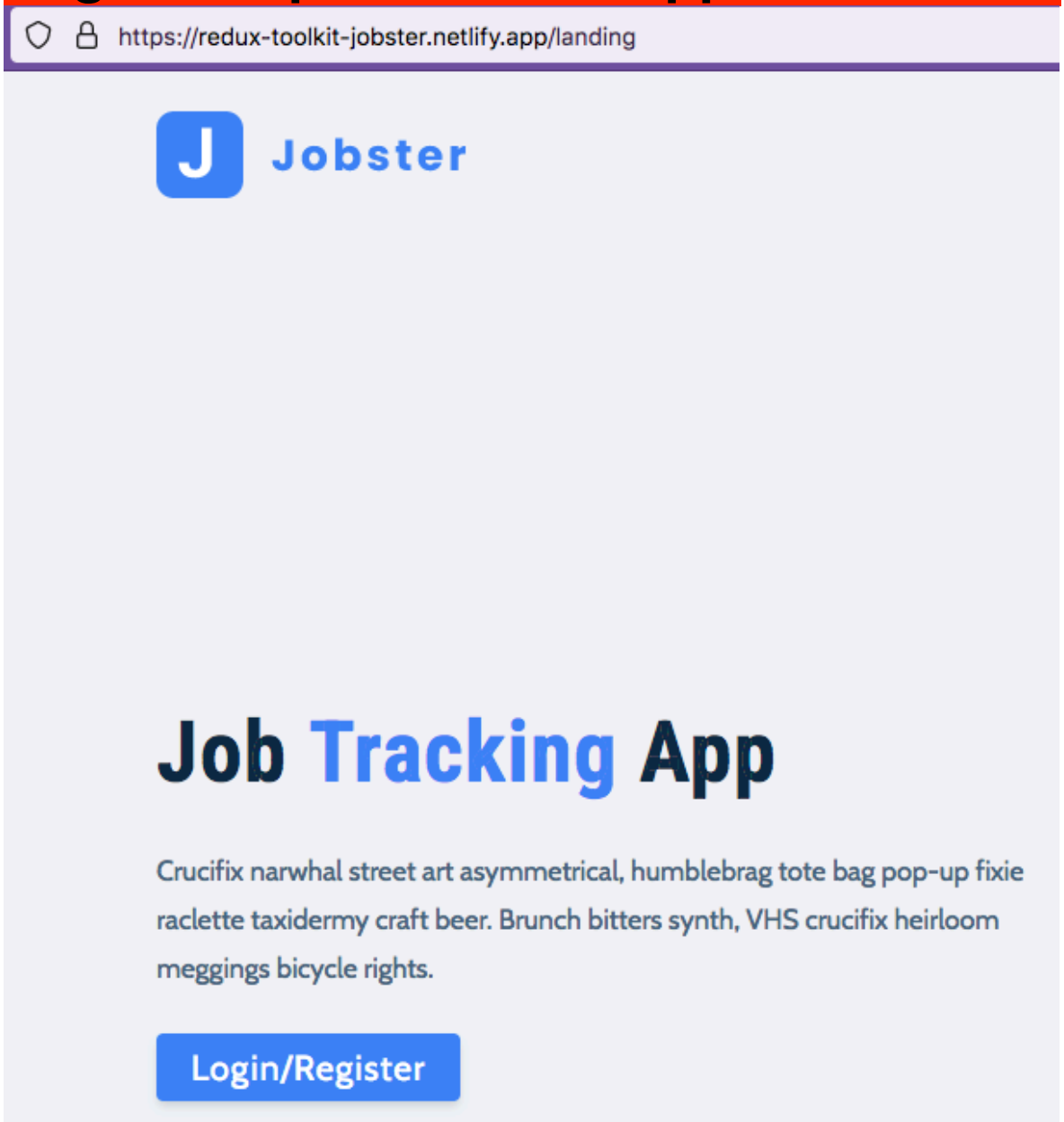
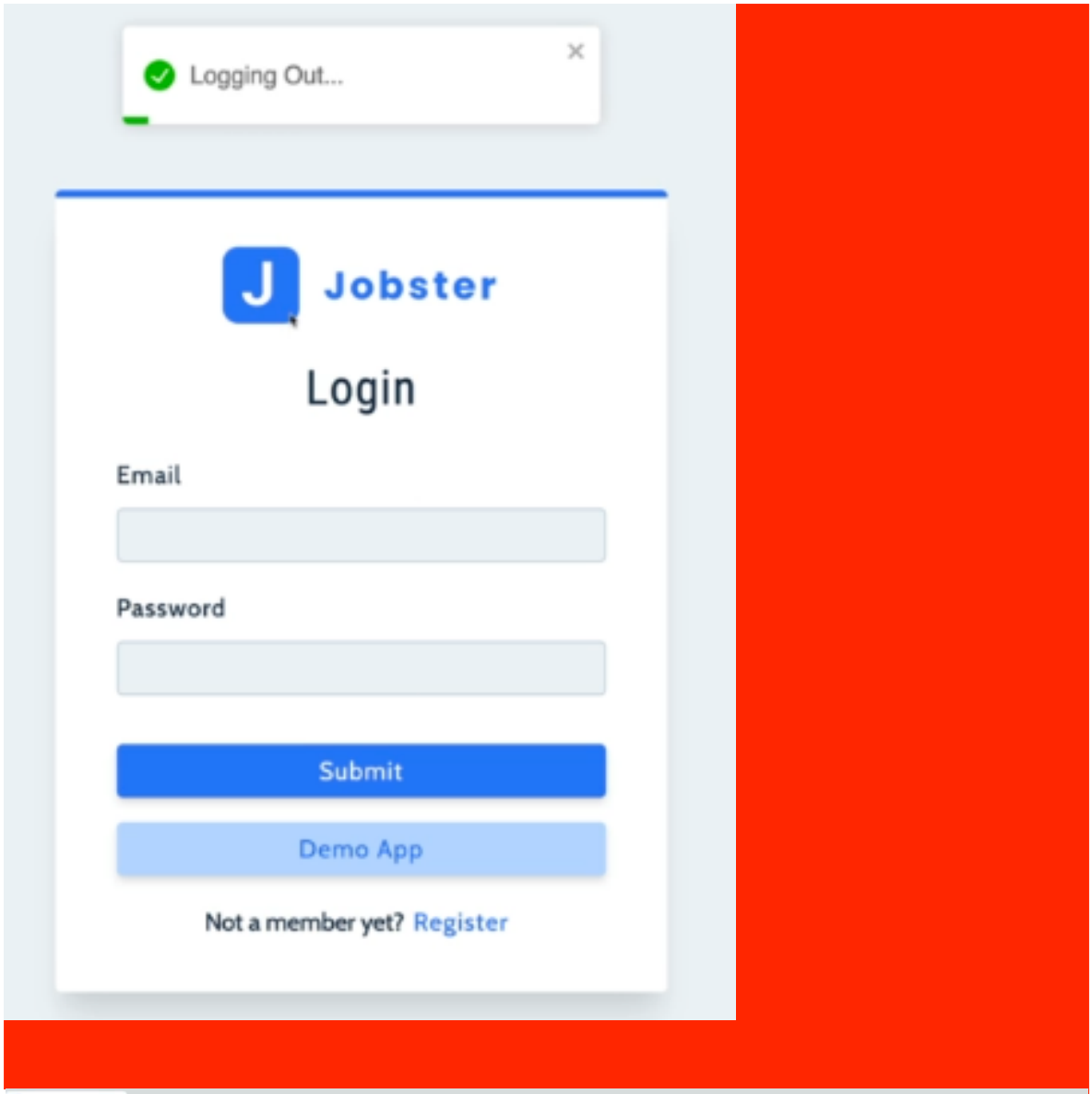
1 - Intro

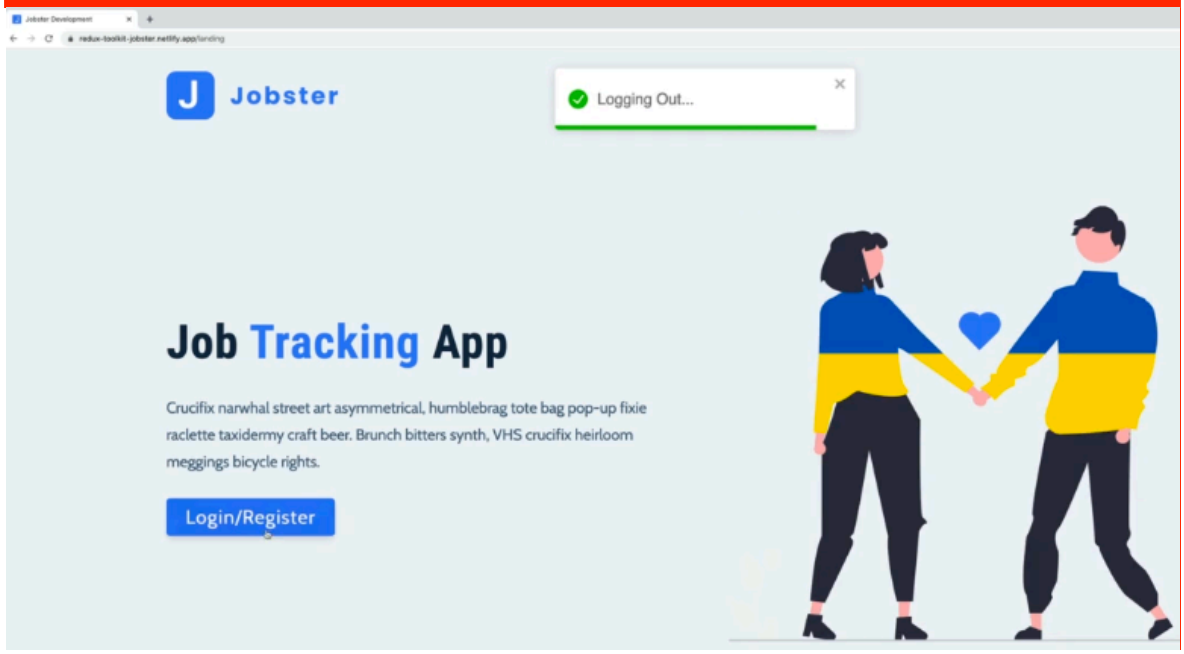
========

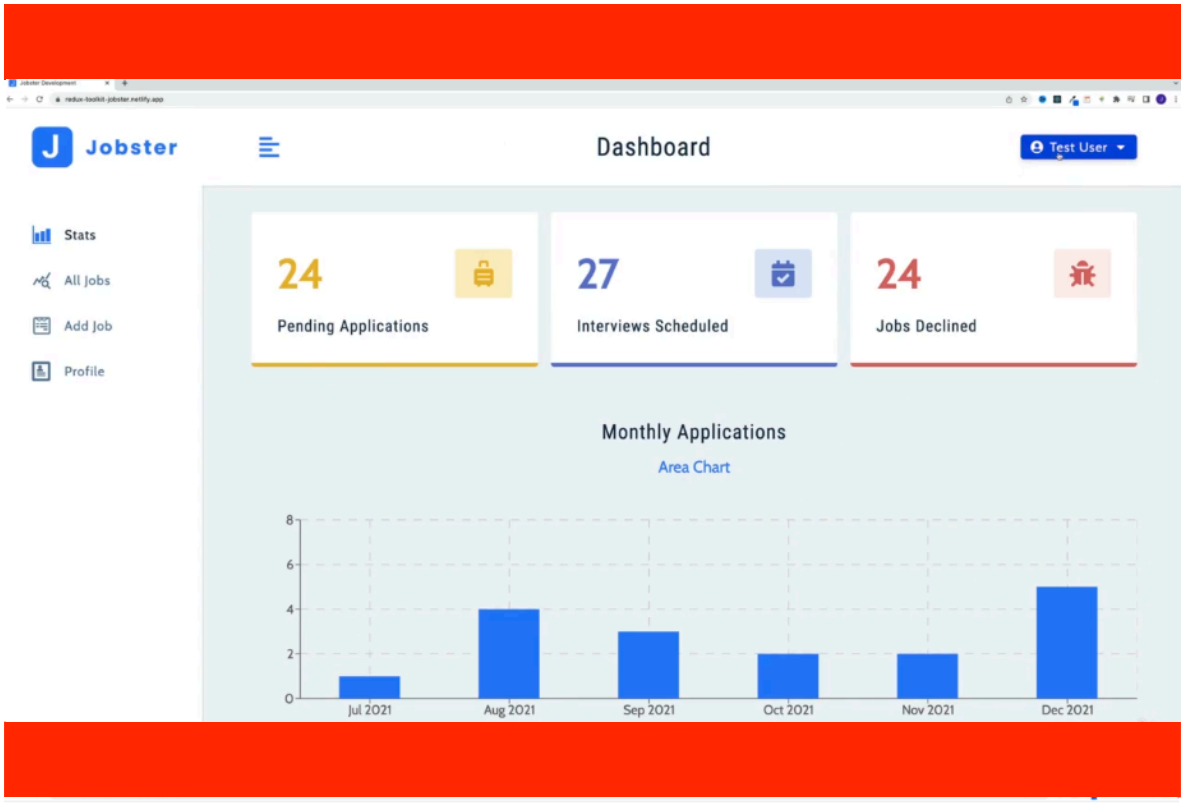
https://redux-toolkit-jobster.netlify.app/landing

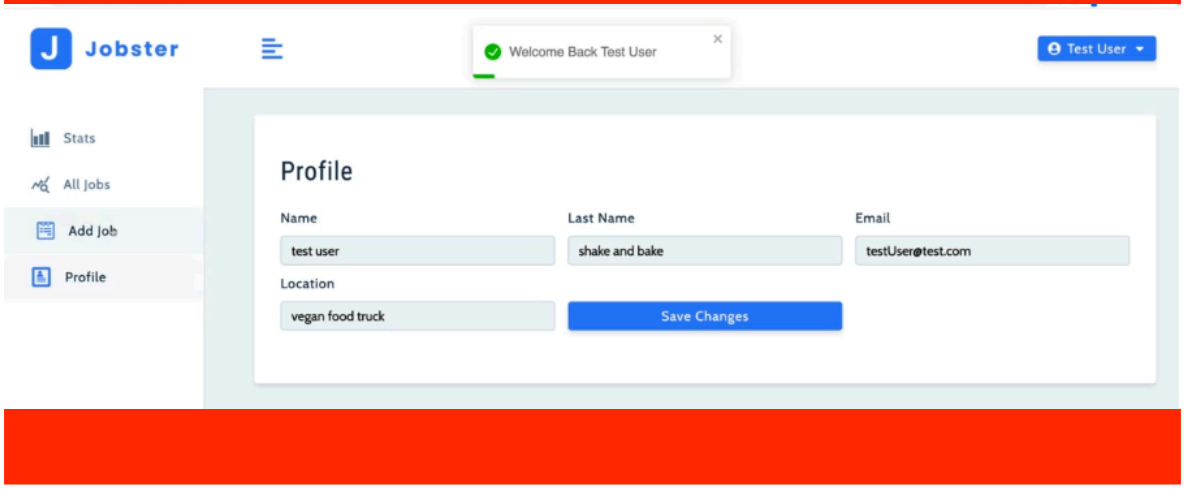
Login and press Demo App Button

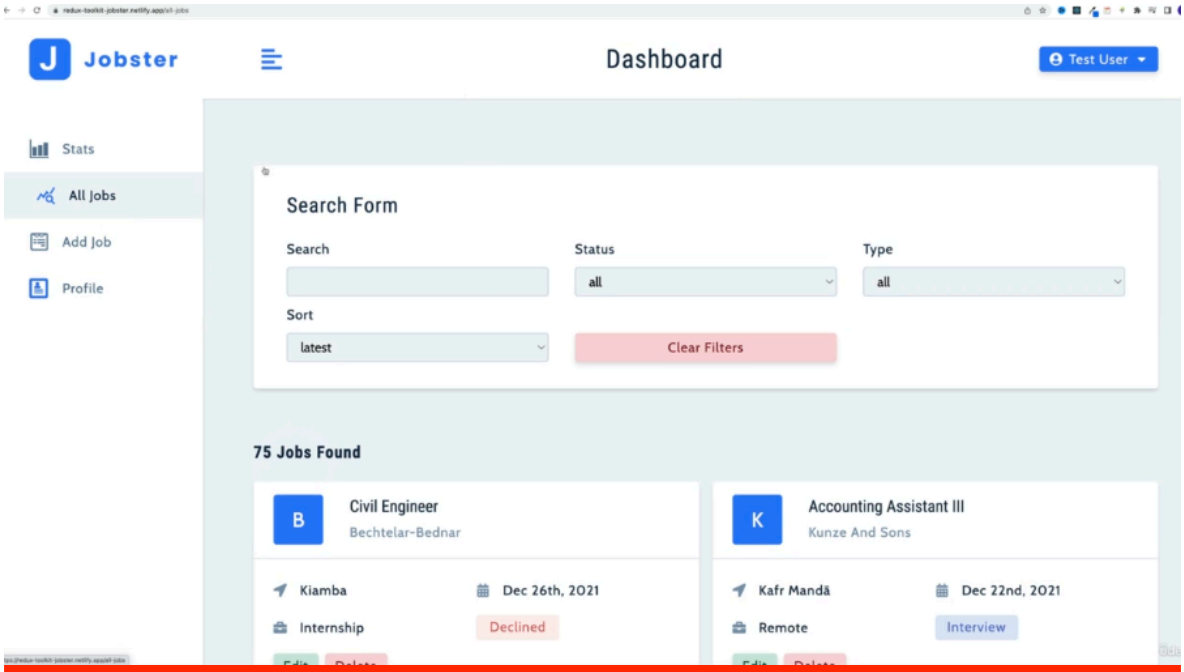



Logging Out...
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register


Jobster
Logging Out...
Job Tracking App
Crucifix narwhal street art asymmetrical, humblebrag tote bag pop-up fixie raclette taxidermy craft beer. Brunch bitters synth, VHS crucifix heirloom meggings bicycle rights.
Login/Register


Jobster
Dashboard
Test User
Stats
All Jobs
Add Job
Profile
24
Pending Applications
27
Interviews Scheduled
24
Jobs Declined
Monthly Applications
Area Chart
8
6
4
2
0
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021


Jobster
Welcome Back Test User
Test User
Stats
All Jobs
Add Job
Profile
Profile
Name
test user
Last Name
shake and bake
Email
testUser@test.com
Location
vegan food truck
Save Changes



2 - Setup

==========

```
# Jobster API

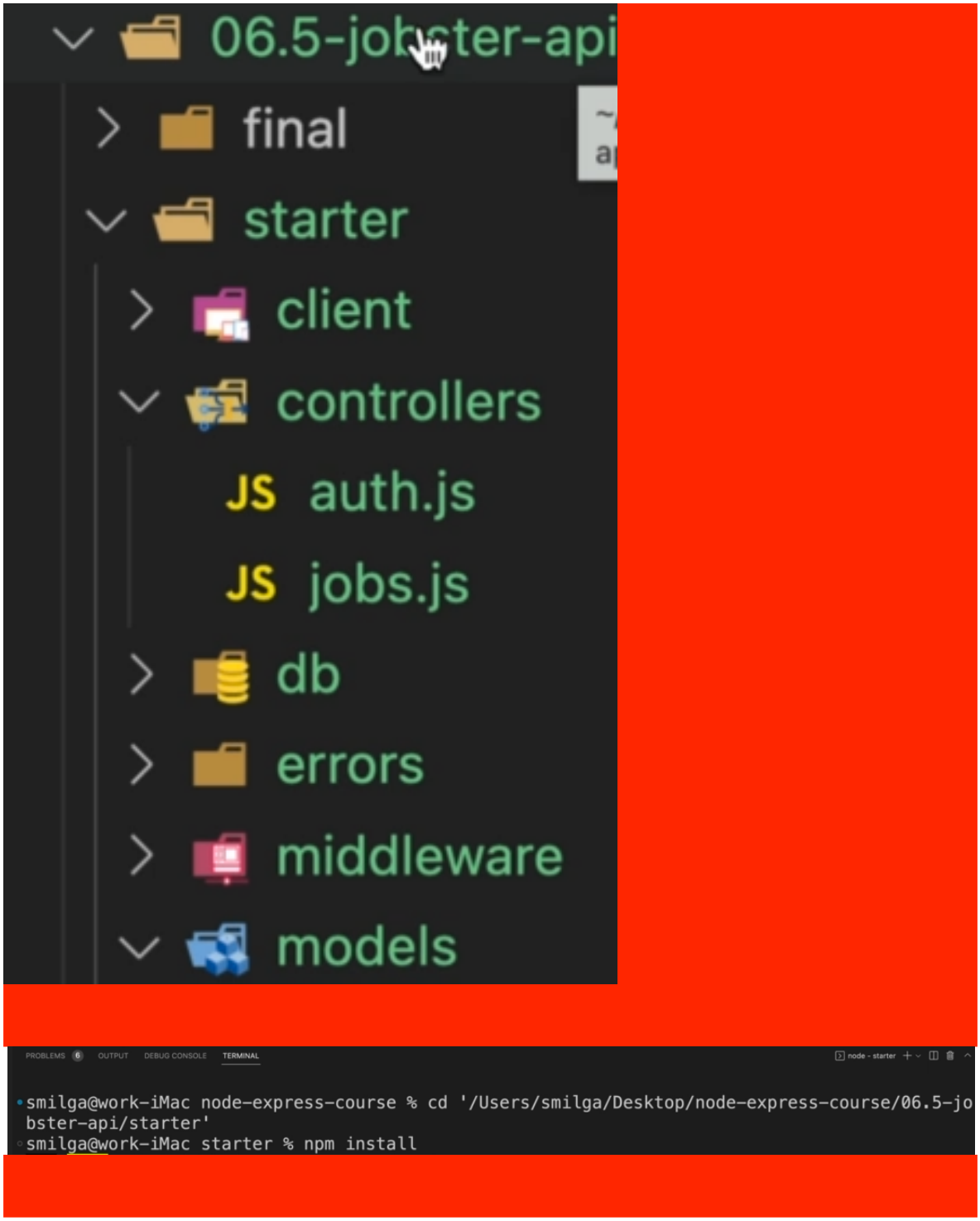
#### Starter

The starter is a copy of jobs-api final project, just with additional data.

#### Setup

- navigate to 06.5-jobster-api/starter
- install dependencies

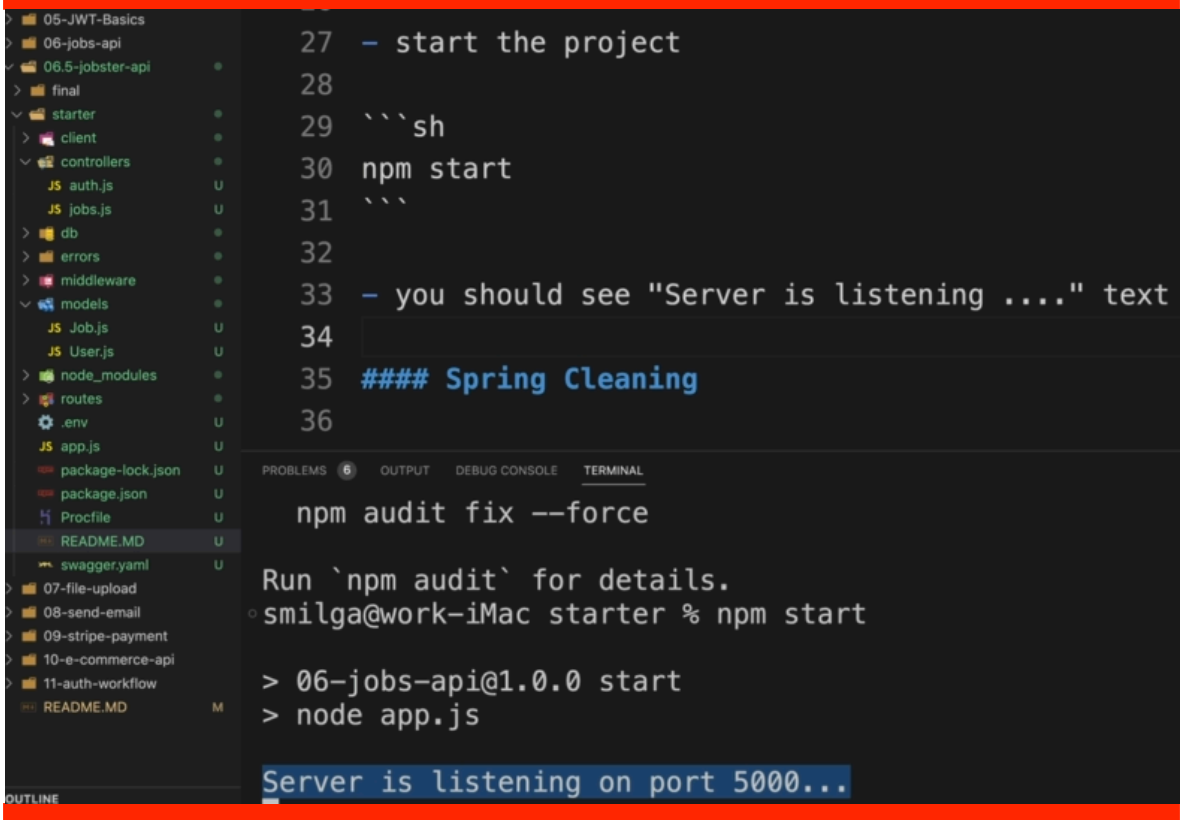
```sh
npm install
```
```


06.5-jobster-api
final
starter
client
controllers
auth.js
jobs.js
db
errors
middleware
models
PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
node - starter
smilga@work-iMac node-express-course % cd '/Users/smilga/Desktop/node-express-course/06.5-jobster-api/starter'
smilga@work-iMac starter % npm install

```
16 - create .env and provide correct values
17 - you can copy from previous project (just change the DB name)
18
19 .env
20
21 ```js
22 MONGO_URI=
23 JWT_SECRET=
24 JWT_LIFETIME=
25 ```
```

create .env file
/06.5-jobster-api/starter/.env

```
06.5-jobster-api > starter > .env
1 MONGO_URI=mongodb+srv://john:<password>@cluster0.4teny.mongodb.net/065-JOBSTER?
  retryWrites=true&w=majority
2 JWT_SECRET=Yq3s6v9y$B&E)H@McQfTjWnZr4u7w!z%
3 JWT_LIFETIME=30d
```



````
#### Spring Cleaning

##### Remove Swagger

- delete swagger.yaml file
- remove these lines of code
- delete swagger.yaml
  app.js

```js
const swaggerUI = require('swagger-ui-express');
const YAML = require('yamljs');
const swaggerDocument = YAML.load('./swagger.yaml');

app.get('/', (req, res) => {
  res.send('<h1>Jobs API</h1><a href="/api-docs">Documentation</a>');
});
app.use('/api-docs', swaggerUI.serve, swaggerUI.setup(swaggerDocument)
  );
```
````

/starter/app.js

README.MD | app.js

```js
require('dotenv').config();
require('express-async-errors');

// extra security packages
const helmet = require('helmet');
const cors = require('cors');
const xss = require('xss-clean');
const rateLimiter = require('express-rate-limit');

// Swagger
const swaggerUI = require('swagger-ui-express');
const YAML = require('yamljs');
const swaggerDocument = YAML.load('./swagger.yaml');

const express = require('express');
const app = express();

const connectDB = require('./db/connect');
const authenticateUser = require('./middleware/authentication');
// routers
const authRouter = require('./routes/auth');
const jobsRouter = require('./routes/jobs');
// error handler
const notFoundMiddleware = require('./middleware/not-found');
const errorHandlerMiddleware = require('./middleware/error-handler');

app.set('trust proxy', 1);
app.use(
  rateLimiter({
    windowMs: 15 * 60 * 1000, // 15 minutes
    max: 100, // limit each IP to 100 requests per windowMs
  })
);
```

```
app.use(express.json());
app.use(helmet());
app.use(cors());
app.use(xss());

app.get('/', (req, res) => {
  res.send('<h1>Jobs API</h1><a href="/api-docs">Documentation</a>');
});
app.use('/api-docs', swaggerUI.serve, swaggerUI.setup(swaggerDocument));

// routes
app.use('/api/v1/auth', authRouter);
app.use('/api/v1/jobs', authenticateUser, jobsRouter);

app.use(notFoundMiddleware);
app.use(errorHandlerMiddleware);

const port = process.env.PORT || 5000;

const start = async () => {
  try {
    await connectDB(process.env.MONGO_URI);
    app.listen(port, () =>
      console.log(`Server is listening on port ${port}...`)
    );
  } catch (error) {
    console.log(error);
  }
};

start();
```

/starter/app.js

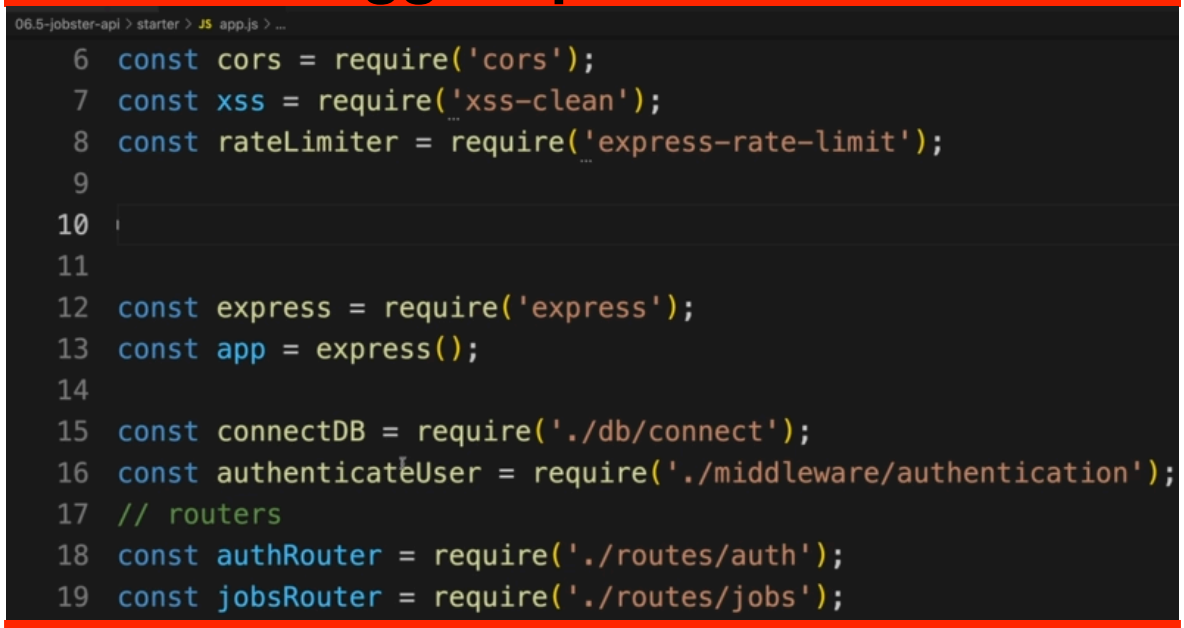
```
06.5-jobster-api > starter > JS app.js > ...
const cors = require('cors');
const xss = require('xss-clean');
const rateLimiter = require('express-rate-limit');

// Swagger
const swaggerUI = require('swagger-ui-express');
const YAML = require('yamljs');
const swaggerDocument = YAML.load('./swagger.yaml');

const express = require('express');
const app = express();

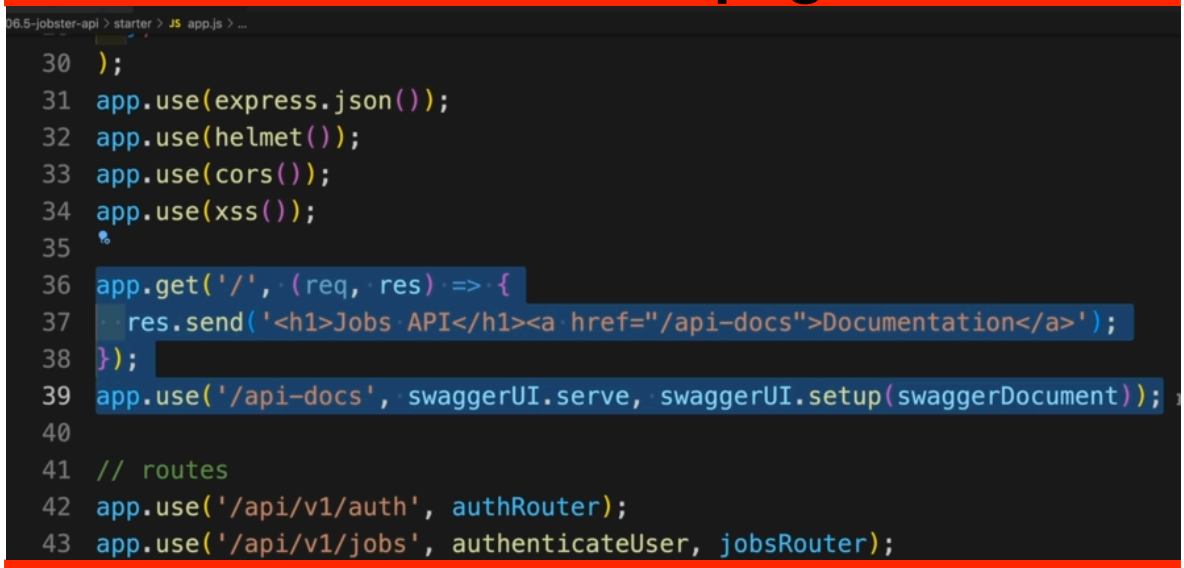
const connectDB = require('./db/connect');
const authenticateUser = require('./middleware/authentication');
```

remove swagger api

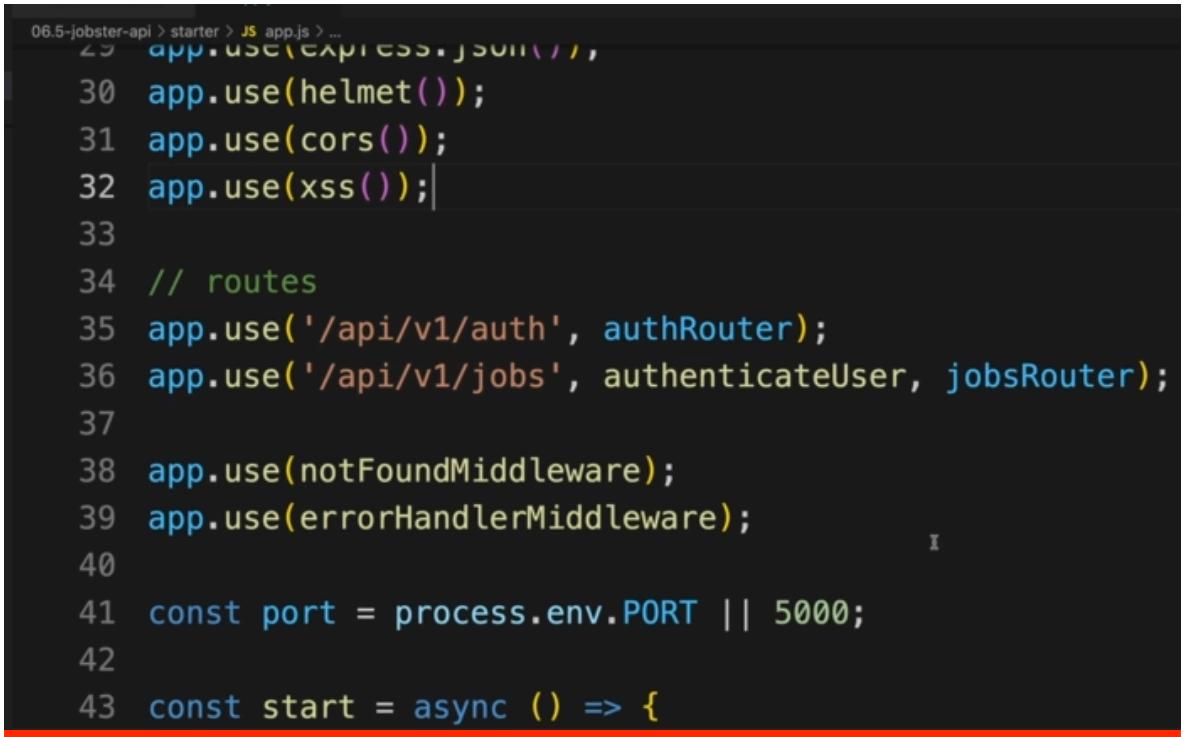


```
06.5-jobster-api > starter > JS app.js > ...
 6  const cors = require('cors');
 7  const xss = require('xss-clean');
 8  const rateLimiter = require('express-rate-limit');
 9
10
11
12  const express = require('express');
13  const app = express();
14
15  const connectDB = require('./db/connect');
16  const authenticateUser = require('./middleware/authentication');
17  // routers
18  const authRouter = require('./routes/auth');
19  const jobsRouter = require('./routes/jobs');
```

remove this from home page



```
06.5-jobster-api > starter > JS app.js > ...
30  );
31  app.use(express.json());
32  app.use(helmet());
33  app.use(cors());
34  app.use(xss());
35
36  app.get('/', (req, res) => {
37    res.send('<h1>Jobs API</h1><a href="/api-docs">Documentation</a>');
38  });
39  app.use('/api-docs', swaggerUI.serve, swaggerUI.setup(swaggerDocument));
40
41  // routes
42  app.use('/api/v1/auth', authRouter);
43  app.use('/api/v1/jobs', authenticateUser, jobsRouter);
```



