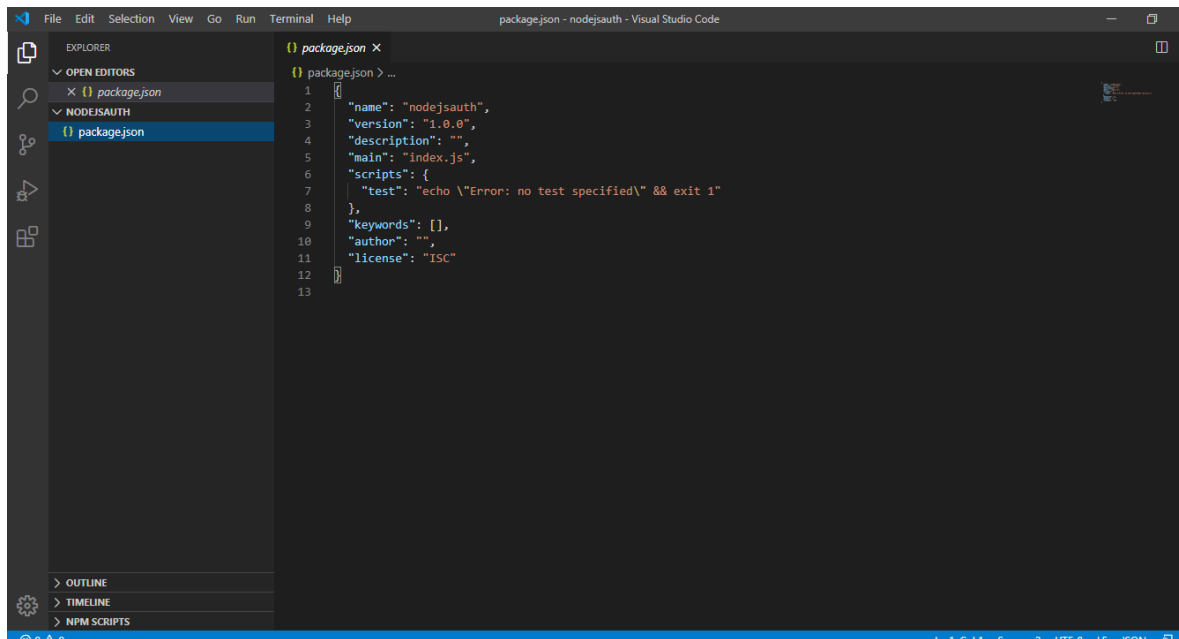


NODEJS AUTHENTICATION PROJECT:

Initially we need .Json file to start a project.

MODULE-1 AUTHENTICATION

So for Json file we use `npm init -y` where npm stands for(Node Package Manager) So the json as:

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'nodejsauth' with a 'package.json' file selected. The main editor area displays the content of 'package.json' with the following JSON structure:

```
1 {}
2   "name": "nodejsauth",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC"
12 }
13
```

Step 2:We install the resources required for the project Such as:

We use `npm i npm i express bcryptjs passport passport-local ejs express-ejs-layouts mongoose connect-flash express-session`

Express->it's our web application server framework

Bcryptjs->To encrypt password because we won't store password in plaintext

Passport->Authentication purpose

Ejs->for layouts

Mongoose->For Database

Connect-Flash->for flash messaging

Nodemon->it's very important to make our task easier because we don't to stop and start the server while making the changes

The screenshot shows the Visual Studio Code interface with the `package.json` file open in the editor. The file contains the following content:

```
{
  "name": "nodejsauth",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "connect-flash": "^0.1.1",
    "ejs": "^3.1.3",
    "express": "^4.17.1",
    "express-ejs-layouts": "^2.5.0",
    "express-session": "^1.17.1",
    "mongoose": "^5.9.16",
    "passport": "^0.4.1",
  }
}
```

The terminal output shows the results of running `npm install`:

```
+ passport@0.4.1
+ mongoose@5.9.16
+ bcryptjs@2.4.3
added 185 packages from 62 contributors and audited 185 packages in 25.964s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\Sai_Santhosh\Desktop\nodejsauth> npm i -D nodemon
```

3) We have assigned a port number and tested it

The screenshot shows the Visual Studio Code interface with the `app.js` file open in the editor. The file contains the following content:

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 5000;
app.listen(PORT, console.log('server started on port'));
```

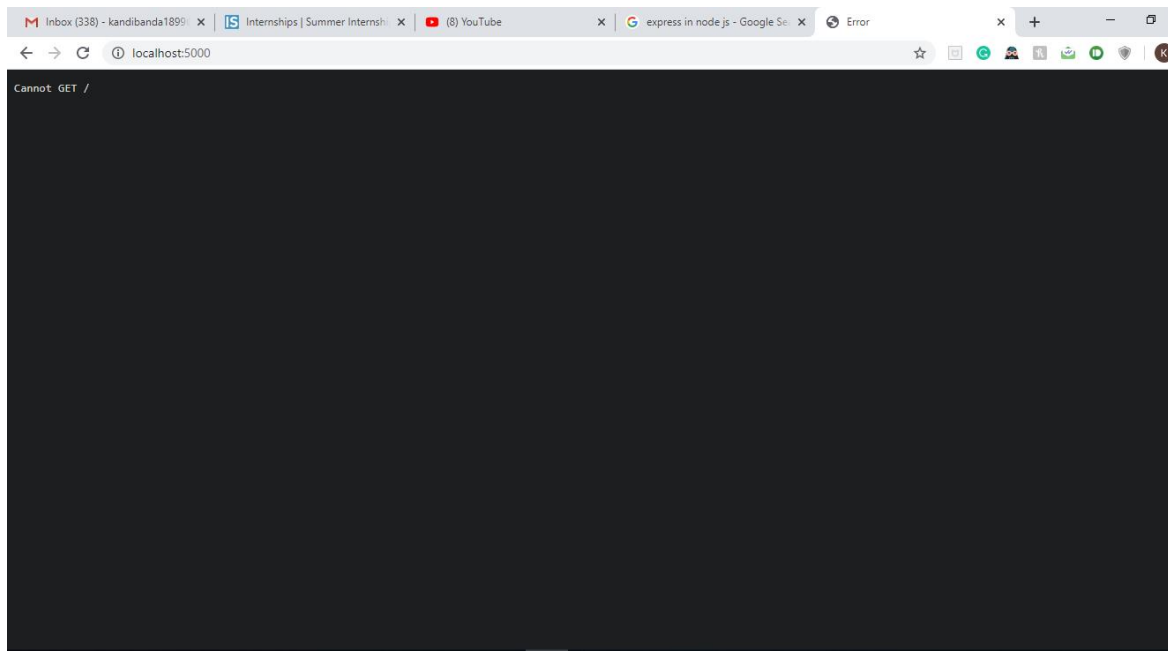
The terminal output shows the results of running `npm run dev`:

```
PS C:\Users\Sai_Santhosh\Desktop\nodejsauth> npm run dev

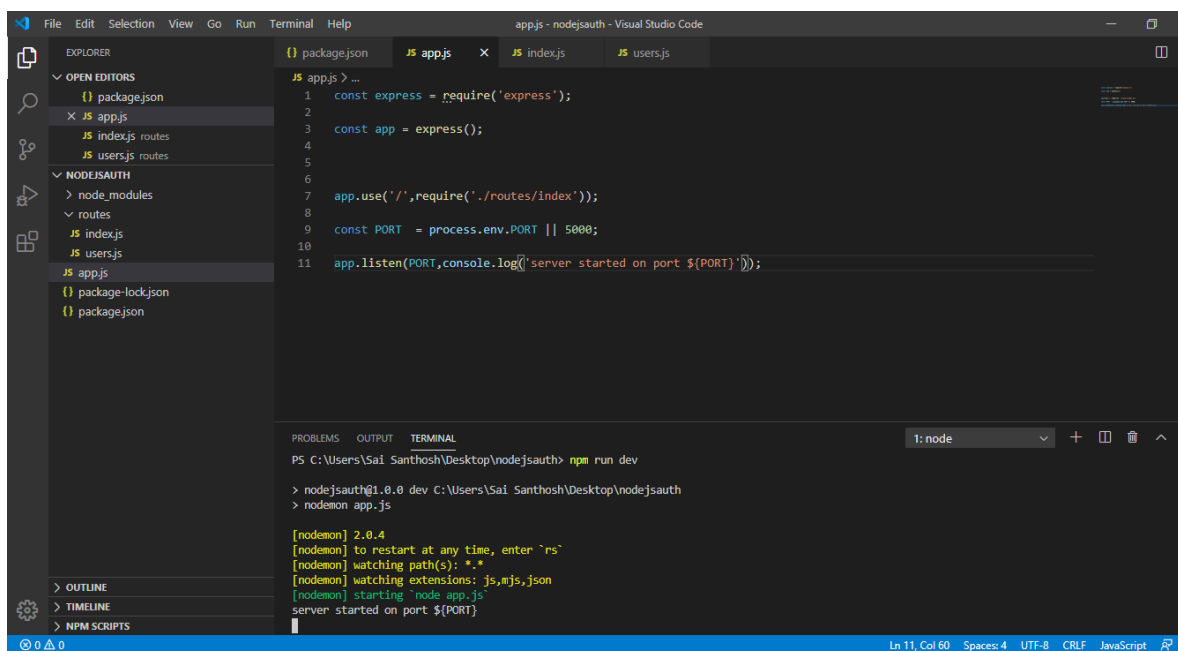
> nodejsauth@1.0.0 dev C:\Users\Sai_Santhosh\Desktop\nodejsauth
> nodemon app.js

