

Department of Computer Science & Engineering  
CSL7090 - Software & Data Engineering (SDE)

# COURSE PROJECT

**Prepared by:** Aviral Tripathi & Ritik Kumar

**Roll no:** m22ma012 & m22ma009

**Instructor:** Dr. Sumit Kalra

# Contents

<b>1</b>	<b>Research Paper</b>	<b>1</b>
1.1	Node.js: Using JavaScript to Build High-Performance Network Programs (IEEE) . . . . .	1
<b>2</b>	<b>Introduction</b>	<b>3</b>
2.1	MERN Stack . . . . .	3
2.1.1	<b>Mongo DB</b> . . . . .	3
2.1.2	Express JS: . . . . .	3
2.1.3	React: . . . . .	3
2.1.4	Node JS: . . . . .	3
<b>3</b>	<b>Installations</b>	<b>4</b>
3.1	Installing Node.JS . . . . .	4
3.2	Initialising a new React project. . . . .	4
3.3	Client side routing . . . . .	6
3.4	Installing Express . . . . .	6
3.5	Install Nodemon . . . . .	7
3.6	Installing CORS . . . . .	8
3.7	Installing JSON Web Token (JWT) . . . . .	8
3.8	Installing cookie-parser . . . . .	8
3.9	react-quill . . . . .	9
3.10	Multer library . . . . .	9
3.11	date-fns . . . . .	9
3.12	dotenv . . . . .	9
3.13	connect-mongo & express-session . . . . .	9
<b>4</b>	<b>Task 2: Setting up MongoDB database</b>	<b>10</b>
4.1	Setting up Mongo DB . . . . .	10
4.2	Installing bcryptjs . . . . .	14
<b>5</b>	<b>Final Results</b>	<b>15</b>
5.1	Starting the App . . . . .	15
<b>6</b>	<b>GitHub Repository Link</b>	<b>20</b>
<b>7</b>	<b>References</b>	<b>21</b>

# 1 Research Paper

## 1.1 Node.js: Using JavaScript to Build High-Performance Network Programs (IEEE)



Figure 1.1: Node JS

The paper explains about the working of Node JS along with its various advantages and how it compares with the 'Multi-threaded Environments', here are some key points from the paper:

- Node JS is a server-side JavaScript environment, for long running server processes.
- It uses the asynchronous event model i.e there is only one thread that is processed at one time (unlike multi-threaded environments where multiple threads can be processed at one time), by switching between different threads for code execution.
- When one thread is busy the processor jumps to another thread

The need for an Asynchronous Event based environment arised because of the disadvantages of multi-threaded environments:

- Deadlock between threads on shared resources
- Less control to developer, as OS itself decides which thread to exec and for how long

### Example for the working of Node JS

If the application writes to a socket and fills the socket's underlying buffer, ordinarily, the socket blocks the application's writes until buffer space becomes available, thus preventing the application from doing any other useful work.

But, if the socket is non-blocking, it instead returns an indication to the application that further writing isn't currently possible, thereby informing the application that it should try again later. Assuming the application has registered interest with the event notification system in that socket, it can go do something else, knowing that it will receive an event when the socket's write buffer has available space.

- JavaScript is a functional language and, as such, supports higher-order functions.
- Every I/O operation is handled by means of higher-order functions — i.e, functions taking functions as a parameter — that specify what to do when there's something to do.

```

var sys = require(\sys"),
    http = require(\http"),
    url = require(\url"),
    path = require(\path"),
    fs = require(\fs");
http.createServer(function(request, response) {
    var uri = url.parse(request.url).pathname;
    var filename = path.join(process.cwd(), uri);
    path.exists(filename, function(exists) {
        if(exists) {
            fs.readFile(filename, function(err, data) {
                response.writeHead(200);
                response.end(data);
            });
        } else {
            response.writeHead(404);
            response.end();
        }
    });
}).listen(8080);
sys.log(\Server running at http://localhost:8080/");

```

### A simple HTTP file server.

- The `http.createServer()` function, which is a wrapper around a low-level efficient HTTP protocol implementation, is passed a function as the only argument.
- This function is invoked whenever data for a new request is ready to be read. Node offers no opportunity to read a file synchronously — the only option is to register another function via `readFile()` that gets invoked whenever data can be read.

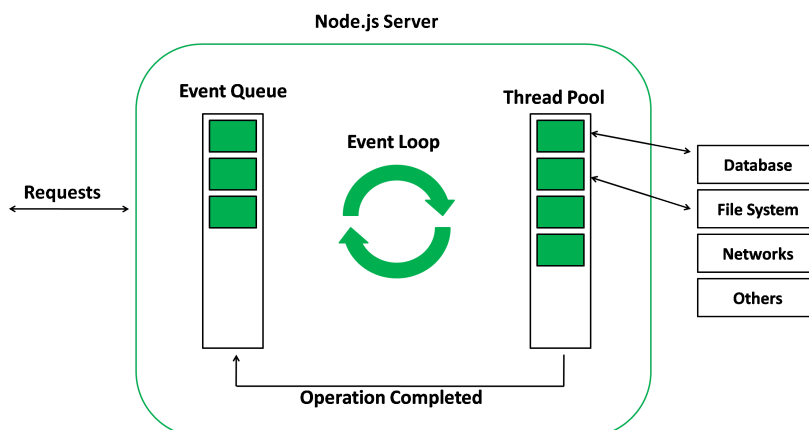


Figure 1.2: Node JS Event Loop

## 2 Introduction

### 2.1 MERN Stack

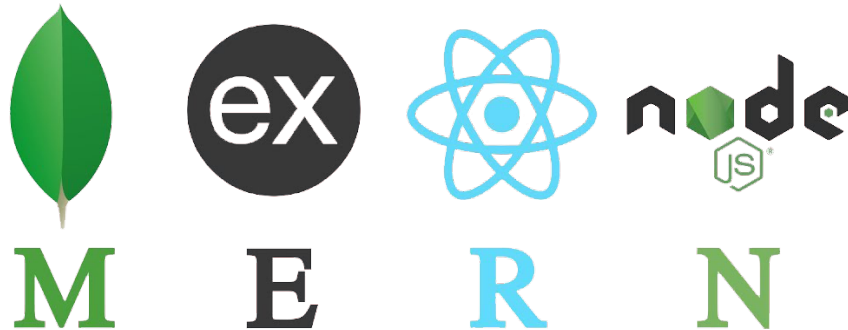


Figure 2.1: MERN Stack

MERN stack is a technology stack comprising of:

- MongoDB Database: for storing data
- Express JS: for backend and Routes
- React: for frontend
- Node JS: JavaScript execution environment (backend)

#### 2.1.1 Mongo DB

It is a cloud based database service that can be used to store the data such as text, username, passwords, and files, we will be using this database to store the Post contents and user data such as username and passwords.

#### 2.1.2 Express JS:

It helps in creating routes (URLs) and handle requests and responses. It makes it easier to manage the server-side logic of your application.

#### 2.1.3 React:

It is used to build the frontend or the user interface of a website

#### 2.1.4 Node JS:

It enables you to use JavaScript not only for frontend development but also for server-side development, in an asynchronous manner.

## 3 Installations

### 3.1 Installing Node.JS

To install node.js, got to the official website of Node, and install the .msi installation file for windows installation. After the download follow the steps of installation wizard to install node.