Remove API Limiter

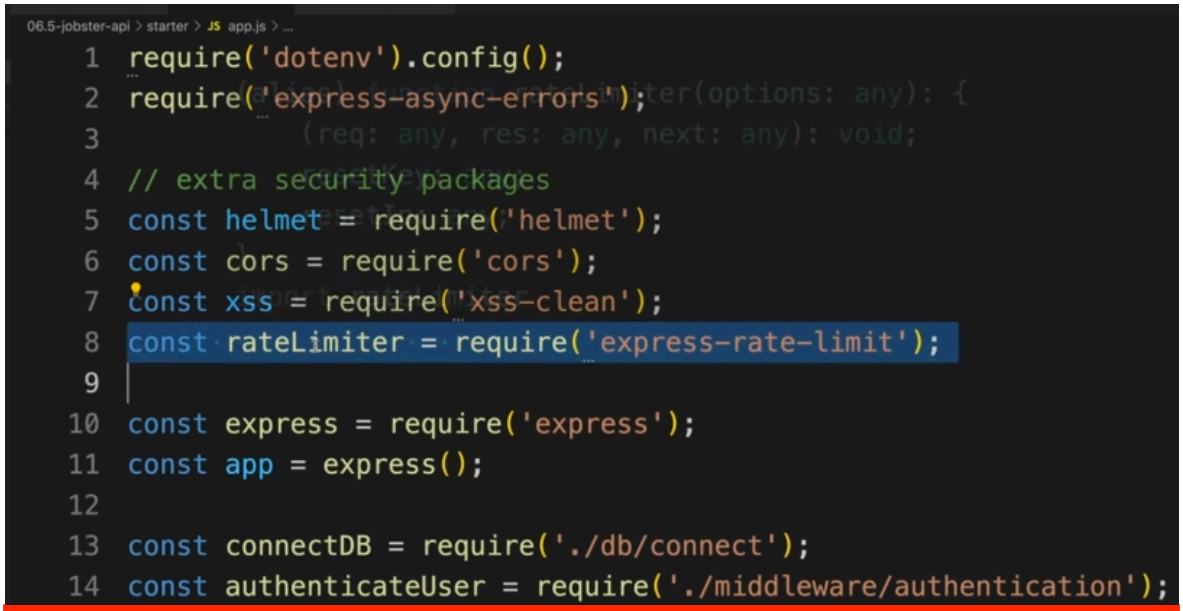
- remove these lines of code

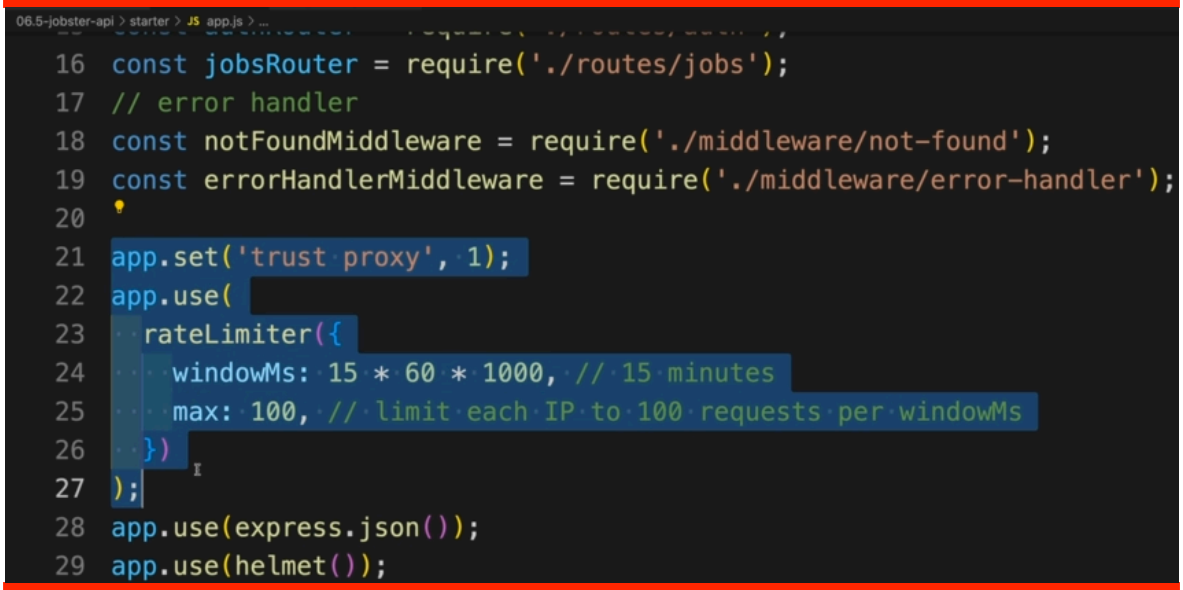
app.js

```js
const rateLimiter = require('express-rate-limit');

app.set('trust proxy', 1);
app.use(
  rateLimiter({
    windowMs: 15 * 60 * 1000, // 15 minutes
    max: 100, // limit each IP to 100 requests per windowMs
  })
);
```

remove rateLimiter

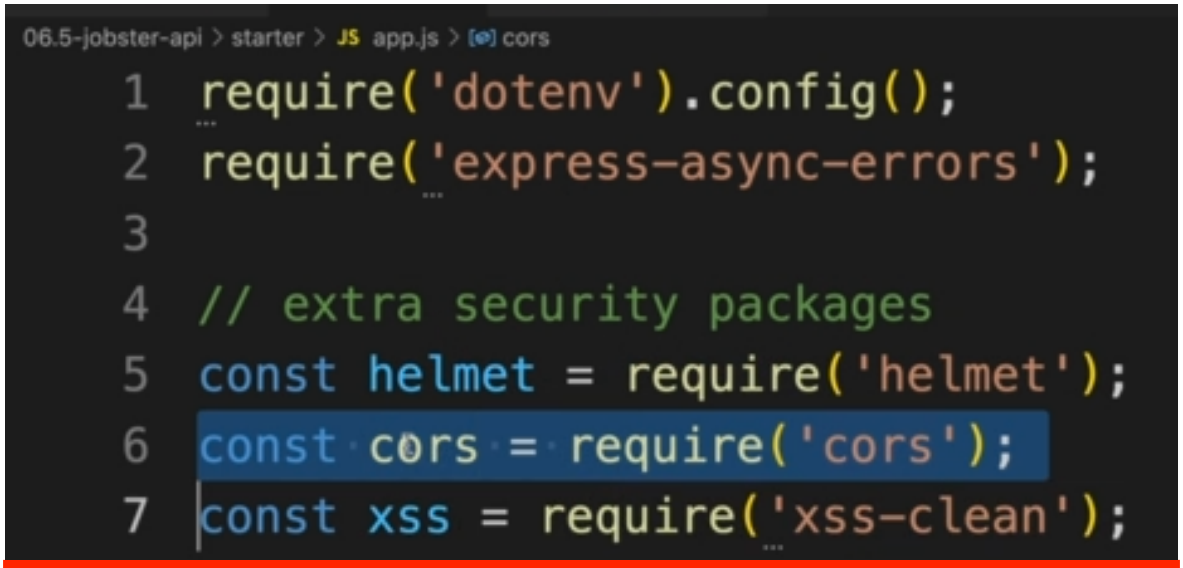




Remove CORS

- we don't want external JS apps to access our API (only our front-end)
- remove these lines of code

```js
const cors = require('cors');
app.use(cors());
```


06.5-jobster-api > starter > JS app.js > cors
1 require('dotenv').config();
2 require('express-async-errors');
3
4 // extra security packages
5 const helmet = require('helmet');
6 const cors = require('cors');
7 const xss = require('xss-clean');


06.5-jobster-api > starter > JS app.js > ...
17 const notFoundMiddleware = require('./middleware/not-found');
18 const errorHandlerMiddleware = require('./middleware/error-handler');
19
20 app.use(express.json());
21 app.use(helmet());
22 app.use(cors());
23 app.use(xss());
24
25 // routes
26 app.use('/api/v1/auth', authRouter);
27 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
28
29 app.use(notFoundMiddleware);
30 app.use(errorHandlerMiddleware);

````
#### Package.json

- add "dev" script with nodemon
- change engines to current version (in my case 16)

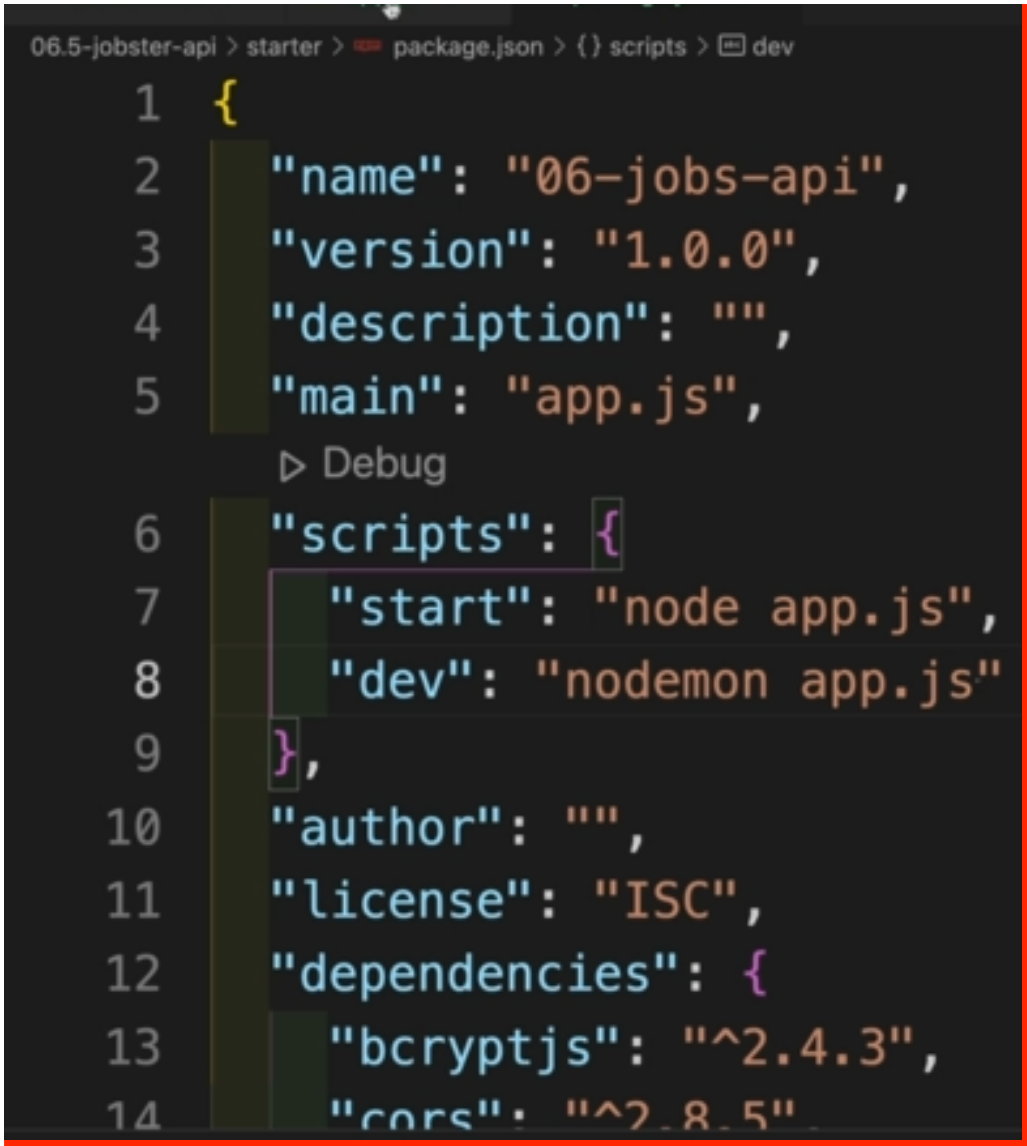
package.json

```js

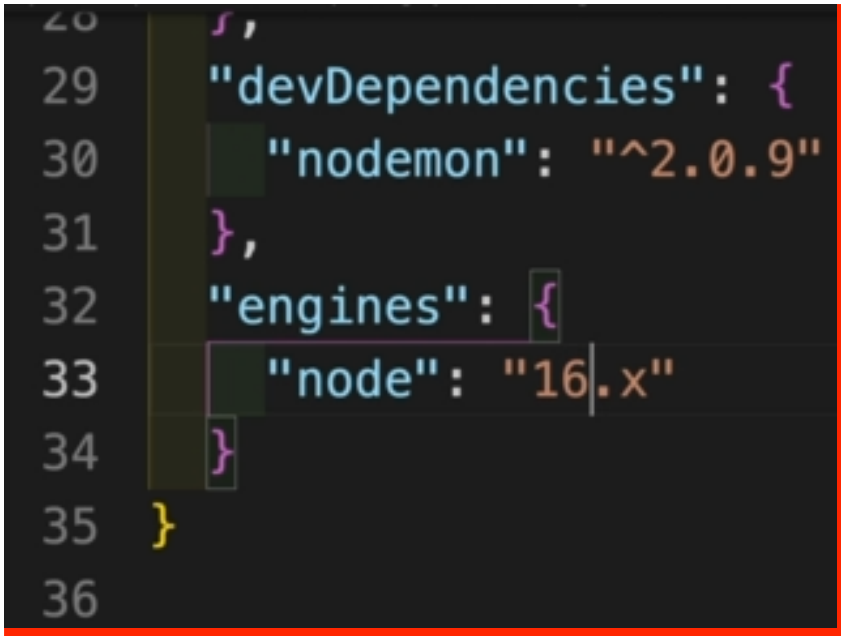
"scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
  },

"engines": {
    "node": "16.x"
  }
```

- restart server with "npm run dev"
````

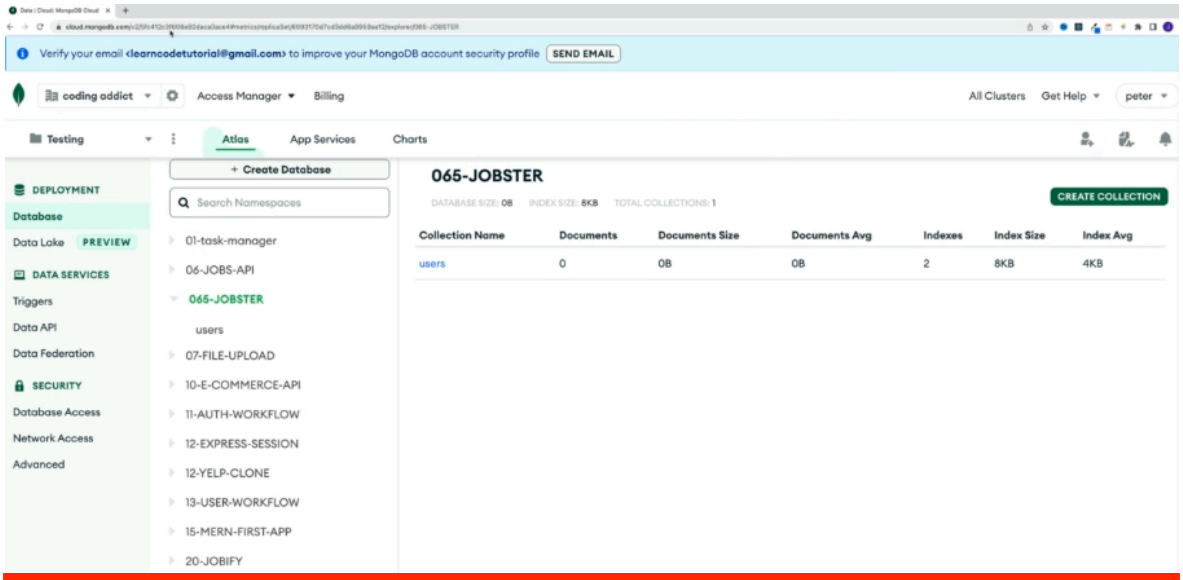


if we want, change it to latest node version

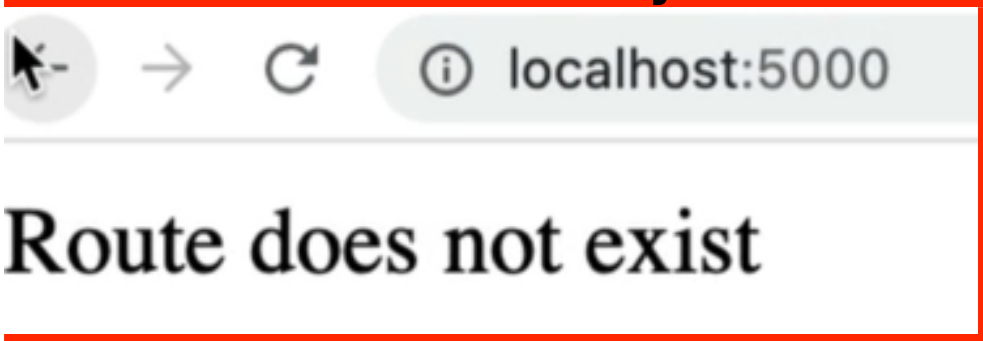



```
{
 "name": "06-jobs-api",
 "version": "1.0.0",
 "description": "",
 "main": "app.js",
 "scripts": {
  "start": "node app.js",
  "dev": "nodemon app.js"
 },
 "author": "",
 "license": "ISC",
 "dependencies": {
  "bcryptjs": "^2.4.3",
  "cors": "^2.8.5",
  "dotenv": "^10.0.0",
```

```
    "express": "^4.17.1",
    "express-async-errors": "^3.1.1",
    "express-rate-limit": "^5.3.0",
    "helmet": "^4.6.0",
    "http-status-codes": "^2.1.4",
    "joi": "^17.4.0",
    "jsonwebtoken": "^8.5.1",
    "moment": "^2.29.4",
    "mongoose": "^6.5.2",
    "rate-limiter": "^0.2.0",
    "swagger-ui-express": "^4.1.6",
    "xss-clean": "^0.1.1",
    "yamljs": "^0.3.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.9"
  }
}
```



check the functionality



4 - Client Folder

==========

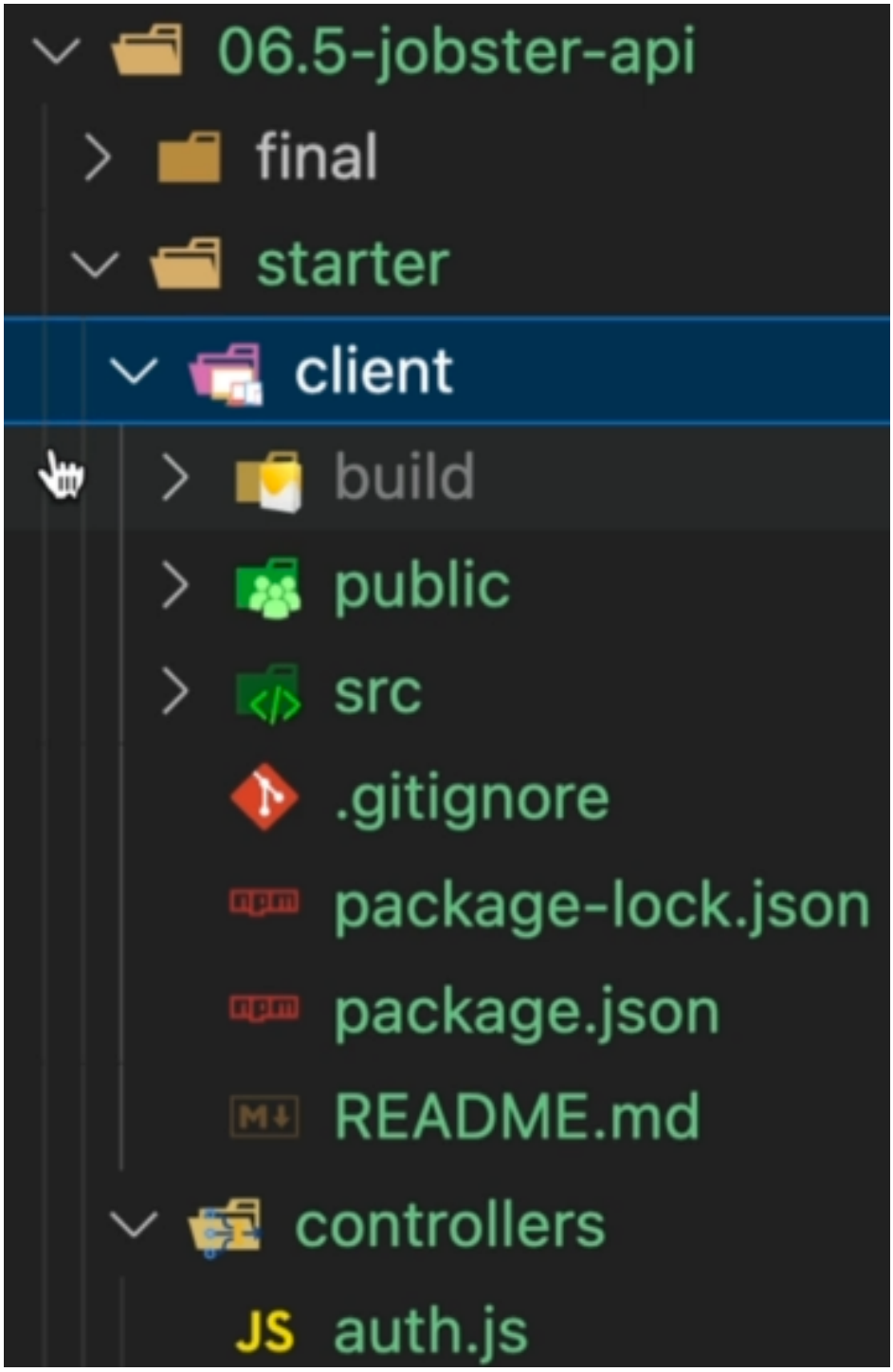
````
##### Client Folder

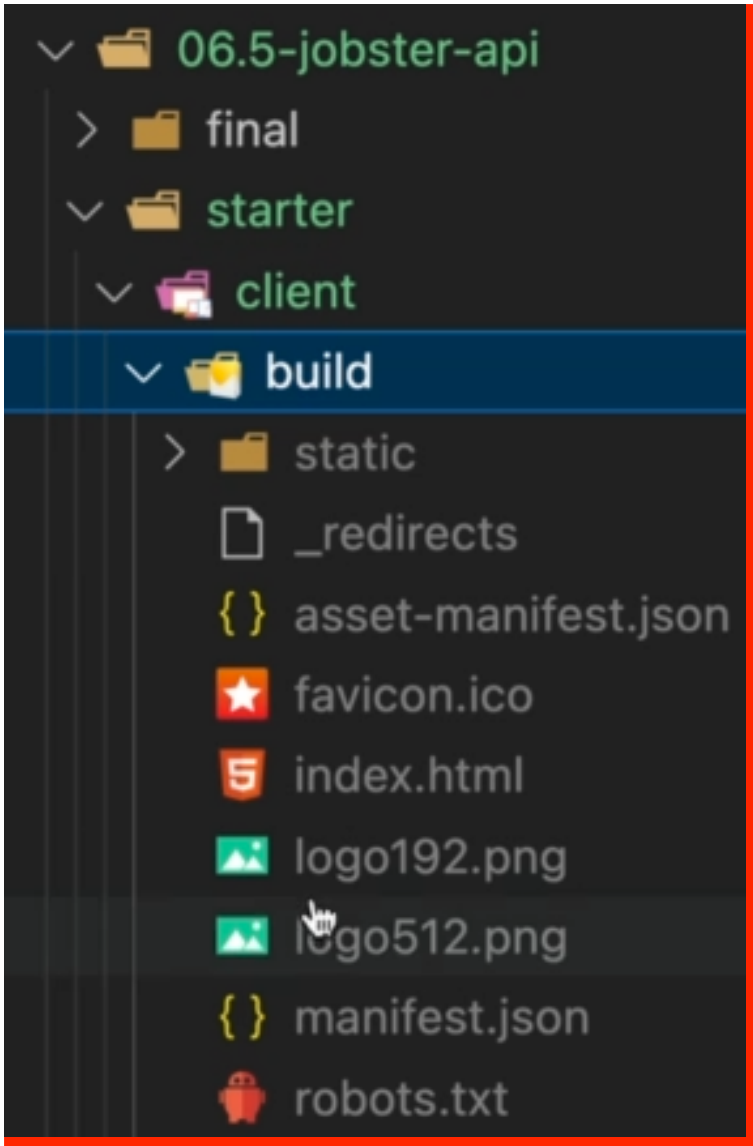
- let's explore client folder
- open client folder
- it's a react app built with CRA
- it's the same as in my React Course (JOBSTER APP),
  just base url points to our current server (instead of heroku app)

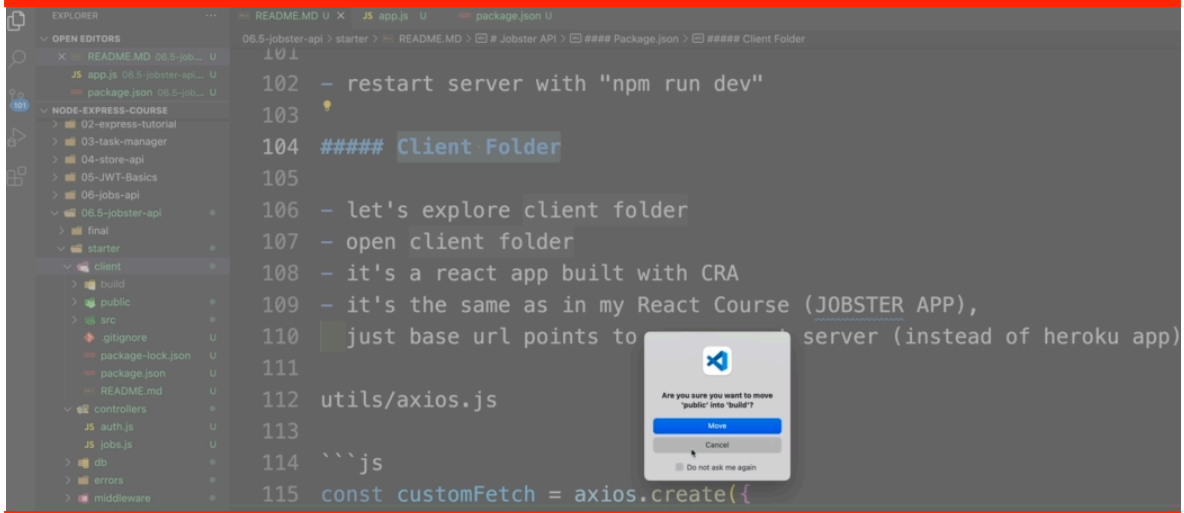
utils/axios.js

```js
const customFetch = axios.create({
  baseURL: '/api/v1',
});
```

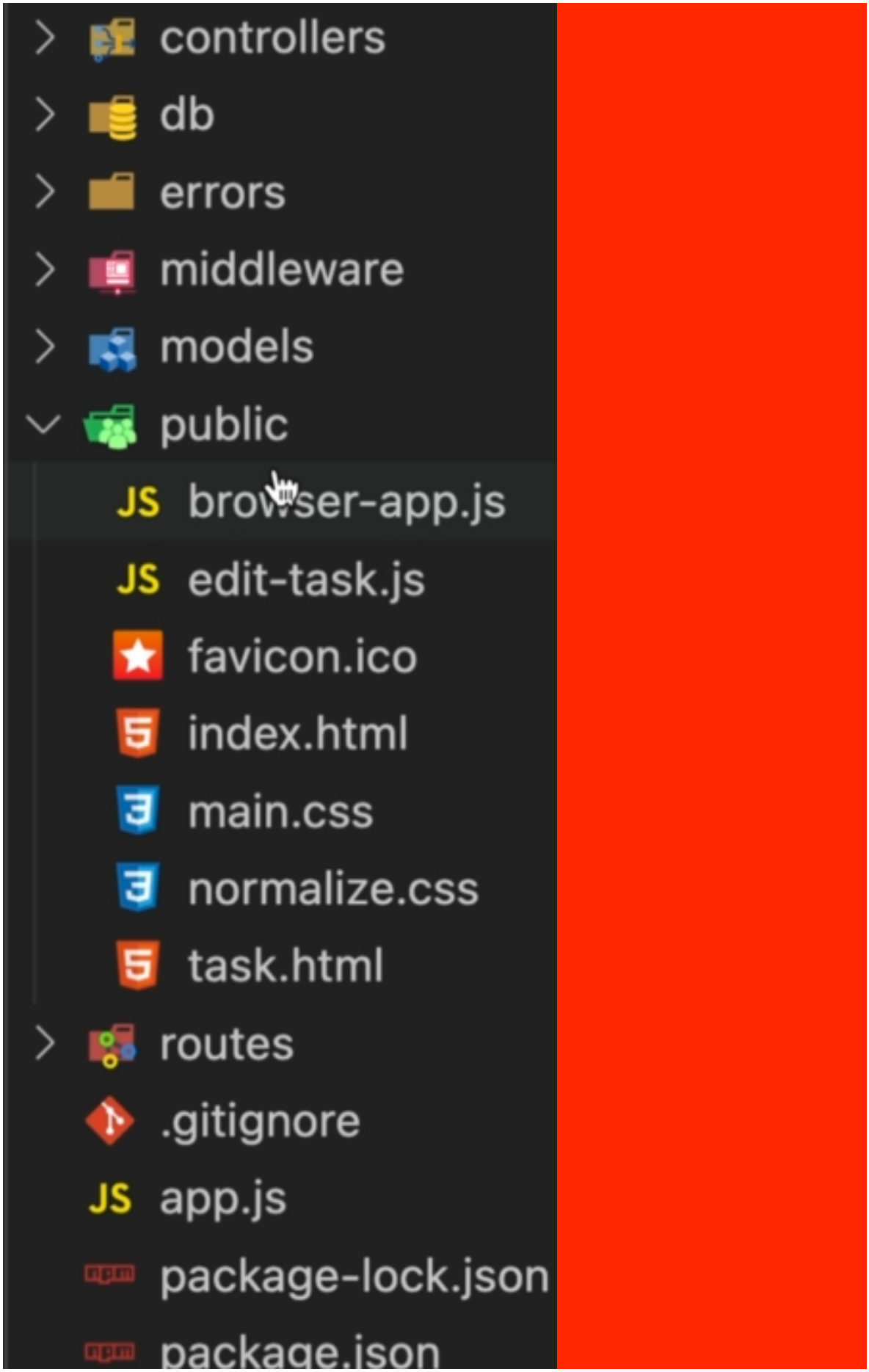
- notice the build folder (production ready application)
- in CRA we can create build folder by running "npm run build"
- that's the one we will use for our front-end
````


06.5-jobster-api
final
starter
client
build
public
src
.gitignore
package-lock.json
package.json
README.md
controllers
auth.js


06.5-jobster-api
final
starter
client
build
static
_redirects
asset-manifest.json
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt


- restart server with "npm run dev"
Client Folder
- let's explore client folder
- open client folder
- it's a react app built with CRA
- it's the same as in my React Course (JOBSTER APP),
just base url points to server (instead of heroku app)
utils/axios.js
```js
const customFetch = axios.create({
Are you sure you want to move 'public' into 'build'?
Move
Cancel
Do not ask me again
```

previous project, we created in public folder


controllers
db
errors
middleware
models
public
browser-app.js
edit-task.js
favicon.ico
index.html
main.css
normalize.css
task.html
routes
.gitignore
app.js
package-lock.json
package.json

5 - Setup Front-End

==========

````
##### Setup Front-End

- require "path" module
- setup express static (as first middleware)
  to serve static assets from client/build
- so now instead of public folder we are using client/build

app.js

```js
const path = require('path');

app.use(express.static(path.resolve(__dirname, './client/build')));

// place as first middleware
app.use(express.json());
app.use(helmet());
app.use(cors());
app.use(xss());
```

- serve index.html for all routes (apart from API)
- front-end routes pick's it up from there

```js
app.use('/api/v1/auth', authRouter);
app.use('/api/v1/jobs', authenticateUser, jobsRouter);

// serve index.html
app.get('*', (req, res) => {
  res.sendFile(path.resolve(__dirname, './client/build', 'index.html')
    );
});

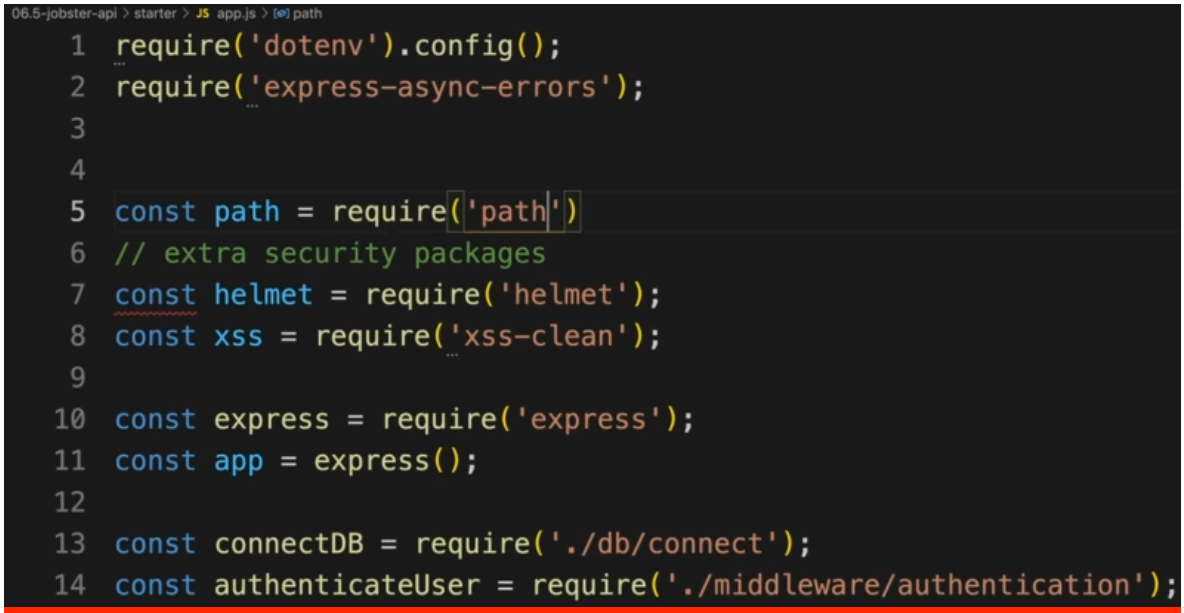
app.use(notFoundMiddleware);
app.use(errorHandlerMiddleware);
```

- navigate to localhost:5000
- clear local storage (if necessary)
````

app.js

======

const path = require('path');



```
06.5-jobster-api > starter > JS app.js > (e) path
1  require('dotenv').config();
2  require('express-async-errors');
3
4
5  const path = require('path')
6  // extra security packages
7  const helmet = require('helmet');
8  const xss = require('xss-clean');
9
10 const express = require('express');
11 const app = express();
12
13 const connectDB = require('./db/connect');
14 const authenticateUser = require('./middleware/authentication');
```

```
app.use(express.static(path.resolve(__dirname, './client/build')));
```

```
06.5-jobster-api > starter > JS app.js > ...
16 const jobsRouter = require('./routes/jobs');
17 // error handler
18 const notFoundMiddleware = require('./middleware/not-found');
19 const errorHandlerMiddleware = require('./middleware/error-handler');
20
21 app.use(express.static(path.resolve(__dirname, './client/build')));
22 app.use(express.json());
23 app.use(helmet());
24
25 app.use(xss());
```

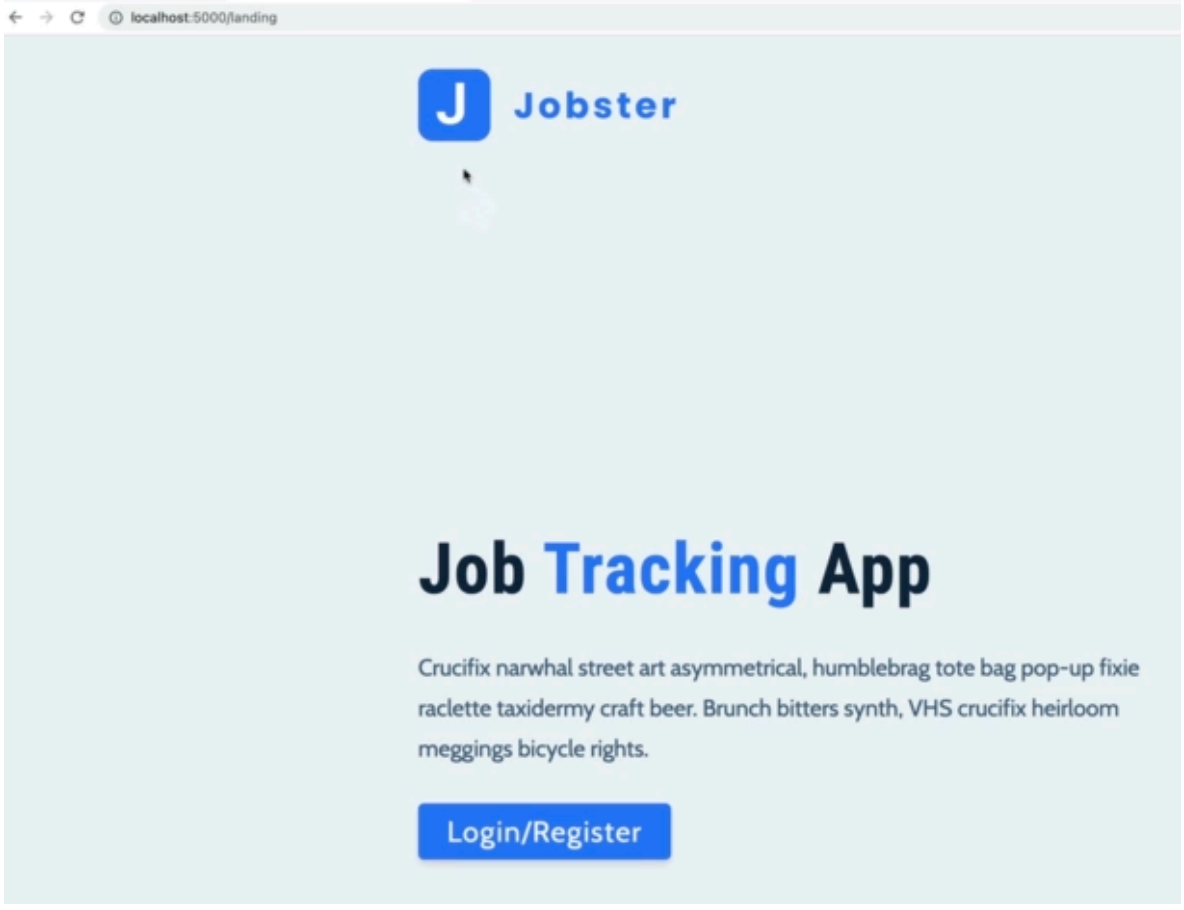
```
app.get('*', (req, res) => {
  res.sendFile(path.resolve(__dirname, './client/build', 'index.html'));
});
```

06.5-jobster-api > starter > JS app.js > app.get('*') callback

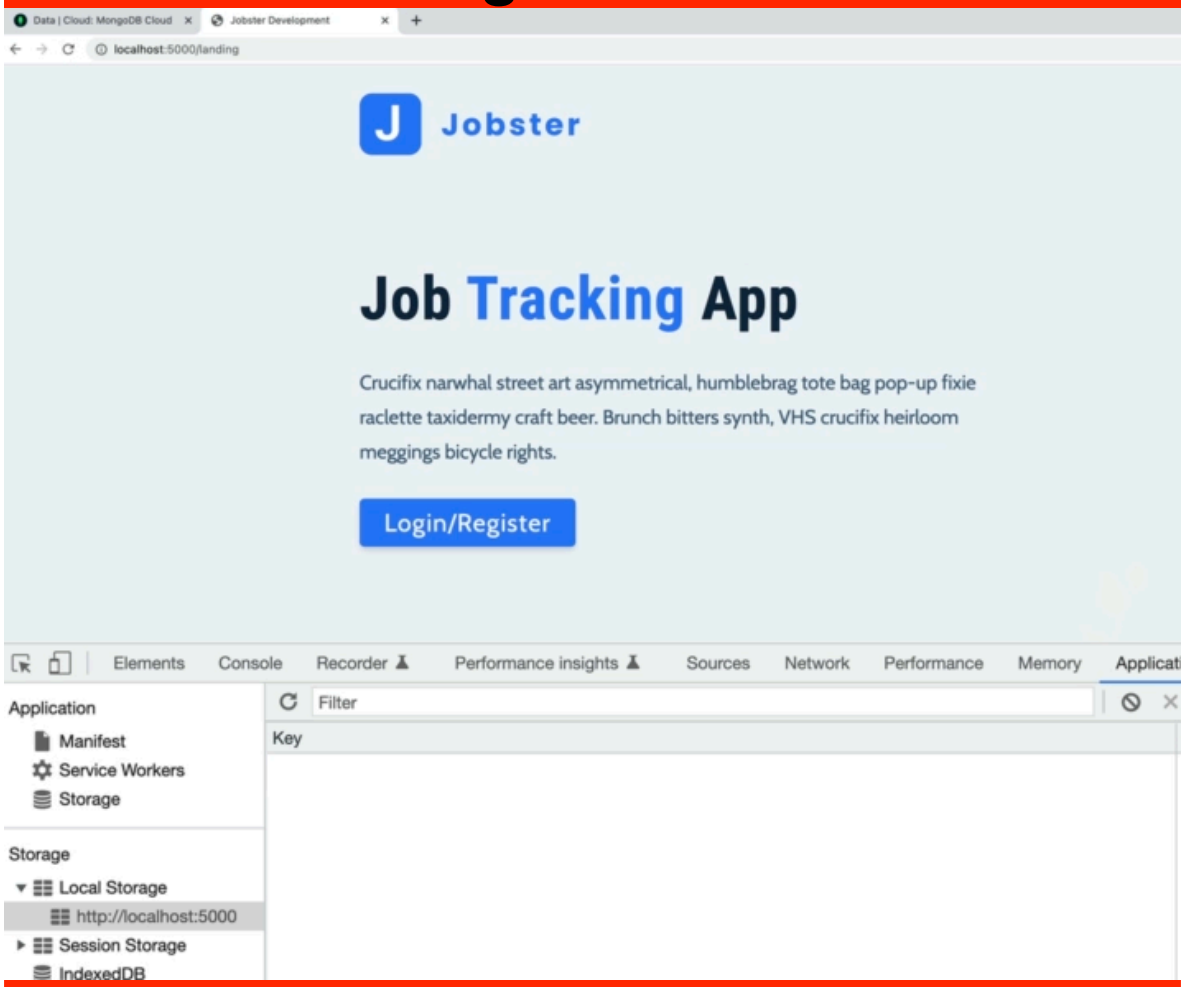
```
23 app.use(helmet());
24
25 app.use(xss());
26
27 // routes
28 app.use('/api/v1/auth', authRouter);
29 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
30
31 app.get('*', (req, res) => {
32   res.sendFile(path.resolve(__dirname, './client/build', 'index.html'));
33 });
34
35 app.use(notFoundMiddleware);
36 app.use(errorHandlerMiddleware);
```

```
smilga@work-iMac starter % npm run dev
```

if we are seeing the running page, we are moving in correct direction



clear local storage of the browser



app.js

```
=======
require('dotenv').config();
require('express-async-errors');

const path = require('path');
// extra security packages
const helmet = require('helmet');
```

```
const xss = require('xss-clean');

const express = require('express');
const app = express();

const connectDB = require('./db/
connect');
const authenticateUser = require('./
middleware/authentication');
// routers
const authRouter = require('./routes/
auth');
const jobsRouter = require('./routes/
jobs');
// error handler
const notFoundMiddleware = require('./
middleware/not-found');
const errorHandlerMiddleware =
require('./middleware/error-handler');

app.use(express.static(path.resolve(__
dirname, './client/build')));
app.use(express.json());
app.use(helmet());
```

```
app.use(xss());

// routes
app.use('/api/v1/auth', authRouter);
app.use('/api/v1/jobs',
authenticateUser, jobsRouter);

app.get('*', (req, res) => {
  res.sendFile(path.resolve(__dirname,
'./client/build', 'index.html'));
});

app.use(notFoundMiddleware);
app.use(errorHandlerMiddleware);

const port = process.env.PORT || 5000;

const start = async () => {
  try {
    await
connectDB(process.env.MONGO_URI);
    app.listen(port, () =>
      console.log(`Server is listening on
```