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
server started on port
```

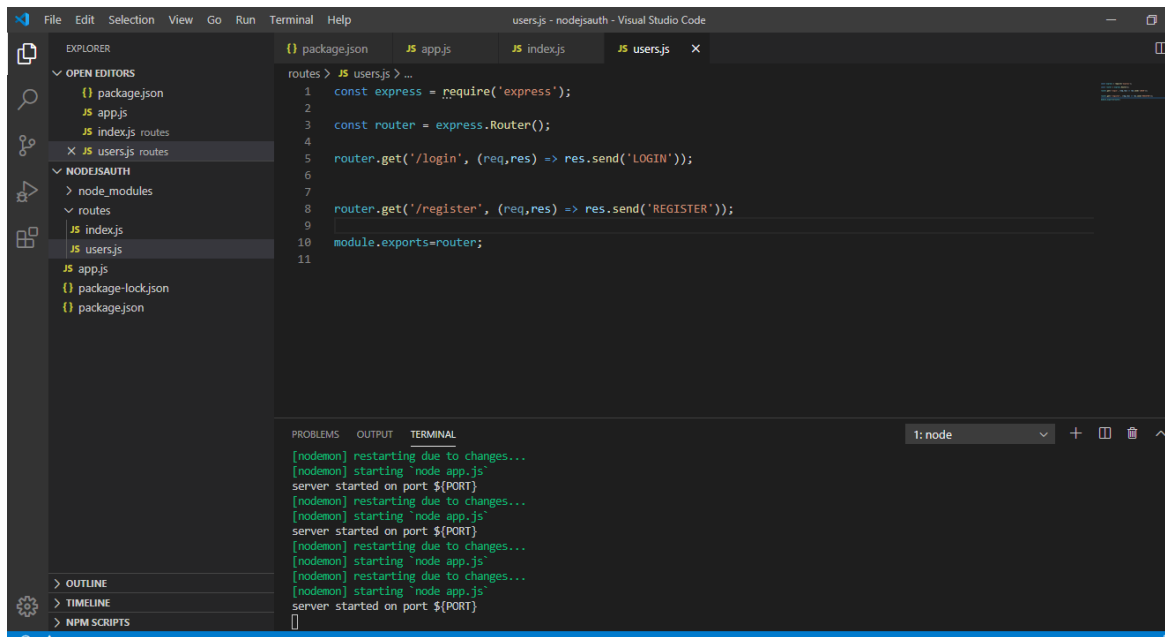
At present we don't have any routes so if we go localhost:5000 we will get it



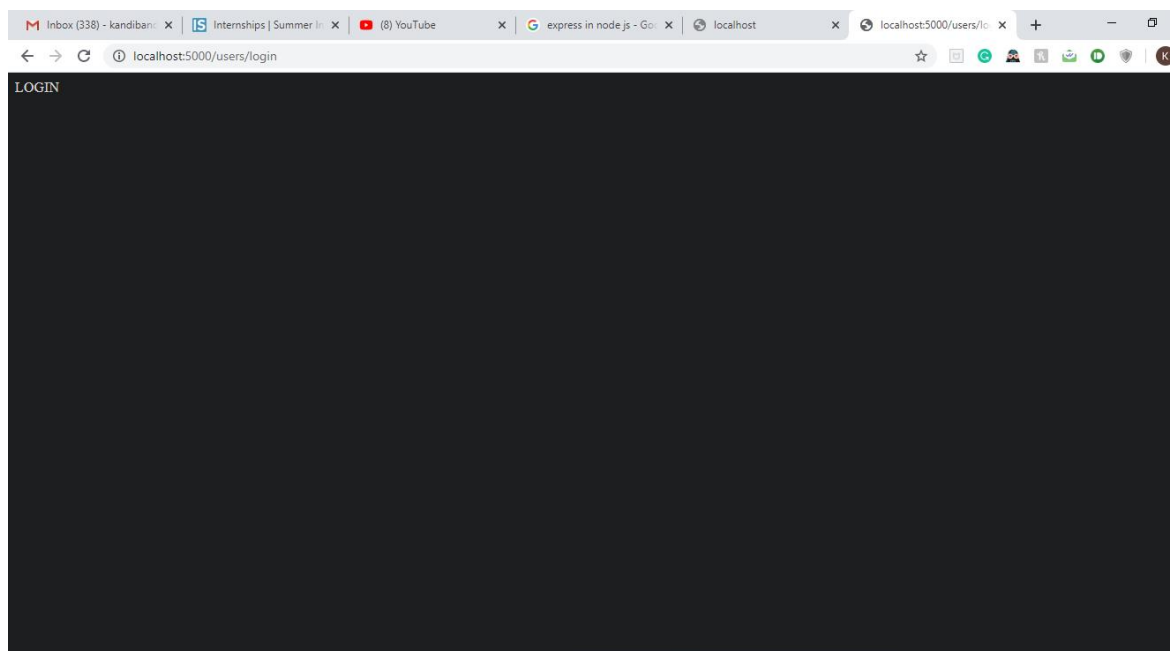
4)After that we are creating routes folder and then we will connect our index page to local host



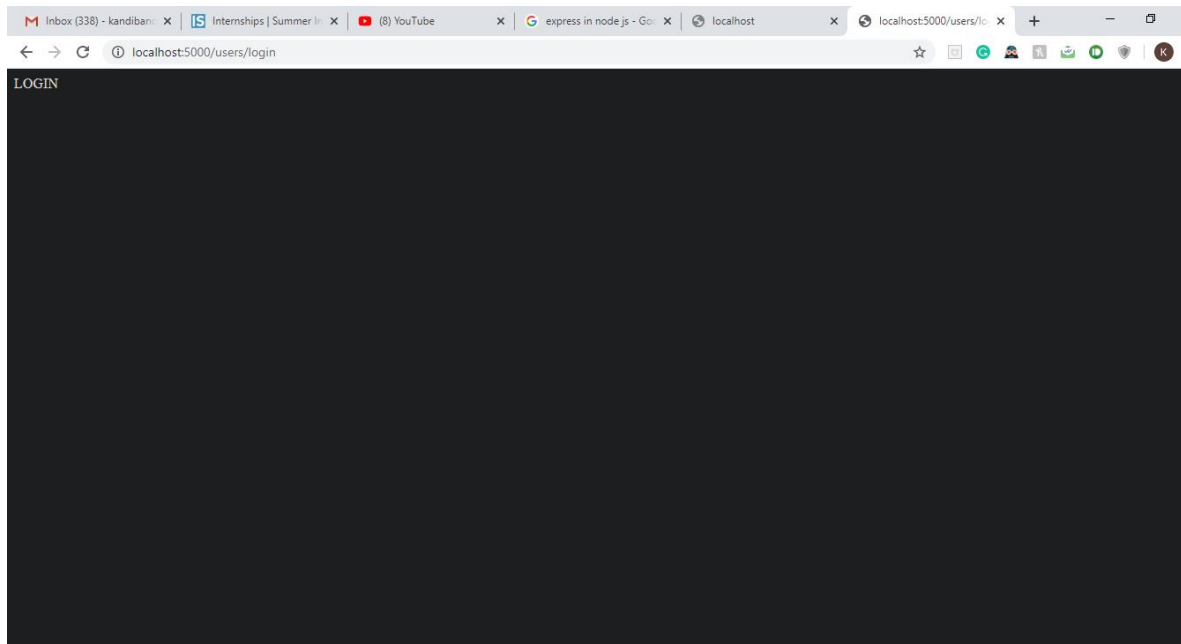
5)we will two pages such as login and register such as



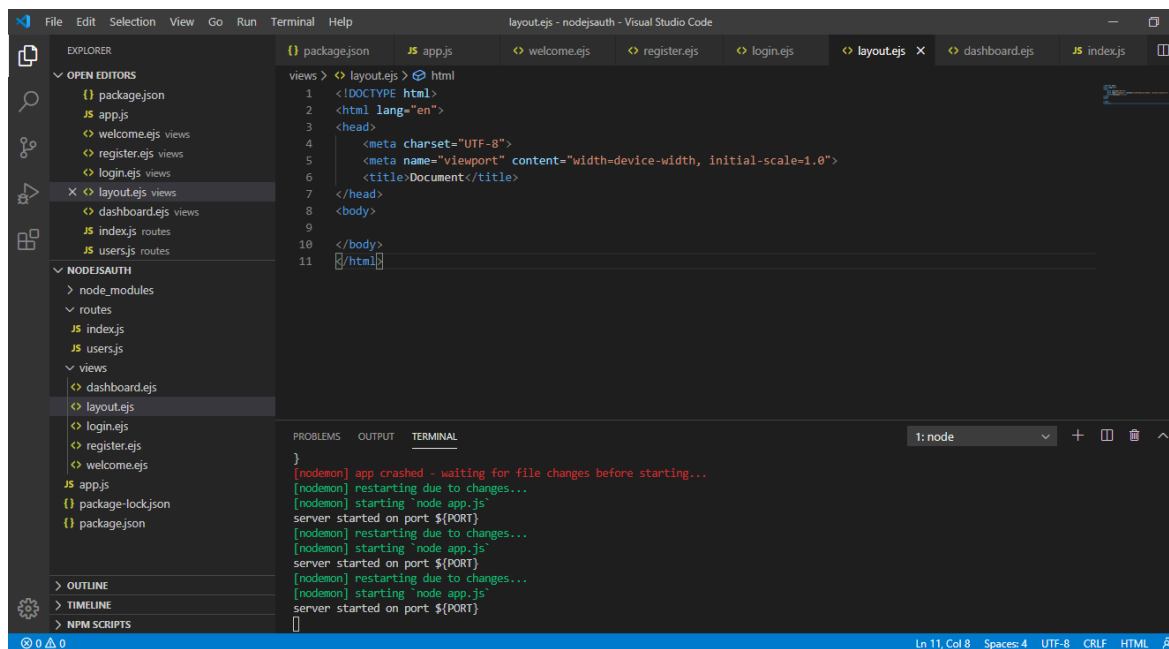
THE LOGIN PAGE SUCH AS:



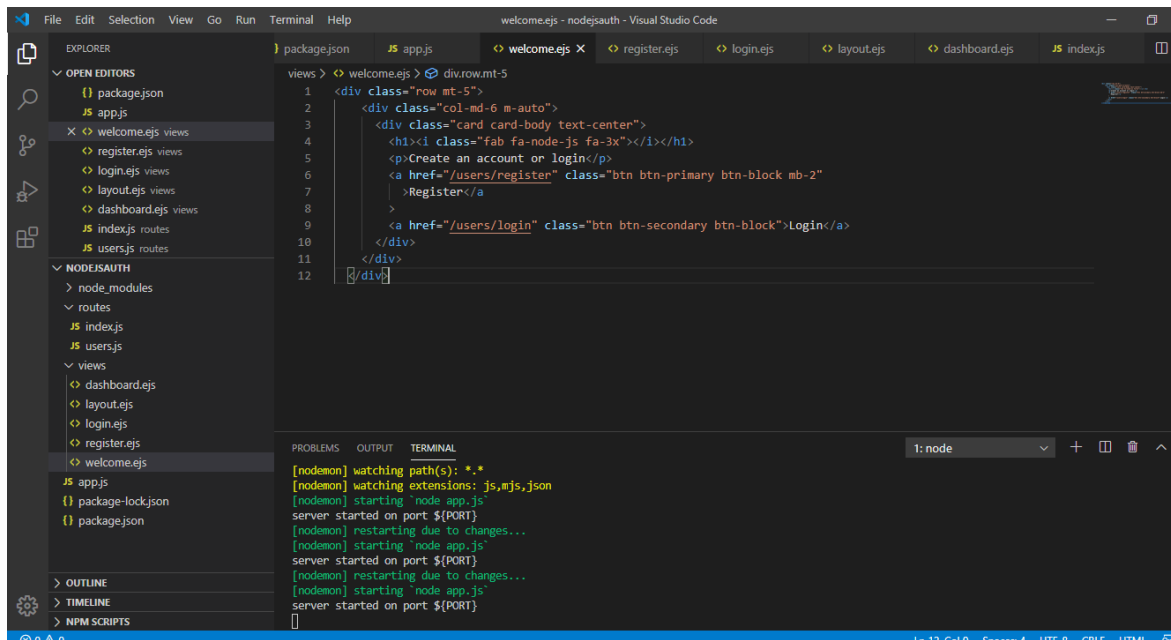
THE REGISTER PAGE



We also create view folder in which we have login, signup, register, dashboard etc.

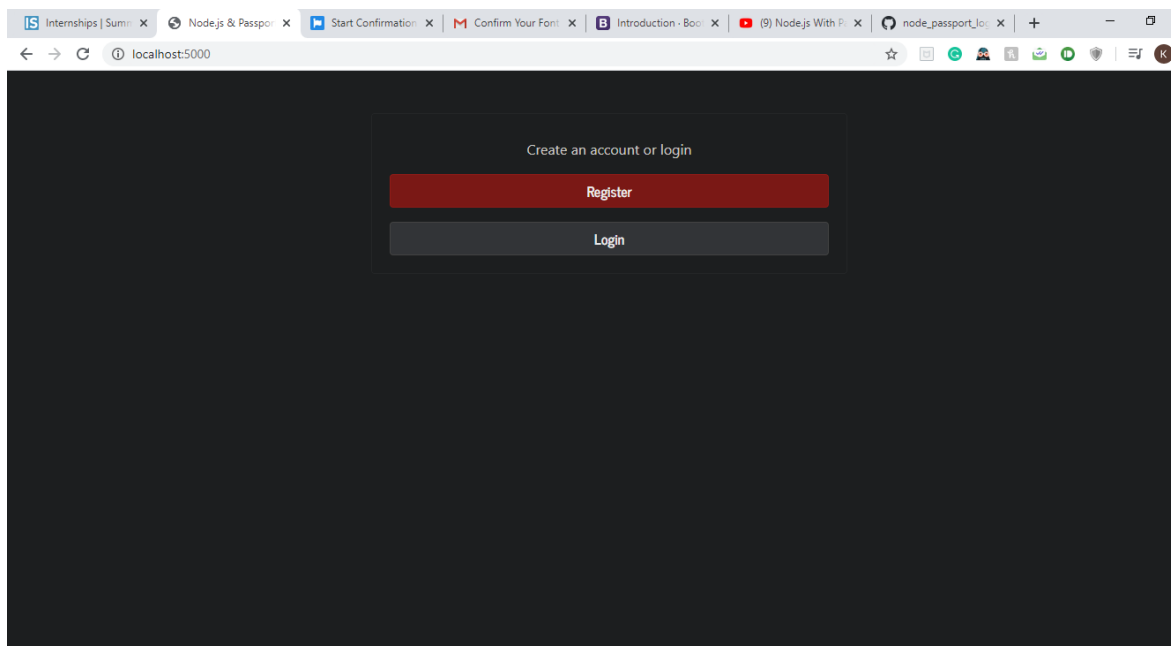


I used bootstrap for the welcome page such as:

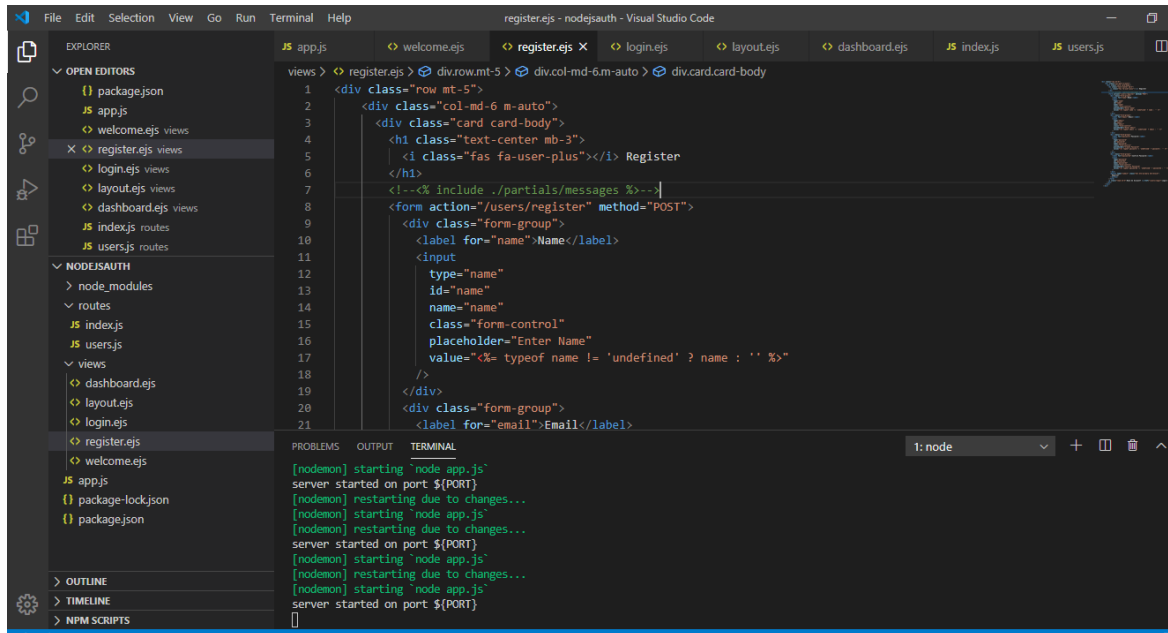