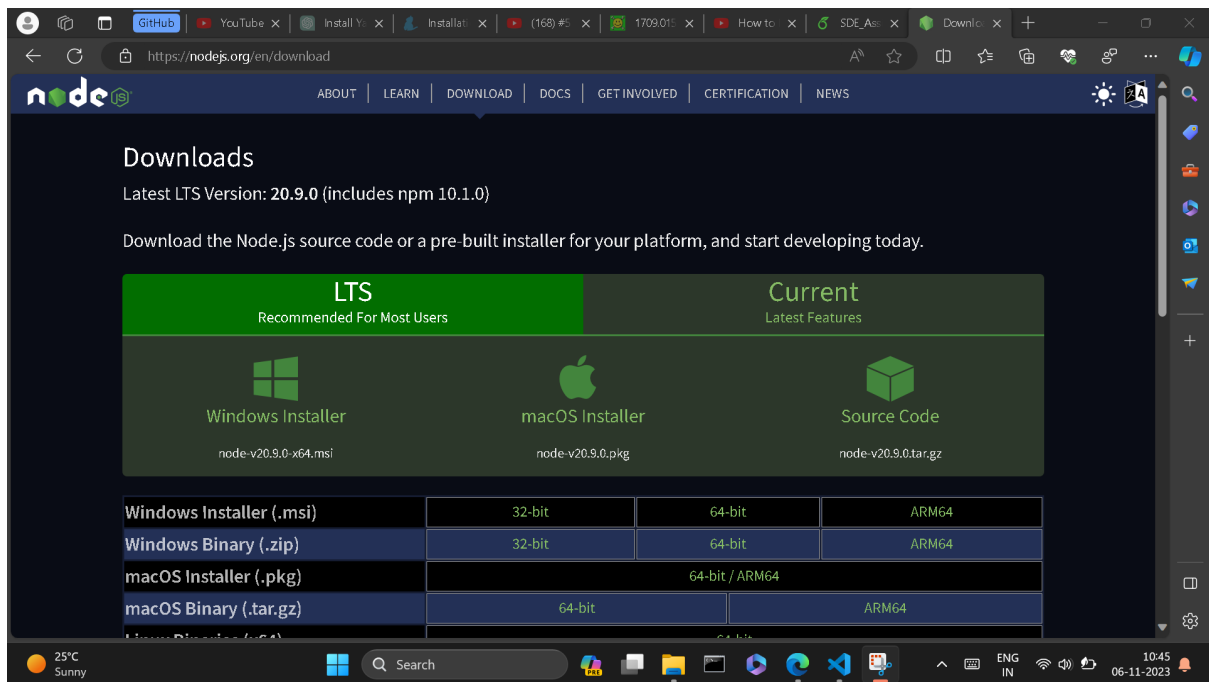


Figure 3.1: Node JS download



Figure 3.2: Verifying Node JS installation

### 3.2 Initialising a new React project.

To initialise a new react project (web-app) we need to run the following commands

```
Set-ExecutionPolicy -Scope CurrentUser RemoteSigned  
yarn create react-app .
```

The first command is used to change the execution policy for the current user (scripts and configuration files that can be run for current user), the RemoteSigned keyword allows to download the scripts without requiring a digital signature

The second command is to define the predefined structure and configuration of a react app (the dot in the end signifies the current directory) and also install the required dependencies, and provides a basic app template overall.

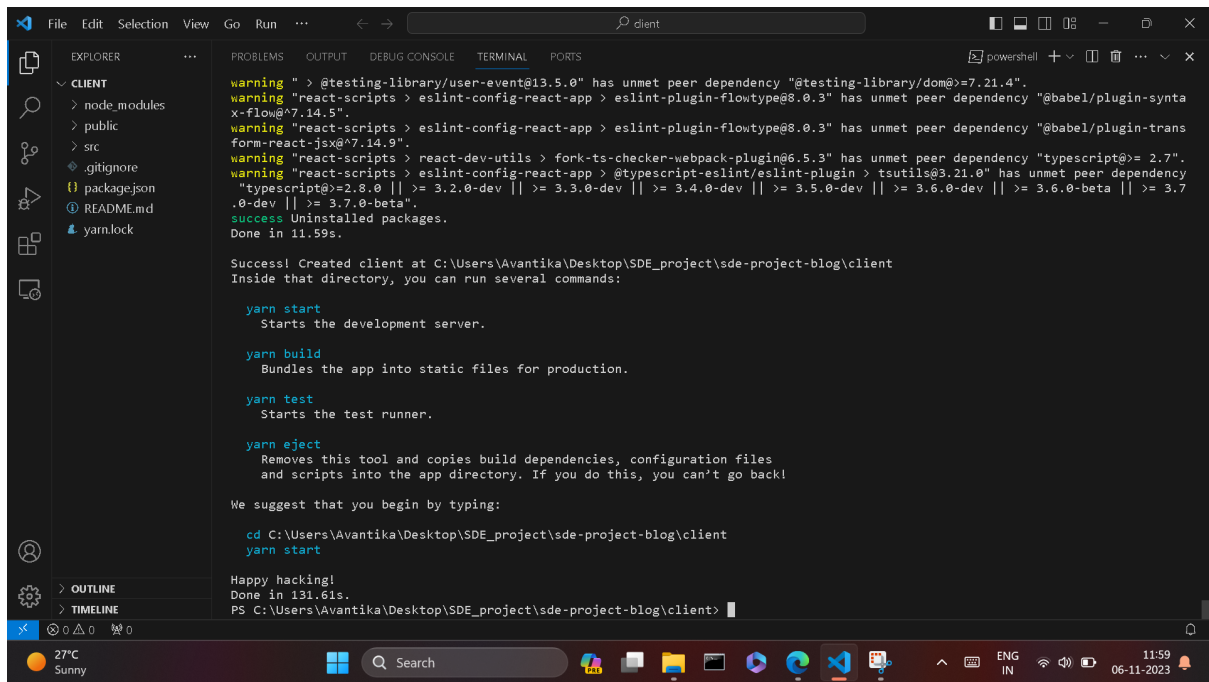


Figure 3.3: Create React App (CRA)

To start the react app run the command:

```
yarn start
```

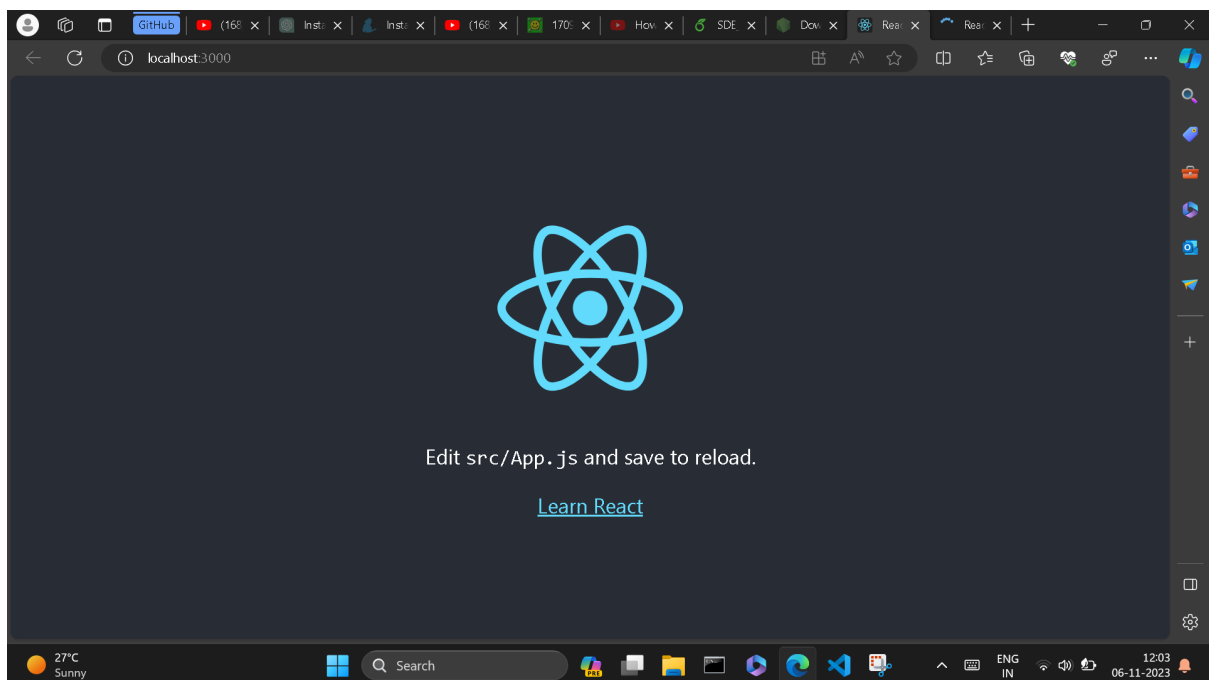


Figure 3.4: React started

### 3.3 Client side routing

Client side routing works well for small or single page websites, it helps in faster navigation through the website, as it loads the web app at once and then dynamically updates the content as per the user request.

i am using **react-router-dom** library to implement client-side routing.

```
yarn react-router-dom
```

```
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\client> yarn add react-router-dom
peer dependency "@testing-library/dom@>=7.21.4".
warning "react-scripts > eslint-config-react-app > eslint-plugin-flowtype@8.0.3" has unmet peer dependency "@babel/plugin-syntax-flow@^7.14.5".
warning "react-scripts > eslint-config-react-app > eslint-plugin-flowtype@8.0.3" has unmet peer dependency "@babel/plugin-transform-react-jsx@^7.14.9".
warning "react-scripts > react-dev-utils > fork-ts-checker-webpack-plugin@6.5.3" has unmet peer dependency "typescript@>= 2.7".
warning "react-scripts > eslint-config-react-app > @typescript-eslint/eslint-plugin > tsutils@3.21.0" has unmet peer dependency "typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta".
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 2 new dependencies.
info Direct dependencies
├─ react-router-dom@6.18.0
└─
info All dependencies
├─ react-router-dom@6.18.0
└─ react-router@6.18.0
Done in 24.96s.
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\client>
```