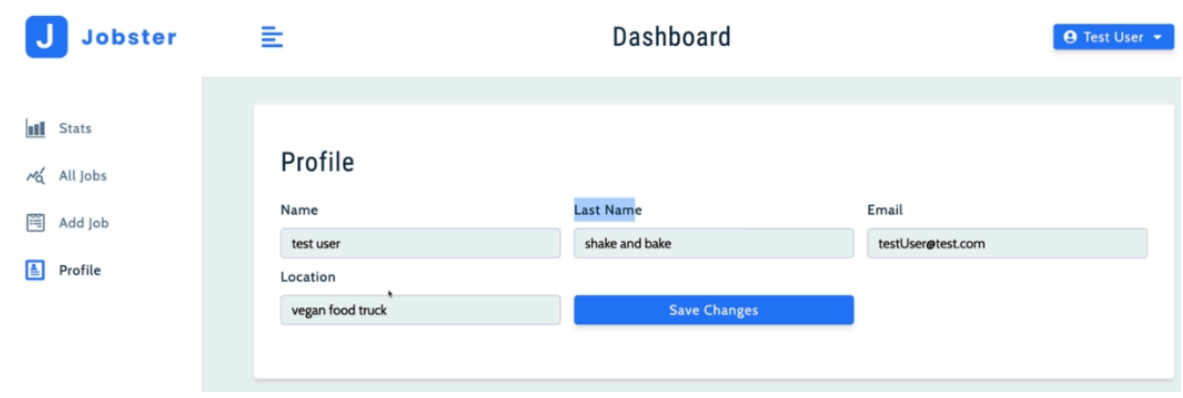
```
port ${port}...`)
    );
  } catch (error) {
    console.log(error);
  }
};

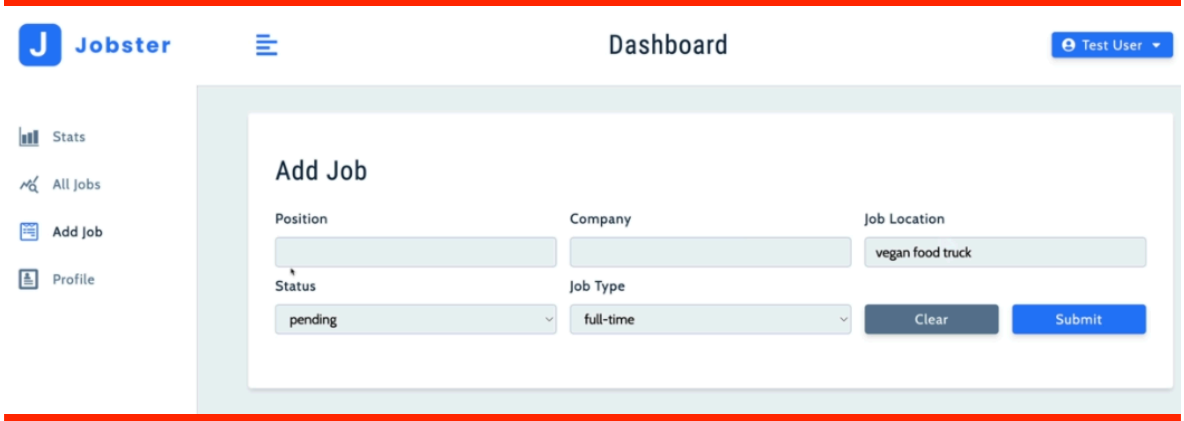
start();
```

6 - Modify User Model

```
164  #### Modify User Model
165
166  - add following properties
167
168  models/User.js
169
170  ```js
171  lastName: {
172      type: String,
173      trim: true,
174      maxlength: 20,
175      default: 'lastName',
176    },
177    location: {
178      type: String,
179      trim: true,
180      maxlength: 20,
181      default: 'my city',
182    },
183
184  ```
```

we have to create new fields lastName and location in Profile and Add Job pages

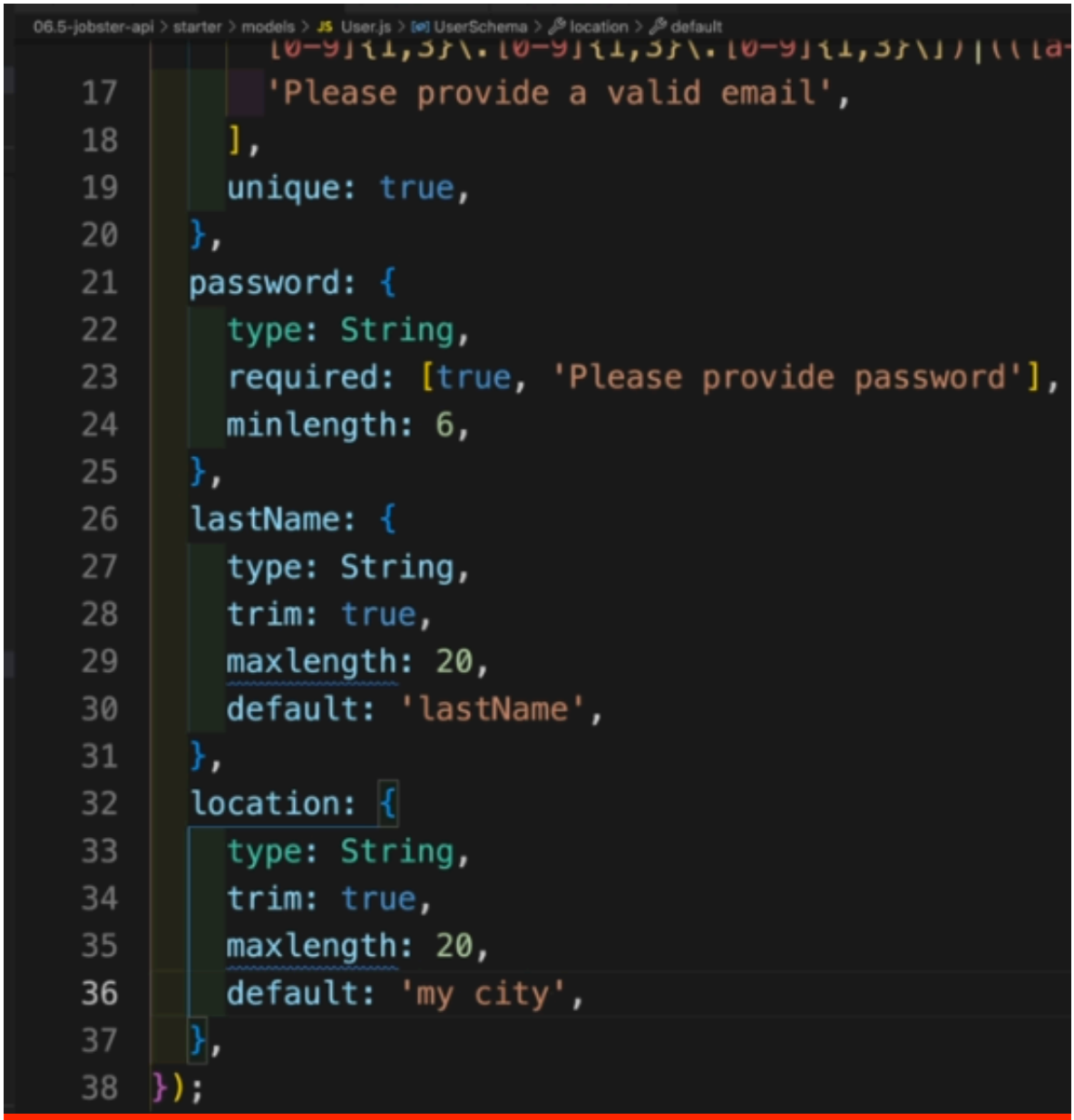




/starter/models/User.js

==========

```
lastName: {
    type: String,
    trim: true,
    maxlength: 20,
    default: 'lastName',
  },
  location: {
    type: String,
    trim: true,
    maxlength: 20,
    default: 'my city',
  },
```




```
const mongoose =
require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const UserSchema = new
mongoose.Schema({
```

```
  name: {
    type: String,
    required: [true, 'Please provide
name'],
    maxlength: 50,
    minlength: 3,
  },
  email: {
    type: String,
    required: [true, 'Please provide
email'],
    match: [
      /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:
\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]
{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-
Z\-0-9]+\.)+[a-zA-Z]{2,}))$/,
      'Please provide a valid email',
    ],
    unique: true,
  },
  password: {
    type: String,
    required: [true, 'Please provide
password'],
```

```
    minlength: 6,
  },
  lastName: {
    type: String,
    trim: true,
    maxlength: 20,
    default: 'lastName',
  },
  location: {
    type: String,
    trim: true,
    maxlength: 20,
    default: 'my city',
  },
});

UserSchema.pre('save', async function
() {
  const salt = await bcrypt.genSalt(10);
  this.password = await
bcrypt.hash(this.password, salt);
});

UserSchema.methods.createJWT =
```

```
function () {
  return jwt.sign(
    { userId: this._id, name: this.name },
    process.env.JWT_SECRET,
    {
      expiresIn:
process.env.JWT_LIFETIME,
    }
  );
};

UserSchema.methods.comparePasswo
rd = async function
(canditatePassword) {
  const isMatch = await
bcrypt.compare(canditatePassword,
this.password);
  return isMatch;
};

module.exports =
mongoose.model('User', UserSchema);
```

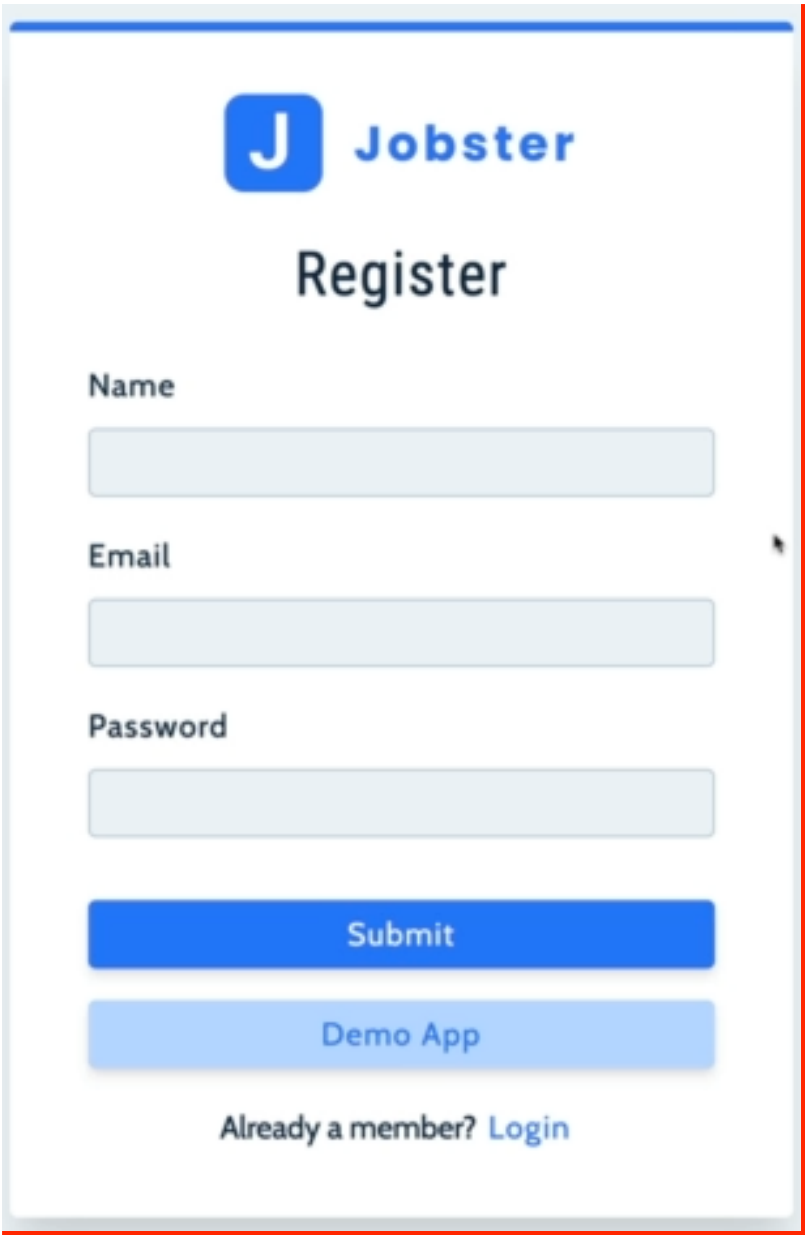
7 - Modify Register and Login

Modify Response in Register and Login

- change the response structure in register and login controllers (just keep old StatusCodes)

controllers/auth.js

```js
res.status(StatusCodes.CREATED).json({
  user: {
    email: user.email,
    lastName: user.lastName,
    location: user.location,
    name: user.name,
    token,
  },
});
```

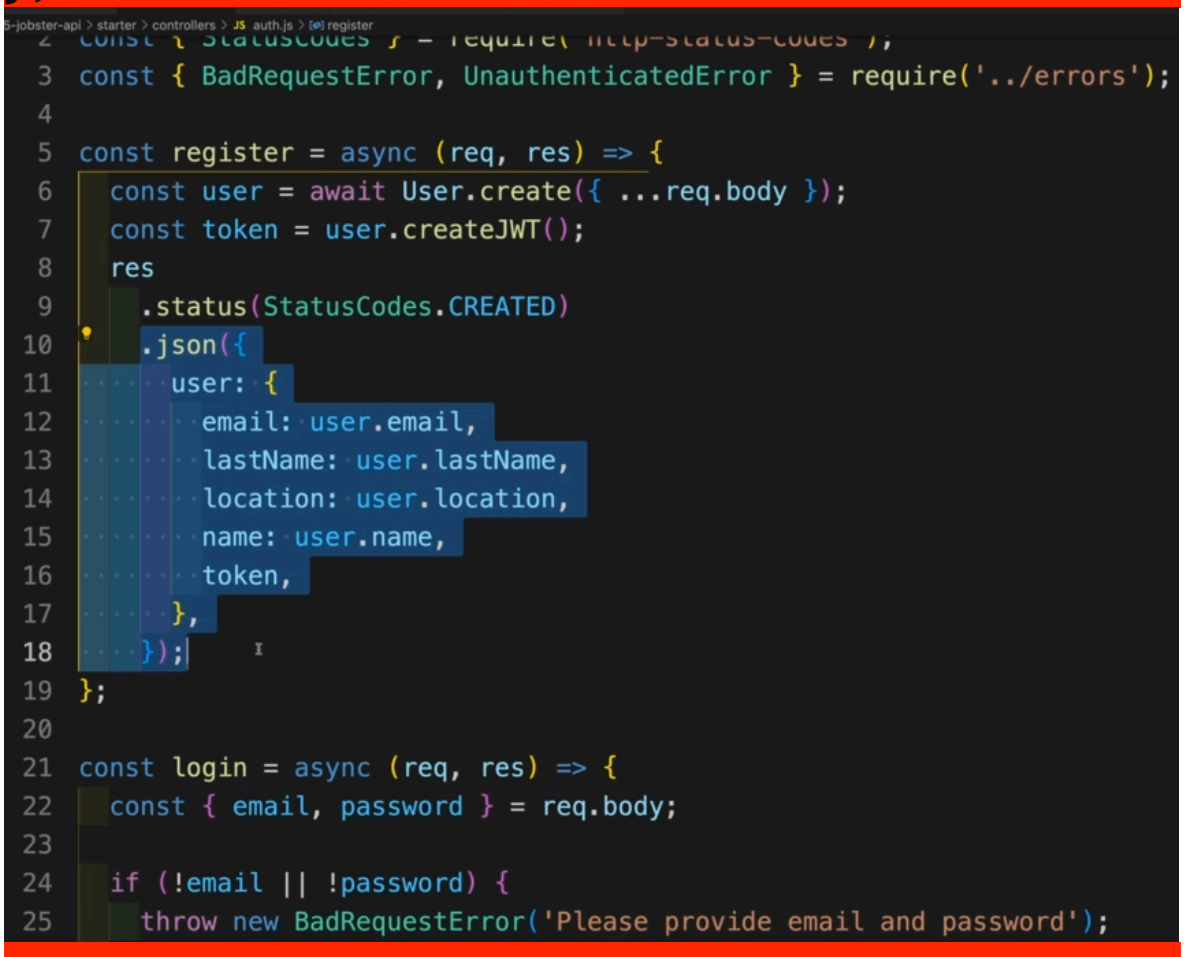


/starter/controllers/auth.js

==========

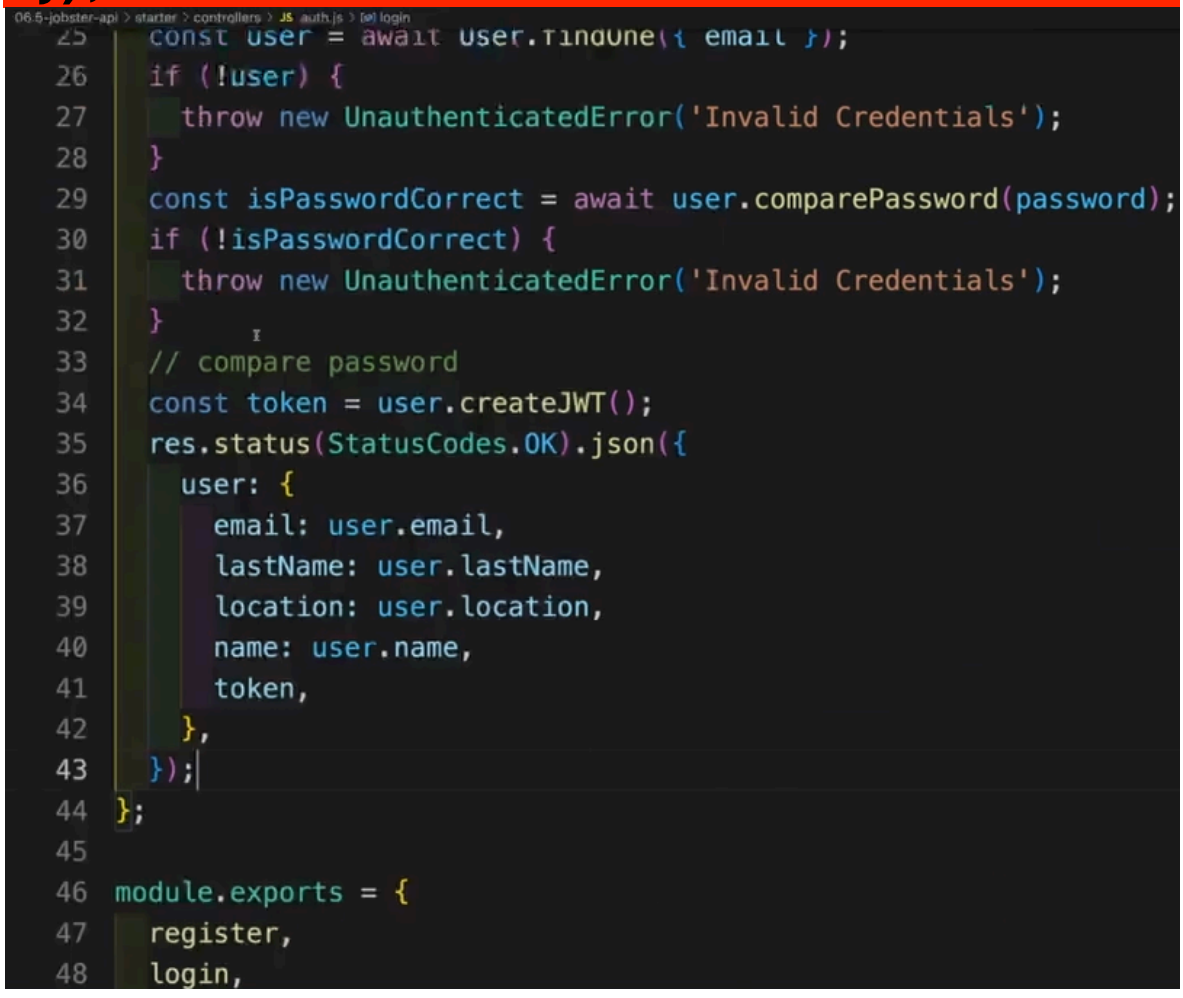
```
const register = async (req, res) => {
  const user = await
User.create({ ...req.body });
  const token = user.createJWT();
```

```
res.status(StatusCodes.CREATED).jso
n({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};
```

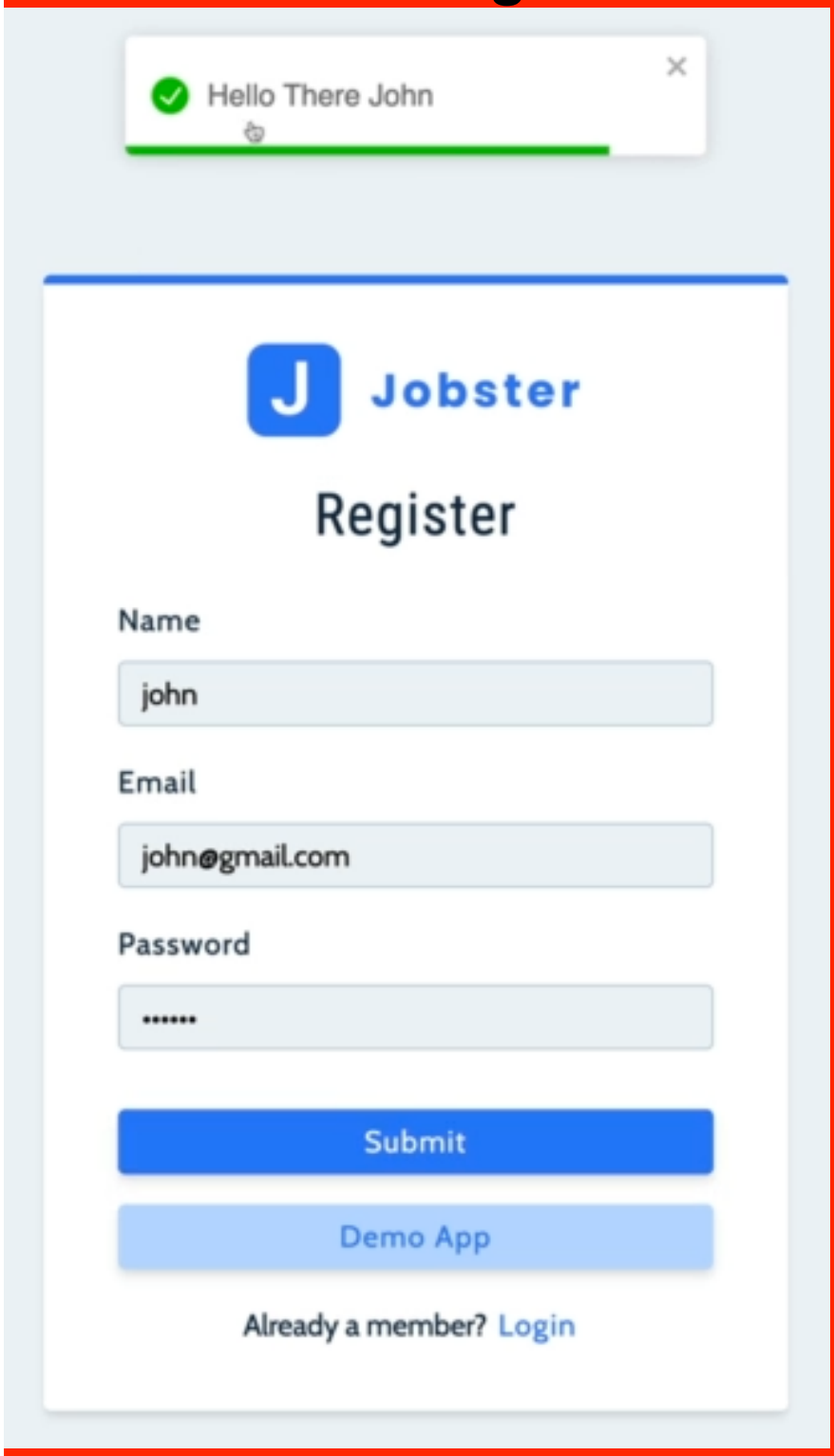


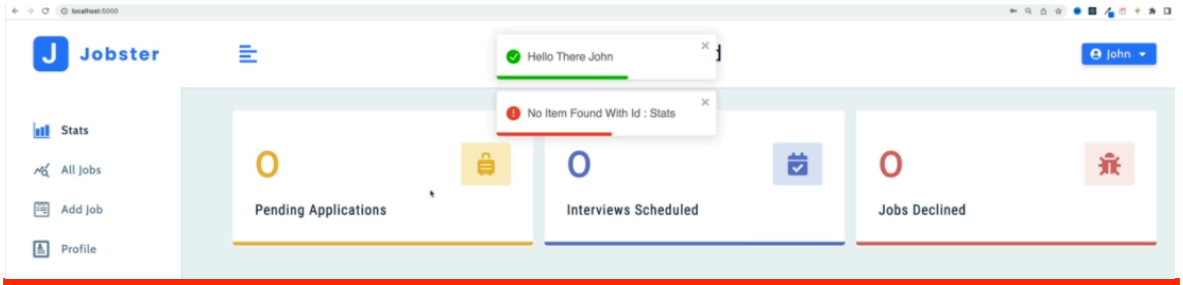
we have to do same thing for login

```
res.status(StatusCodes.OK).json({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
```

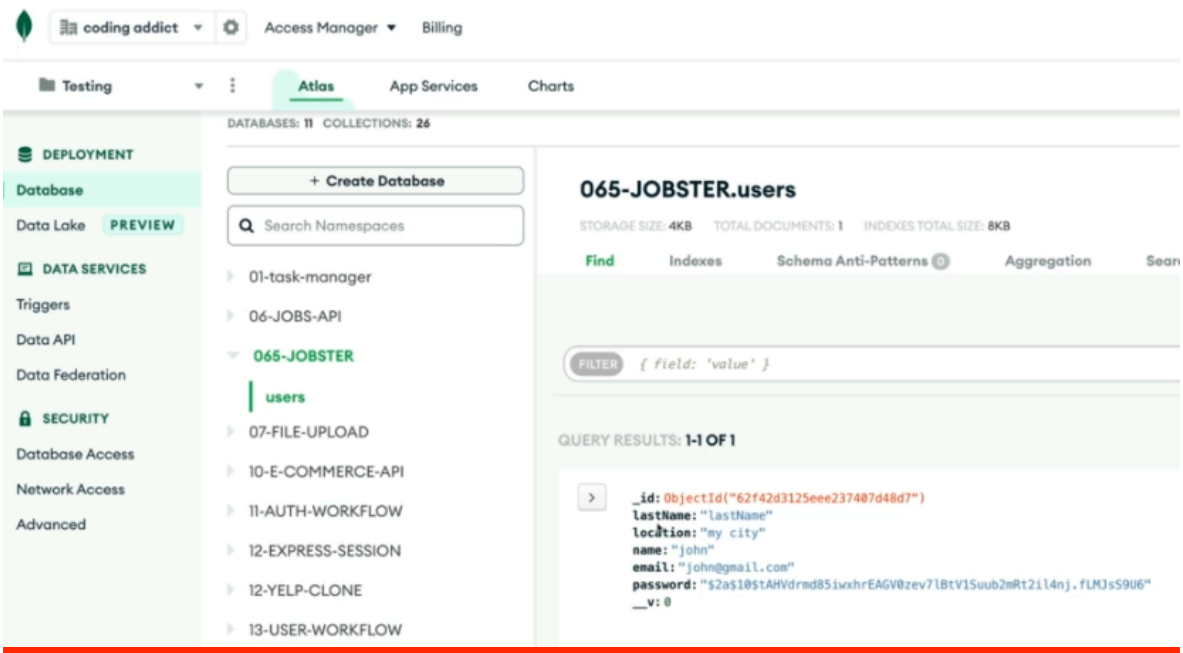


localhost:5000/register

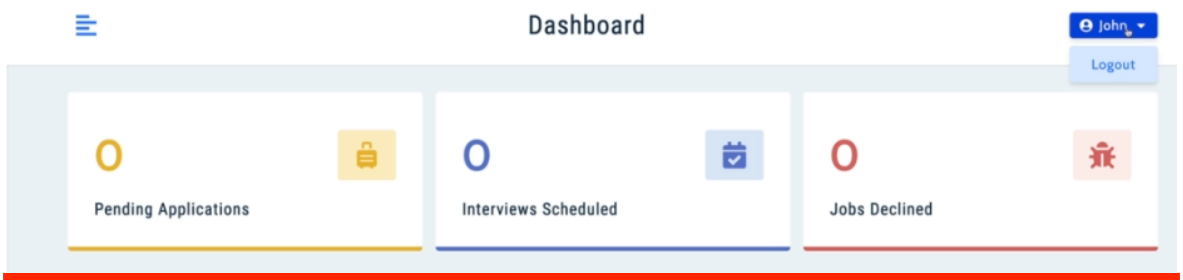


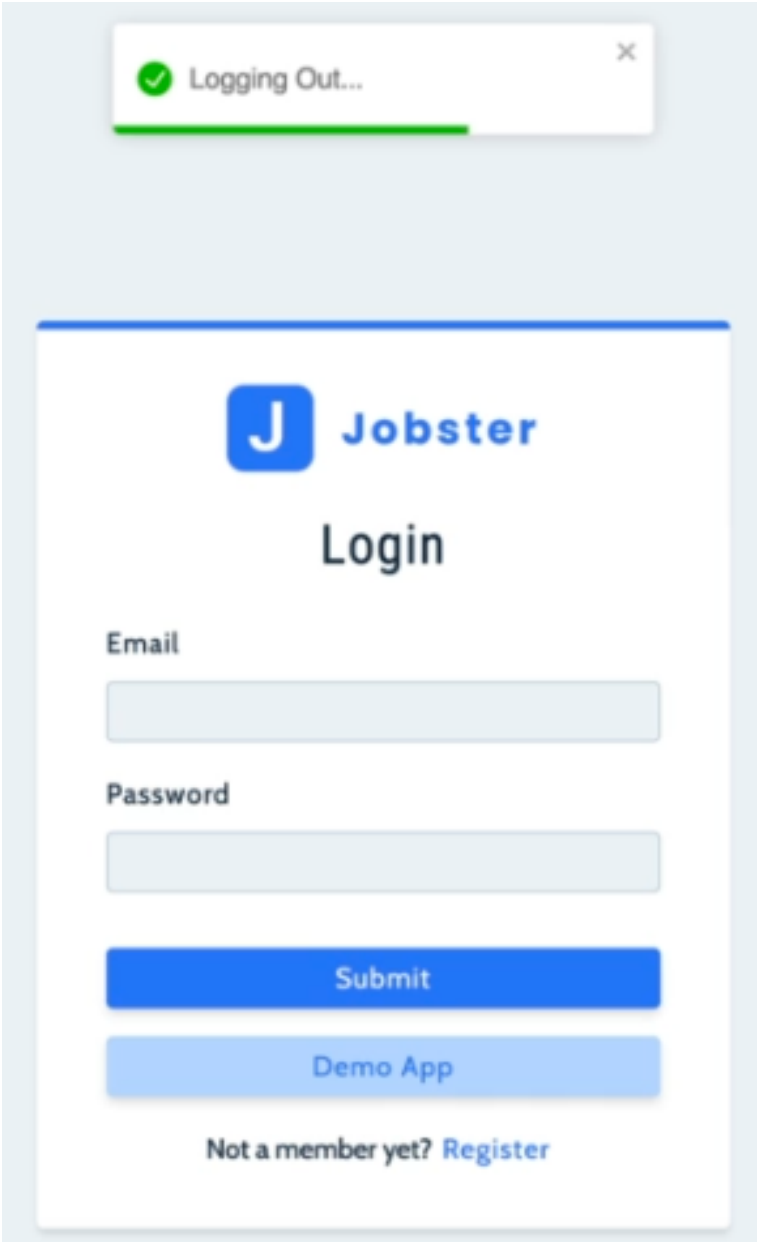


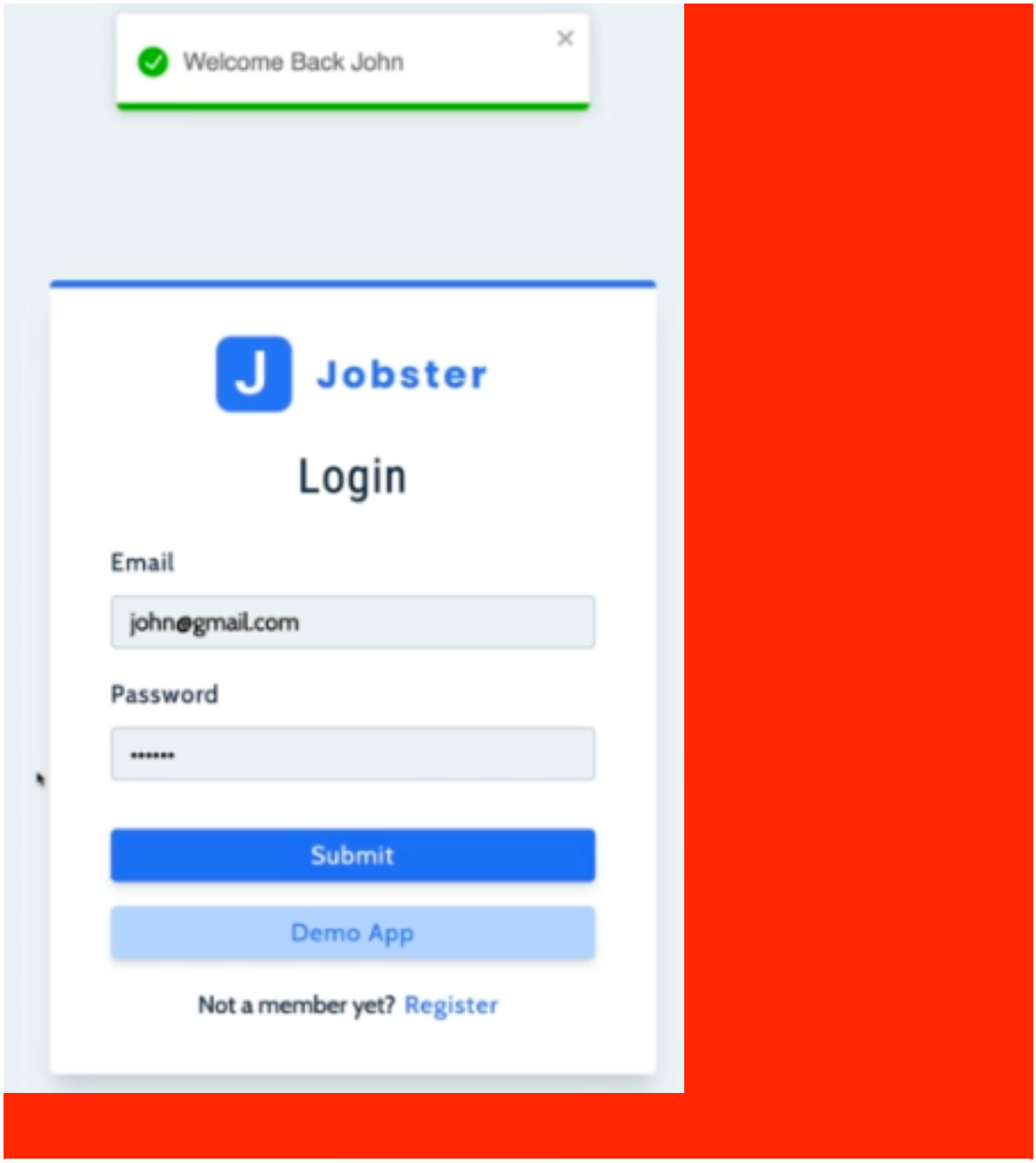
the server should respond as per front end requirement

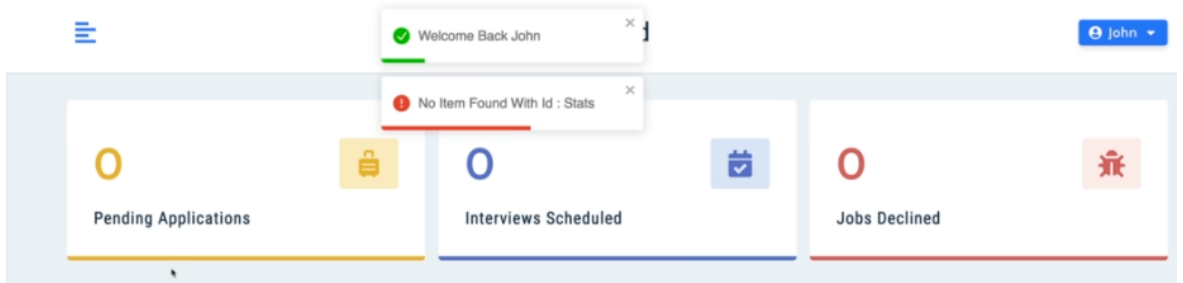


logout




Logging Out...
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register






```
const User = require('../models/User');
const { StatusCodes } = require('http-
```

```
status-codes');
const { BadRequestError,
UnauthenticatedError } = require('../
errors');

const register = async (req, res) => {
  const user = await
User.create({ ...req.body });
  const token = user.createJWT();

res.status(StatusCodes.CREATED).jso
n({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};

const login = async (req, res) => {
  const { email, password } = req.body;
```

```
  if (!email || !password) {
    throw new BadRequestError('Please
provide email and password');
  }
  const user = await
User.findOne({ email });
  if (!user) {
    throw new
UnauthenticatedError('Invalid
Credentials');
  }
  const isPasswordCorrect = await
user.comparePassword(password);
  if (!isPasswordCorrect) {
    throw new
UnauthenticatedError('Invalid
Credentials');
  }
  // compare password
  const token = user.createJWT();
  res.status(StatusCodes.OK).json({
    user: {
      email: user.email,
```

```
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};

module.exports = {
  register,
  login,
};
```

