion: a not-to-do list.

It helps to weed out the unwanted stuff and focus on the things that matter most. Creating a not-to-do list should take no more than half an hour. Once you get it, however, you'll recognize that your to-do list shrinks to its half over time.

Schedule a one-on-one meeting with yourself and compile a simple list of tasks and activities that you want to get rid of. Cut the nonsense.

---

### Logged Exercises

Username	Description	Duration	Date	Actions
jay	run	9	2020-11-23	<a href="#">edit</a>   <a href="#">delete</a>
jay	bike ride	30	2020-11-27	<a href="#">edit</a>   <a href="#">delete</a>
jay	run	90	2020-11-25	<a href="#">edit</a>   <a href="#">delete</a>
tarun	swim	30	2020-11-24	<a href="#">edit</a>   <a href="#">delete</a>
Satyam	walk	60	2020-11-27	<a href="#">edit</a>   <a href="#">delete</a>

Edit and Delete button are made so that user can update its data whenever he/she feels like