Figure 3.5: react-router-dom Installed

### 3.4 Installing Express

The main app is contained in the 'client' directory and the backend APIs are managed in 'api' directory, we change the directory to 'api' and run this command to install 'Express'

```
yarn add express
```



```

● PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\api> yarn add express
yarn add v1.22.19
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 36 new dependencies.
info Direct dependencies
├─ express@4.18.2
info All dependencies
├─ accepts@1.3.8
├─ array-flatten@1.1.1
├─ body-parser@1.20.1
├─ call-bind@1.0.5
├─ content-disposition@0.5.4
├─ cookie-signature@1.0.6
├─ cookie@0.5.0
├─ define-data-property@1.1.1
├─ ee-first@1.1.1
├─ express@4.18.2
├─ finalhandler@1.2.0
├─ forwarded@0.2.0
├─ fresh@0.5.2
├─ http-errors@2.0.0
├─ http-status@1.6.2
├─ iconv-lite@0.6.3
├─ ipaddr.js@1.9.1
├─ merge-descriptors@1.0.1
├─ on-finished@2.4.1
├─ on-headers@1.0.2
├─ parseurl@1.3.3
├─ path-to-regexp@0.1.7
├─ qs@6.11.0
├─ range-parser@1.2.1
├─ raw-body@2.5.1
├─ safe-buffer@5.2.1
├─ send@0.18.0
├─ serve-static@1.15.0
├─ setprototypeof@1.2.0
├─ statuses@1.5.0
├─ toidentifier@1.0.1
├─ type-is@1.6.18
├─ unpipe@1.0.0
├─ utils-merge@1.0.1
├─ vary@1.1.2
├─ wds@2.0.1
├─ which@2.0.2
├─ wrap-ansi@7.0.0
├─ y18n@5.0.8
└─ yargs@17.7.2

```

Figure 3.6: Express installed in 'api' directory

### 3.5 Install Nodemon

Nodemon helps to automatically restart the Node.js server (app) whenever we edit/change a file, following is the command to install 'nodemon'.

```
yarn global add nodemon
```

```

PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\api> yarn global add nodemon
yarn global v1.22.19
warning package.json: No license field
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Installed "nodemon@3.0.1" with binaries:
  - nodemon
Done in 5.89s.
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\api> 

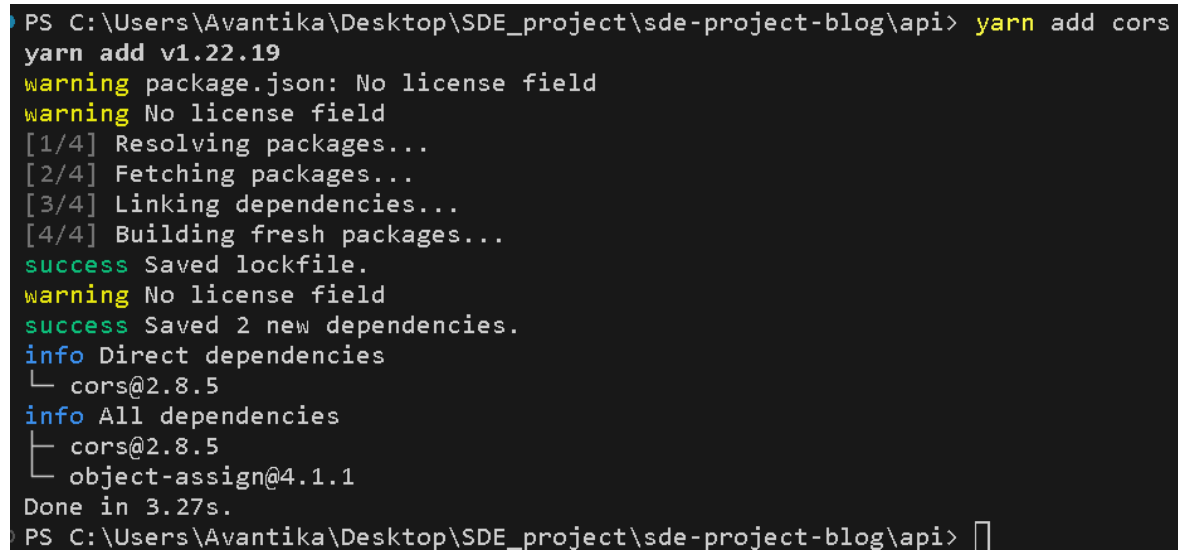
```

Figure 3.7: Nodemon installed

### 3.6 Installing CORS

CORS (Cross Origin Resource Sharing) is used to prevent web pages hosted at one server to make requests at the other server, but when required this constrain can be relaxed as well.

```
yarn add cors
```



```
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\api> yarn add cors
yarn add v1.22.19
warning package.json: No license field
warning No license field
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
warning No license field
success Saved 2 new dependencies.
info Direct dependencies
└─ cors@2.8.5
info All dependencies
└─ cors@2.8.5
   └─ object-assign@4.1.1
Done in 3.27s.
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\api> █
```

Figure 3.8: Cors installed in 'api' directory

### 3.7 Installing JSON Web Token (JWT)

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.

```
yarn add jsonwebtoken
```

### 3.8 Installing cookie-parser

whenever a user visits the website, and their browser sends cookies, the app uses cookie-parser to automatically understand and organize those cookies.

```
yarn add cookie-parser
```

### 3.9 react-quill

React Quill is a text editor that will be used to edit or write posts in the web-app as well as format the text on the web page itself.

```
yarn add react-quill
```

### 3.10 Multer library

Multer library is used to for handling file uploads in Express and Node

```
yarn add multer
```

### 3.11 date-fns

This library is used to deal with dates in JavaScript, it will be used to display the date when a post is published

```
yarn add date-fns
```

### 3.12 dotenv

Dotenv is used to load environment variables from a .env file. This is useful for storing configuration settings and sensitive information (MongoDB connection string with password) without hardcoding them in your code.

### 3.13 connect-mongo & express-session

express-session is used for session management and connect-mongo is used along with it to store the user session data in MongoDB database instead of server memory this is used to prevent data loss in case of server restart

```
npm i connect-mongo  
npm i express-session
```

## 4 Task 2: Setting up MongoDB database



Figure 4.1: MongoDB

### 4.1 Setting up Mongo DB

MongoDB is a database that we will be using for storing the various user information such as username and passwords in it, as well as the posts. To set up MongoDB, we need to first go to the official website of MongoDB and create a 'New Project' by clicking on the button at top right corner.

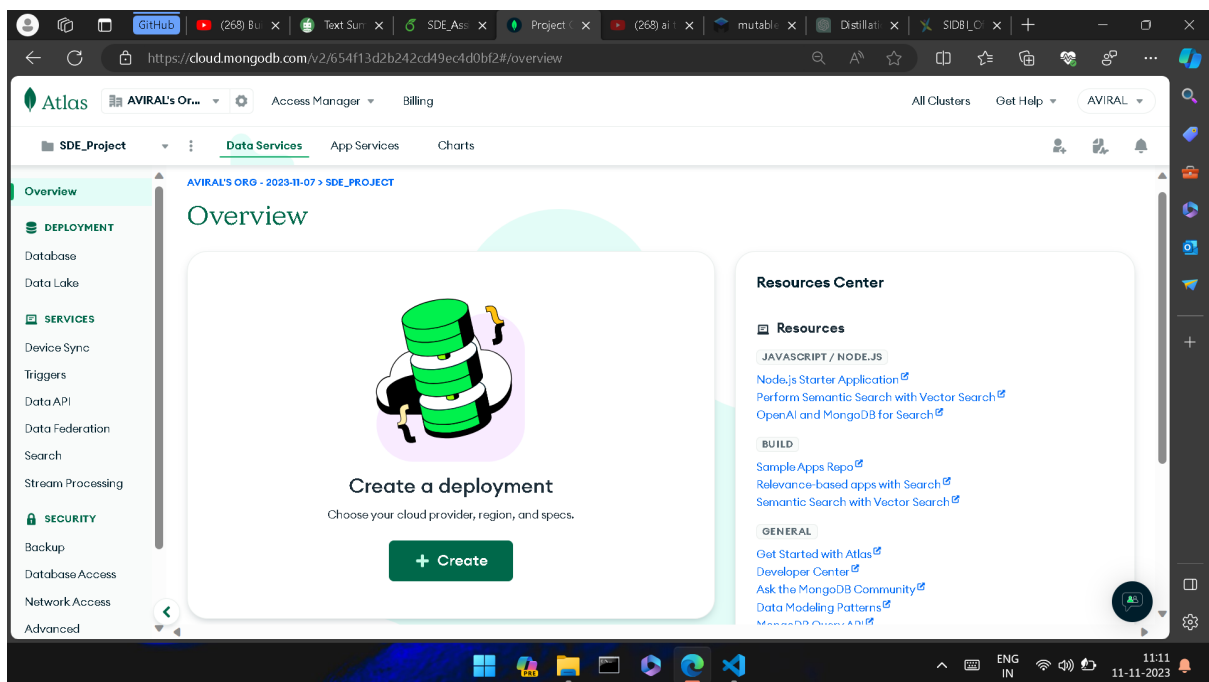


Figure 4.2: MongoDB New project (SDE\_Node\_JS\_Blog) created

click on "create" button to create a new cluster, and follow the required steps. Then to connect with the code we need to install mongoose