8 - Create Test User

==========

````
#### Create Test User

- test front-end request
- in postman or front-end
- make sure email and password are the same (or change the front-end)

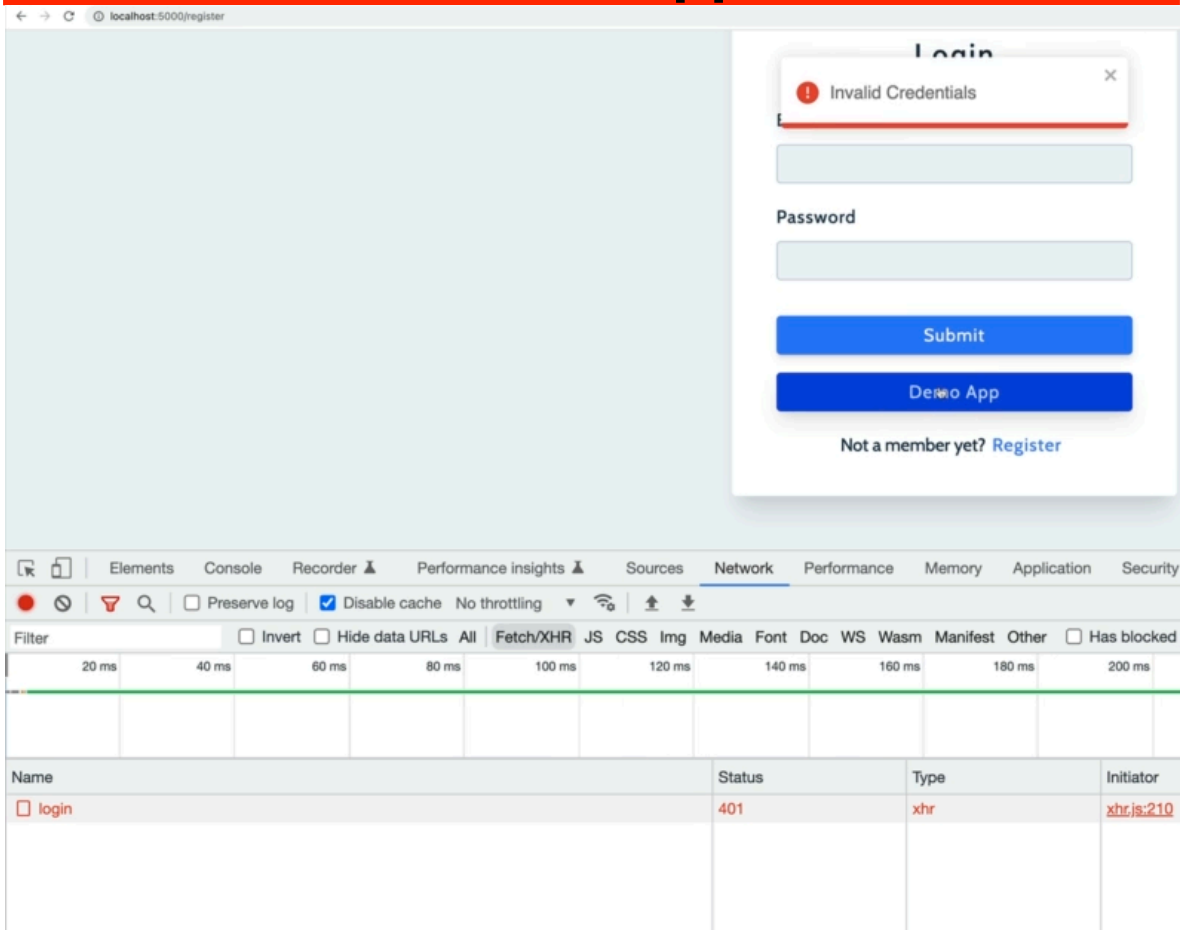
```js
{
    "name":"demo user",
    "email":"testUser@test.com",
    "password":"secret"
}
```

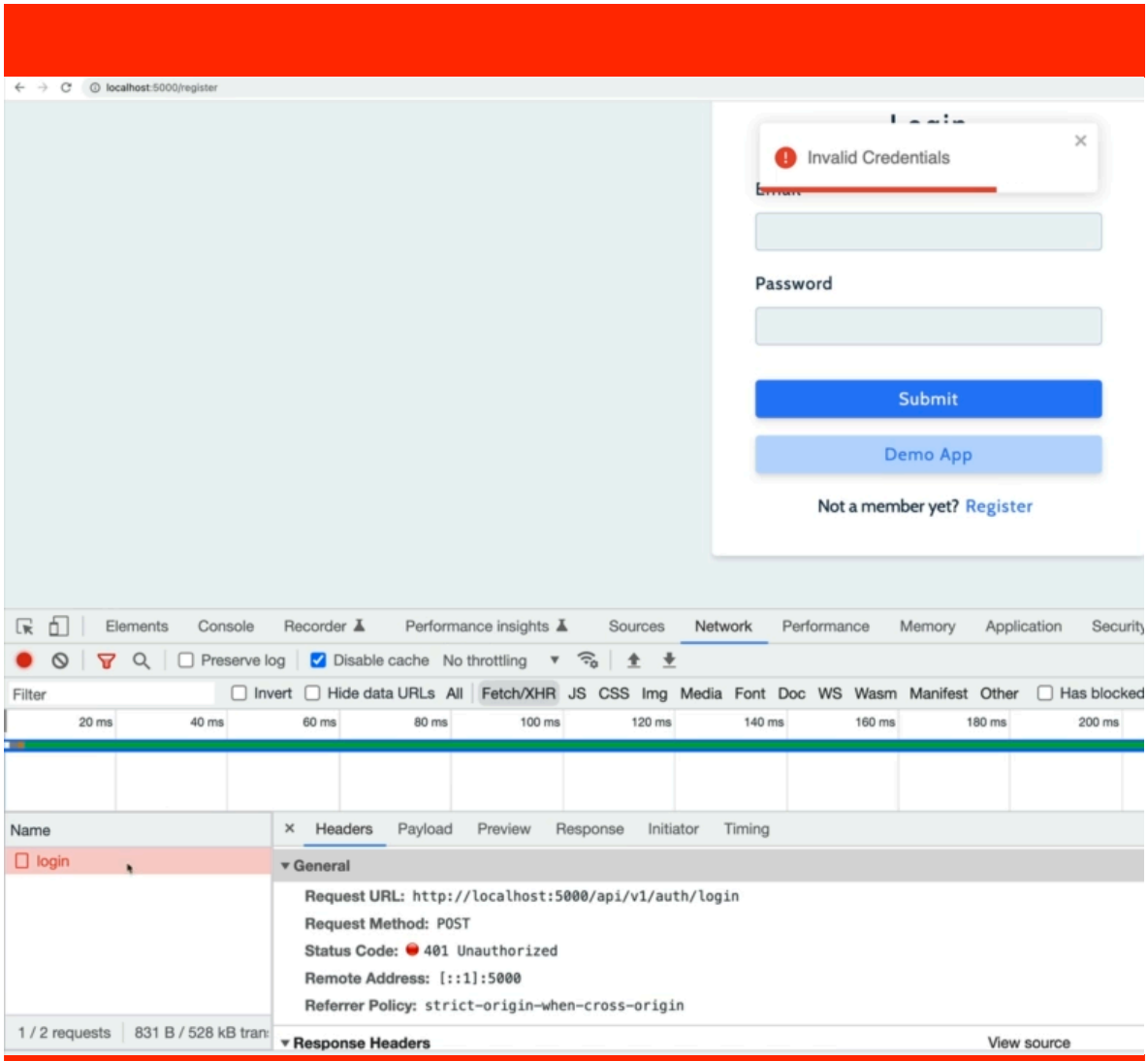
- navigate to client/src/pages/Register.js
- make sure email and password match your test user

```js
<button
  type='button'
  className='btn btn-block btn-hipster'
  disabled={isLoading}
  onClick={() =>
  dispatch(loginUser({ email: 'testUser@test.com', password: 'secret' }))}
    >

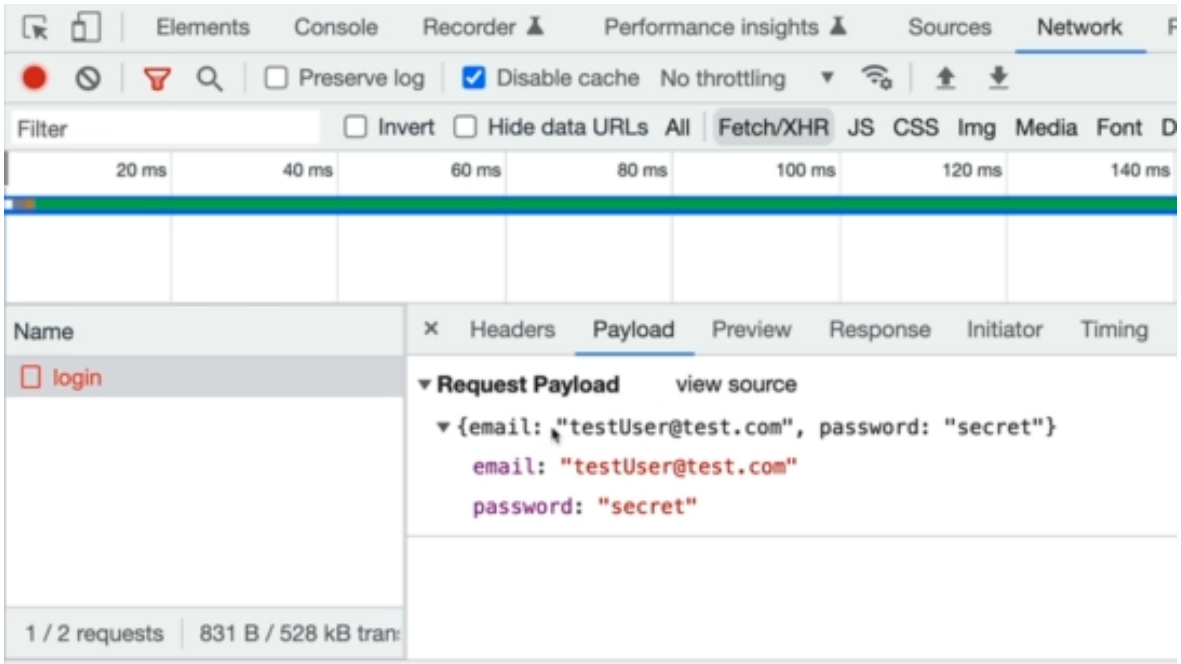
```
````

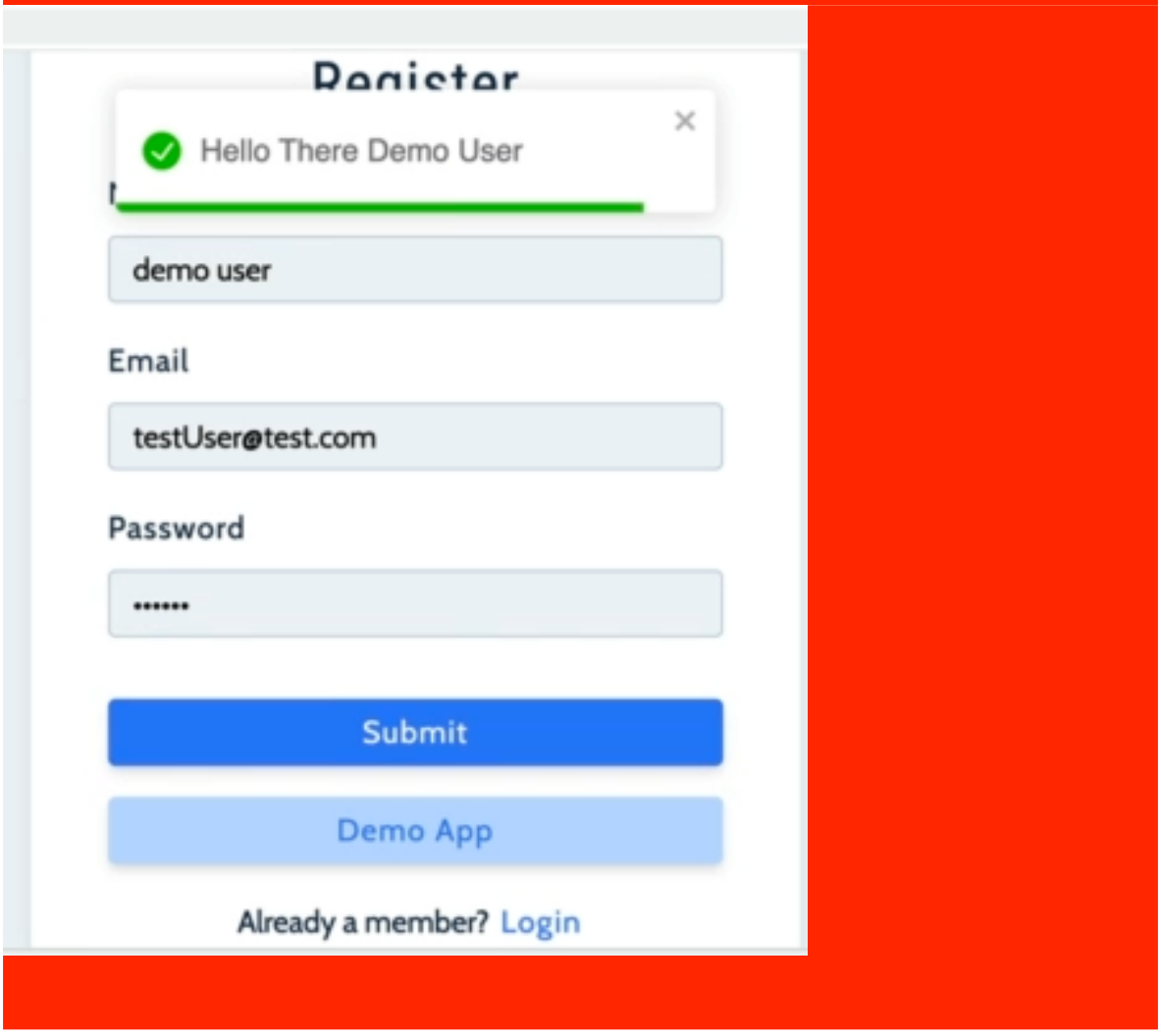
when we click Demo App

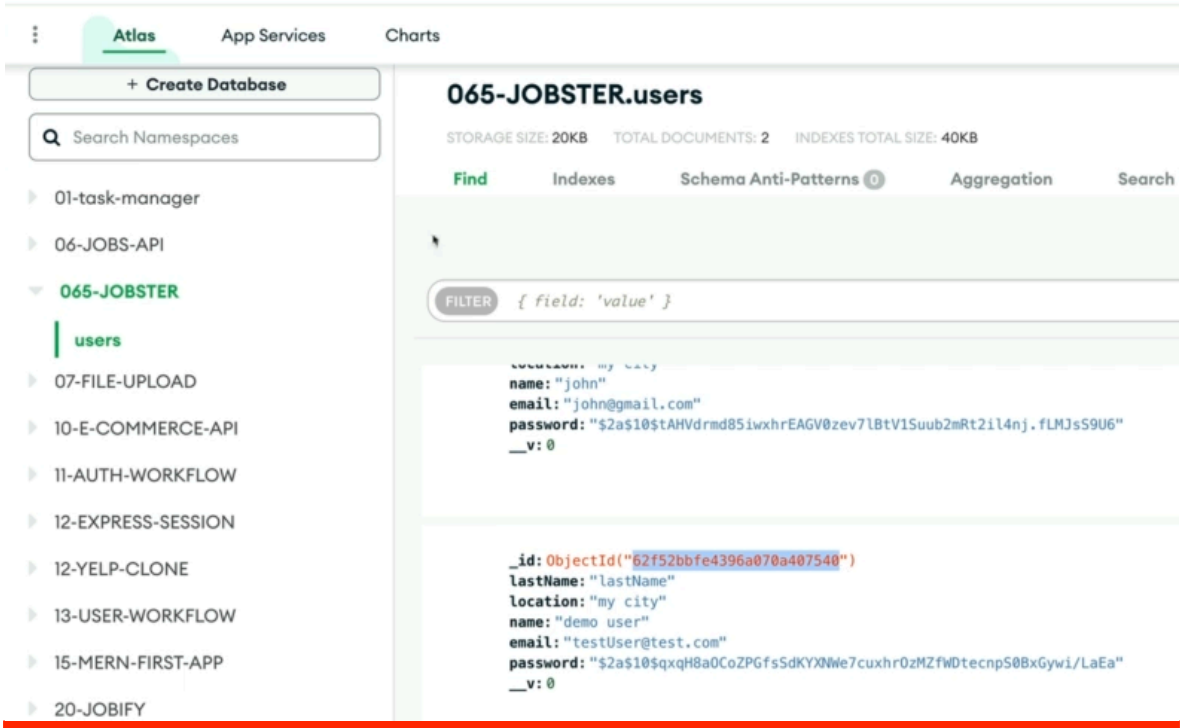


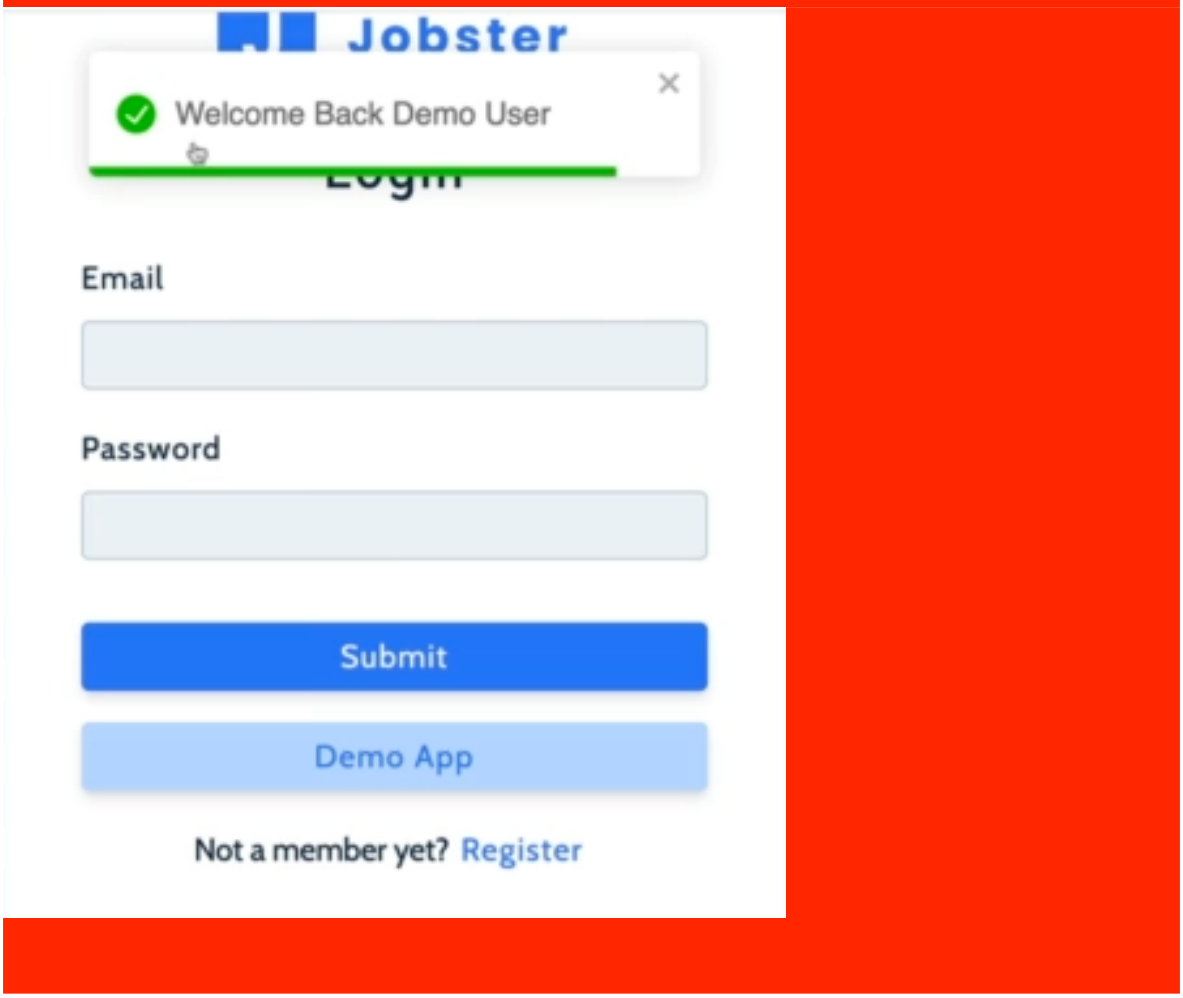


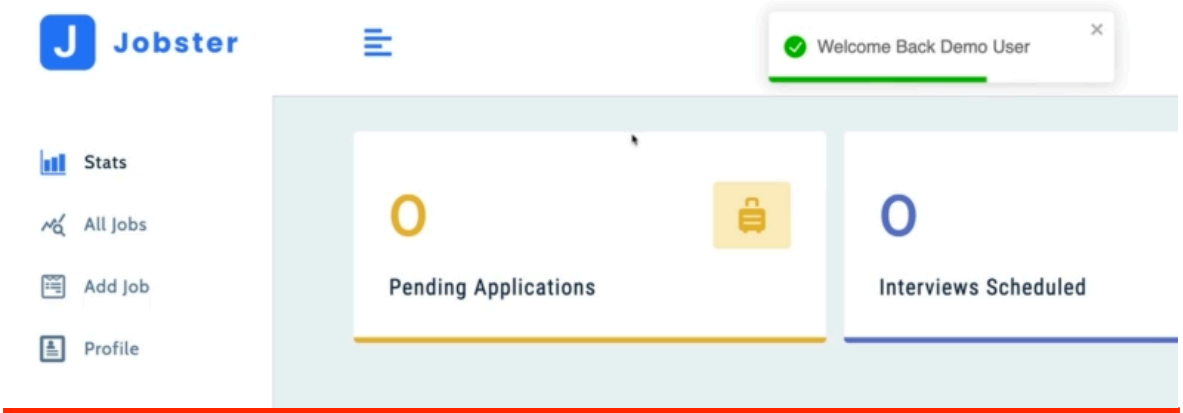
so we have to create this user in database and login with restrictions


Elements
Console
Recorder
Performance insights
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms
140 ms
Name
login
Headers
Payload
Preview
Response
Initiator
Timing
Request Payload
view source
{email: "testUser@test.com", password: "secret"}
email: "testUser@test.com"
password: "secret"
1 / 2 requests
831 B / 528 kB tran


Register
Hello There Demo User
demo user
Email
testUser@test.com
Password
••••••
Submit
Demo App
Already a member? Login


Atlas
App Services
Charts
+ Create Database
Search Namespaces
01-task-manager
06-JOBS-API
065-JOBSTER
users
07-FILE-UPLOAD
10-E-COMMERCE-API
11-AUTH-WORKFLOW
12-EXPRESS-SESSION
12-YELP-CLONE
13-USER-WORKFLOW
15-MERN-FIRST-APP
20-JOBIFY
065-JOBSTER.users
STORAGE SIZE: 20KB
TOTAL DOCUMENTS: 2
INDEXES TOTAL SIZE: 40KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search
FILTER
{ field: 'value' }
name: "john"
email: "john@gmail.com"
password: "$2a$10$tAHVdrmd85iwxhrEAGV0zev7lBtV1Suub2mRt2il4nj.fLMJsS9U6"
__v: 0
_id: ObjectId("62f52bbfe4396a070a407540")
lastName: "lastName"
location: "my city"
name: "demo user"
email: "testUser@test.com"
password: "$2a$10$qxqH8aOCoZPGfsSdKYXNWe7cuxhrOzMZfWDtecnpS0BxGywi/LaEa"
__v: 0


Jobster
Welcome Back Demo User
Email
Password
Submit
Demo App
Not a member yet? Register



9 - Update User - Setup

==========

````
#### Update User Functionality

- import authenticateUser middleware
- setup updateUser route (protected route)

routes/auth.js

```js
const express = require('express');
const router = express.Router();
const authenticateUser = require('../middleware/authentication');
const { register, login, updateUser } = require('../controllers/auth');
router.post('/register', register);
router.post('/login', login);
router.patch('/updateUser', authenticateUser, updateUser);

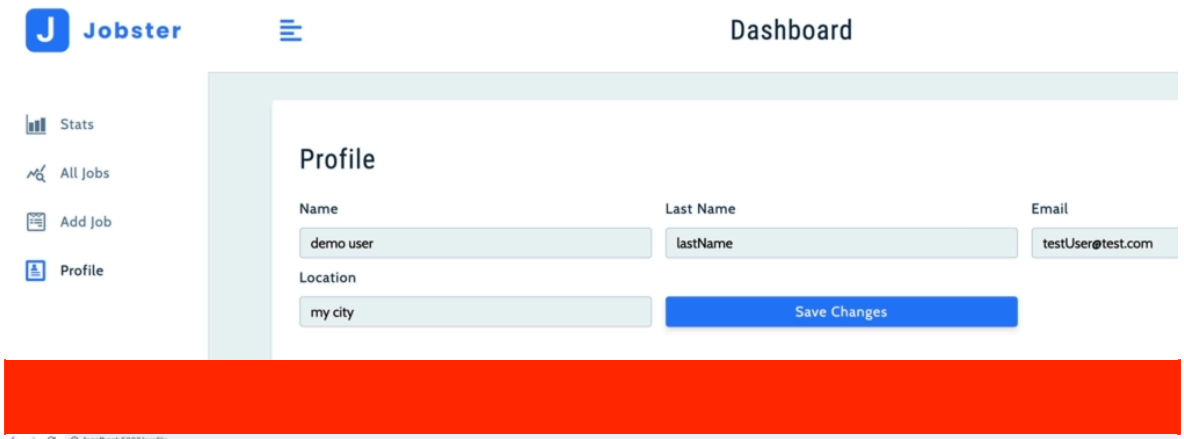
module.exports = router;
```

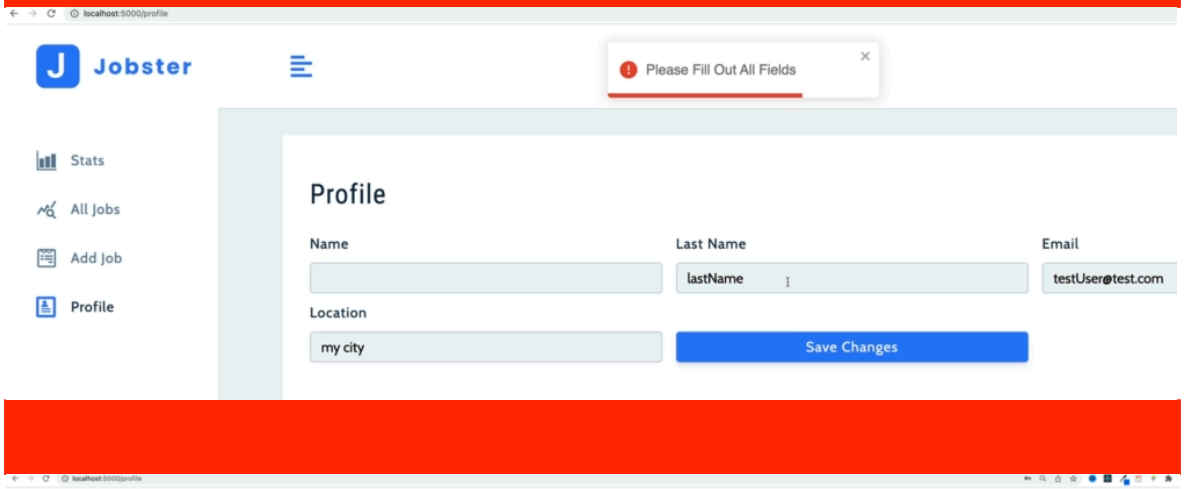
- create updateUser controller
- setup export

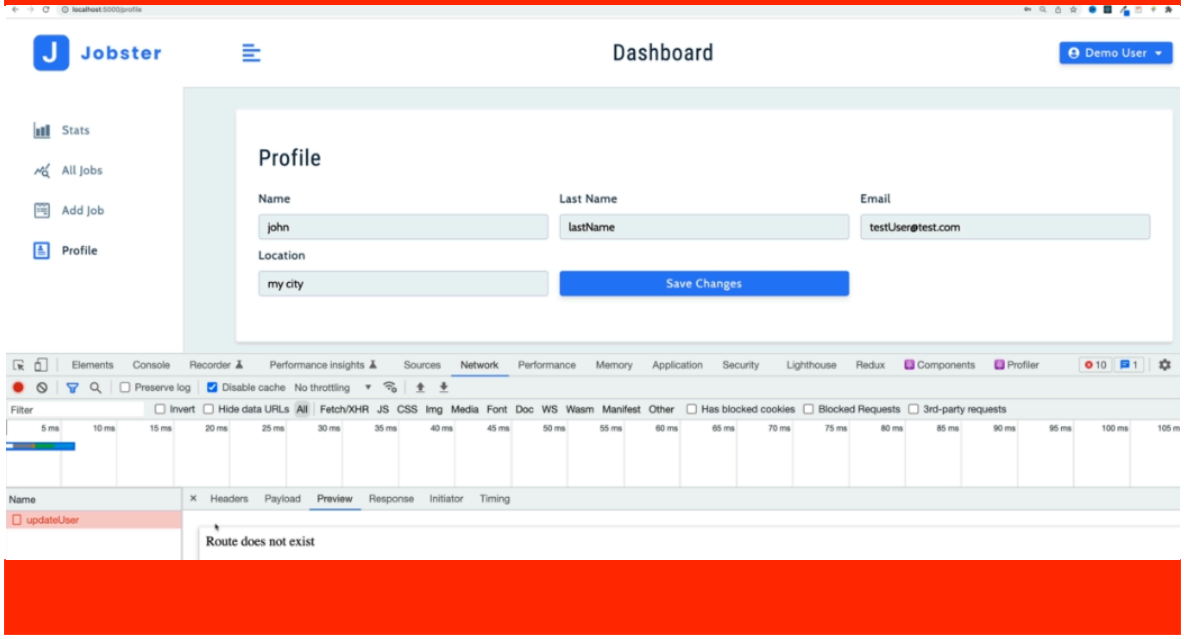
controllers/auth.js

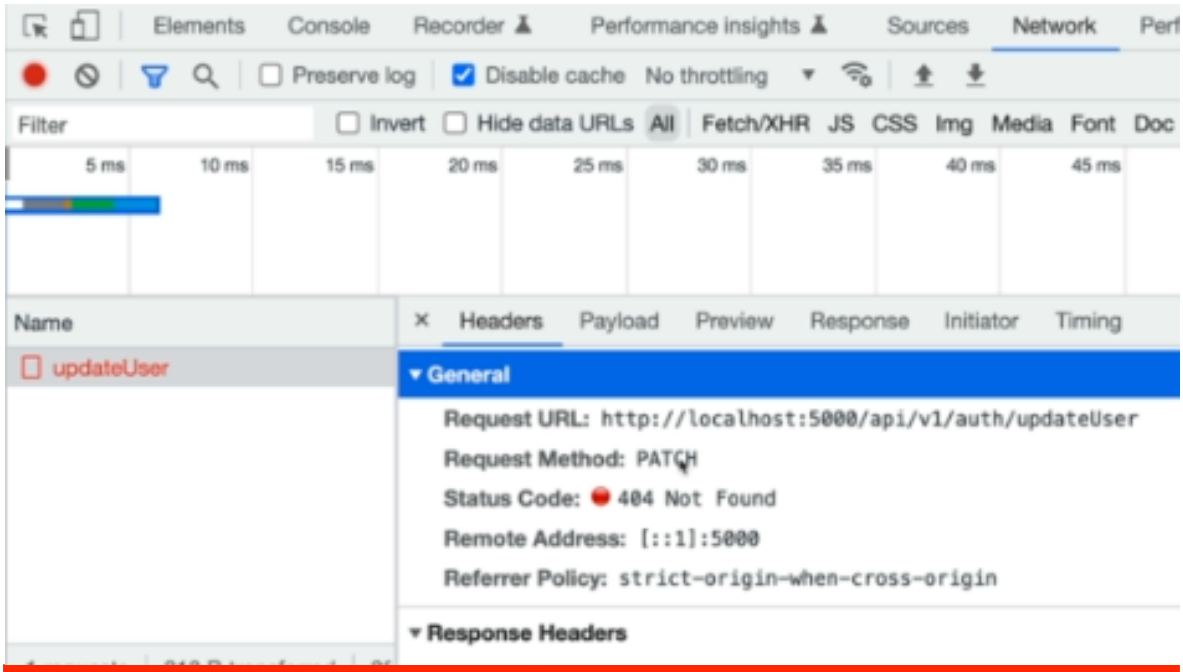
```js
const updateUser = async (req, res) => {
  console.log(req.user);
  console.log(req.body);
};
```
````

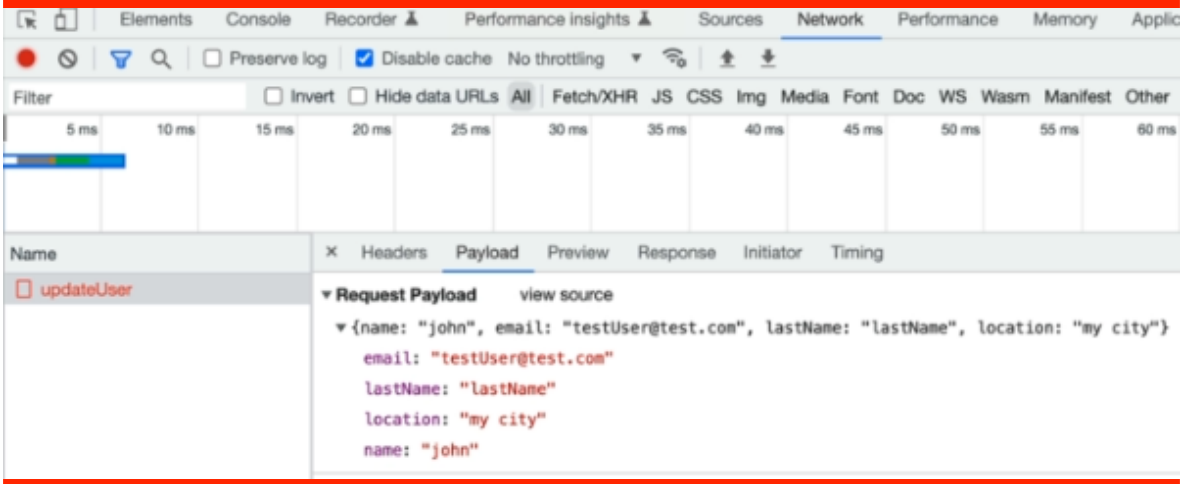
Profile Page


Jobster
Dashboard
Stats
All Jobs
Add Job
Profile
Profile
Name
demo user
Last Name
lastName
Email
testUser@test.com
Location
my city
Save Changes

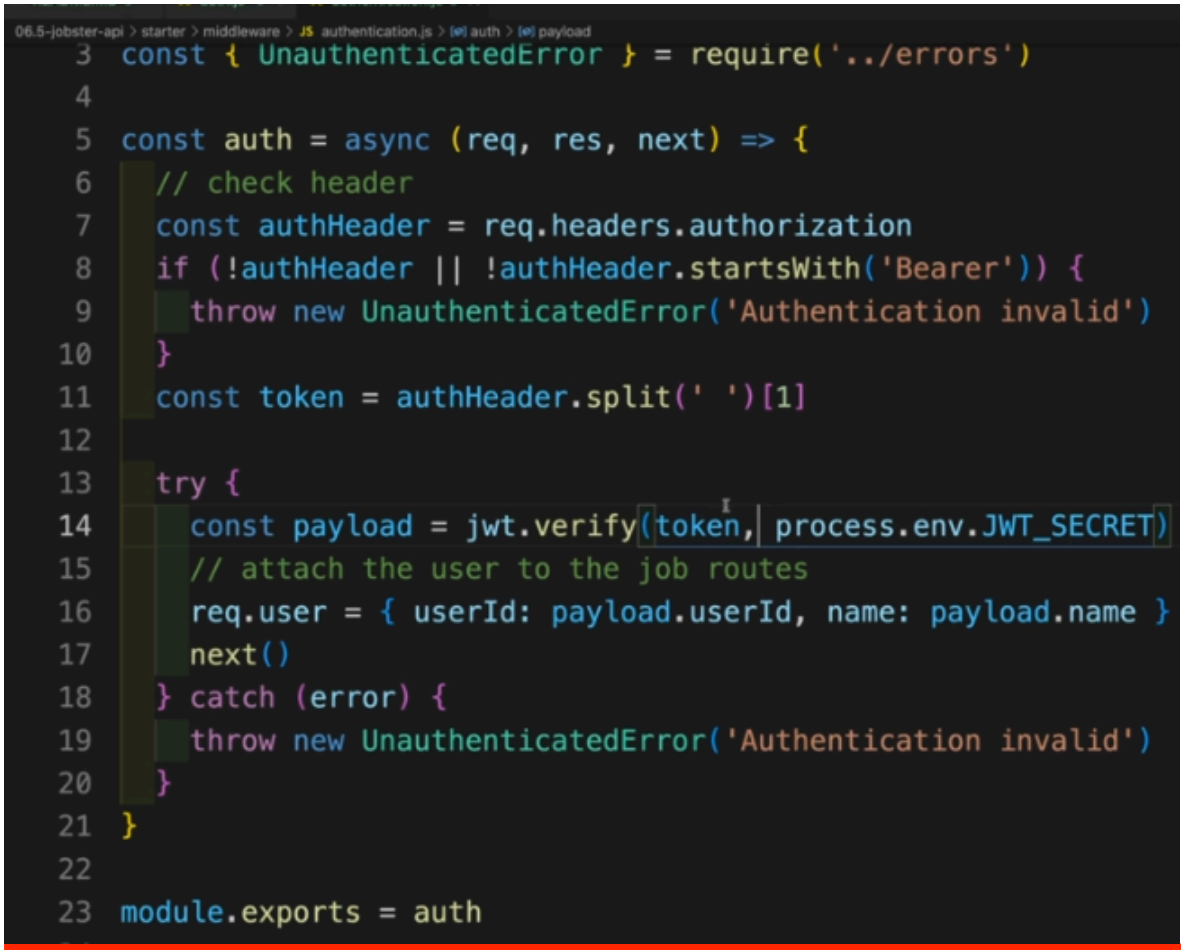

localhost:5000/profile
Jobster
Please Fill Out All Fields
Stats
All Jobs
Add Job
Profile
Profile
Name
Last Name
lastName
Email
testUser@test.com
Location
my city
Save Changes


Jobster
Dashboard
Demo User
Stats
All Jobs
Add Job
Profile
Profile
Name
john
Last Name
lastName
Email
testUser@test.com
Location
my city
Save Changes
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Redux
Components
Profiler
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
Name
updateUser
Headers
Payload
Preview
Response
Initiator
Timing
Route does not exist

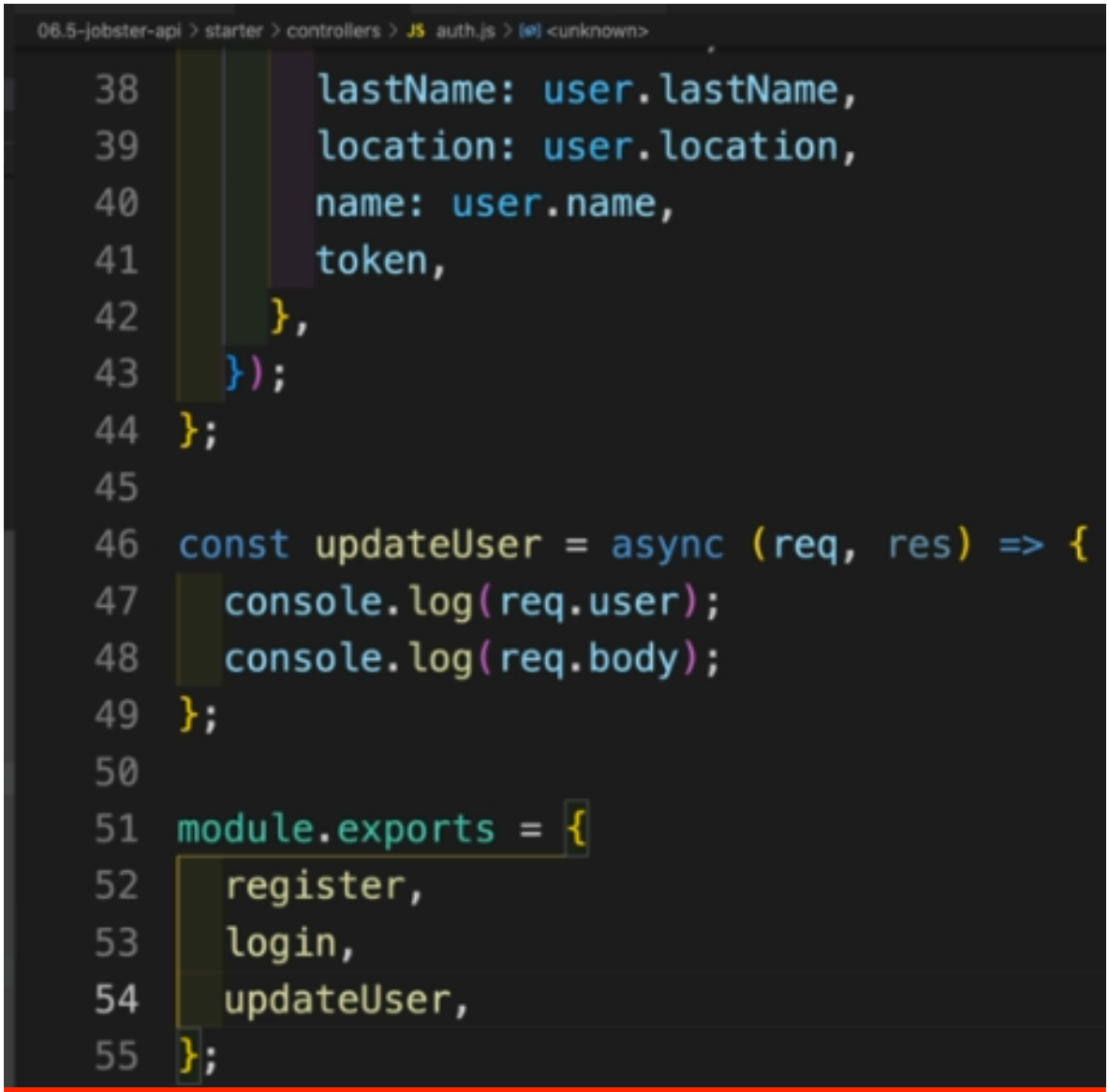




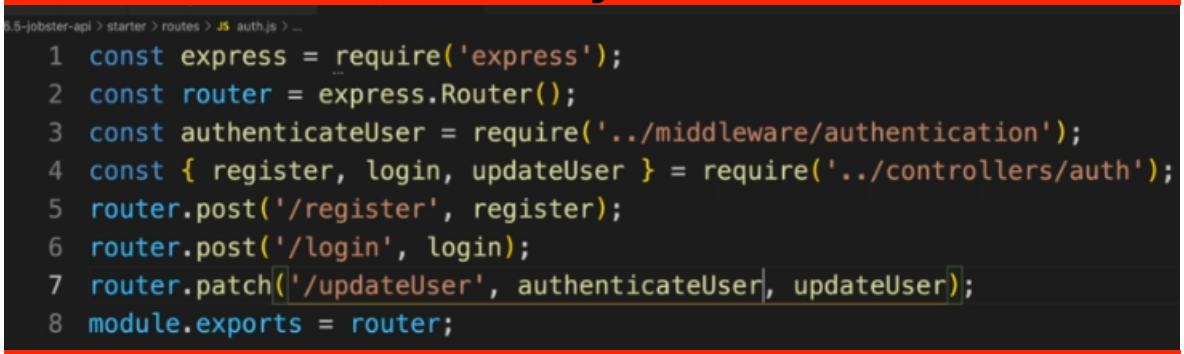
we are getting user from middleware authentication.js file