```
views > welcome.ejs > div.row.mt-5
1 <div class="row mt-5">
2 <div class="col-md-6 m-auto">
3 <div class="card card-body text-center">
4 <h1><i class="fab fa-node-js fa-3x"></i></h1>
5 <p>Create an account or login</p>
6 <a href="/users/register" class="btn btn-primary btn-block mb-2">
7   Register</a>
8 <a href="/users/login" class="btn btn-secondary btn-block">Login</a>
9 </div>
10 </div>
11 </div>
12 </div>
```

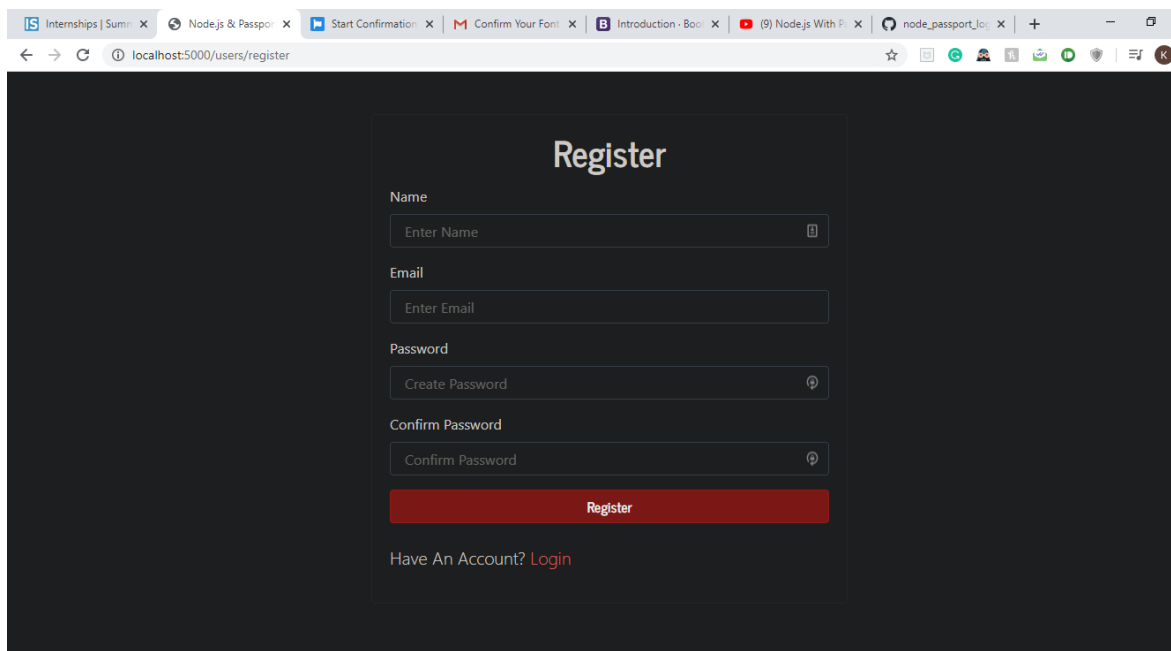
And the welcome page looks like:



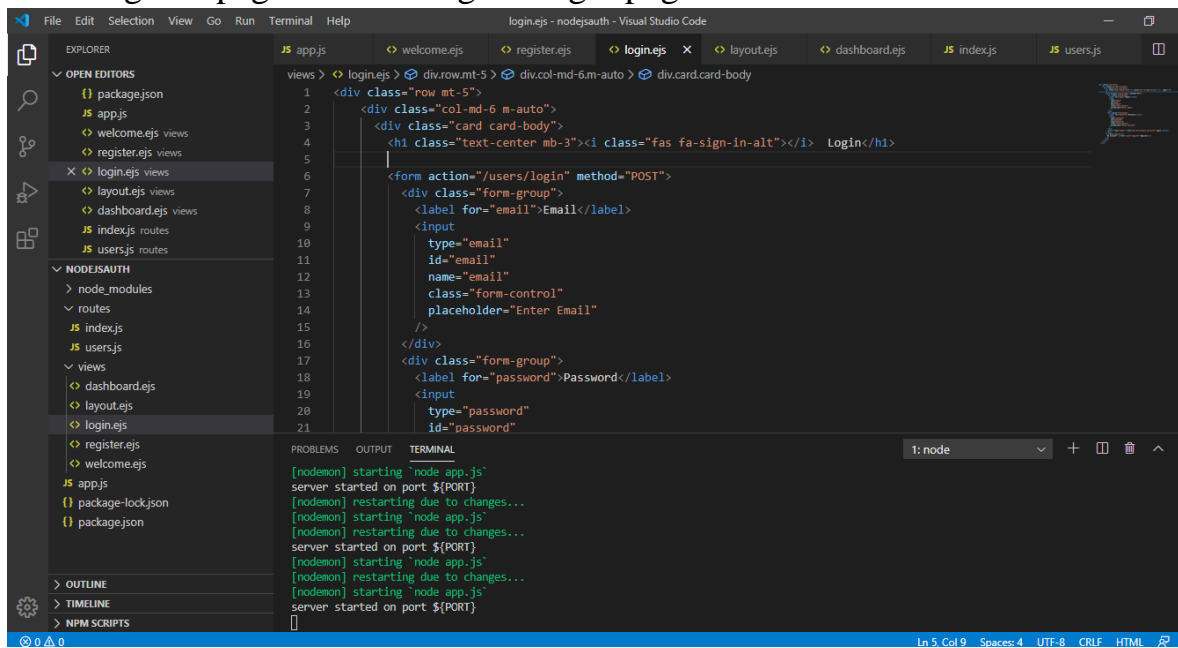
After this I have designed a register page



The register page looks like



After register page I have designed login page:



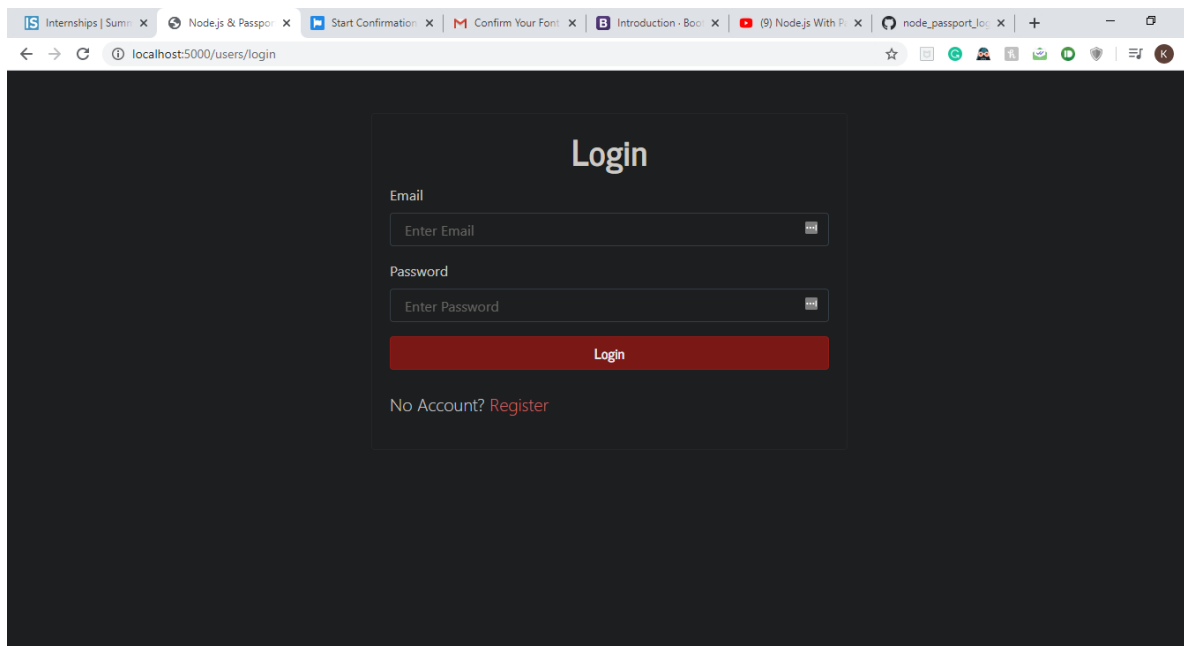
The screenshot shows the Visual Studio Code editor with the `login.ejs` file open. The file contains the following HTML code:

```
1 <div class="row mt-5">
2   <div class="col-md-6 m-auto">
3     <div class="card card-body">
4       <h1 class="text-center mb-3"><i class="fas fa-sign-in-alt"></i> Login</h1>
5
6       <form action="/users/login" method="POST">
7         <div class="form-group">
8           <label for="email">Email</label>
9           <input
10             type="email"
11             id="email"
12             name="email"
13             class="form-control"
14             placeholder="Enter Email"
15           />
16         </div>
17         <div class="form-group">
18           <label for="password">Password</label>
19           <input
20             type="password"
21             id="password"
```

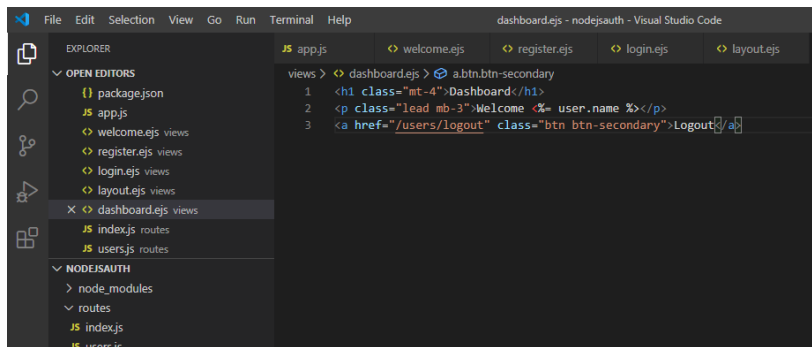
The terminal at the bottom shows the following output:

```
[nodemon] starting `node app.js`
server started on port ${PORT}
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
server started on port ${PORT}
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
server started on port ${PORT}
```

The login page looks like

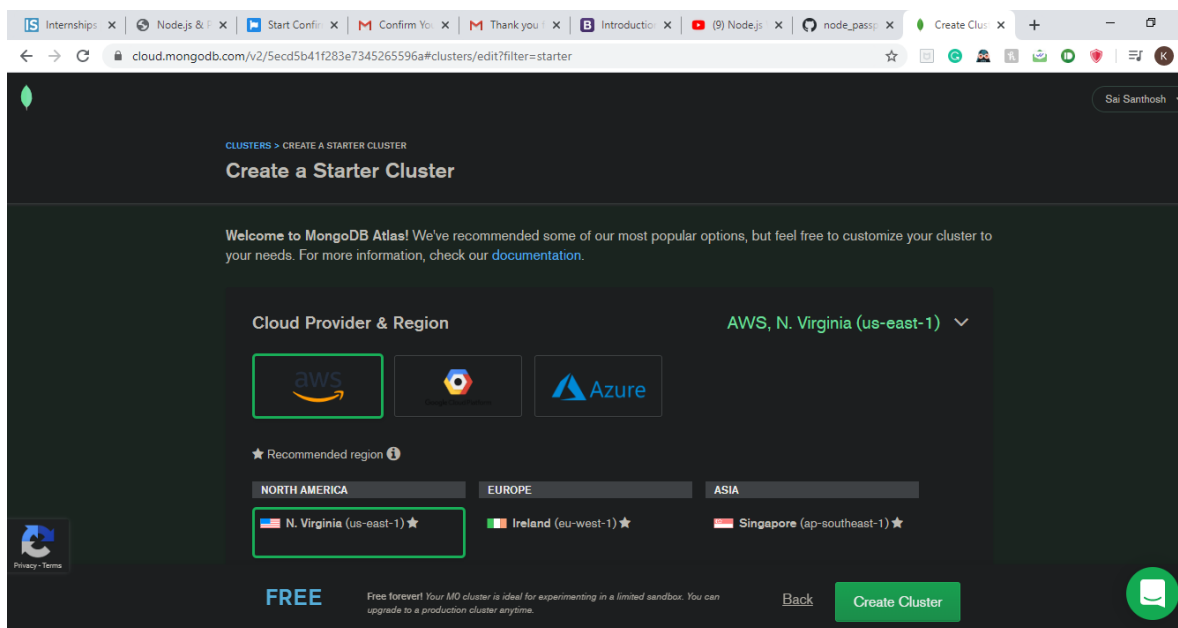


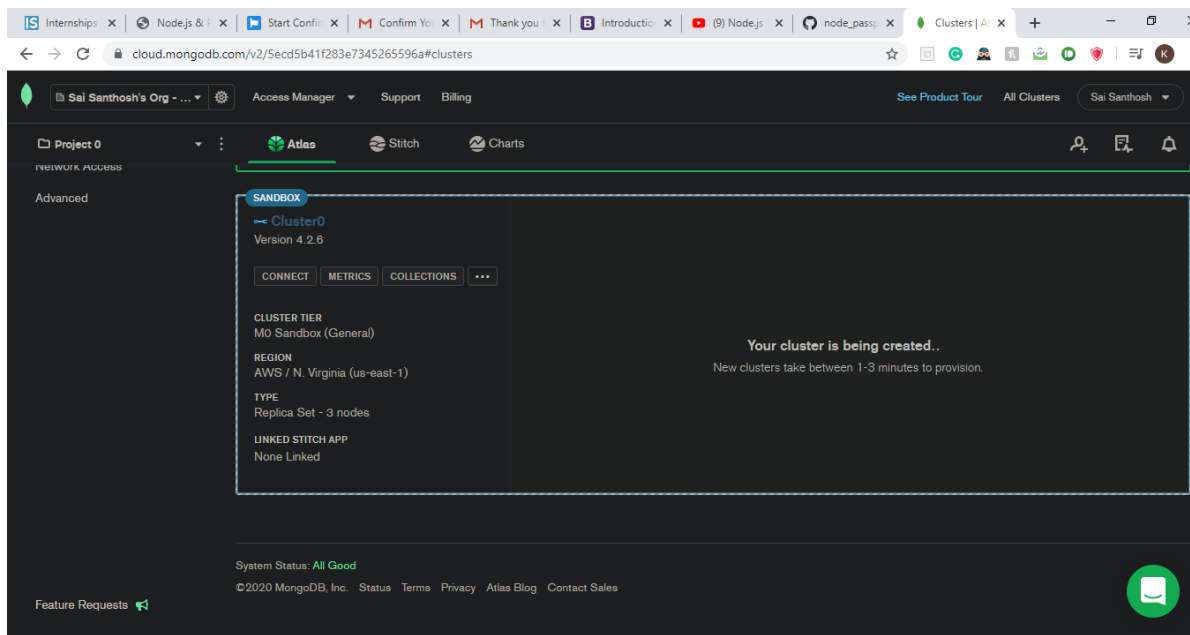
As I discussed earlier I have designed a dashboard page



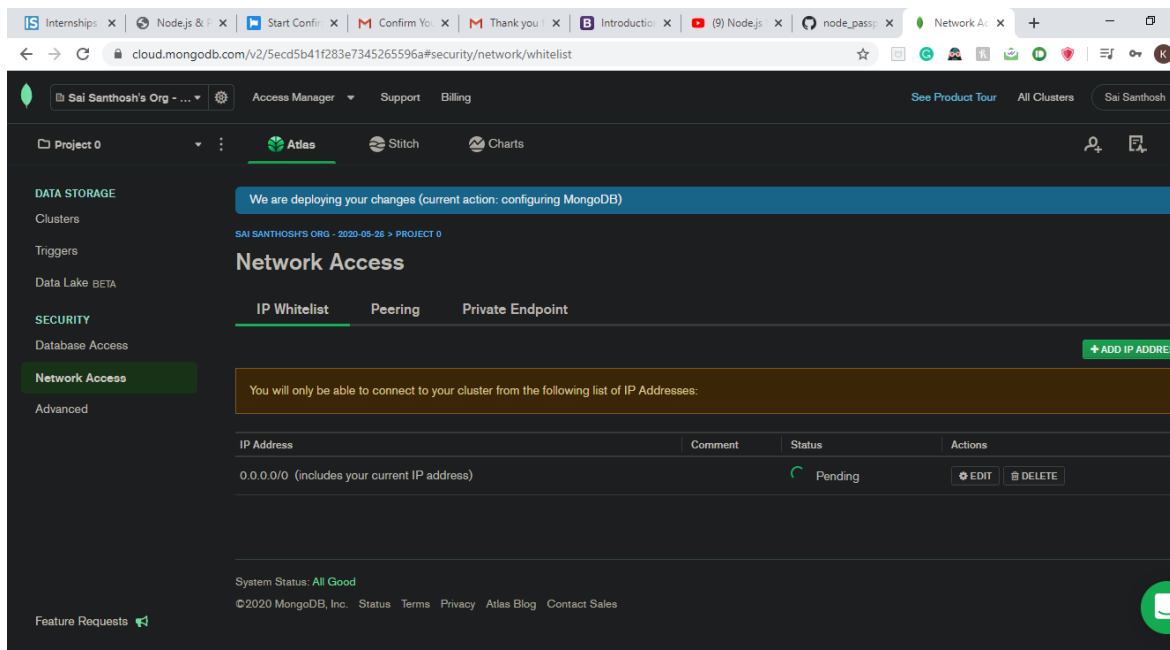
After designing the dashboard we use mongoose basically used to attach database with web pages

I use Mongoose here it goes after creating a cluster

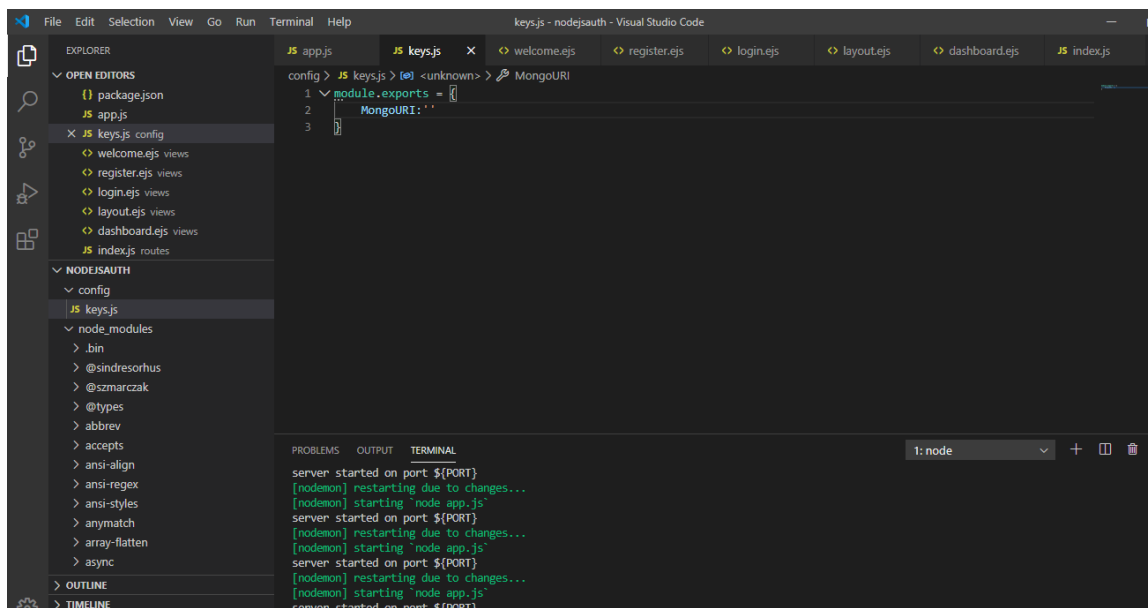
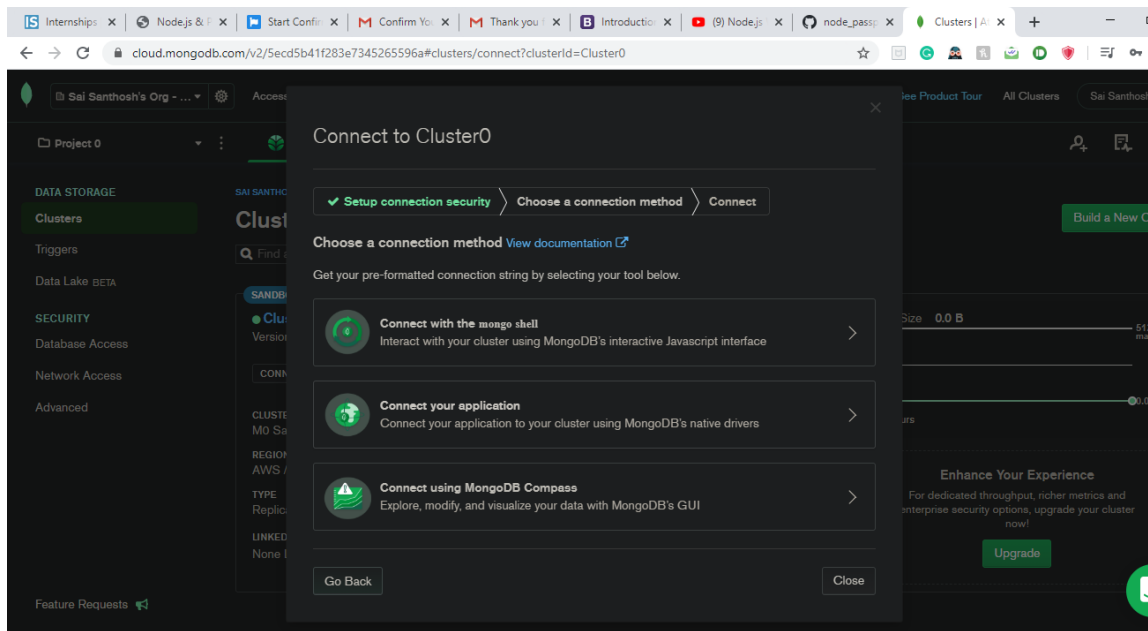




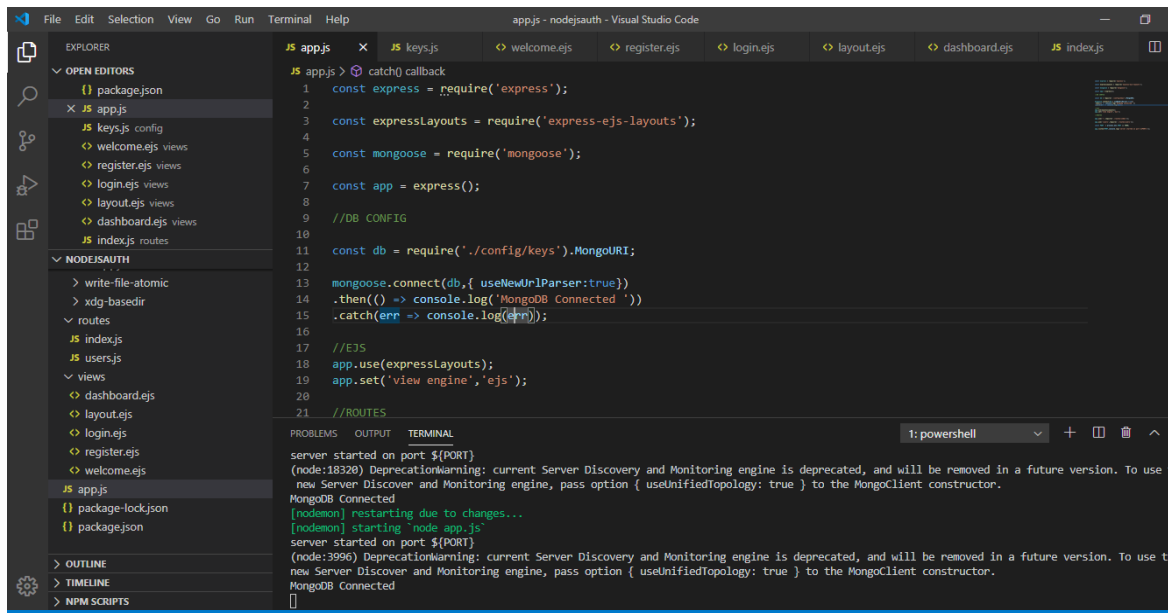
After registering a user in MongoDB and adding N/W access such as 0.0.0.0



Connecting MongoDB with Our application



I have connected the Database.

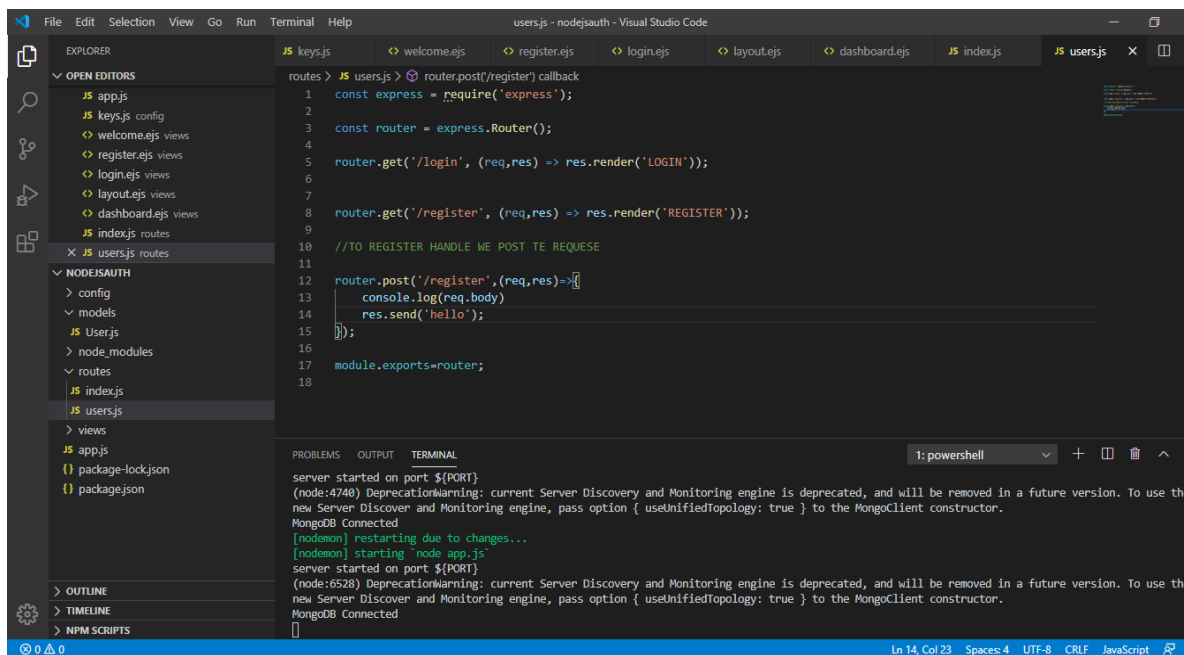