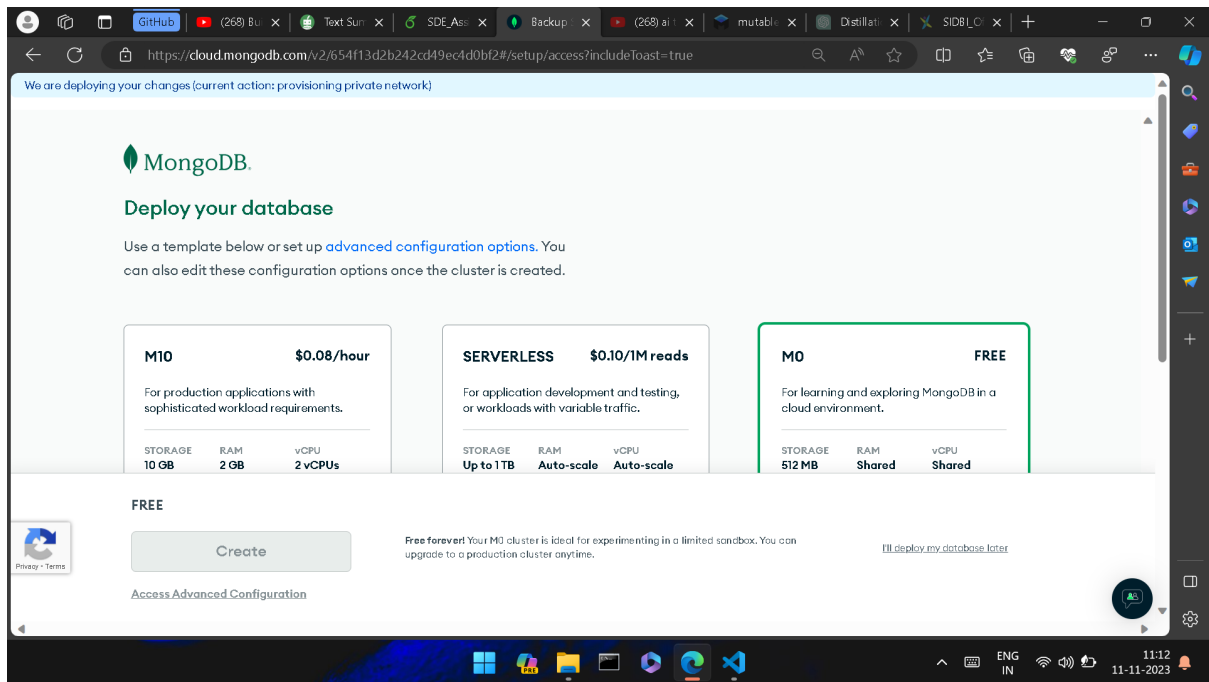


Figure 4.3: MongoDB Deployment using 'free' version

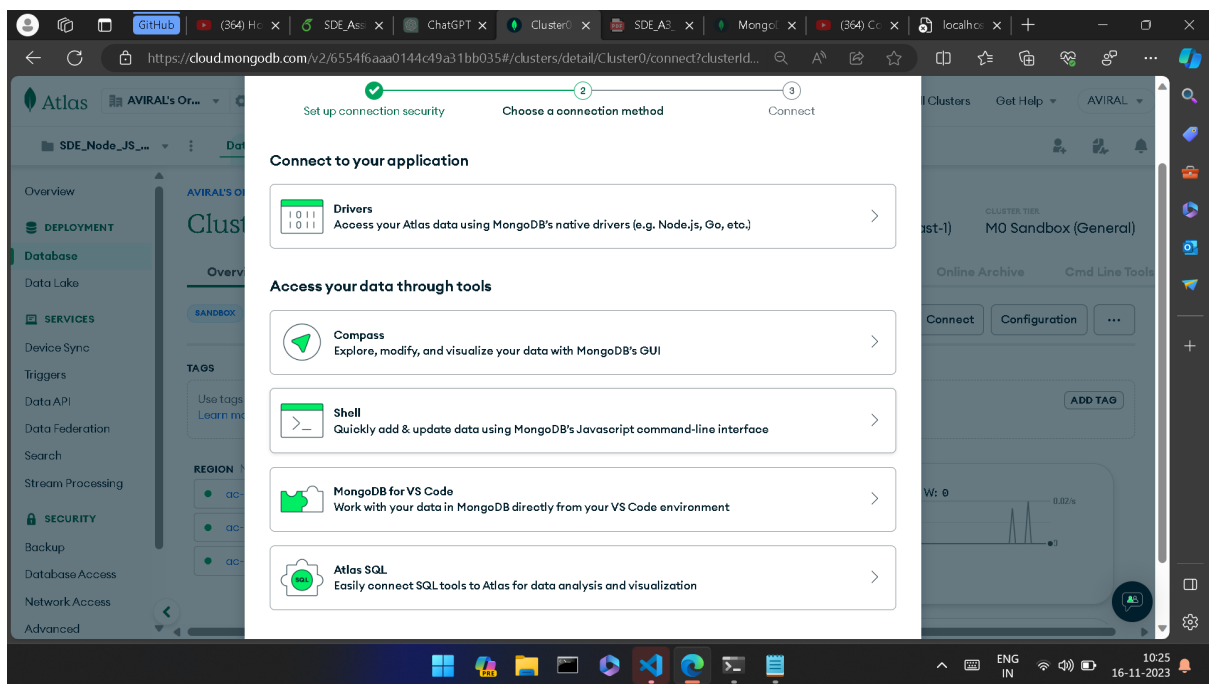


Figure 4.4: connecting MongoDB to VS Code

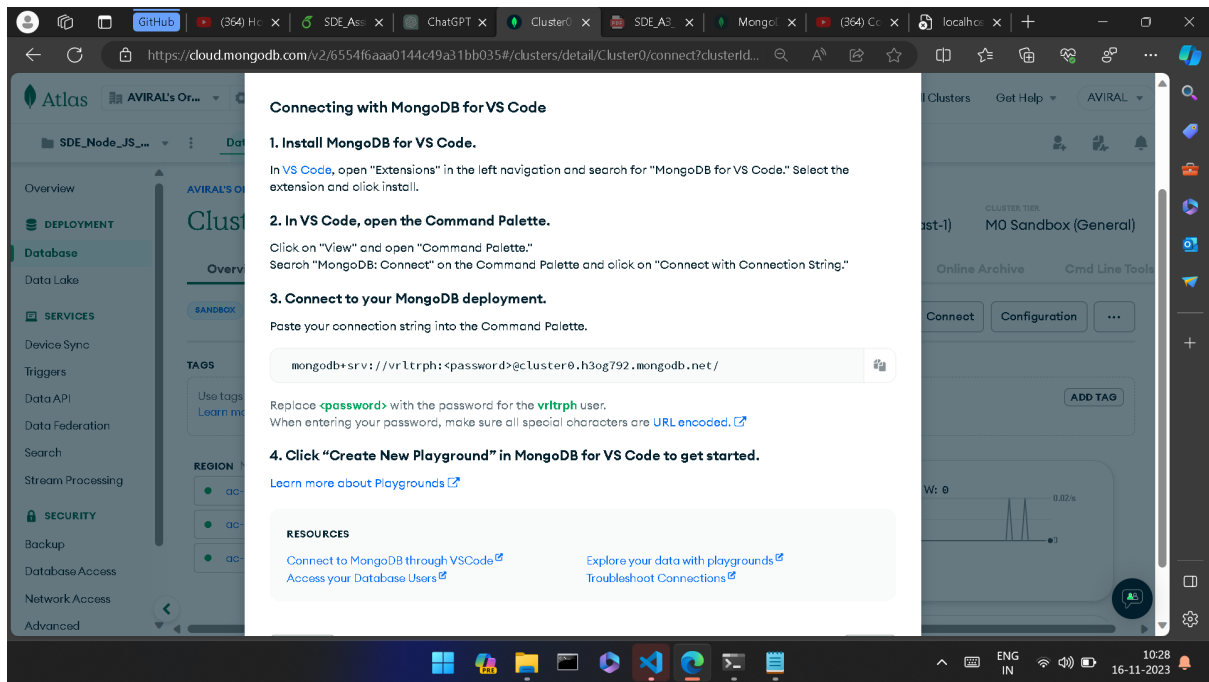


Figure 4.5: Copy the connection string and put it in .env file in VS Code

```
yarn add mongose
```

```
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\client> yarn add mongose
yarn add v1.22.19
[1/4] Resolving packages...
info There appears to be trouble with your network connection. Retrying...
[2/4] Fetching packages...
[3/4] Linking dependencies...
warning " > @testing-library/user-event@13.5.0" has unmet peer dependency "@testing-library/dom@>=7.21.4".
warning "react-scripts > eslint-config-react-app > eslint-plugin-flowtype@8.0.3" has unmet peer dependency "@babel/plugin-syntax-flow@^7.14.5".
warning "react-scripts > eslint-config-react-app > eslint-plugin-flowtype@8.0.3" has unmet peer dependency "@babel/plugin-transform-react-jsx@^7.14.9".
warning "react-scripts > react-dev-utils > fork-ts-checker-webpack-plugin@6.5.3" has unmet peer dependency "typescript@>= 2.7".
warning "react-scripts > eslint-config-react-app > @typescript-eslint/eslint-plugin > tsutils@3.21.0" has unmet peer dependency "typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta".
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 1 new dependency.
info Direct dependencies
└─ mongose@0.0.2-security
info All dependencies
└─ mongose@0.0.2-security
Done in 45.61s.
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\client>
```