/controllers/auth.js

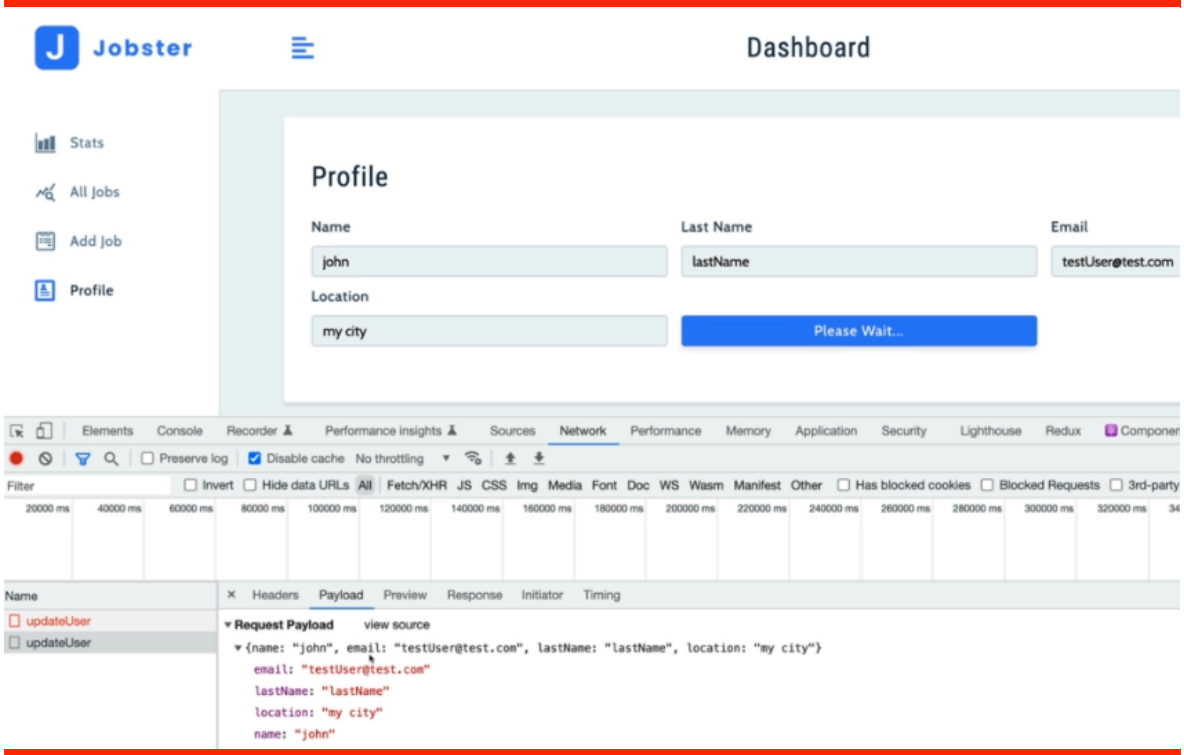


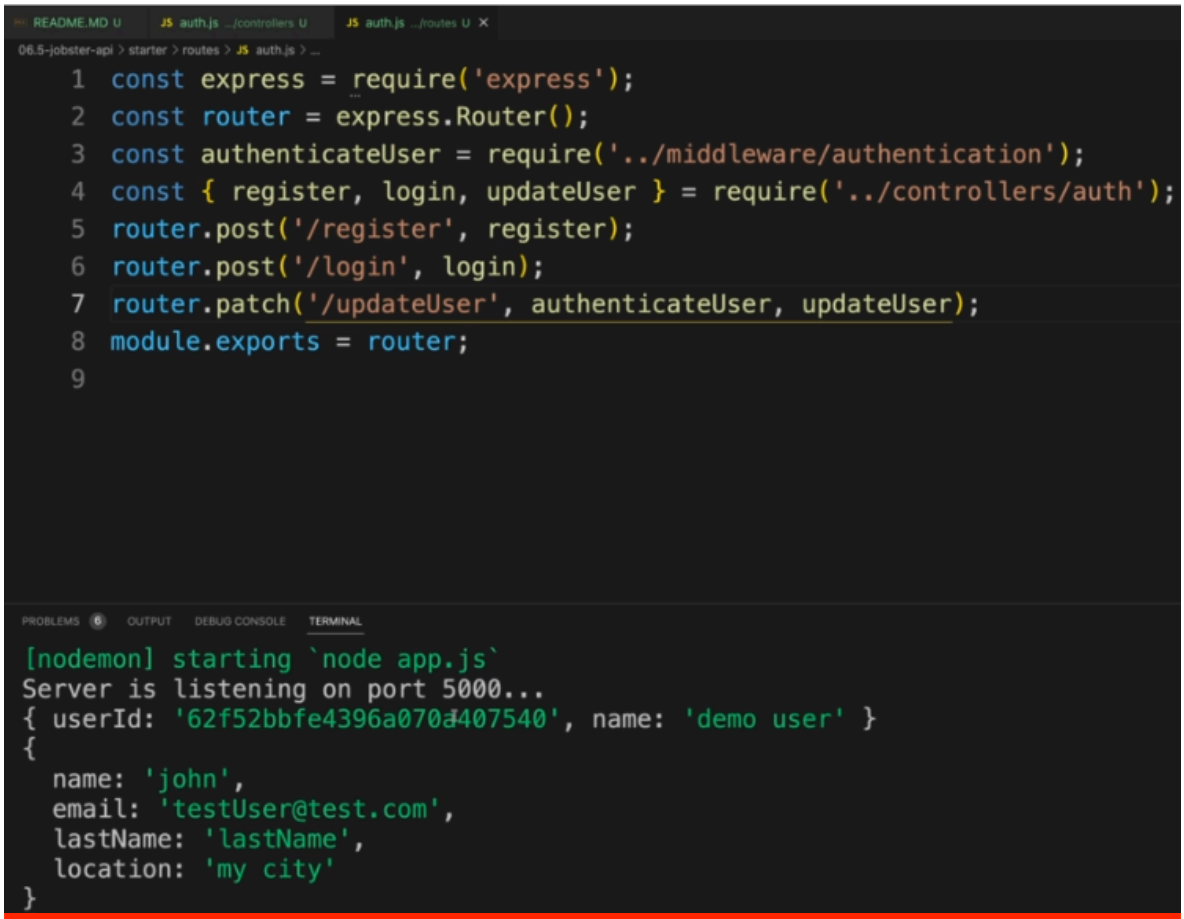
/starter/routes/auth.js



const express = require('express');

```
const router = express.Router();
const authenticateUser = require('../
middleware/authentication');
const { register, login, updateUser } =
require('../controllers/auth');
router.post('/register', register);
router.post('/login',  login);
router.patch('/updateUser',
authenticateUser,  updateUser);
module.exports = router;
```





10 - Update User (Complete)

==========

````
- complete updateUser

controllers/auth.js

```js
const updateUser = async (req, res) => {
  const { email, name, lastName, location } = req.body;
  if (!email || !name || !lastName || !location) {
    throw new BadRequestError('Please provide all values');
  }
  const user = await User.findOne({ _id: req.user.userId });

  user.email = email;
  user.name = name;
  user.lastName = lastName;
  user.location = location;

  await user.save();

  const token = user.createJWT();

  res.status(StatusCodes.OK).json({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};
```
````

/controllers/auth.js

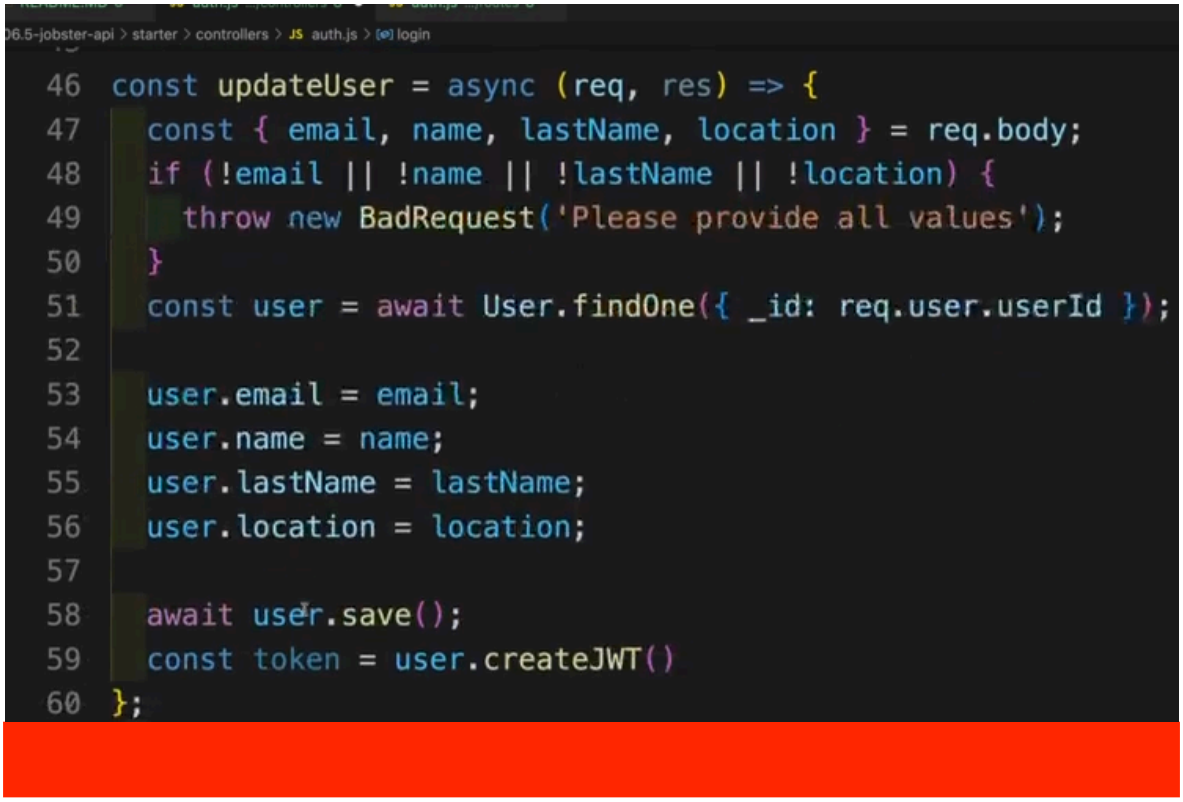
==========

```
const updateUser = async (req, res) =>
{
  const { email, name, lastName,
location } = req.body;
  if (!email || !name || !lastName || !
location) {
```

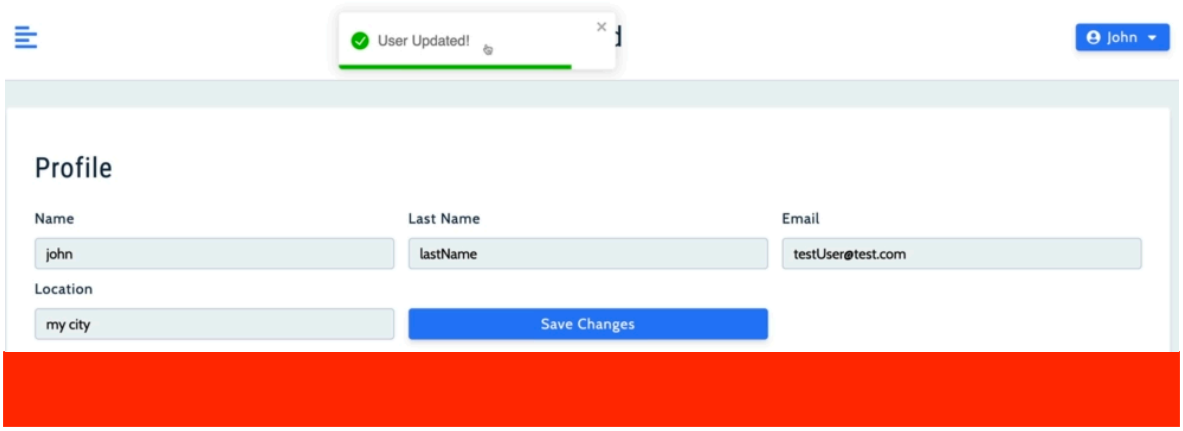
```
    throw new BadRequest('Please
provide all values');
  }
  const user = await User.findOne({ _id:
req.user.userId });

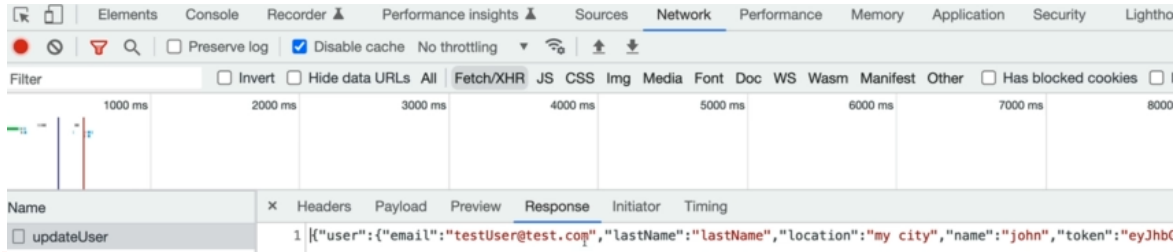
  user.email = email;
  user.name = name;
  user.lastName = lastName;
  user.location = location;

  await user.save();
  const token = user.createJWT();
  res.status(StatusCodes.OK).json({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};
```


06.5-jobster-api > starter > controllers > JS auth.js > login
46 const updateUser = async (req, res) => {
47 const { email, name, lastName, location } = req.body;
48 if (!email || !name || !lastName || !location) {
49 throw new BadRequest('Please provide all values');
50 }
51 const user = await User.findOne({ _id: req.user.userId });
52
53 user.email = email;
54 user.name = name;
55 user.lastName = lastName;
56 user.location = location;
57
58 await user.save();
59 const token = user.createJWT()
60 };

```
06.5-jobster-api > starter > controllers > JS auth.js > updateUser
54    user.name = name;
55    user.lastName = lastName;
56    user.location = location;
57
58    await user.save();
59    const token = user.createJWT();
60    res.status(StatusCodes.OK).json({
61      user: {
62        email: user.email,
63        lastName: user.lastName,
64        location: user.location,
65        name: user.name,
66        token,
67      },
68    });
69  };
```





/controllers/auth.js

==========

```
const User = require('../models/User');
const { StatusCodes } = require('http-status-codes');
const { BadRequestError, UnauthenticatedError } = require('../errors');

const register = async (req, res) => {
  const user = await User.create({ ...req.body });
  const token = user.createJWT();

res.status(StatusCodes.CREATED).json({
```

```
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};

const login = async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    throw new BadRequestError('Please
provide email and password');
  }
  const user = await
User.findOne({ email });
  if (!user) {
    throw new
UnauthenticatedError('Invalid
Credentials');
  }
```

```
  const isPasswordCorrect = await
user.comparePassword(password);
  if (!isPasswordCorrect) {
    throw new
UnauthenticatedError('Invalid
Credentials');
  }
  // compare password
  const token = user.createJWT();
  res.status(StatusCodes.OK).json({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
};

const updateUser = async (req, res) =>
{
  const { email, name, lastName,
location } = req.body;
```

```
  if (!email || !name || !lastName || !
location) {
    throw new BadRequest('Please
provide all values');
  }
  const user = await User.findOne({ _id:
req.user.userId });

  user.email = email;
  user.name = name;
  user.lastName = lastName;
  user.location = location;

  await user.save();
  const token = user.createJWT();
  res.status(StatusCodes.OK).json({
    user: {
      email: user.email,
      lastName: user.lastName,
      location: user.location,
      name: user.name,
      token,
    },
  });
```

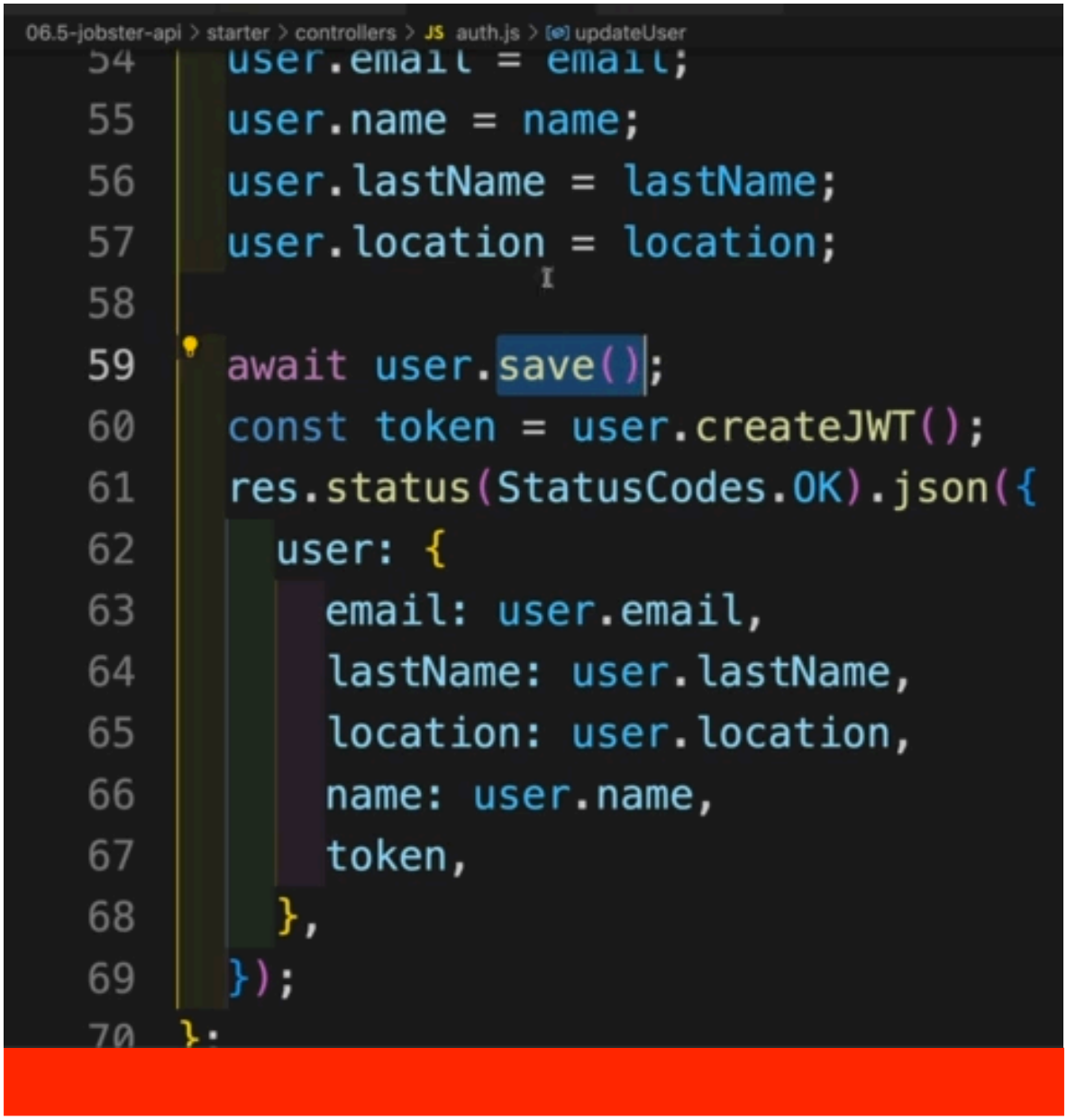
```
};

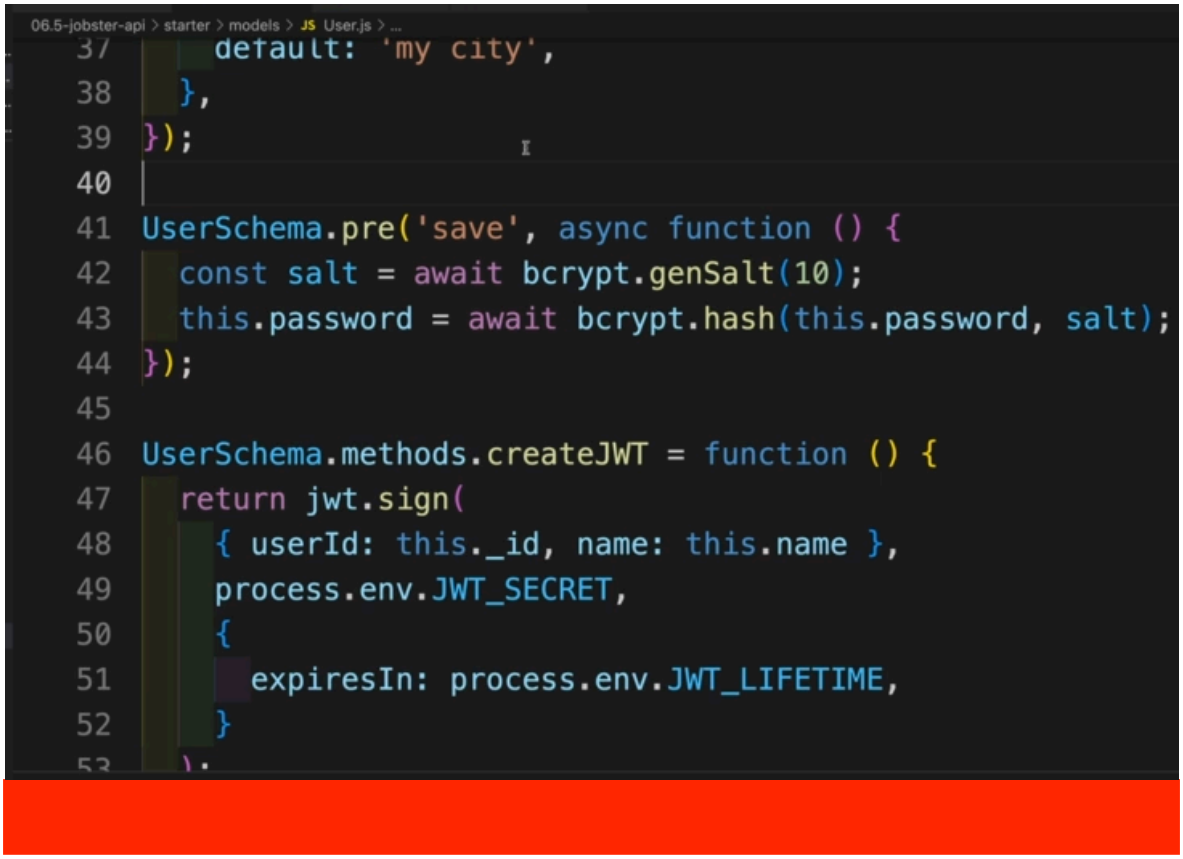
module.exports = {
  register,
  login,
  updateUser,
};
```

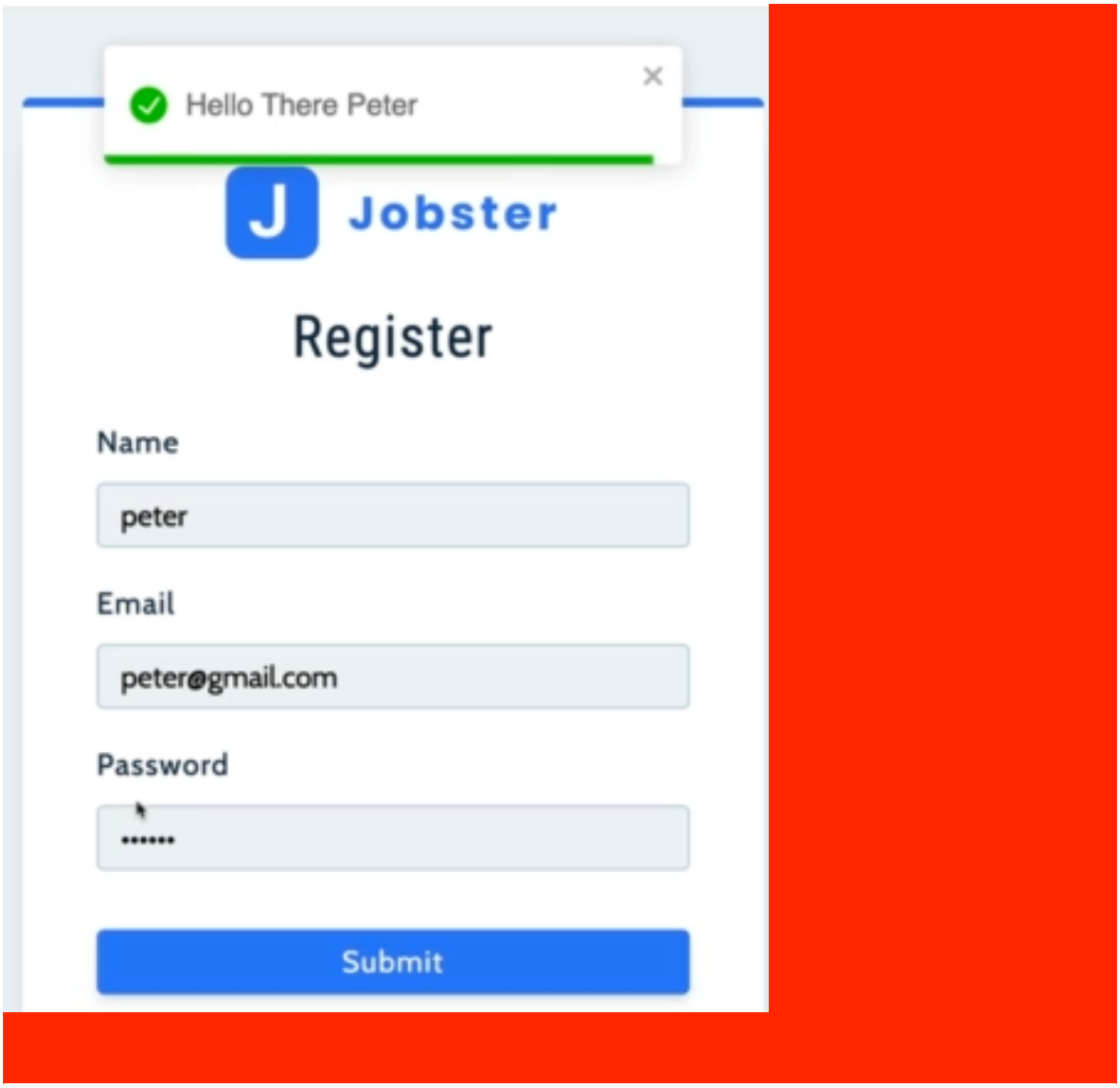
11 - Password Gotcha

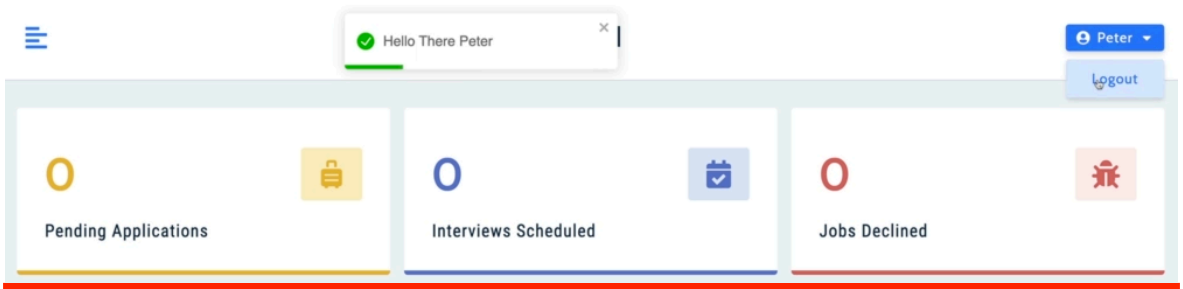
==========

when we update, original password is not working, becuase of user.save( )

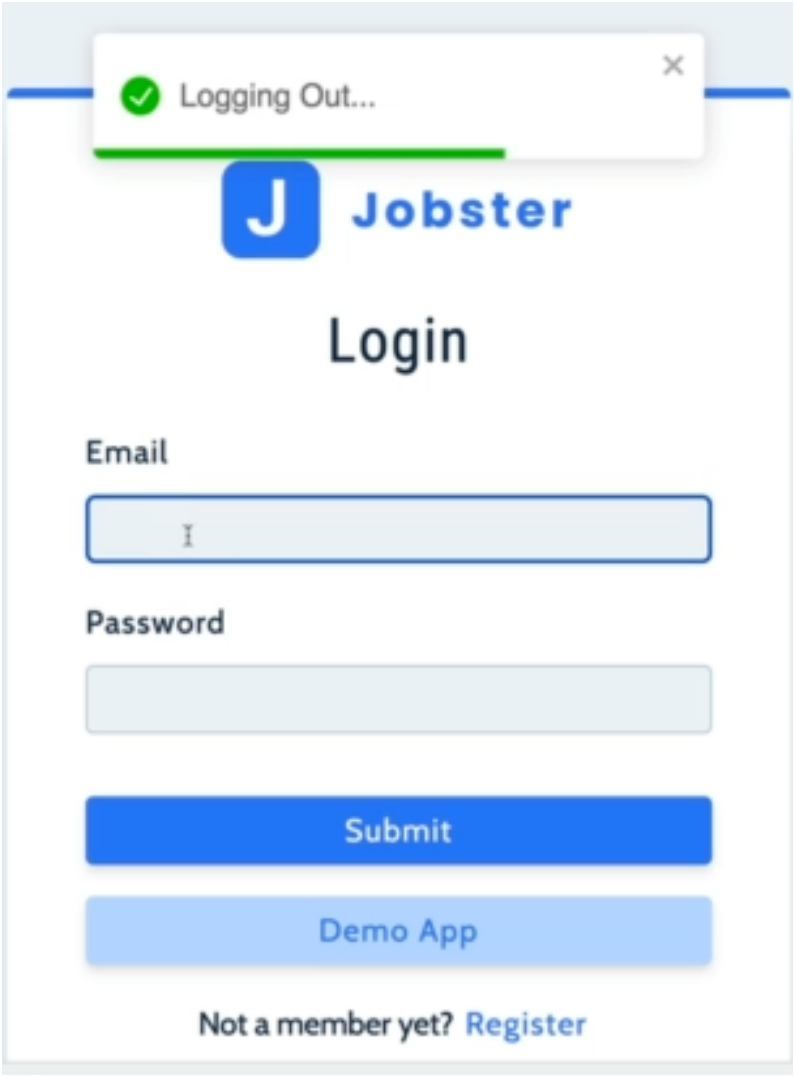

06.5-jobster-api > starter > controllers > JS auth.js > updateUser
user.email = email;
user.name = name;
user.lastName = lastName;
user.location = location;
await user.save();
const token = user.createJWT();
res.status(StatusCodes.OK).json({
user: {
email: user.email,
lastName: user.lastName,
location: user.location,
name: user.name,
token,
},
});

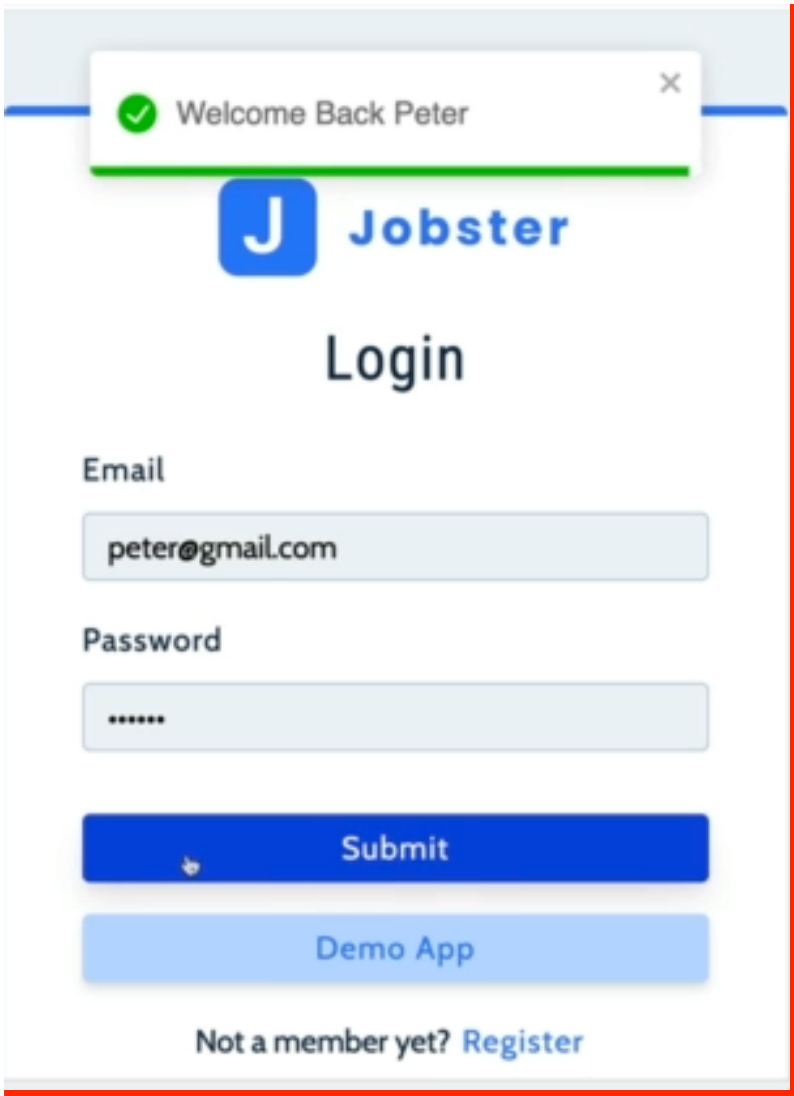

06.5-jobster-api > starter > models > JS User.js > ...
default: 'my city',
},
});
UserSchema.pre('save', async function () {
const salt = await bcrypt.genSalt(10);
this.password = await bcrypt.hash(this.password, salt);
});
UserSchema.methods.createJWT = function () {
return jwt.sign(
{ userId: this._id, name: this.name },
process.env.JWT_SECRET,
{
expiresIn: process.env.JWT_LIFETIME,
}
),