```
app.js
1 const express = require('express');
2
3 const expressLayouts = require('express-ejs-layouts');
4
5 const mongoose = require('mongoose');
6
7 const app = express();
8
9 //DB CONFIG
10
11 const db = require('./config/keys').MongoURI;
12
13 mongoose.connect(db,{ useNewUrlParser:true})
14 .then(() => console.log('MongoDB Connected '))
15 .catch(err => console.log(err));
16
17 //EJS
18 app.use(expressLayouts);
19 app.set('view engine','ejs');
20
21 //ROUTES
```

server started on port 3000
(node:18328) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB Connected
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
server started on port 3000
(node:3996) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB Connected

We have connected the database

We have given a token such that it sends hello message in return



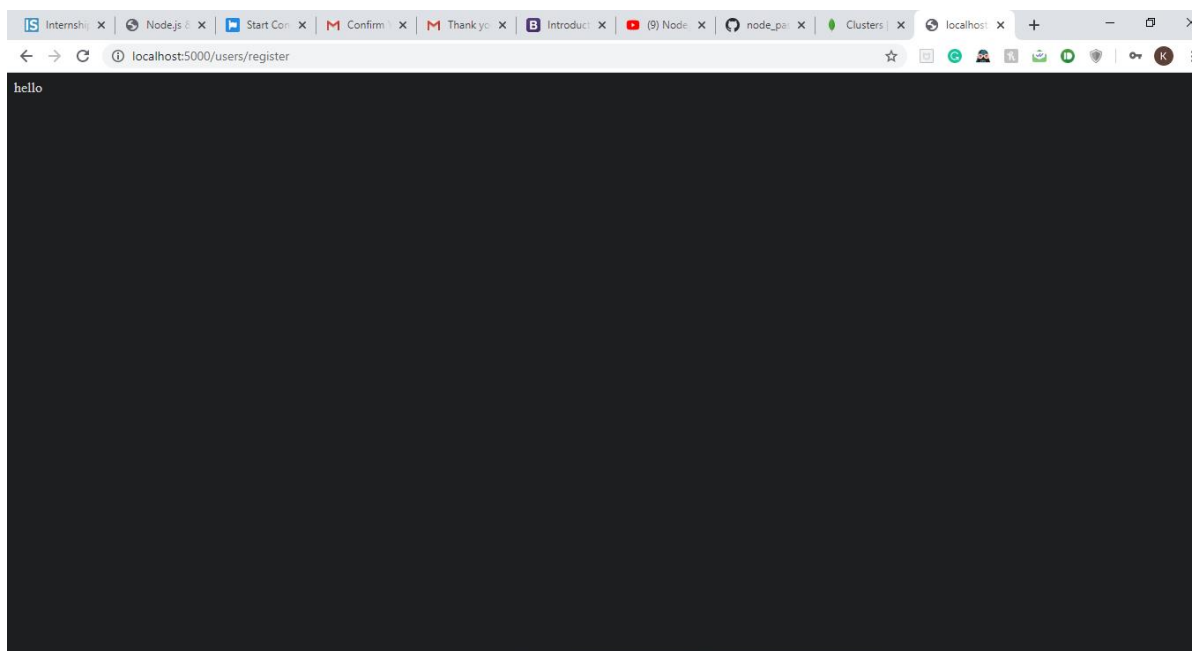
```
users.js
1 const express = require('express');
2
3 const router = express.Router();
4
5 router.get('/login', (req,res) => res.render('LOGIN'));
6
7 router.get('/register', (req,res) => res.render('REGISTER'));
8
9 //TO REGISTER HANDLE WE POST THE REQUEST
10
11
12 router.post('/register', (req,res) => {
13   console.log(req.body)
14   res.send('hello');
15 });
16
17 module.exports=router;
```

server started on port 3000
(node:4748) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB Connected
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
server started on port 3000
(node:6528) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB Connected

The screenshot shows a web browser window with the address bar displaying 'localhost:5000/users/register'. The browser has several tabs open, including 'Intership', 'Node.js', 'Start Con', 'Confirm', 'Thank you', 'Introduc', '(9) Node', 'node_pa', 'Clusters', and 'Node.js'. The registration form is centered on a dark background and contains the following fields and elements:

- Register** (Title)
- Name**: Input field containing 'k sai santhosh'.
- Email**: Input field containing 'kandibanda651@gmail.com'.
- Password**: Input field with masked characters '....' and a password icon.
- Confirm Password**: Input field with masked characters '....' and a password icon.
- Register** (Red button)
- Have An Account? [Login](#)** (Text with a link)

As soon as we register we got a reply message as hello



Till the time we haven't done the validations but I have given the same password as we see in the terminal we will get the register credentials as shown:

```
routes > JS users.js > router.post('/register') callback
1  const express = require('express');
2
3  const router = express.Router();
4
5  router.get('/login', (req,res) => res.render('LOGIN'));
6
7
8  router.get('/register', (req,res) => res.render('REGISTER'));
9
10 //TO REGISTER HANDLE WE POST TE REQUESE
11
12 router.post('/register',(req,res)=>{
13   console.log(req.body)
14   res.send('hello');
15 });
16
17 module.exports=router;
18
```

```
server started on port ${PORT}
(node:6528) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB Connected
[Object: null prototype] {
  name: 'k sai santhosh',
  email: 'kandibanda651@gmail.com',
  password: 'sasa',
  password2: 'sasa'
}
```

After this I have given all the validations that are required for the registering a user.

```
routes > JS users.js > router.post('/register') callback
11
12 router.post('/register',(req,res)=>{
13   const {name,email,password,password2} = req.body;
14   let errors=[];
15   //CHECKING THE REQUIRED FIELDS
16
17   if(!name || !email || !password || !password2){
18     errors.push({msg: 'please fill in all details!'});
19   }
20
21   //GIVING VALIDATIONS SUCH AS IN THIS CASE CONFIRM PASSWORD IS SAME AS PASSWORD
22
23   if(password != password2) {
24     errors.push({msg: 'passwords do not match!'});
25   }
26
27   //IN THIS CASE WE ARE CHECKING WHEATHER THE PASSWORD IS GREATER THAN 6 CHARACTERS OR NOT
28
29
30
31
```

```
[nodemon] starting 'node app.js'
server started on port ${PORT}
(node:19852) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
server started on port ${PORT}
(node:18308) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB Connected
```

I have designed in such a way if a user makes mistake while entering the form then user will again get the registration form.

```
22 //GIVING VALIDATIONS SUCH AS IN THIS CASE CONFIRM PASSWORD IS SAME AS PASSWORD
23
24 if(password != password2) {
25   errors.push({msg: 'passwords do not match'});
26 }
27
28 //IN THIS CASE WE ARE CHECKING WHEATHER THE PASSWORD IS GREATER THAN 6 CHARACTERS OR NOT
29
30 if(password.length < 6) {
31   errors.push({msg: 'password should be atleast 6 characters'});
32 }
33
34 if(errors.length > 0) {
35   res.render('register', {
36     errors,
37     name,
38     email,
39     password,
40     password2
41   });
42 }
```

terminal

```
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
server started on port ${PORT}
Error: querySrv ECONNREFUSED _mongodb._tcp.cluster0-0rk8o.mongodb.net
at QueryRedwrap.onresolve [as oncomplete] (dns.js:202:19) {
  errno: 'ECONNREFUSED',
  code: 'ECONNREFUSED',
  syscall: 'querySrv',
  hostname: '_mongodb._tcp.cluster0-0rk8o.mongodb.net'
}
```

Now we will check for all the changes we have done initially,

In the previous case th form is submitted although we have password < 6 characters.

Now the for is not submitted because it doesn't match our criteria.

Register

Name
k sai santhosh

Email
kandibanda651@gmail.com

Password

Confirm Password

Register

Have An Account? [Login](#)