Figure 4.6: MongoDB installed on our computer

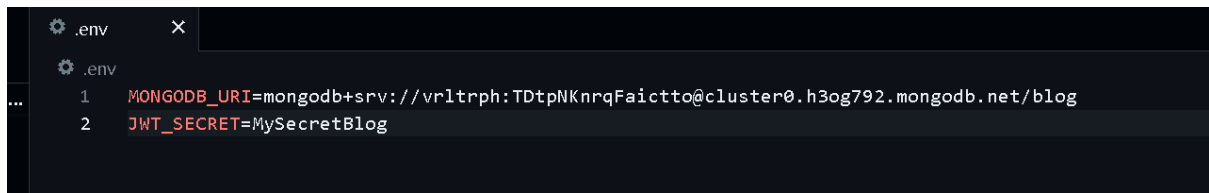


Figure 4.7: .env file in VS Code

This code is then used to connect with the database:

```
const mongoose = require('mongoose');
const connectDB = async () => {

  try {
    mongoose.set('strictQuery', false);
    const conn = await mongoose.connect(process.env.MONGODB_URI);
    console.log(`Database Connected: ${conn.connection.host}`);
  } catch (error) {
    console.log(error);
  }

}

module.exports = connectDB;
```

once we are connected to MongoDB, we will see the data in it in the 'collections' tab:

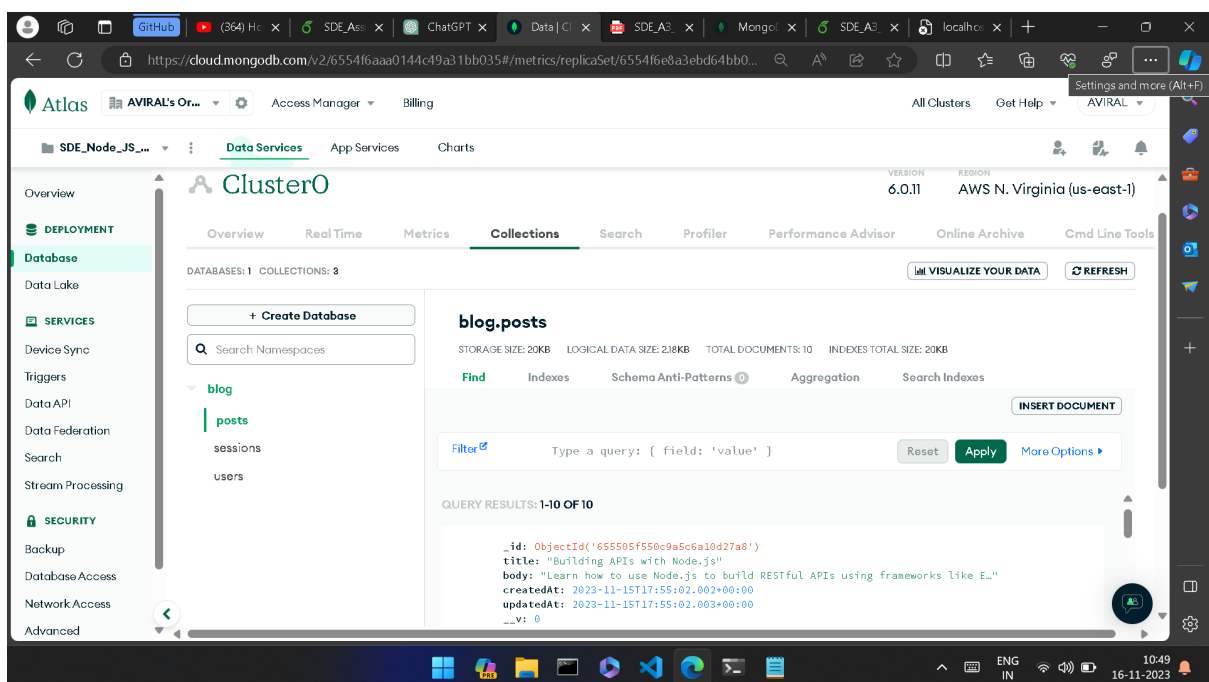


Figure 4.8: MongoDB Collections

## 4.2 Installing bcryptjs

In the database the passwords are directly exposed to anyone who has an access to the database, we use bcryptjs to encrypt (hash) the passwords, so they are no seen as plane text or strings.

```
yarn add bcryptjs
```

```
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\client\src\pages> yarn add bcryptjs
yarn add v1.22.19
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
warning " > @testing-library/user-event@13.5.0" has unmet peer dependency "@testing-library/dom@
>=7.21.4".
warning "react-scripts > eslint-config-react-app > eslint-plugin-flowtype@8.0.3" has unmet peer
dependency "@babel/plugin-syntax-flow@^7.14.5".
warning "react-scripts > eslint-config-react-app > eslint-plugin-flowtype@8.0.3" has unmet peer
dependency "@babel/plugin-transform-react-jsx@^7.14.9".
warning "react-scripts > react-dev-utils > fork-ts-checker-webpack-plugin@6.5.3" has unmet peer
dependency "typescript@>= 2.7".
warning "react-scripts > eslint-config-react-app > @typescript-eslint/eslint-plugin > tsutils@3.
21.0" has unmet peer dependency "typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-
dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta".
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 1 new dependency.
info Direct dependencies
└─ bcryptjs@2.4.3
info All dependencies
└─ bcryptjs@2.4.3
Done in 30.25s.
PS C:\Users\Avantika\Desktop\SDE_project\sde-project-blog\client\src\pages> █
```

Figure 4.9: Bcryptjs installation



## 5 Final Results

After all the front-end and back-end implementation our MERN blog is ready to use, and have the following features:

- A user can Register with unique 'username' and 'password'.
- A user can login with previously registered username and password.
- A Registered user can 'create', 'edit', and 'delete' the posts in the blog.
- All the data such as: usernames, passwords, post contents, and a user's session data is stored in Mongo DB database.

### 5.1 Starting the App

To start the app we just need to navigate to the project directory in PowerShell and type: ***npm run dev***, this command will run the 'dev' script for the project present in packages.json file

```
PS C:\Users\Avantika\Desktop\NodeJs-Express-EJS-MongoDB--Blog-SDE_Project> npm run dev
> nodejs-blog@1.0.0 dev
> nodemon app.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
App listening on port 5000
Database Connected: ac-fphzobd-shard-00-00.h3og792.mongodb.net
```

Figure 5.1: Starting the MERN blog app

As the app is running on port 5000 (by default), we go to the browser and type 'localhost:5000' and press enter to display the homepage of the blog