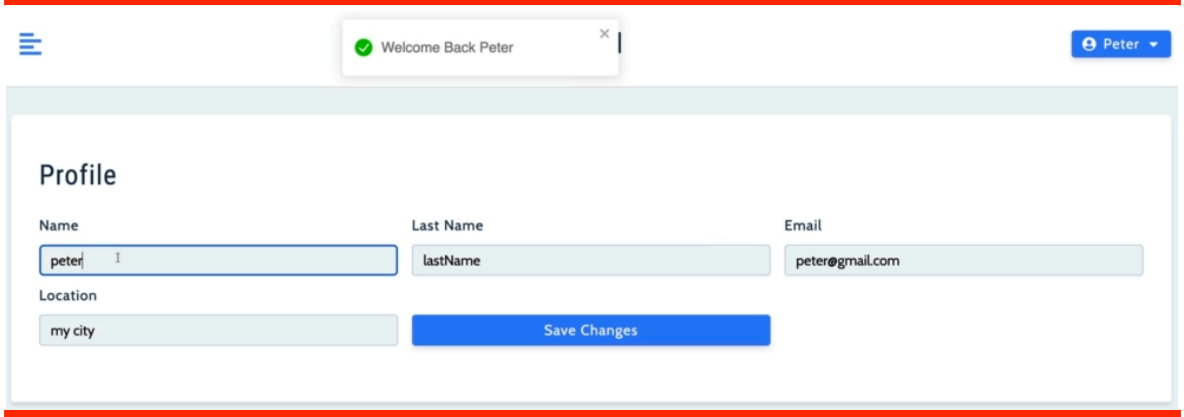




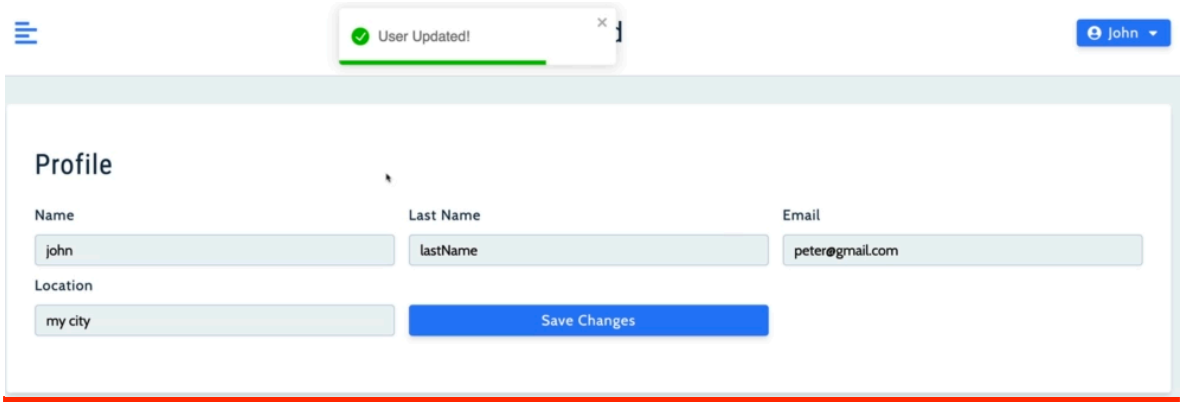
log out and log in one more time


Logging Out...
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register

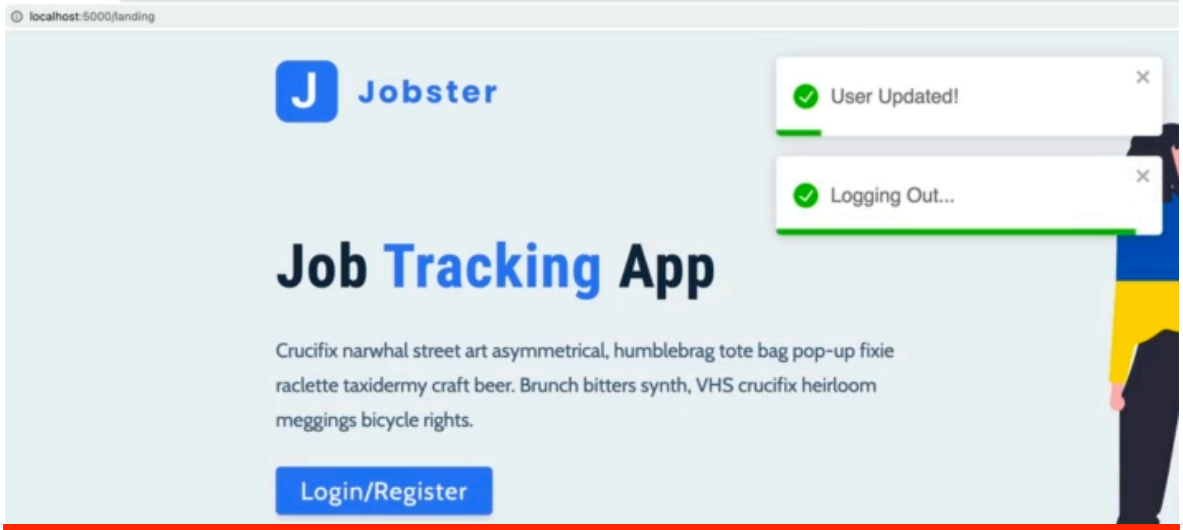




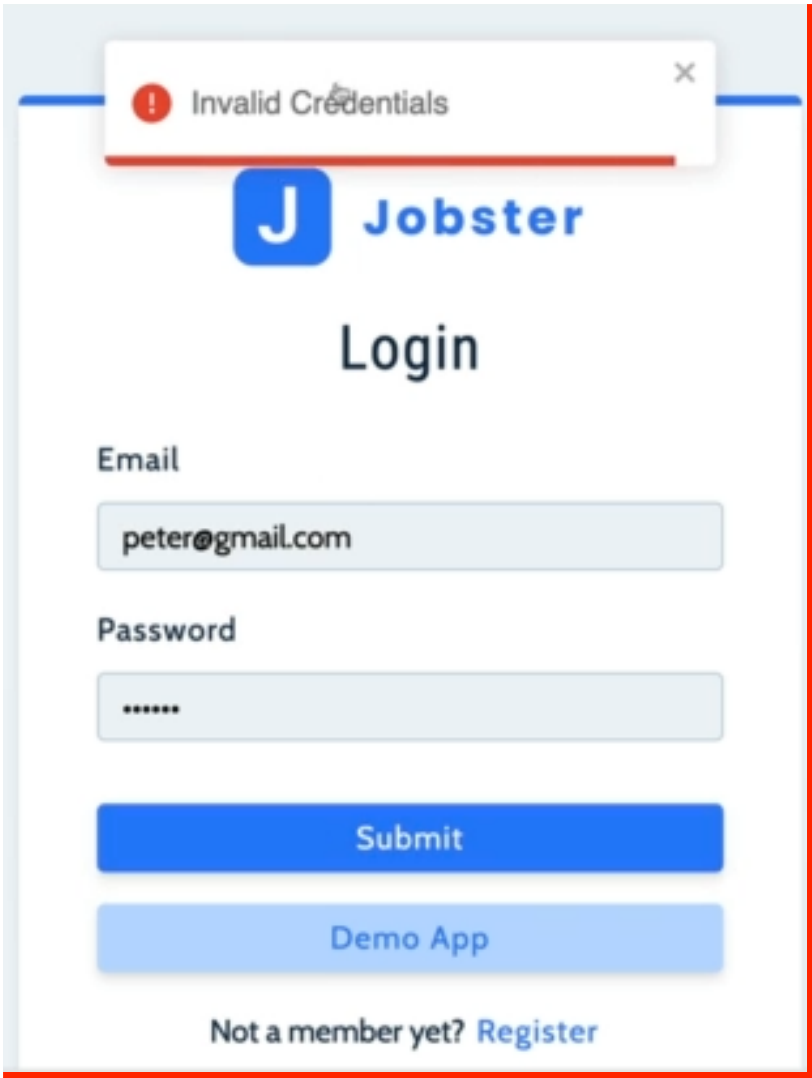
if we change name to john



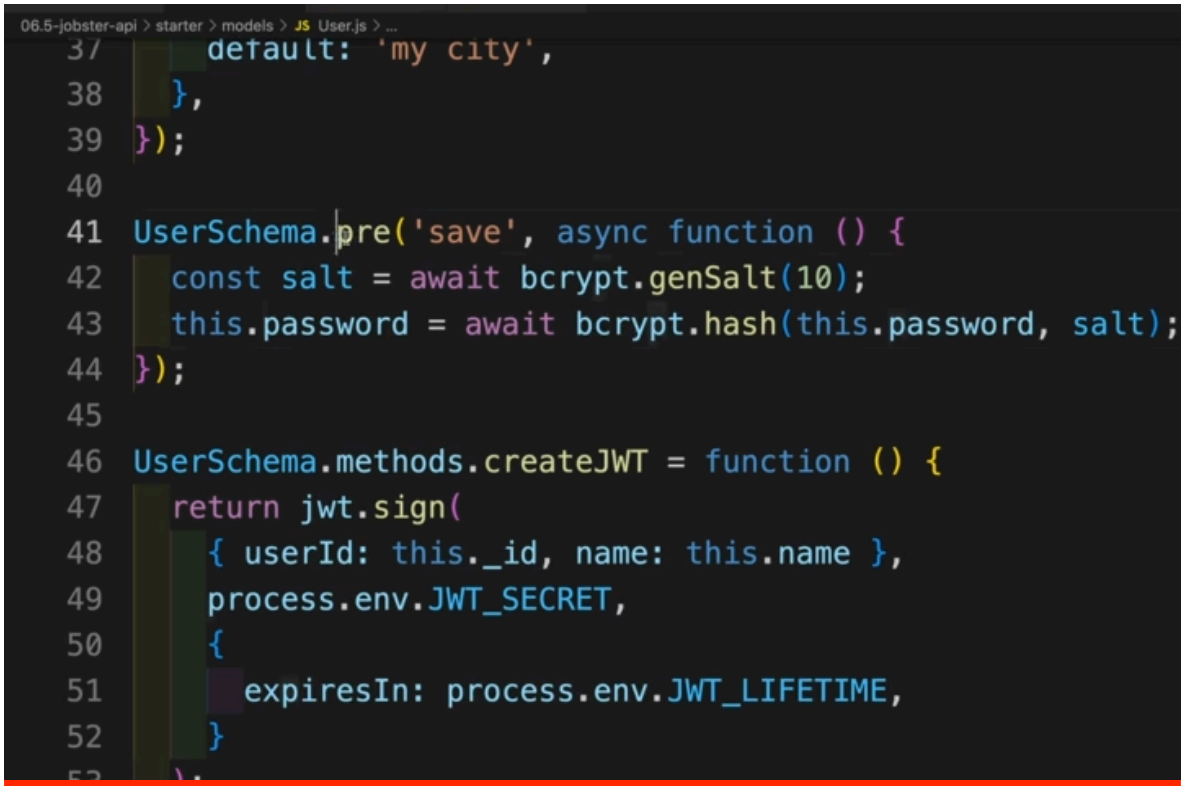
logout

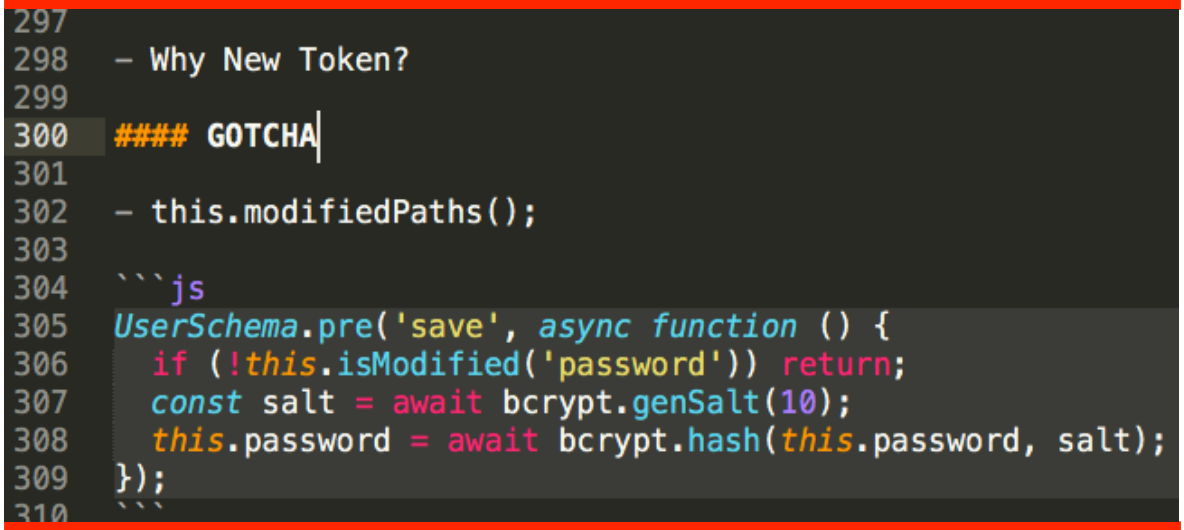


we can not login no more



becuase when we update, we call save, so it was hashed on more time

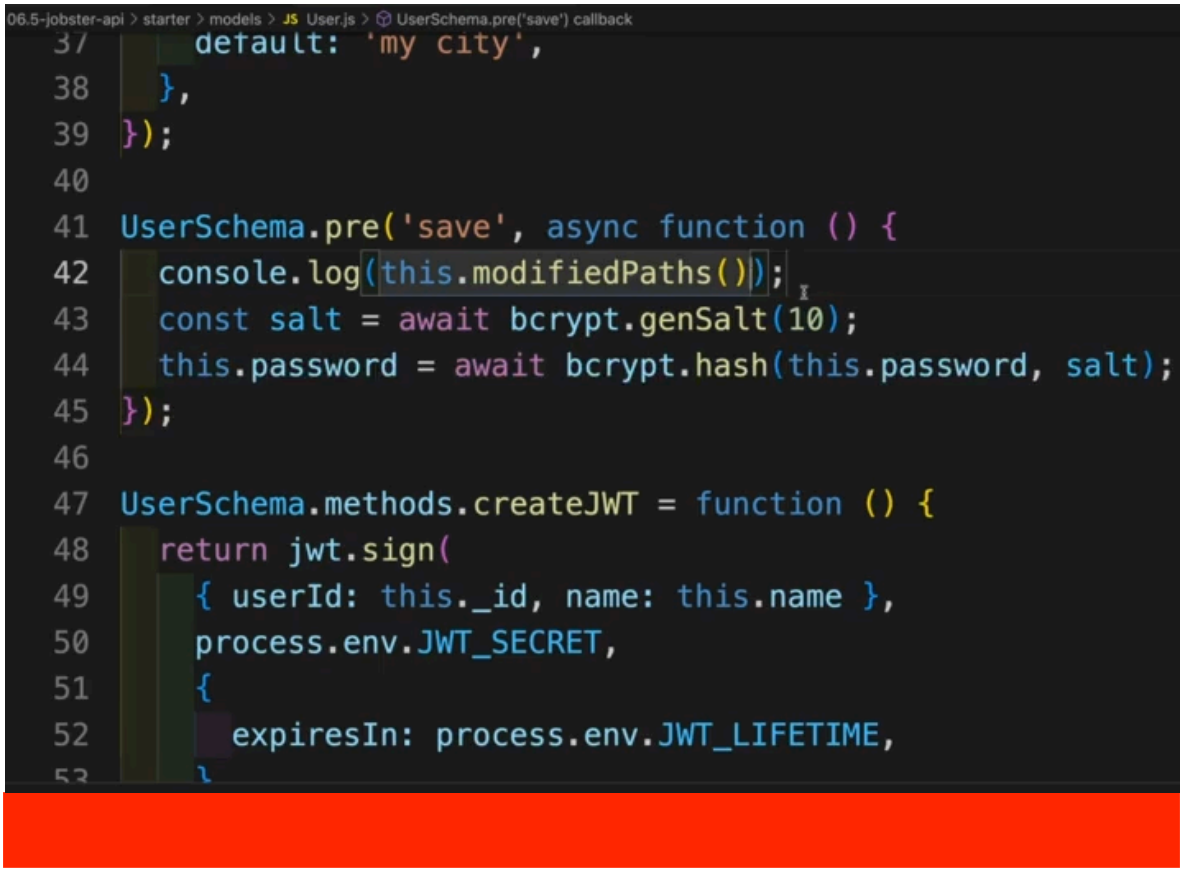


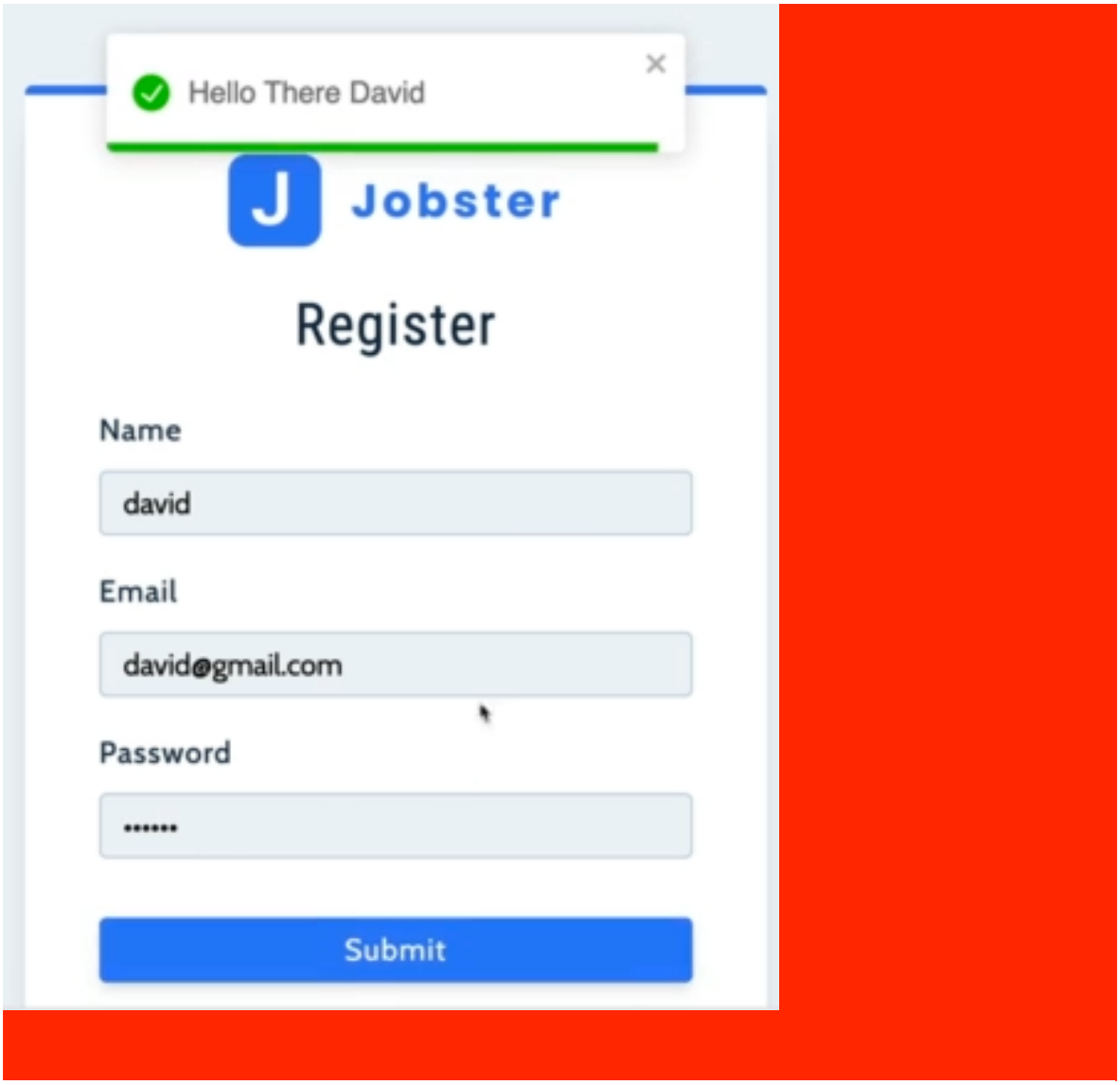


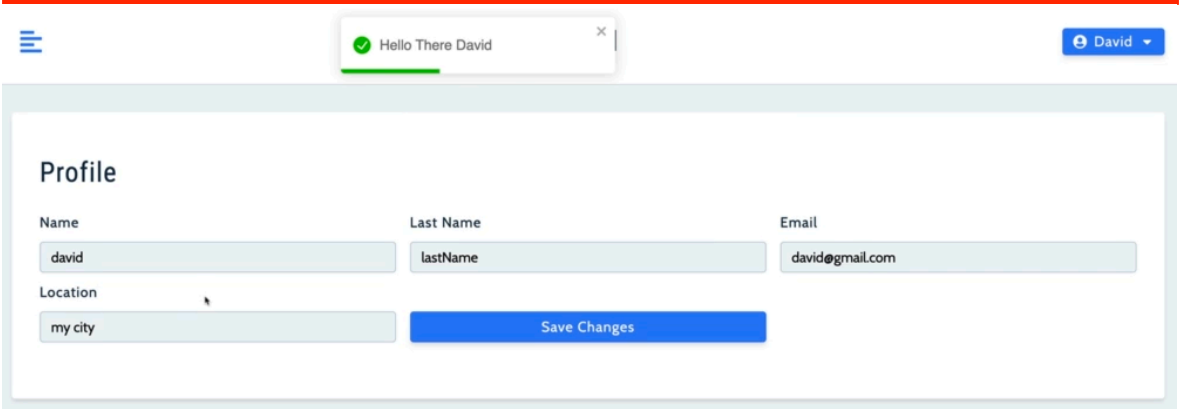
/models/User.js

==========

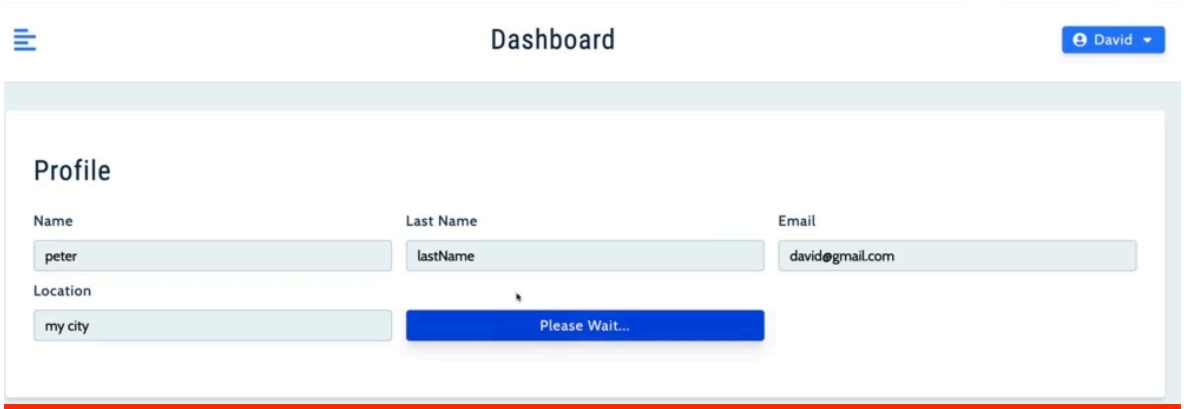
if (!this.isModified('password')) return;


06.5-jobster-api > starter > models > JS User.js > UserSchema.pre('save') callback
default: 'my city',
},
});
UserSchema.pre('save', async function () {
console.log(this.modifiedPaths());
const salt = await bcrypt.genSalt(10);
this.password = await bcrypt.hash(this.password, salt);
});
UserSchema.methods.createJWT = function () {
return jwt.sign(
{ userId: this._id, name: this.name },
process.env.JWT_SECRET,
{
expiresIn: process.env.JWT_LIFETIME,





change name to peter



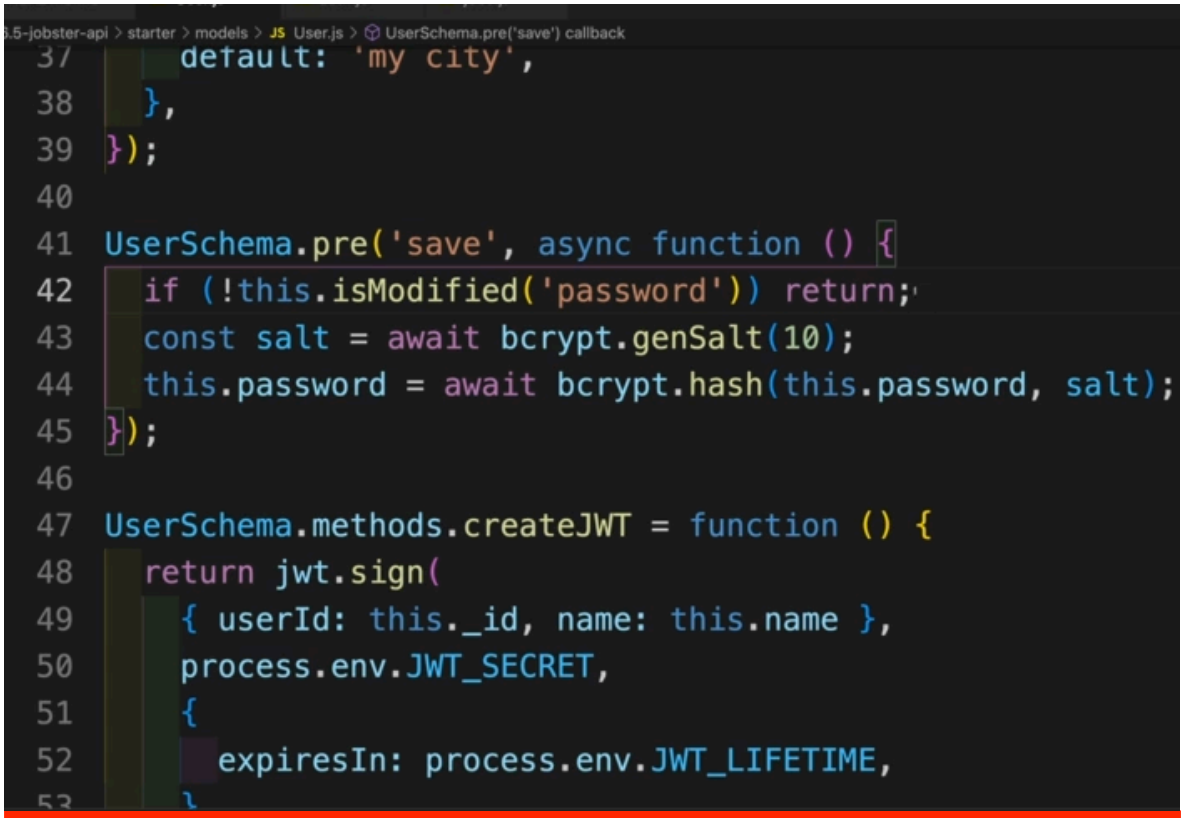
README.MD | User.js | auth.js | jobs.js

06.5-jobster-api > starter > models > User.js > ...

```
37      default: 'my city',
38    },
39  });
40
41  UserSchema.pre('save', async function () {
42    console.log(this.modifiedPaths());
43    const salt = await bcrypt.genSalt(10);
44    this.password = await bcrypt.hash(this.password, salt);
45  });
46
47  UserSchema.methods.createJWT = function () {
48    return jwt.sign(
49      { userId: this._id, name: this.name },
50      process.env.JWT_SECRET,
51      {
52        expiresIn: process.env.JWT_LIFETIME,
53      }
```

PROBLEMS 11 | OUTPUT | DEBUG CONSOLE | TERMINAL

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Server is listening on port 5000...
[ 'name', 'email', 'password' ]
{ userId: '62f949b8122692388e022065', testUser: false }
[ 'name' ]
```




```
const mongoose =
require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const UserSchema = new
mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Please provide
name'],
    maxlength: 50,
```

```
    minlength: 3,
  },
  email: {
    type: String,
    required: [true, 'Please provide
email'],
    match: [
      /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:
\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]
{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-
Z\-0-9]+\.)+[a-zA-Z]{2,}))$/,
      'Please provide a valid email',
    ],
    unique: true,
  },
  password: {
    type: String,
    required: [true, 'Please provide
password'],
    minlength: 6,
  },
  lastName: {
    type: String,
    trim: true,
```

```
    maxlength: 20,
    default: 'lastName',
  },
  location: {
    type: String,
    trim: true,
    maxlength: 20,
    default: 'my city',
  },
});

UserSchema.pre('save', async function
() {
  if (!this.isModified('password')) return;
  const salt = await bcrypt.genSalt(10);
  this.password = await
bcrypt.hash(this.password, salt);
});

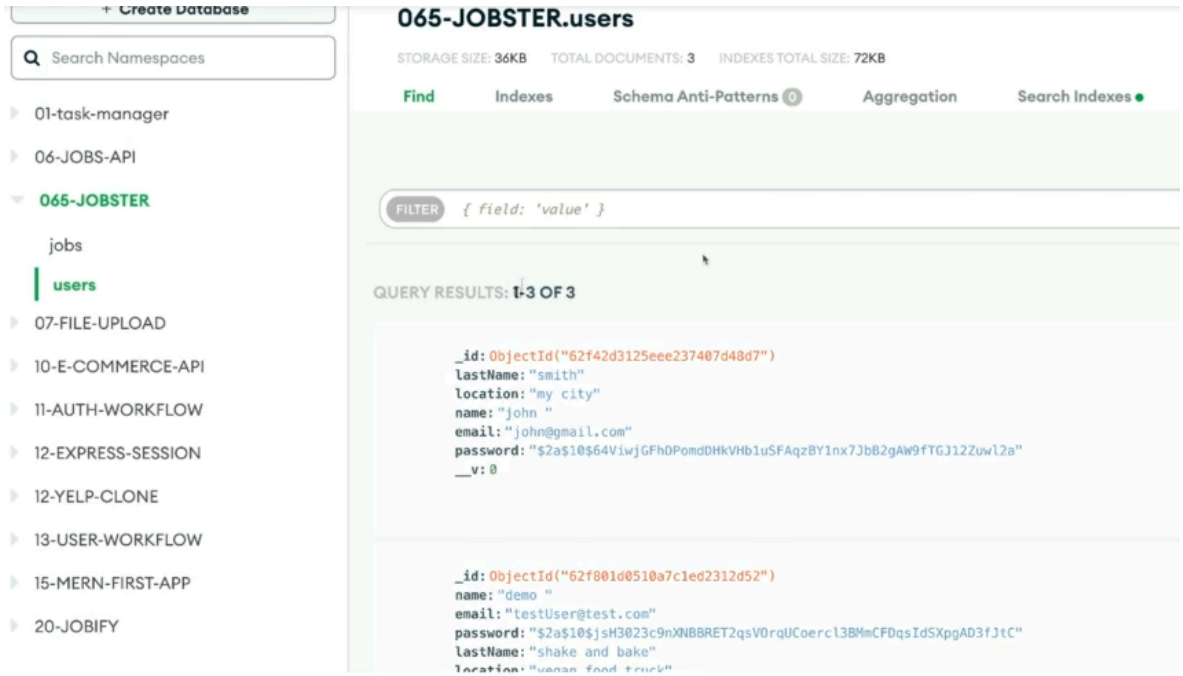
UserSchema.methods.createJWT =
function () {
  return jwt.sign(
    { userId: this._id, name: this.name },
    process.env.JWT_SECRET,
```

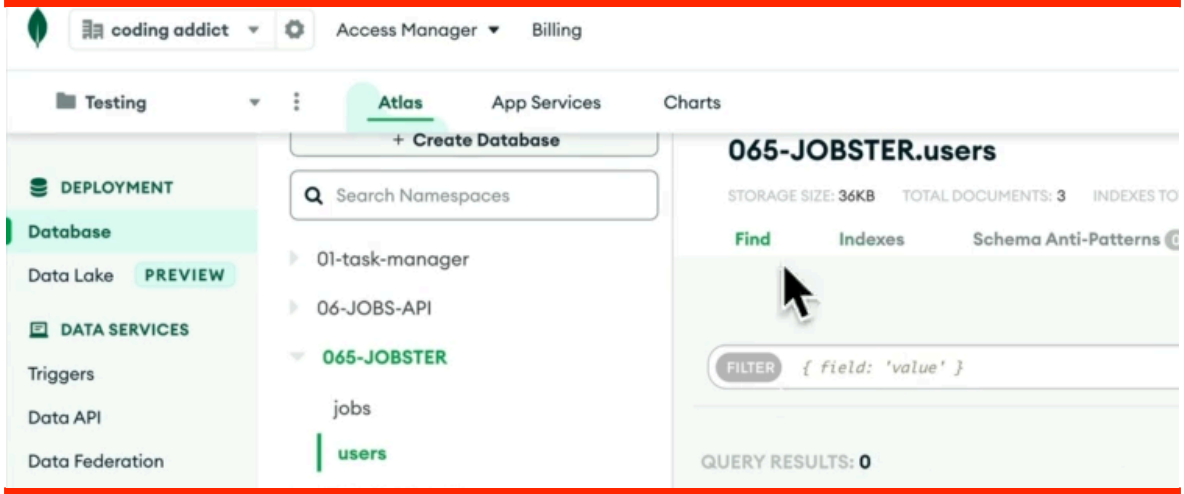
```
    {
      expiresIn:
process.env.JWT_LIFETIME,
    }
  );
};

UserSchema.methods.comparePasswo
rd = async function
(canditatePassword) {
  const isMatch = await
bcrypt.compare(canditatePassword,
this.password);
  return isMatch;
};

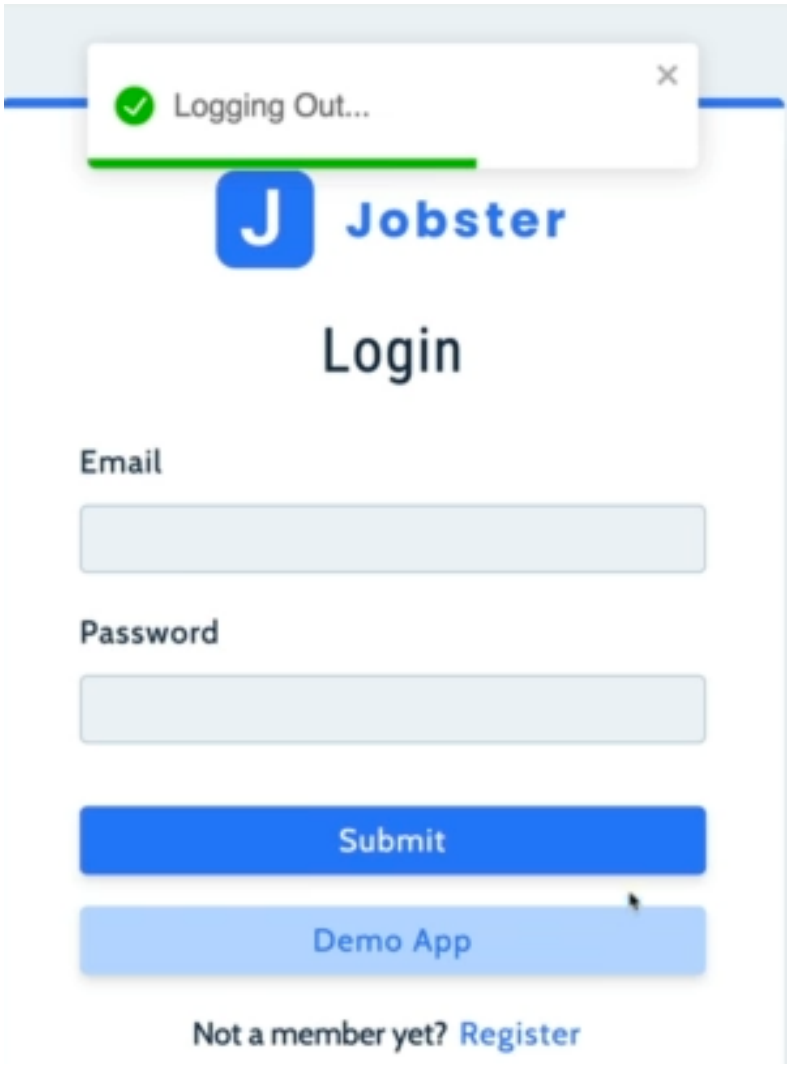
module.exports =
mongoose.model('User', UserSchema);

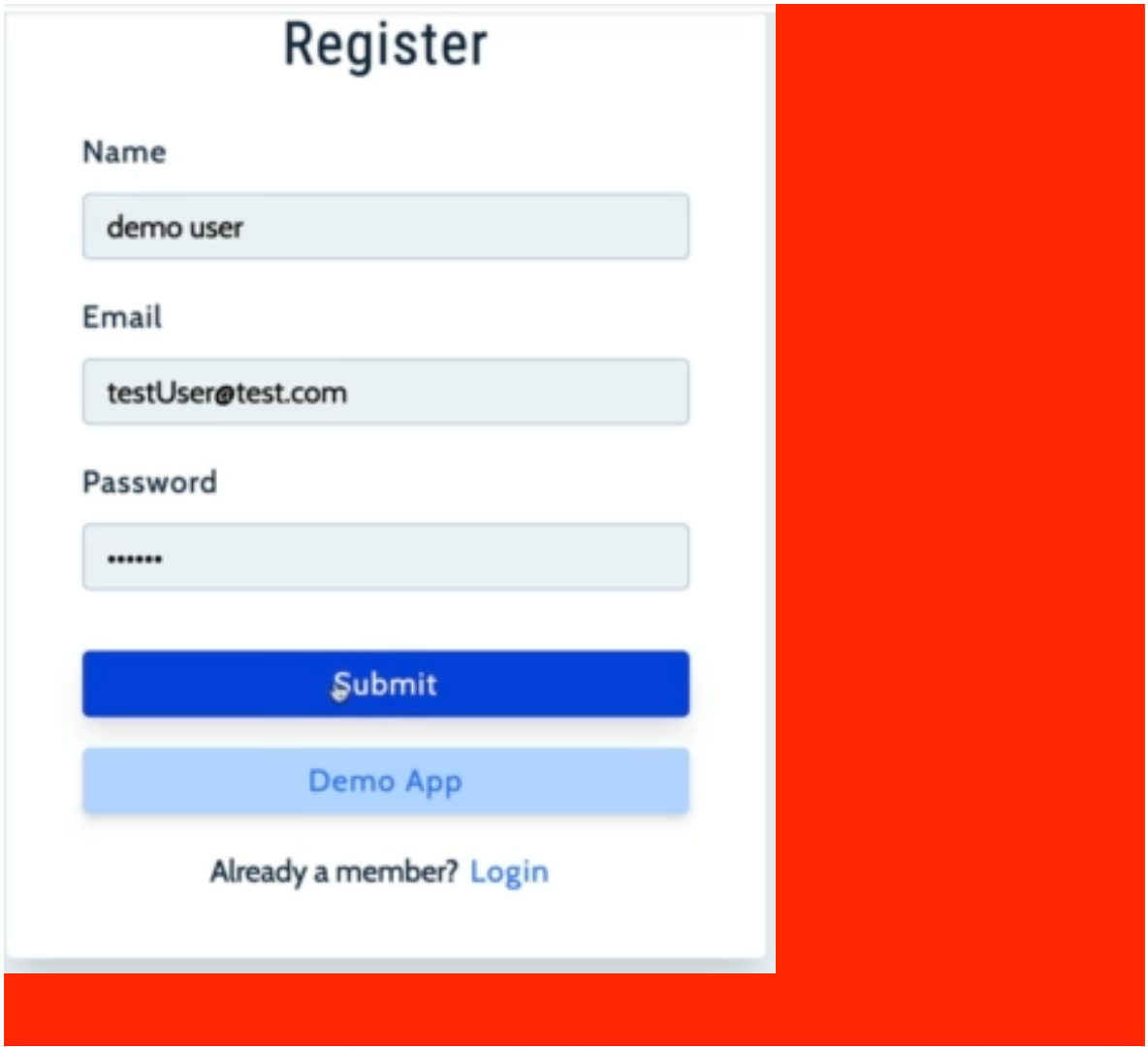
remove all users to check
```

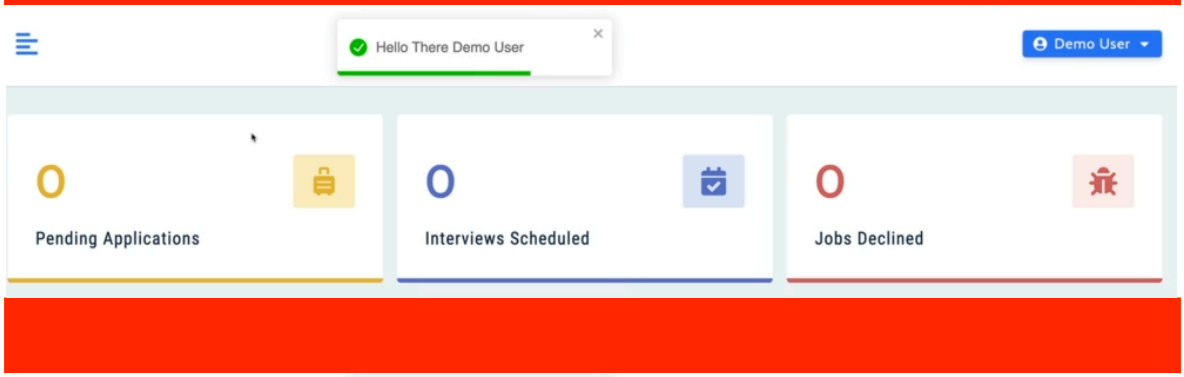


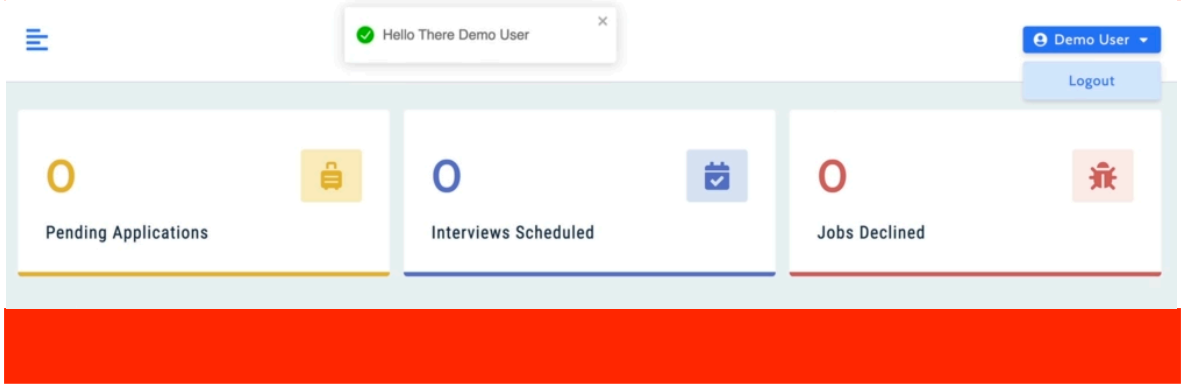


log out and register one more time

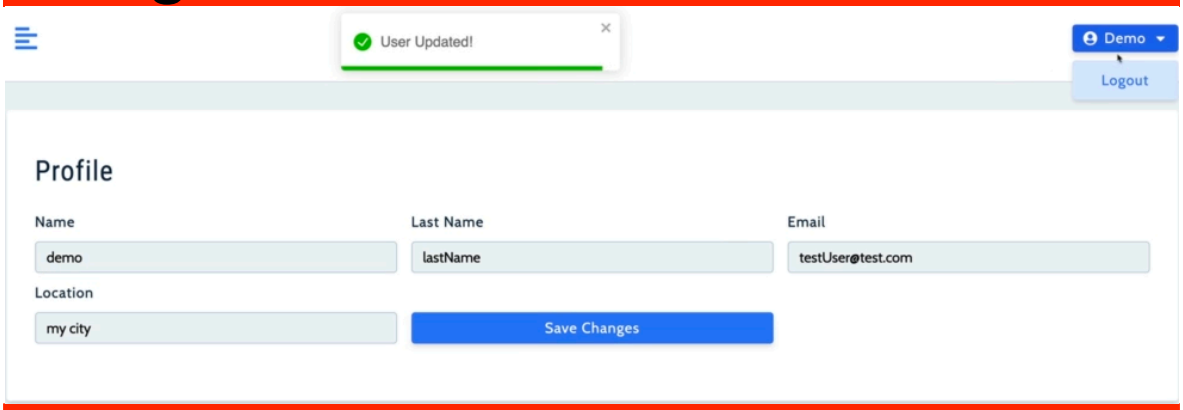

Logging Out...
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register


Register
Name
demo user
Email
testUser@test.com
Password
••••••
Submit
Demo App
Already a member? Login

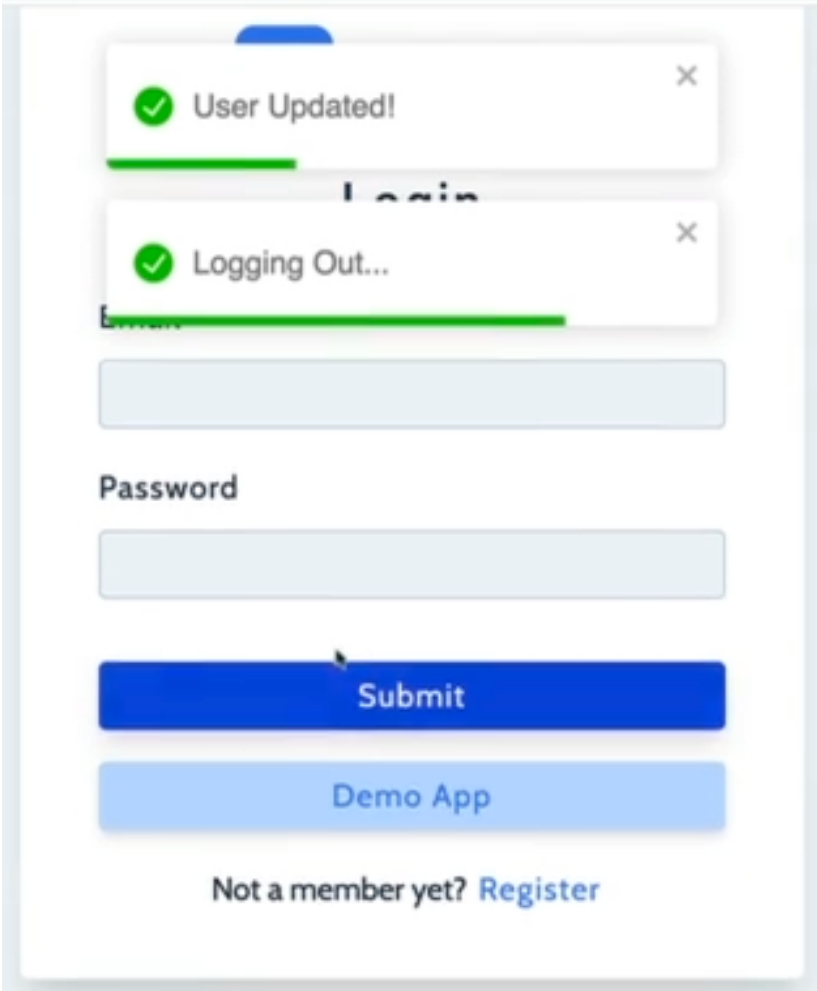

Hello There Demo User
Demo User
0
Pending Applications
0
Interviews Scheduled
0
Jobs Declined

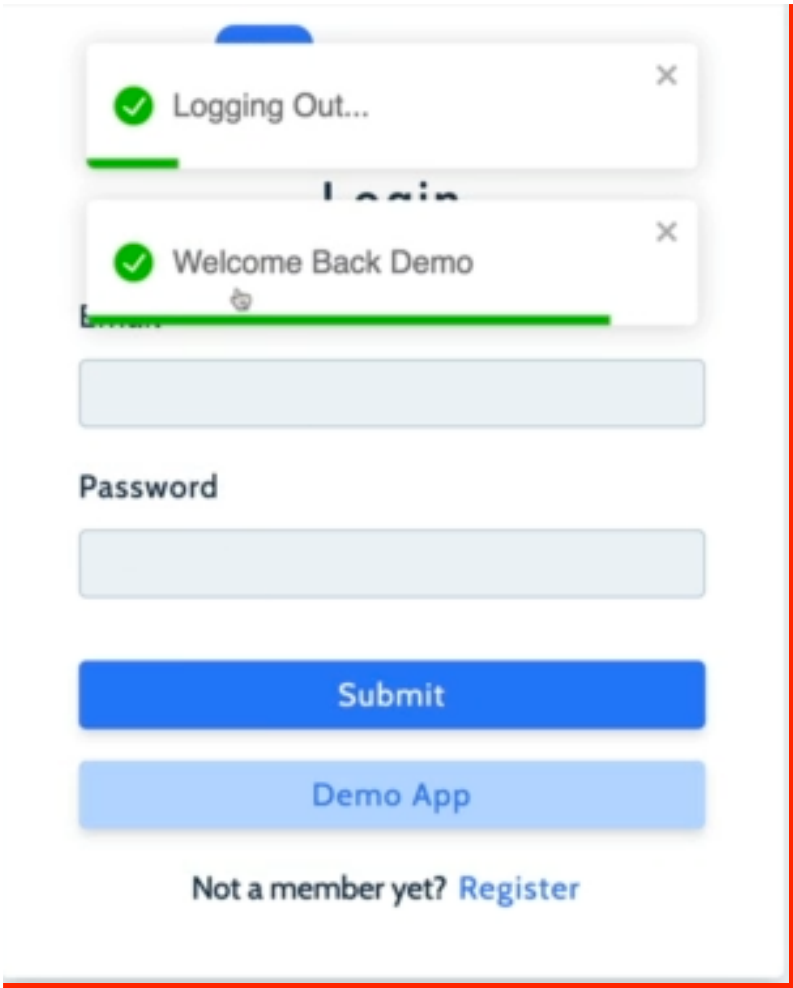

Hello There Demo User
Demo User
Logout
0
Pending Applications
0
Interviews Scheduled
0
Jobs Declined

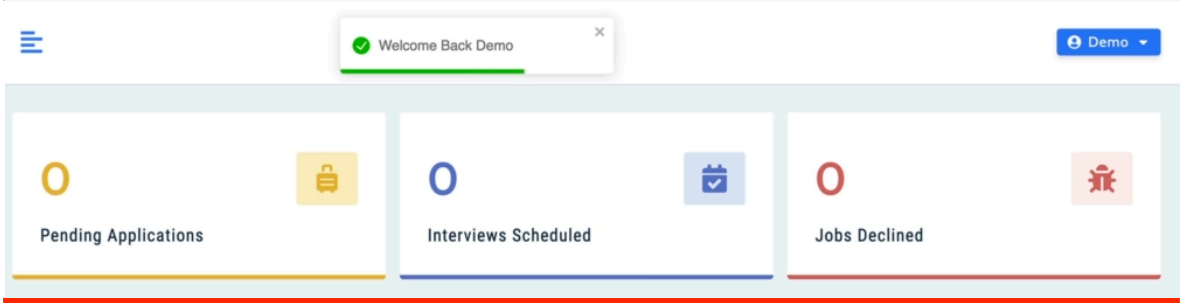
change name to Demo



log out and try Demo App



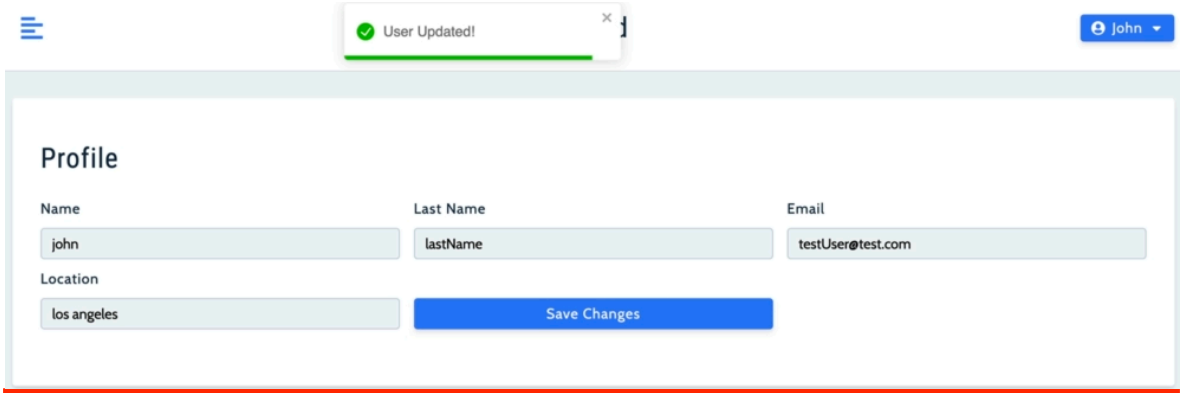




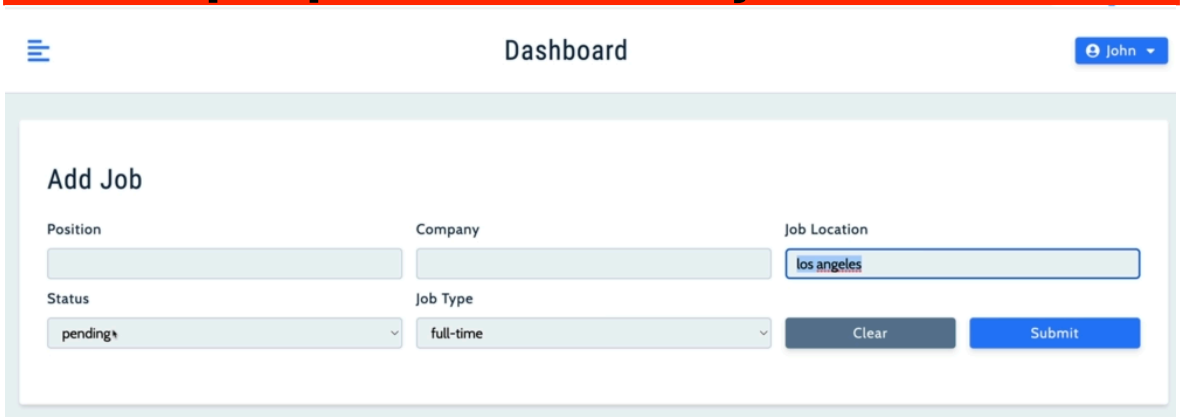
12 - Complete Jobs CRUD Functionality

==========

change the location and save



first we have to add job location and status properties on the jobs

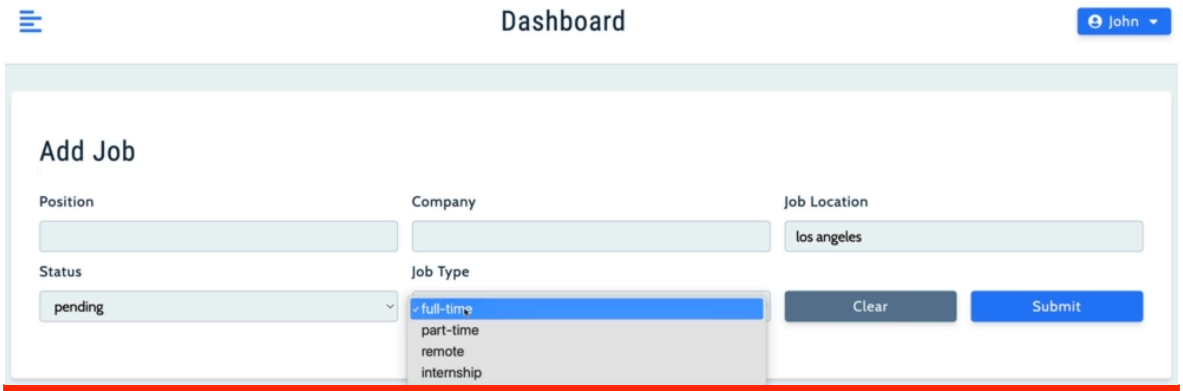


Modify Job Model

- add following properties

models/Job.js

```js
jobType: {
      type: String,
      enum: ['full-time', 'part-time', 'remote', 'internship'],
      default: 'full-time',
    },
    jobLocation: {
      type: String,
      default: 'my city',
      required: true,
    },
```

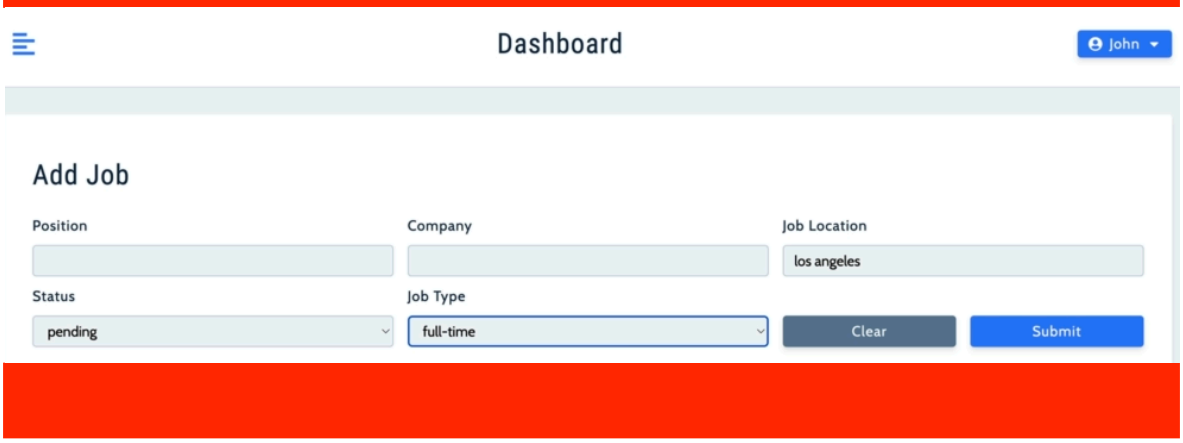


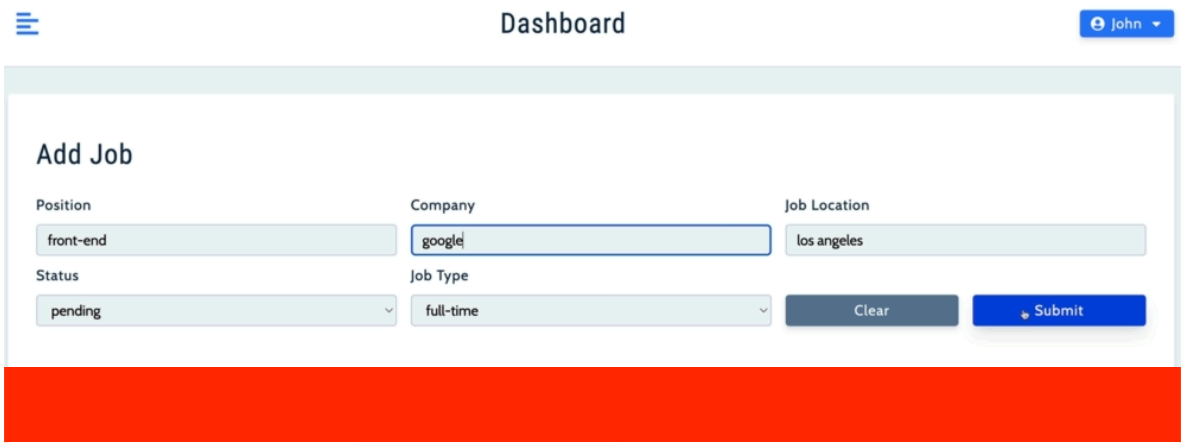
/starter/models/Job.js

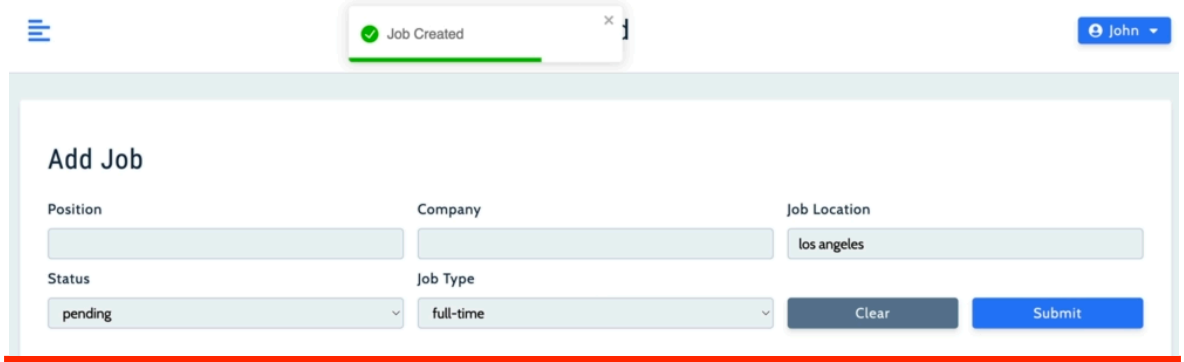
==========

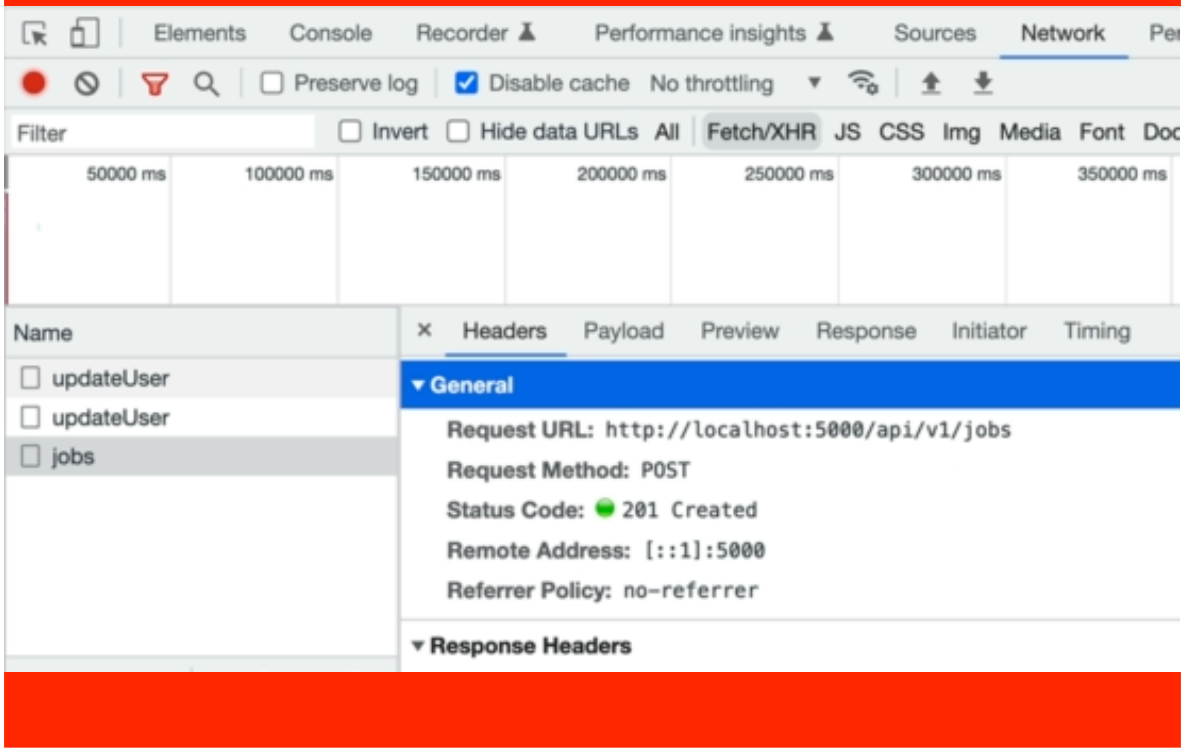
```
jobType: {
      type: String,
      enum: ['full-time', 'part-time',
'remote', 'internship'],
      default: 'full-time',
    },
    jobLocation: {
      type: String,
      default: 'my city',
      required: true,
    },
```

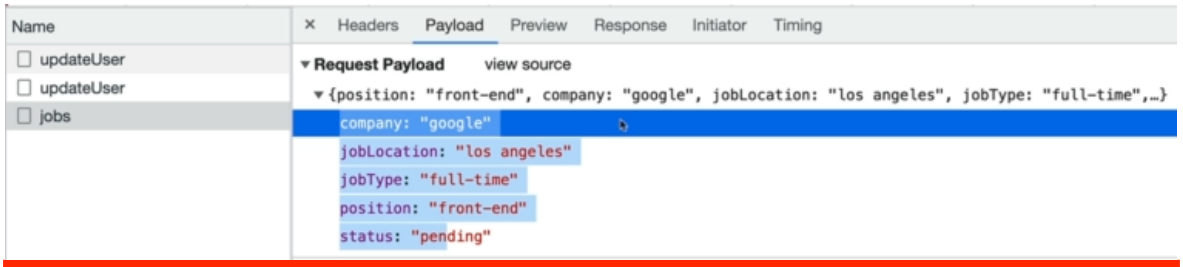
```
5-jobster-api > starter > models > JS Job.js > [⊙] JobSchema
14      },
15      status: {
16        type: String,
17        enum: ['interview', 'declined', 'pending'],
18        default: 'pending',
19      },
20      createdBy: {
21        type: mongoose.Types.ObjectId,
22        ref: 'User',
23        required: [true, 'Please provide user'],
24      },
25      jobType: {
26        type: String,
27        enum: ['full-time', 'part-time', 'remote', 'internship'],
28        default: 'full-time',
29      },
30      jobLocation: {
31        type: String,
32        default: 'my city',
33        required: true,
34      },
35    },
36    { timestamps: true }
37  );
```




Dashboard
John
Add Job
Position
front-end
Company
google
Job Location
los angeles
Status
pending
Job Type
full-time
Clear
Submit

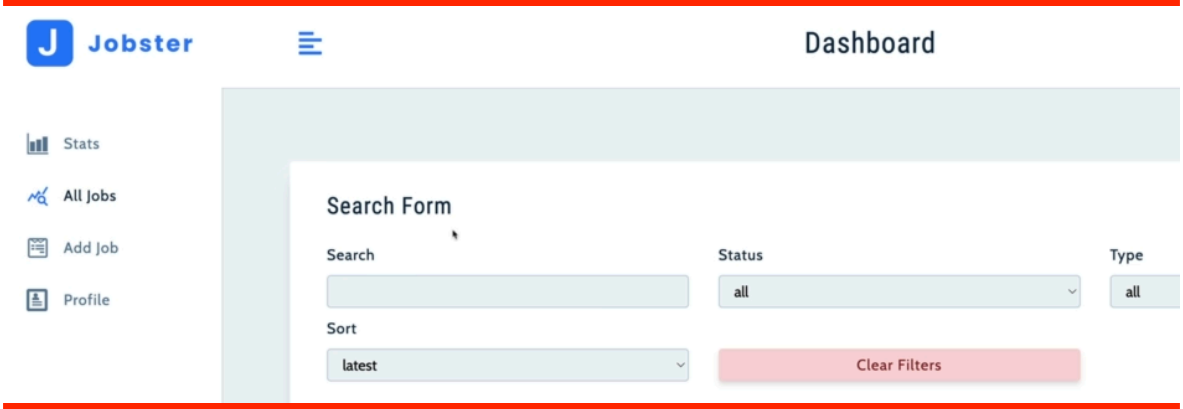

Job Created
John
Add Job
Position
Company
Job Location
los angeles
Status
pending
Job Type
full-time
Clear
Submit

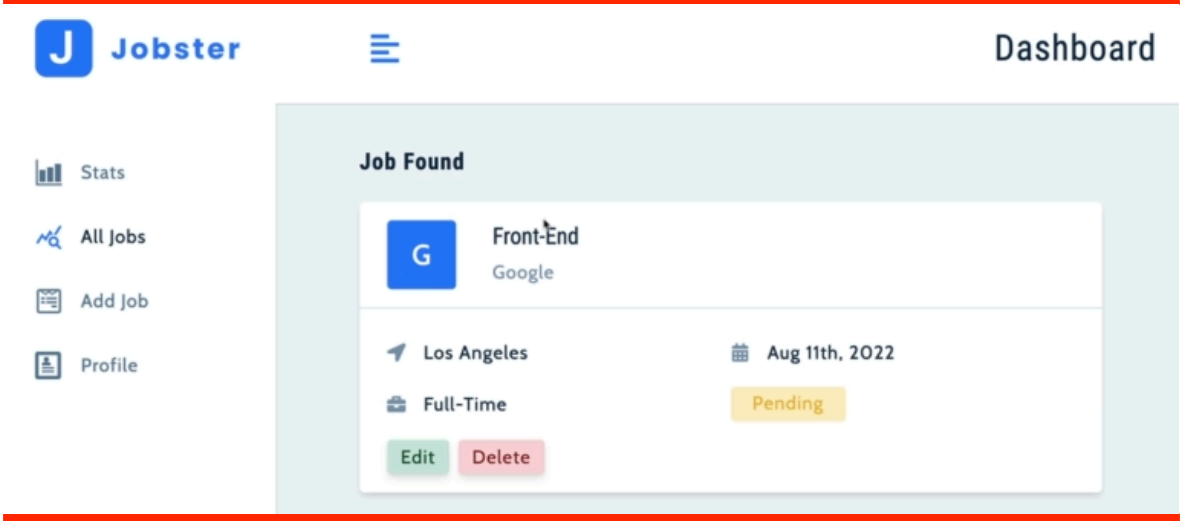

Elements
Console
Recorder
Performance insights
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
50000 ms
100000 ms
150000 ms
200000 ms
250000 ms
300000 ms
350000 ms
Name
updateUser
updateUser
jobs
Headers
Payload
Preview
Response
Initiator
Timing
General
Request URL: http://localhost:5000/api/v1/jobs
Request Method: POST
Status Code: 201 Created
Remote Address: [::1]:5000
Referrer Policy: no-referrer
Response Headers

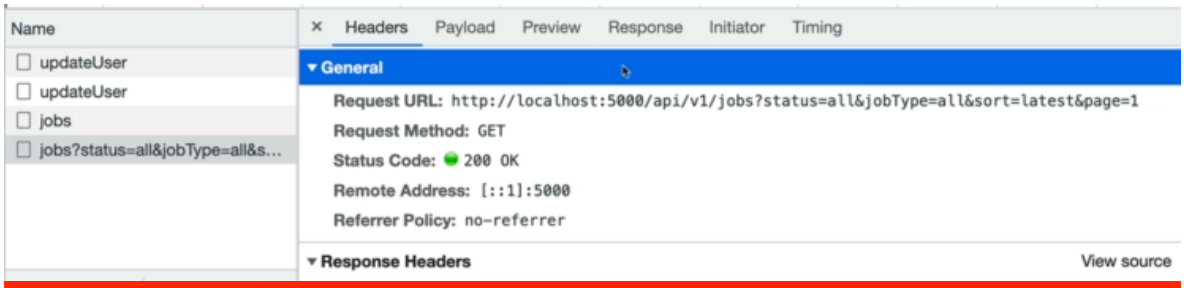

Name
updateUser
updateUser
jobs
Headers
Payload
Preview
Response
Initiator
Timing
Request Payload
view source
{position: "front-end", company: "google", jobLocation: "los angeles", jobType: "full-time",…}
company: "google"
jobLocation: "los angeles"
jobType: "full-time"
position: "front-end"
status: "pending"


Name
updateUser
updateUser
jobs
Headers
Payload
Preview
Response
Initiator
Timing
{"job":{"status":"pending","jobType":"full-time","jobLocation":"los angeles","_id":"62f533a1183fb40c928a58cc","position":"front-end","company":"google","createdBy":"62f52bbfe4396a07

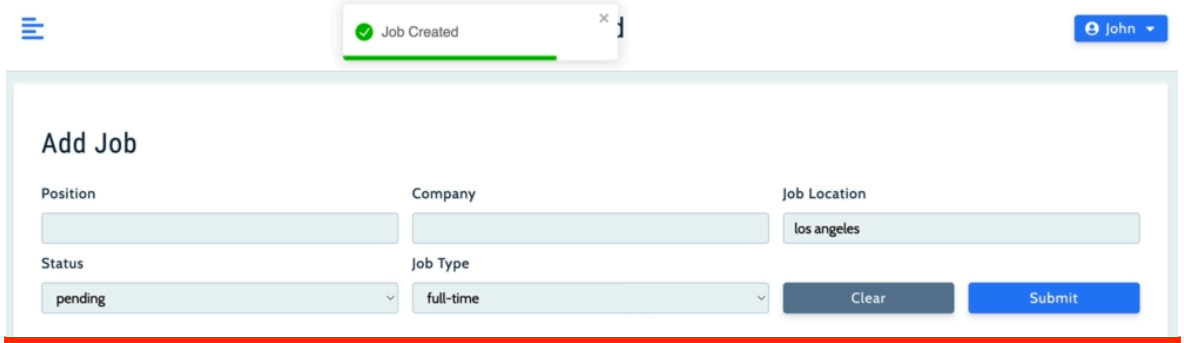
All Jobs

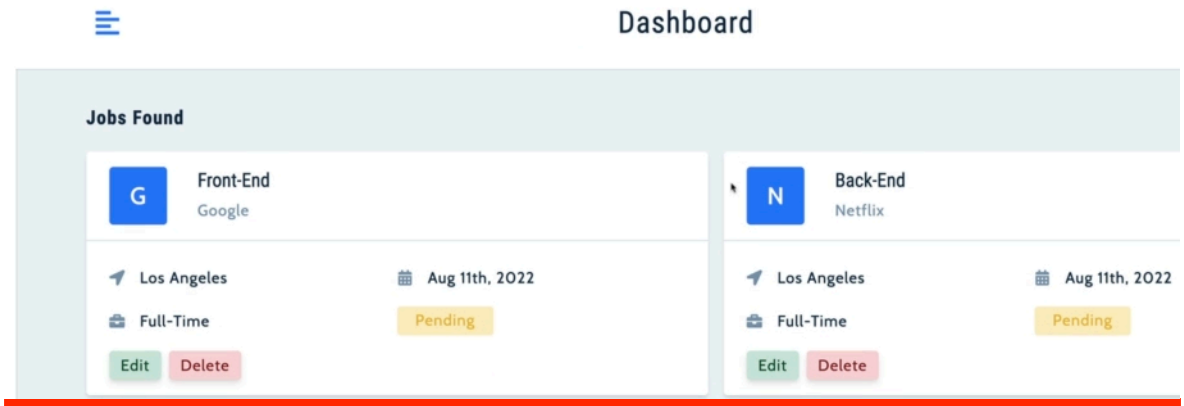

Jobster
Dashboard
Stats
All Jobs
Add Job
Profile
Search Form
Search
Status
all
Type
all
Sort
latest
Clear Filters


Jobster
Dashboard
Stats
All Jobs
Add Job
Profile
Job Found
G
Front-End
Google
Los Angeles
Aug 11th, 2022
Full-Time
Pending
Edit
Delete

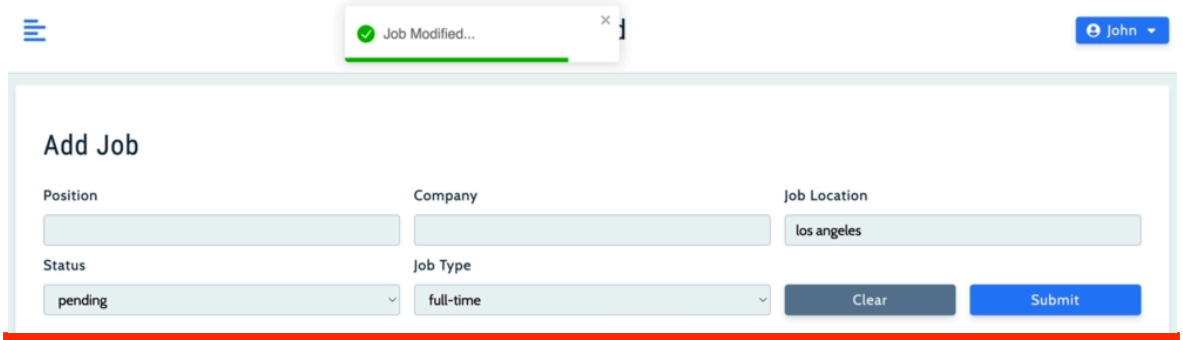


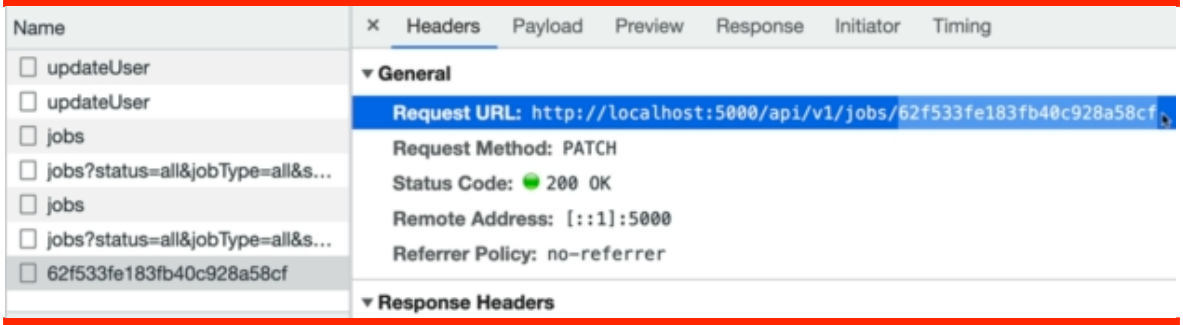
we will add another job

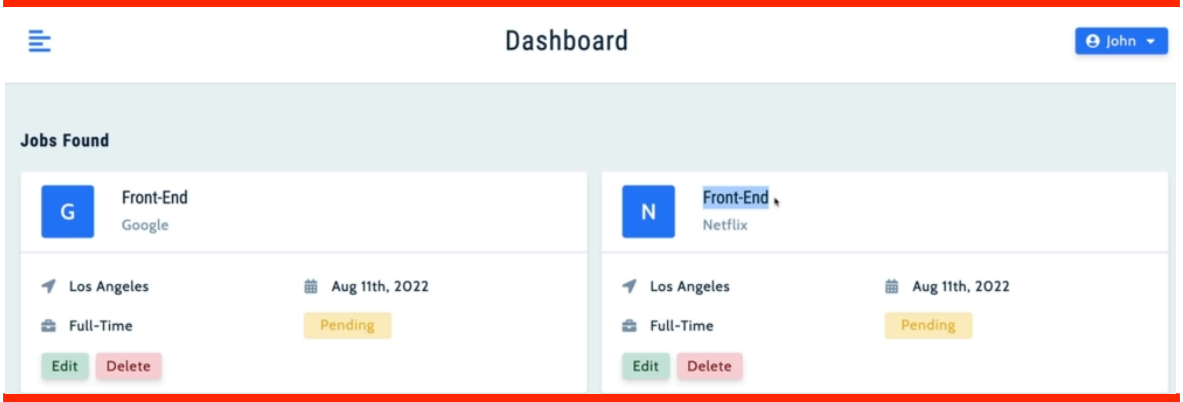




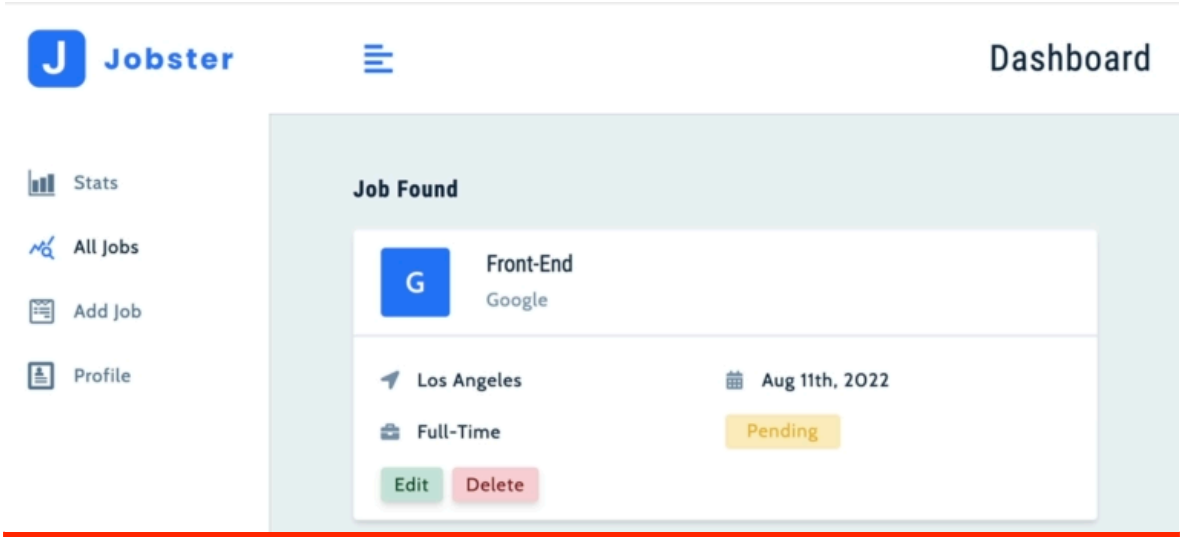
Edit Back-End job and change to Front-End

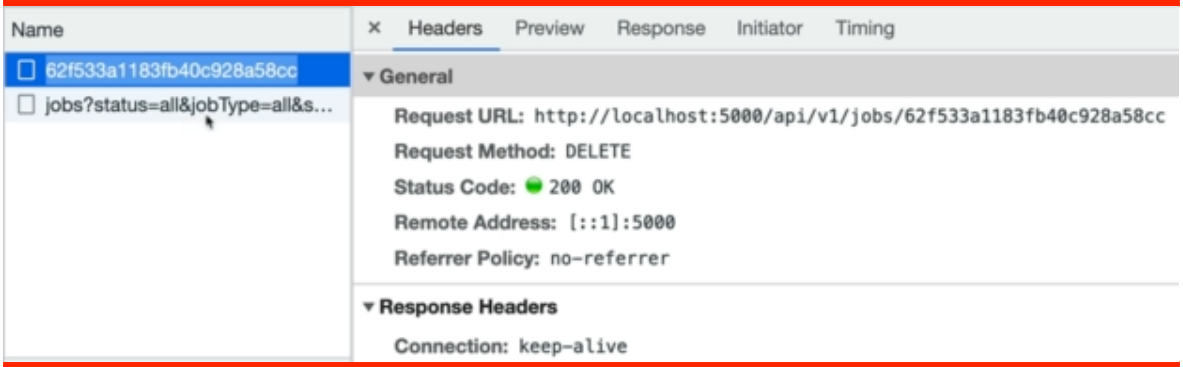




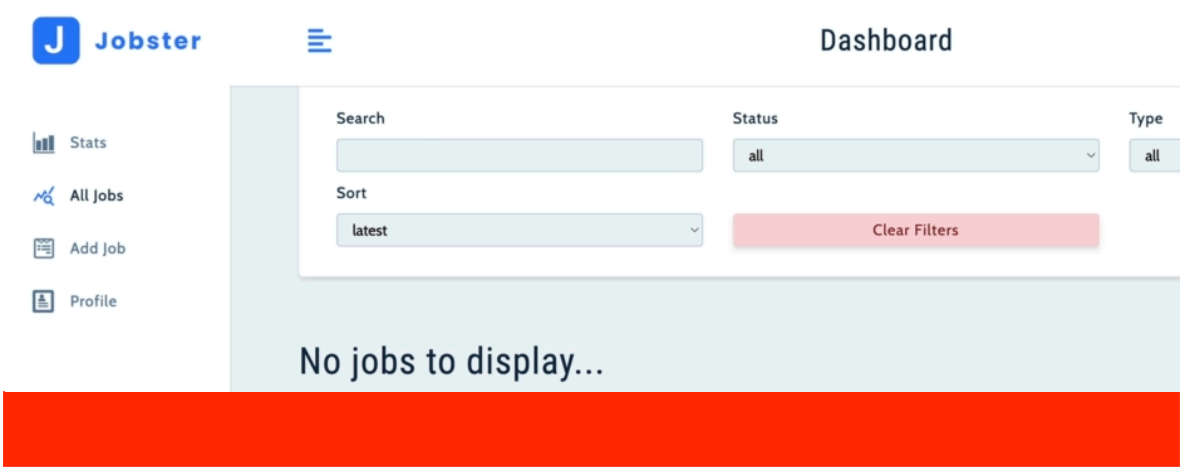


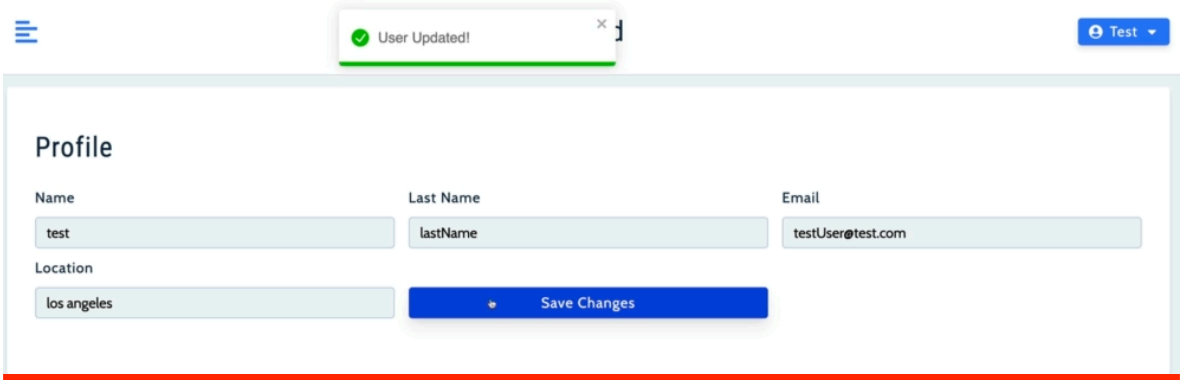
Press Delete to delete the changed record





press All Jobs

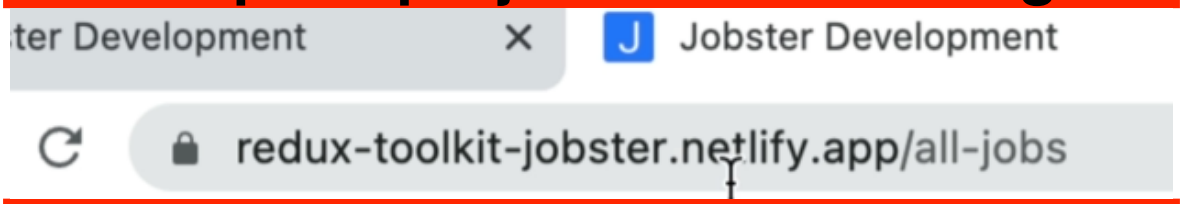




13 - Fake Data - Mockaroo

==========

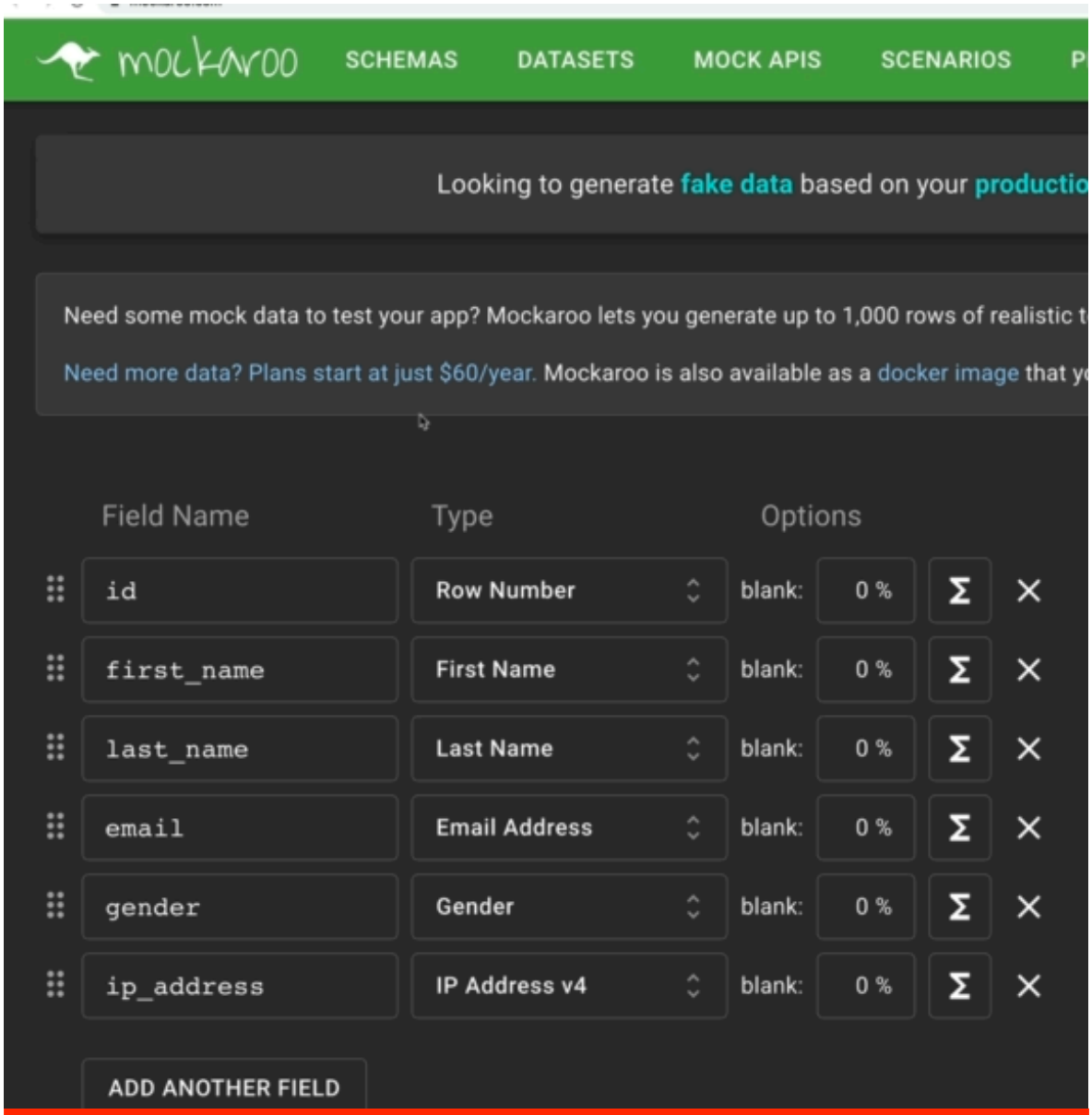
for complete project understanding



https://www.mockaroo.com

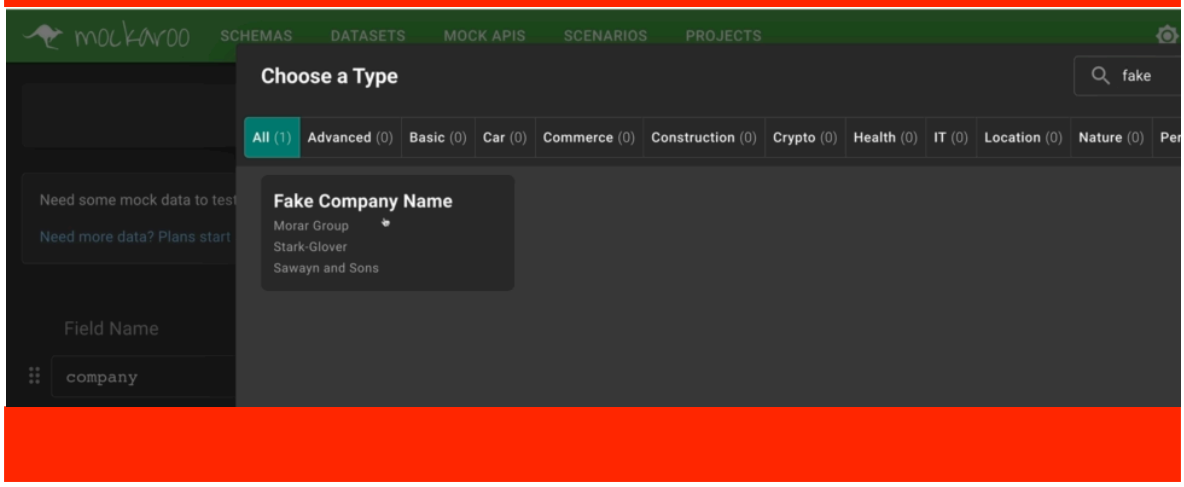
```
#### Setup Mock Data