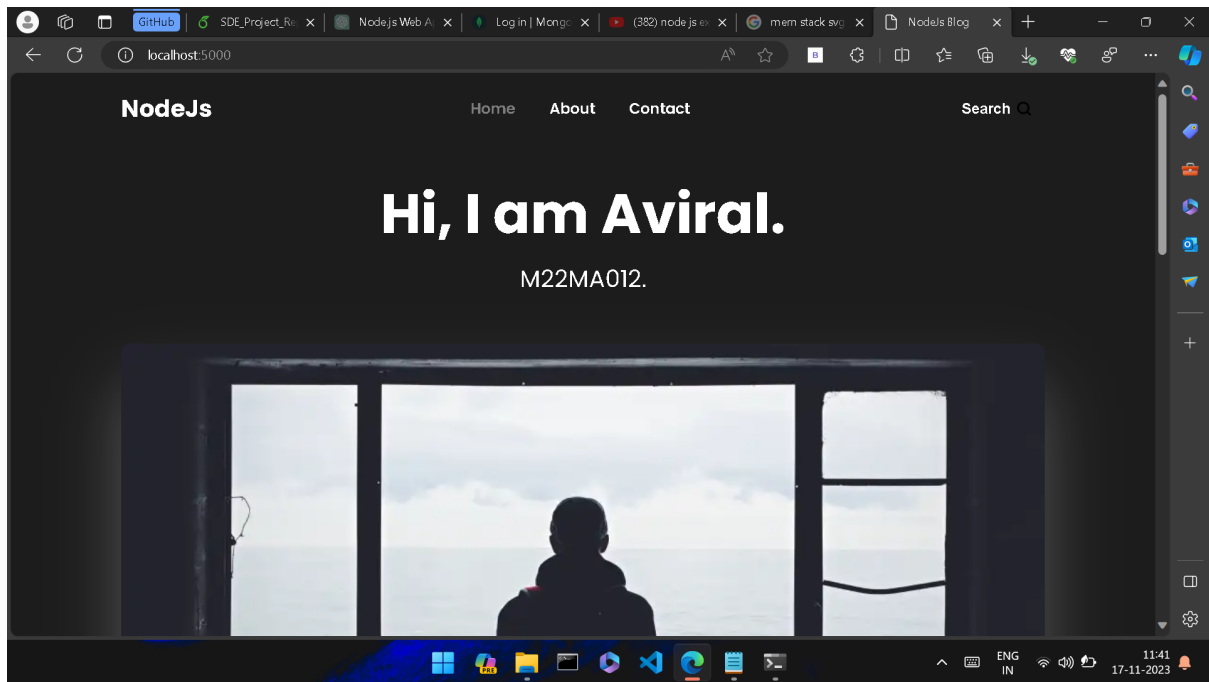


Figure 5.2: Homepage of the MERN blog app

Scrolling further down we can see the posts in the blog

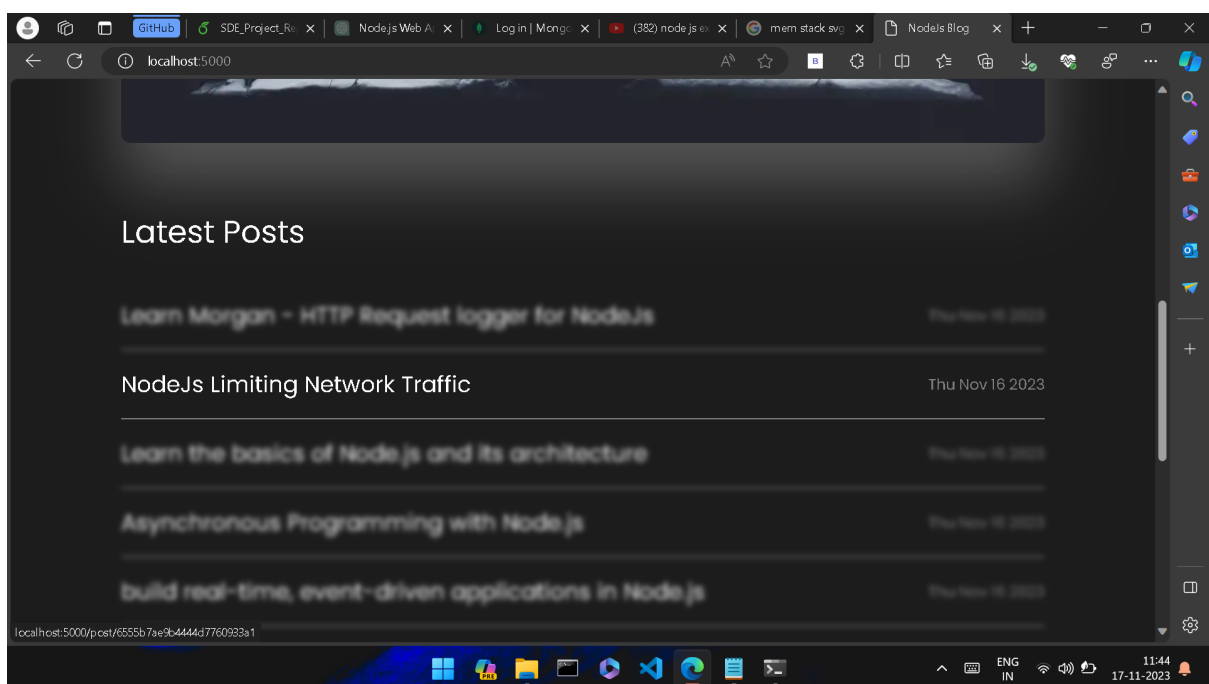


Figure 5.3: Homepage Posts

It can be verified that these posts are stored in the MongoDB (Atlas) database, and can be seen there:

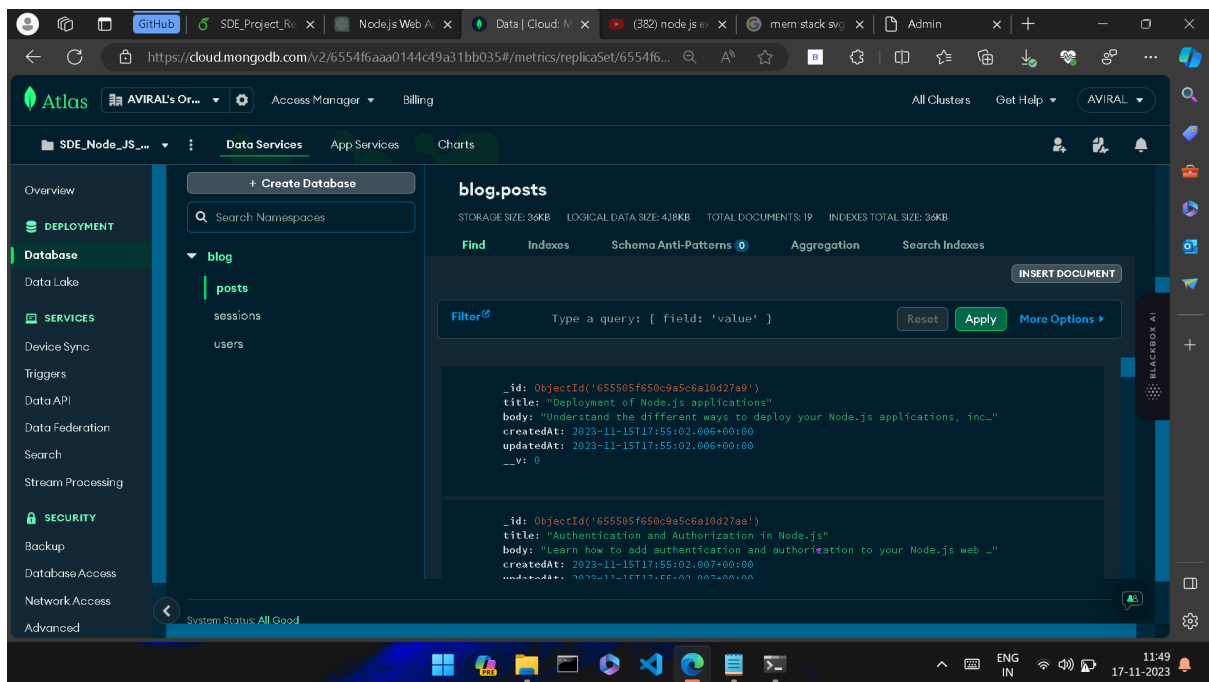


Figure 5.4: Posts in MongoDB (Atlas)

Upon visiting 'localhost:5000/admin' in the browser tab we can see the login and registration forms, if we register a user, we can find its username and password (Hashed by bcrypt) in MongoDB (Atlas)

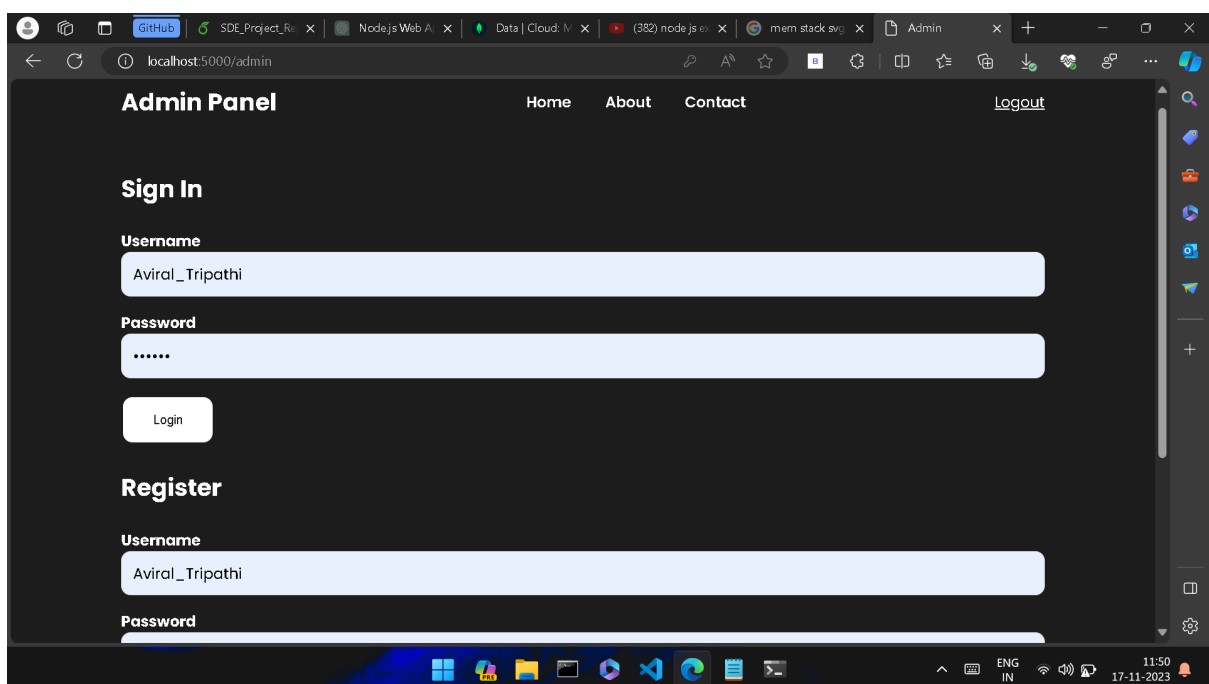


Figure 5.5: Login and Registration forms

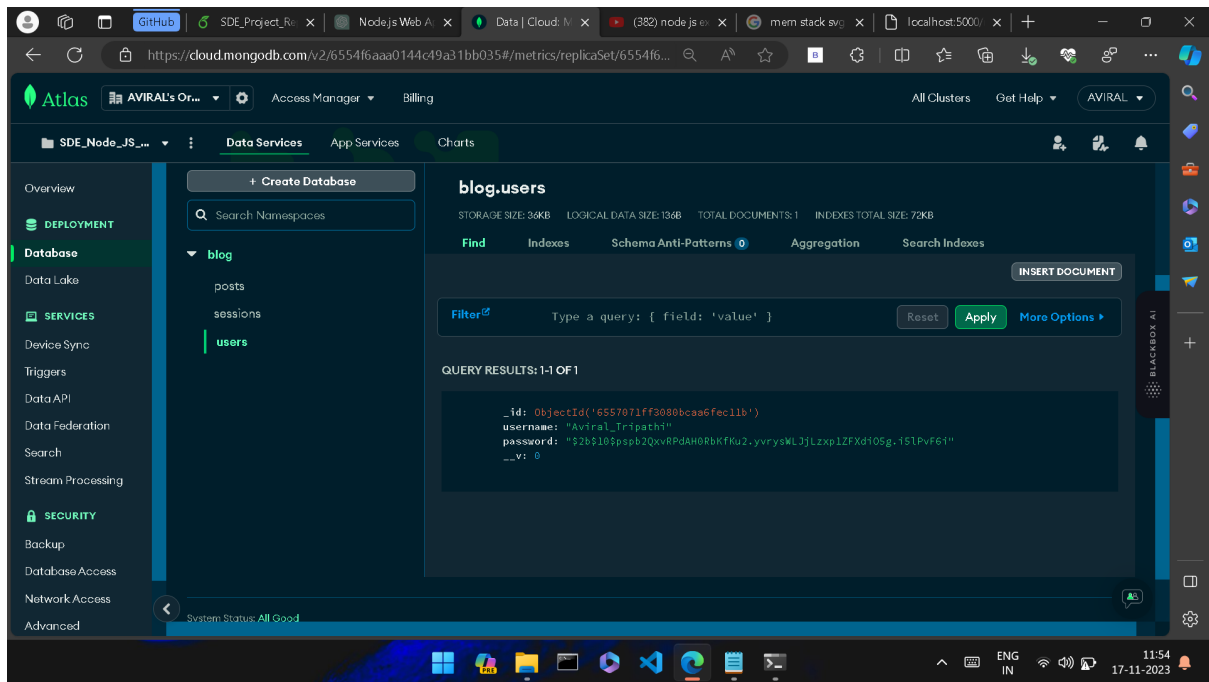


Figure 5.6: Username and Password (Hashed) in Atlas

Upon entering a registered username and password a user can login to 'create', 'edit', and 'delete' the existing posts in the blog.

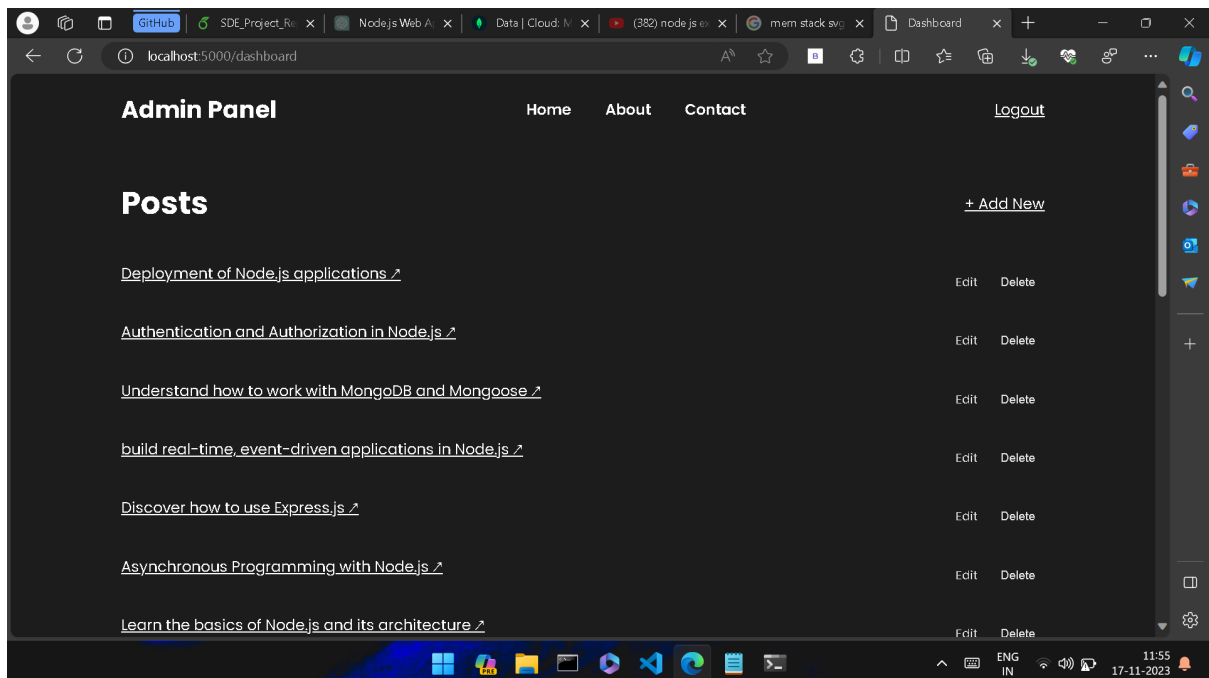


Figure 5.7: Loggedin User Dashboard

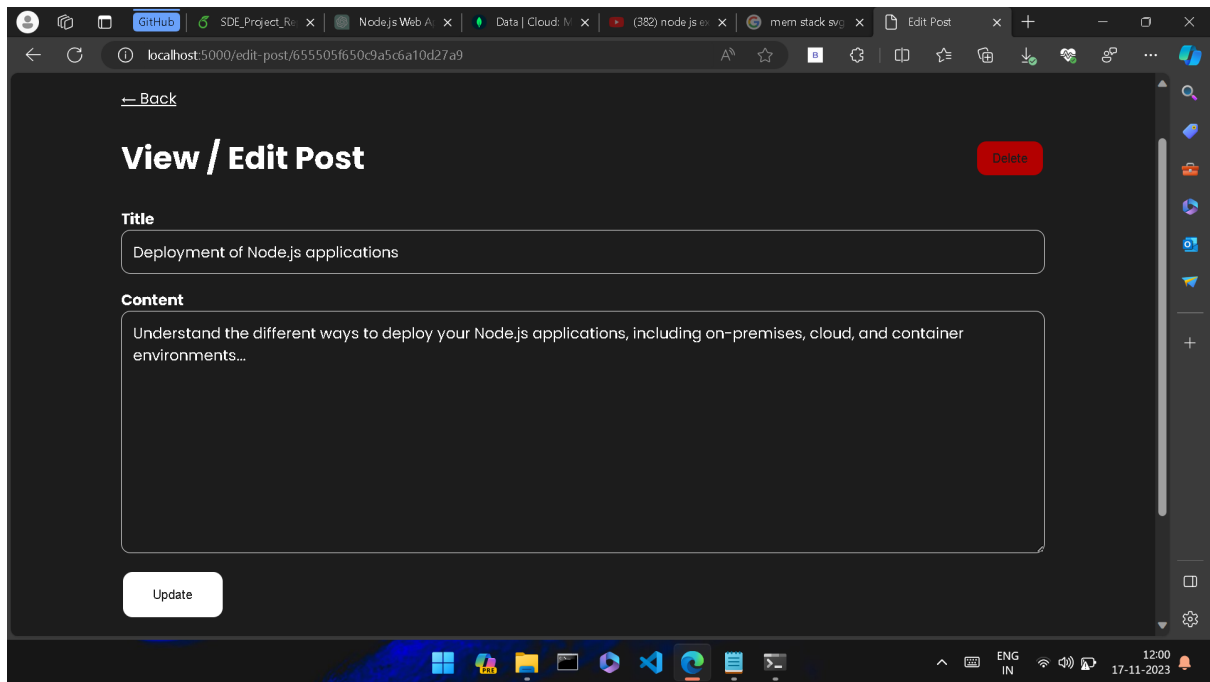


Figure 5.8: Editing or deleting a post

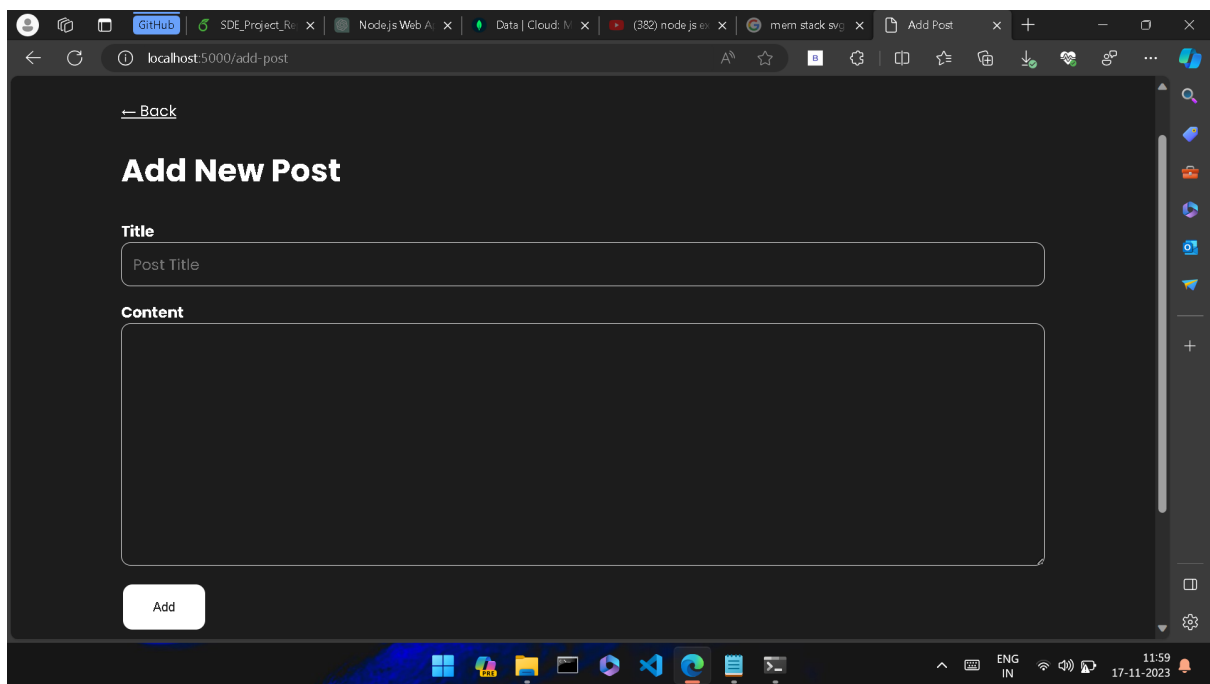


Figure 5.9: Creating a post

## 6 GitHub Repository Link



Figure 6.1: GitHub Link

### GitHub Repository Link

[https://github.com/AviralTripathim22ma012/NodeJS\\_MERN\\_Stack\\_Blog\\_SDE\\_Project/](https://github.com/AviralTripathim22ma012/NodeJS_MERN_Stack_Blog_SDE_Project/)

## 7 References

1. <https://youtu.be/-foo92lFIto?si=1XMIOPTFUxTwPkQ3>
2. <https://youtu.be/gv3FFn0dCIo?si=BC4bG-fz92hbpfd1>
3. <https://youtu.be/MruZEGPibC4?si=fxKXzWD3Dr-ge4Yj>
4. <https://youtu.be/FjuctFNN0FA?si=PRQdkW1fHbEYRup5>
5. [https://youtu.be/Cz-2QzkuCHo?si=m\\_25ddm7ogHtVHvy](https://youtu.be/Cz-2QzkuCHo?si=m_25ddm7ogHtVHvy)
6. [https://youtu.be/9Bnpl6bcev4?si=q5RistigoE2T\\_5Jo](https://youtu.be/9Bnpl6bcev4?si=q5RistigoE2T_5Jo)
7. <https://youtu.be/FmqBk2UTHzE?si=5tjSsXuZ762h06ZV>
8. <https://youtu.be/uCQirwe5UVg?si=sgsjaz5scrUxX3-b>
9. <https://youtu.be/xQVM33SAjLM?si=pK3NygH6vzBZhLTp>
10. <https://youtu.be/gR1Zlu4u58g?si=8LZwaFwg6PtB8Cjg>
11. [https://youtu.be/eopdY\\_rD2FE?si=9vKcccGyZq1jX-lG](https://youtu.be/eopdY_rD2FE?si=9vKcccGyZq1jX-lG)
12. [https://youtu.be/bJSj1a84I20?si=Bo1HYyrHLdekqSC\\_](https://youtu.be/bJSj1a84I20?si=Bo1HYyrHLdekqSC_)
13. <https://youtu.be/1IX6qThkXJ4?si=XxS7sarJdUCh-g4Z>
14. <https://youtu.be/zuye-dYSkS0?si=COq3QIzJiEtY3NFW>
15. <https://youtu.be/TXJsBBK4Lhw?si=gnGgamkgmAjnw4dV>
16. <https://youtu.be/R81g-2r6ynM?si=bgfT3gcz90hZAaXj>
17. <https://youtu.be/JehgY5V0h5c?si=96CkhNHch0Pa1T-j>
18. <https://www.youtube.com/watch?v=cQeCi2hT3is>
19. <https://youtu.be/bJSj1a84I20?si=AsipUC19iwX9NR4r>
20. [https://youtu.be/zQAdZYxbH14?si=VFRKGx3Exkt1d\\_k6](https://youtu.be/zQAdZYxbH14?si=VFRKGx3Exkt1d_k6)
21. <https://youtu.be/xKs2IZZya7c?si=v-cgNS30S3XS0Zql>
22. [https://www.youtube.com/watch?v=FVnA1d\\_TGw0](https://www.youtube.com/watch?v=FVnA1d_TGw0)
23. <https://github.com/RaddyTheBrand/25.NodeJs-Express-EJS-MongoDB--Blog>
24. <https://github.com/dejwid/mern-blog>
25. <https://nodejs.org/en/download>
26. <https://nodejs.org/en/learn/getting-started/how-to-install-nodejs>
27. <https://react.dev/>
28. <https://react.dev/learn>
29. <https://chat.openai.com/>
30. <https://www.mongodb.com/>
31. <https://www.mongodb.com/products/tools/compass>
32. <https://www.mongodb.com/try/download/compass>
33. <https://www.mongodb.com/try/download/compass>
34. <https://www.mongodb.com/try/download/atlascli>

35. <https://www.mongodb.com/try/download/atlas-kubernetes-operator>
36. <https://www.mongodb.com/try/download/mongosync>



## List of Figures

1.1	Node JS . . . . .	1
1.2	Node JS Event Loop . . . . .	2
2.1	MERN Stack . . . . .	3
3.1	Node JS download . . . . .	4
3.2	Verifying Node JS installation . . . . .	4
3.3	Create React App (CRA) . . . . .	5
3.4	React started . . . . .	5
3.5	react-router-dom Installed . . . . .	6
3.6	Express installed in 'api' directory . . . . .	7
3.7	Nodemon installed . . . . .	7
3.8	Cors installed in 'api' directory . . . . .	8
4.1	MongoDB . . . . .	10
4.2	MongoDB New project (SDE_Node_JS_Blog) created . . .	10
4.3	MongoDB Deployment using 'free' version . . . . .	11
4.4	connecting MongoDB to VS Code . . . . .	11
4.5	Copy the connection string and put it in .env file in VS Code	12
4.6	MongoDB installed on our computer . . . . .	12
4.7	.env file in VS Code . . . . .	13
4.8	MongoDB Collections . . . . .	13
4.9	Bycryptjs installation . . . . .	14
5.1	Starting the MERN blog app . . . . .	15
5.2	Homepage of the MERN blog app . . . . .	16
5.3	Homepage Posts . . . . .	16
5.4	Posts in MongoDB (Atlas) . . . . .	17
5.5	Login and Registration forms . . . . .	17
5.6	Username and Password (Hashed) in Atlas . . . . .	18
5.7	Loggedin User Dashboard . . . . .	18
5.8	Editing or deleting a post . . . . .	19
5.9	Creating a post . . . . .	19
6.1	GitHub Link . . . . .	20