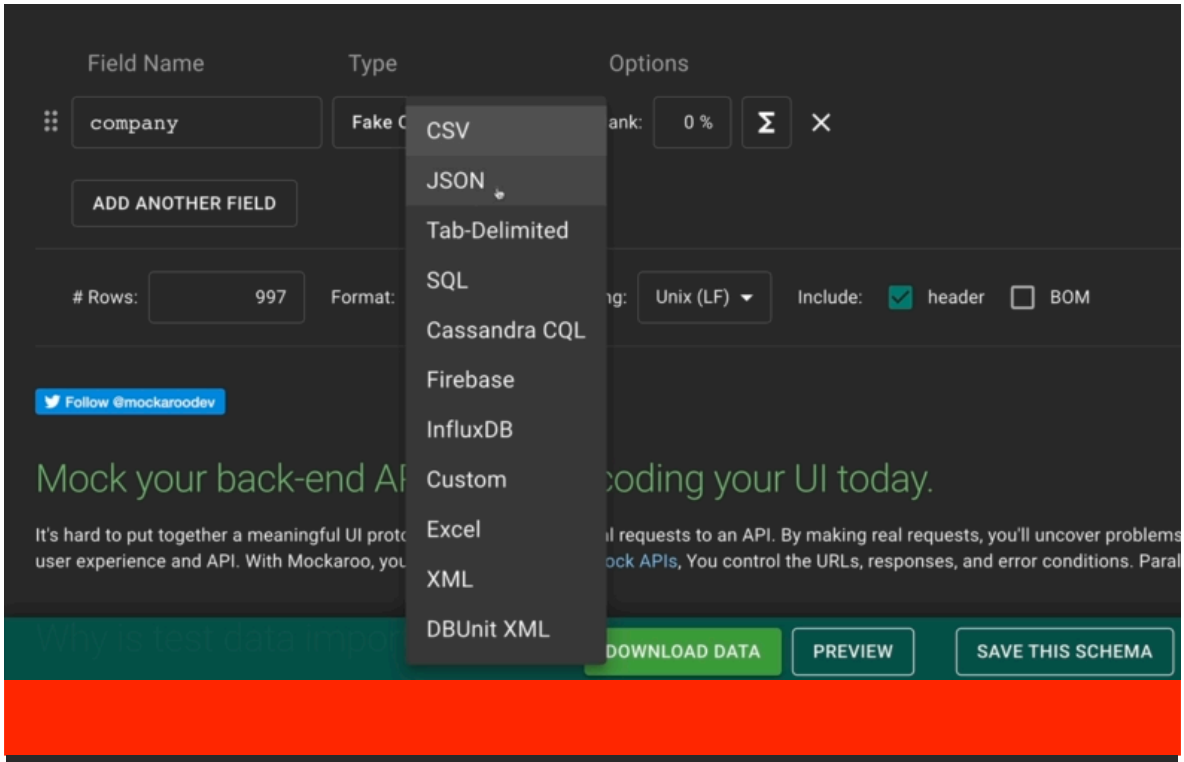
- [Mockaroo](https://www.mockaroo.com/)
- create mock-data.json (root)
- provide test user id
```

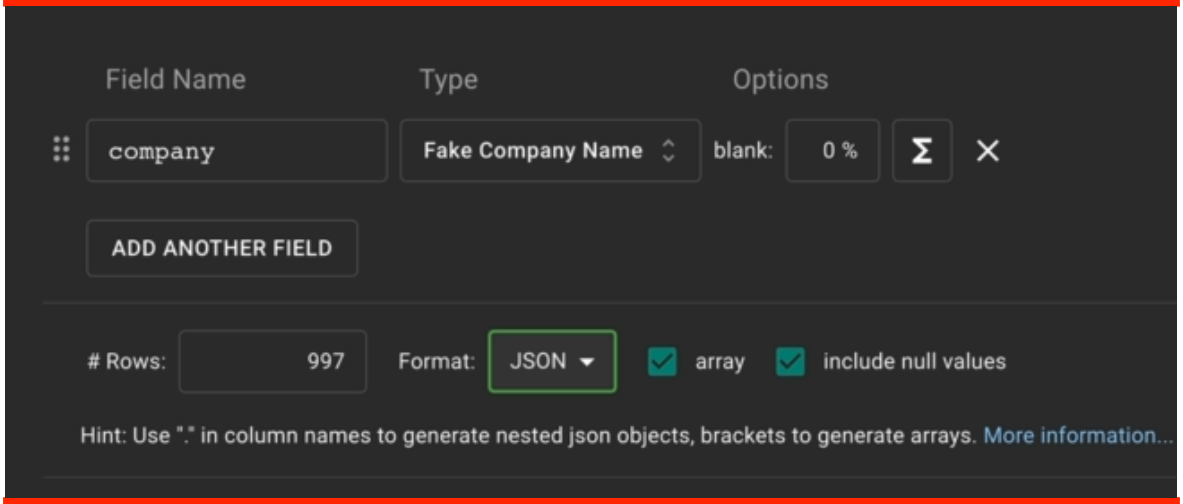


we have to create same as job model


06.5-jobster-api > starter > models > Job.js > JobSchema > jobType
const JobSchema = new mongoose.Schema(
{
company: {
type: String,
required: [true, 'Please provide company name'],
maxlength: 50,
},
position: {
type: String,
required: [true, 'Please provide position'],
maxlength: 100,
},
status: {
type: String,
enum: ['interview', 'declined', 'pending'],
default: 'pending',
},
createdBy: {
type: mongoose.Types.ObjectId,
ref: 'User',
required: [true, 'Please provide user'],
},
jobType: {

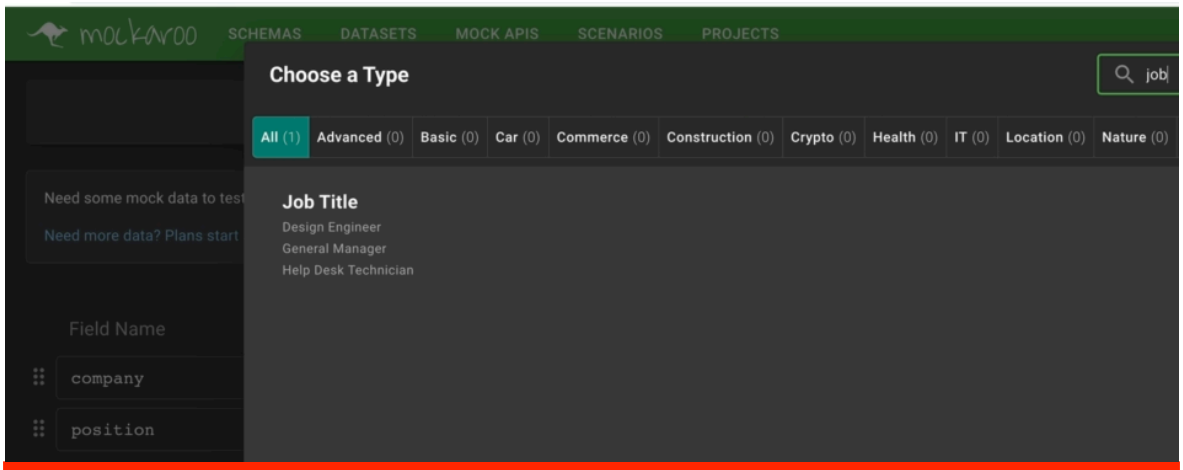

mockaroo
SCHEMAS
DATASETS
MOCK APIS
SCENARIOS
PROJECTS
Choose a Type
fake
All (1)
Advanced (0)
Basic (0)
Car (0)
Commerce (0)
Construction (0)
Crypto (0)
Health (0)
IT (0)
Location (0)
Nature (0)
Need some mock data to test
Need more data? Plans start
Fake Company Name
Morar Group
Stark-Glover
Sawayn and Sons
Field Name
company

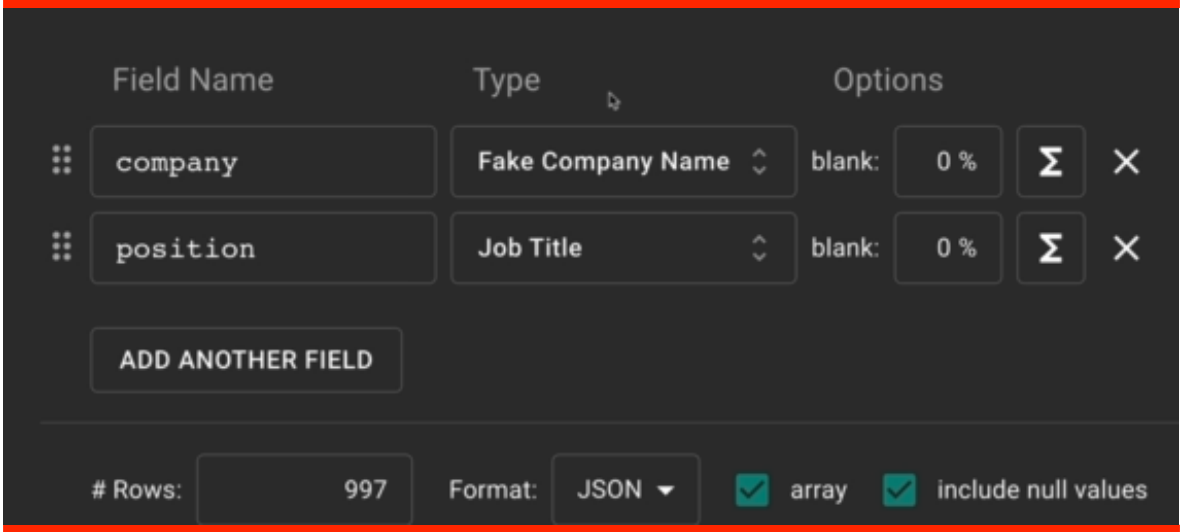


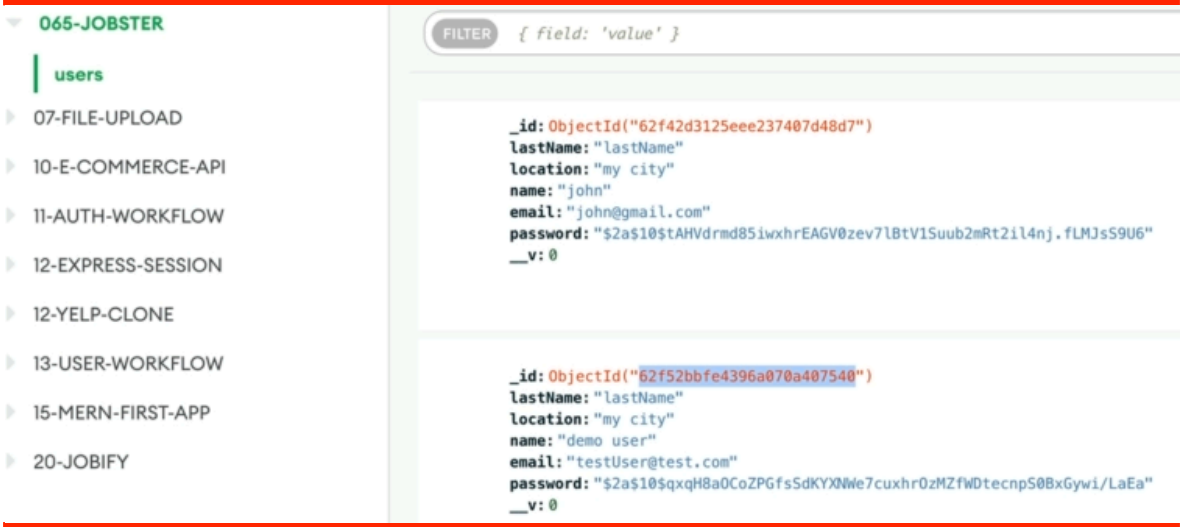


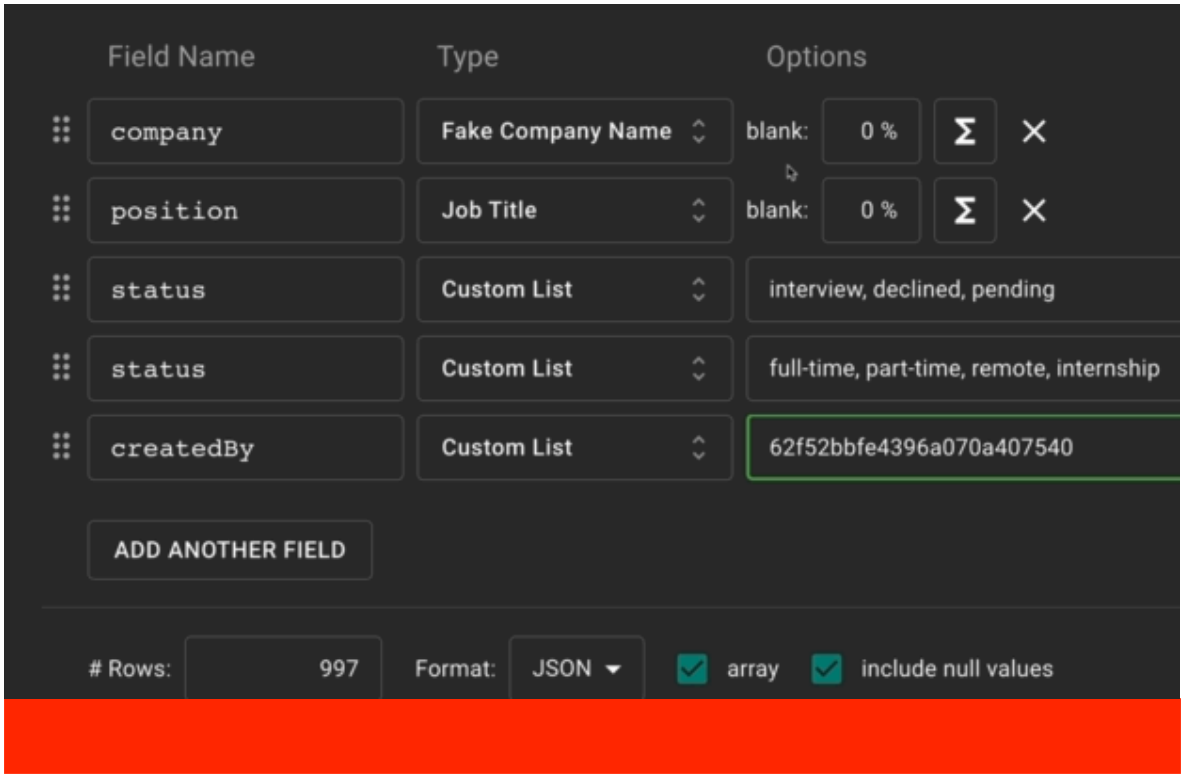
generating array of objects

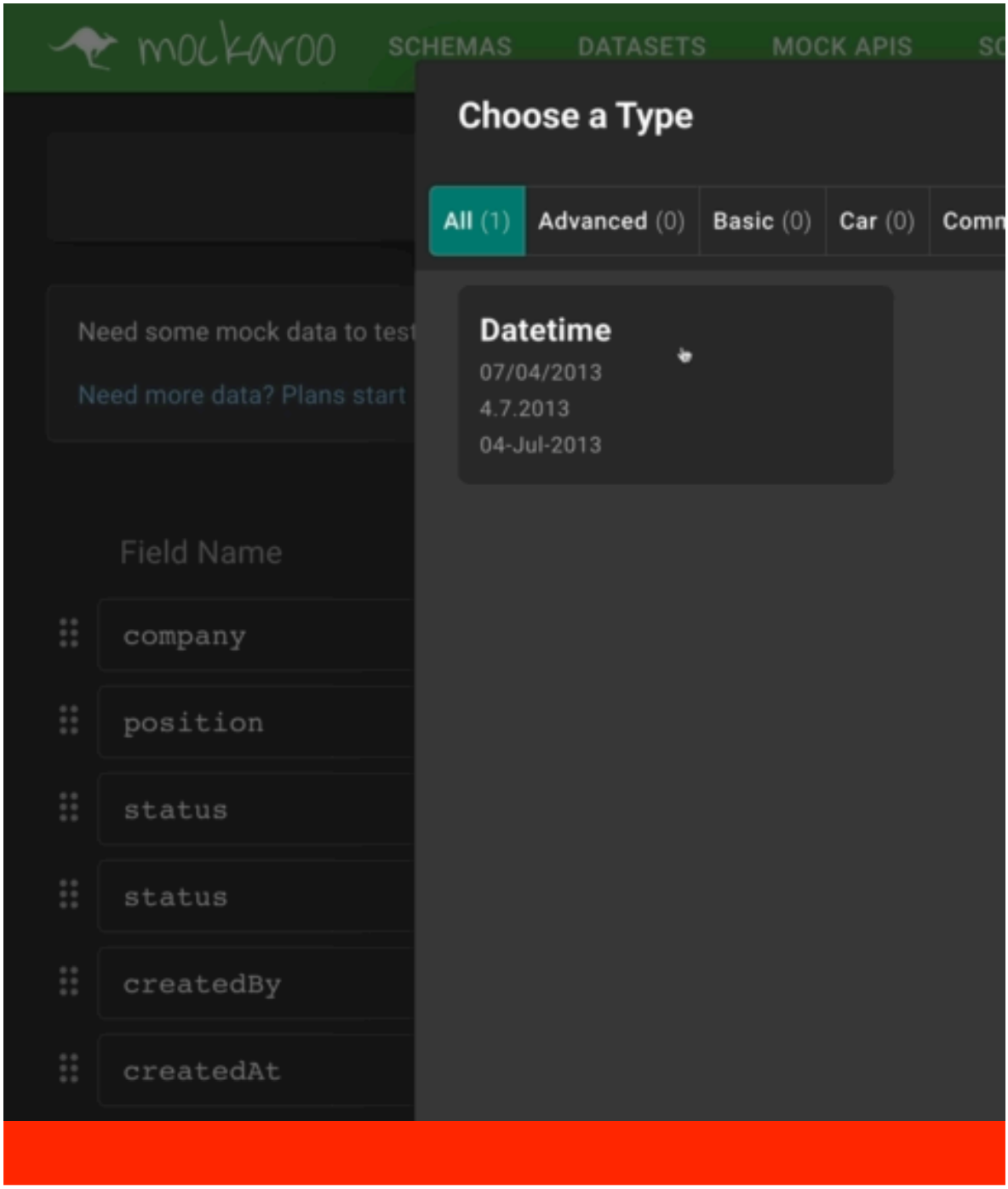

mockaroo
SCHEMAS
DATASETS
Preview
[{
"company": "Moen LLC"
}, {
"company": "Rodriguez LLC"
}, {
"company": "Hodkiewicz-Robel"
}, {
"company": "Ritchie and Sons"
}, {
"company": "Bashirian LLC"
}, {
"company": "Hand and Sons"
}, {
"company": "Sipes Inc"
}, {
"company": "Olson, Schulist and Dicki"
}, {
"company": "Ferry, Heidenreich and Upton"
}, {
"company": "Rowe LLC"
}, {
"company": "Witting-Williamson"
}, {
"company": "Kilback-Hickle"
}, {
"company": "Crona, Jones and Adams"
}, {
"company": "Bernhard-McDermott"
}, {
showing first 100 rows

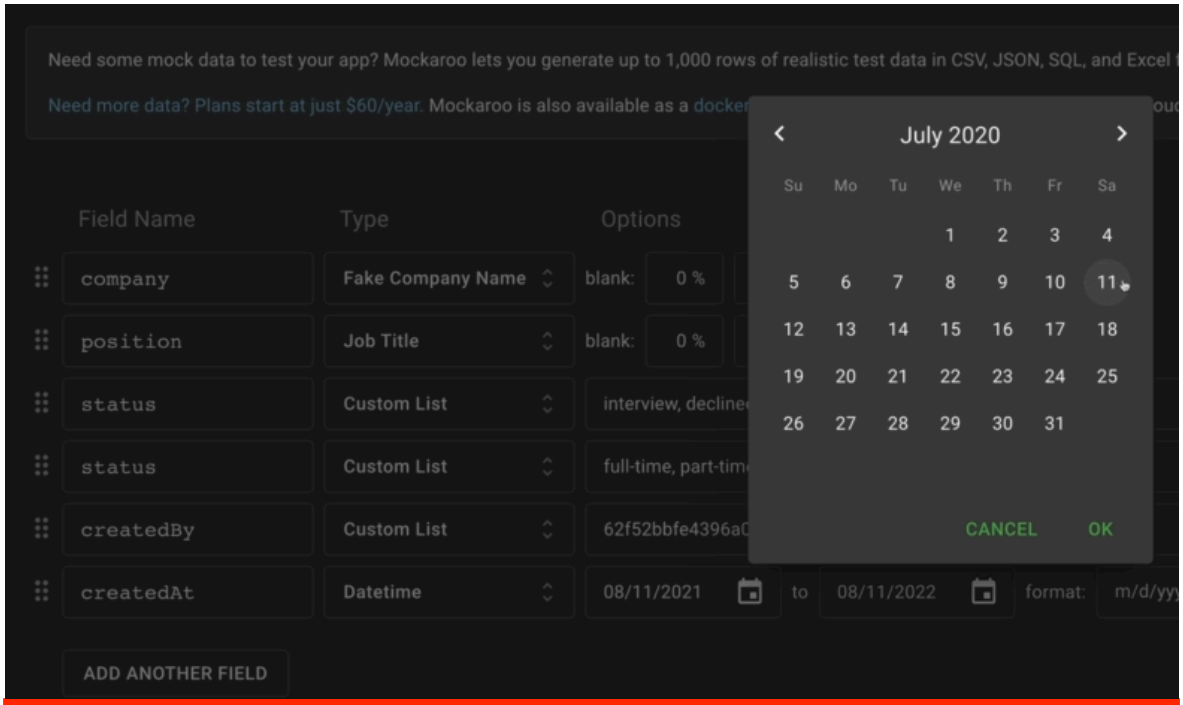

mockaroo
SCHEMAS
DATASETS
MOCK APIS
SCENARIOS
PROJECTS
Choose a Type
job
All (1)
Advanced (0)
Basic (0)
Car (0)
Commerce (0)
Construction (0)
Crypto (0)
Health (0)
IT (0)
Location (0)
Nature (0)
Need some mock data to test
Need more data? Plans start
Job Title
Design Engineer
General Manager
Help Desk Technician
Field Name
company
position

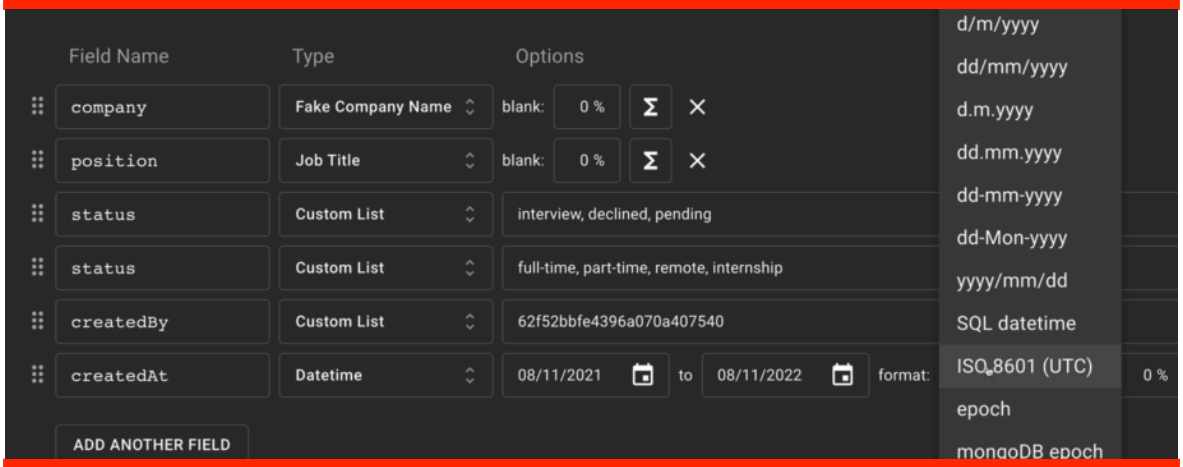

Field Name
Type
Options
company
Fake Company Name
blank:
0 %
position
Job Title
blank:
0 %
ADD ANOTHER FIELD
Rows:
997
Format:
JSON
array
include null values


065-JOBSTER
users
07-FILE-UPLOAD
10-E-COMMERCE-API
11-AUTH-WORKFLOW
12-EXPRESS-SESSION
12-YELP-CLONE
13-USER-WORKFLOW
15-MERN-FIRST-APP
20-JOBIFY
FILTER
{ field: 'value' }
_id: ObjectId("62f42d3125eee237407d48d7")
lastName: "lastName"
location: "my city"
name: "john"
email: "john@gmail.com"
password: "$2a$10$tAHVdrmd85iwxhrEAGV0zev7lBtV1Suub2mRt2il4nj.fLMJsS9U6"
__v: 0
_id: ObjectId("62f52bbfe4396a070a407540")
lastName: "lastName"
location: "my city"
name: "demo user"
email: "testUser@test.com"
password: "$2a$10$qxqH8aOCoZPGfsSdKYXNWe7cuxhrOzMZfWDtecnpS0BxGywi/LaEa"
__v: 0

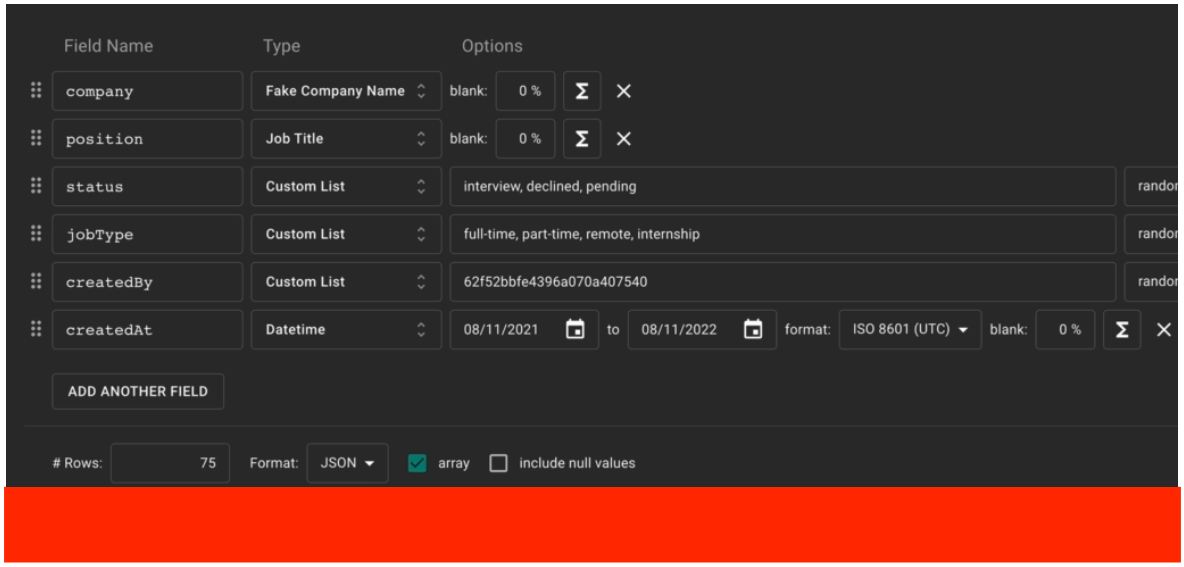

Field Name
Type
Options
company
Fake Company Name
blank:
0 %
position
Job Title
blank:
0 %
status
Custom List
interview, declined, pending
status
Custom List
full-time, part-time, remote, internship
createdBy
Custom List
62f52bbfe4396a070a407540
ADD ANOTHER FIELD
Rows:
997
Format:
JSON
array
include null values


mockaroo
SCHEMAS
DATASETS
MOCK APIS
Choose a Type
All (1)
Advanced (0)
Basic (0)
Car (0)
Need some mock data to test
Need more data? Plans start
Datetime
07/04/2013
4.7.2013
04-Jul-2013
Field Name
company
position
status
status
createdBy
createdAt





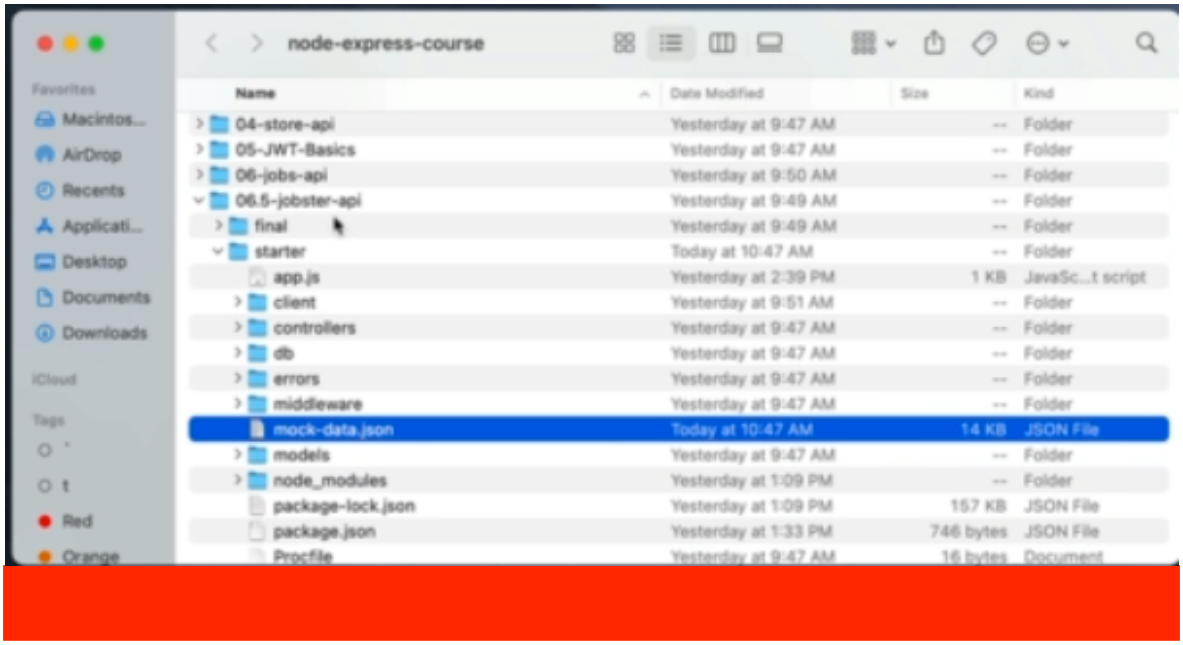
disable the include null values check box and reduce the total number of rows to 75

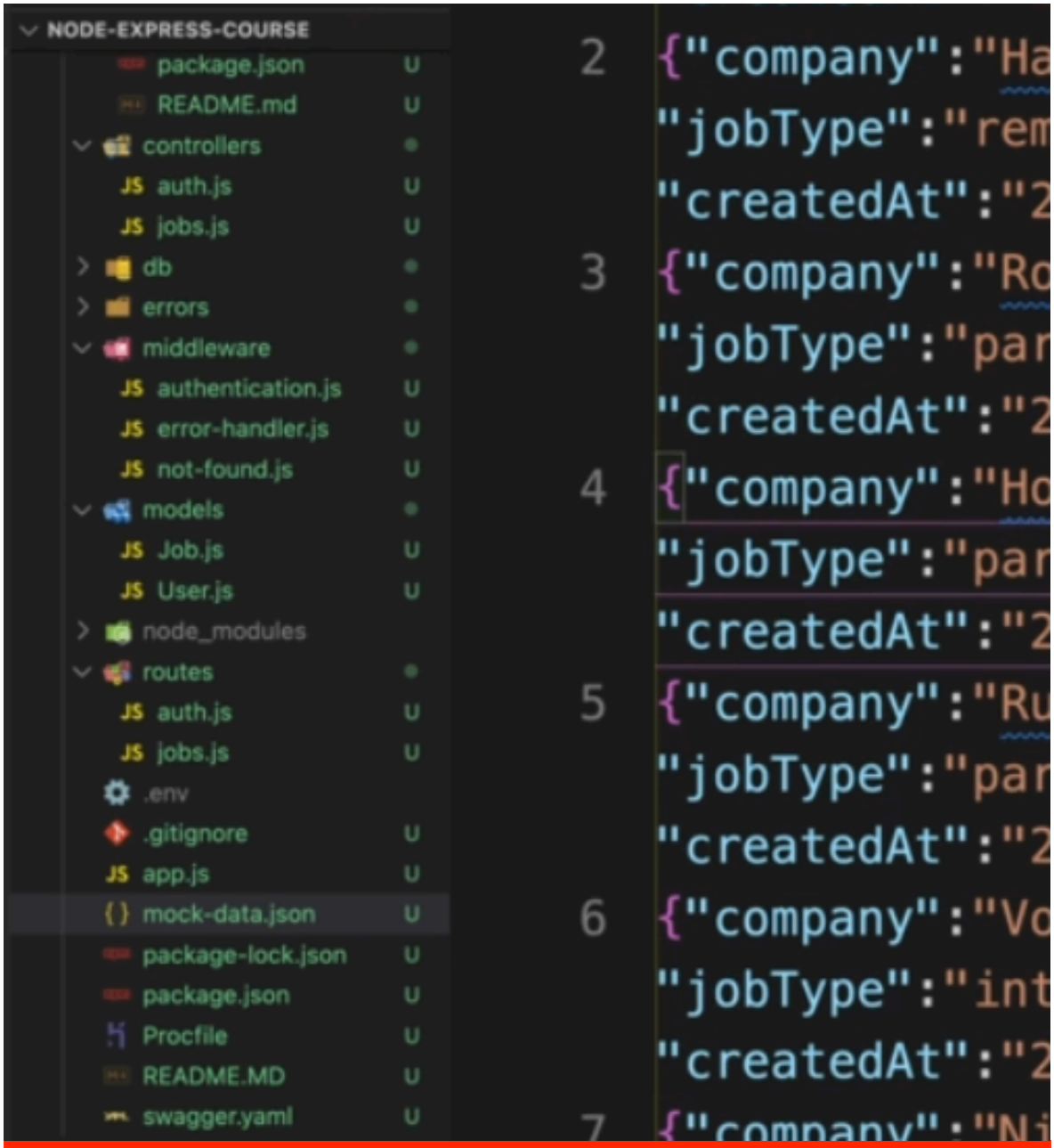

Field Name
Type
Options
company
Fake Company Name
blank:
0 %
position
Job Title
blank:
0 %
status
Custom List
interview, declined, pending
jobType
Custom List
full-time, part-time, remote, internship
createdBy
Custom List
62f52bbfe4396a070a407540
createdAt
Datetime
08/11/2021
to
08/11/2022
format:
ISO 8601 (UTC)
blank:
0 %
ADD ANOTHER FIELD
Rows:
75
Format:
JSON
array
include null values

Preview


```
[{
  "company": "Heathcote Inc",
  "position": "Assistant Media Planner",
  "status": "interview",
  "jobType": "full-time",
  "createdBy": "62f52bbfe4396a070a407540",
  "createdAt": "2022-05-02T11:31:58Z"
}, {
  "company": "Kuhic, Turcotte and Reichel",
  "position": "General Manager",
  "status": "pending",
  "jobType": "internship",
  "createdBy": "62f52bbfe4396a070a407540",
  "createdAt": "2022-01-13T07:13:25Z"
}, {
  "company": "Cummings-Rath",
  "position": "VP Accounting",
  "status": "declined",
  "jobType": "remote",
  "createdBy": "62f52bbfe4396a070a407540",
  "createdAt": "2022-04-29T20:58:27Z"
}, {
  "company": "Beier, O'Reilly and Cole",
  "position": "Administrative Assistant I",
  "status": "declined",
  "jobType": "full-time",
  "createdBy": "62f52bbfe4396a070a407540",
  "createdAt": "2022-02-11T06:49:57Z"
}, {
```


download the file and rename and copy it in your project


node-express-course
Favorites
Macintos...
AirDrop
Recents
Applicati...
Desktop
Documents
Downloads
iCloud
Tags
t
Red
Orange
Name
Date Modified
Size
Kind
04-store-api
Yesterday at 9:47 AM
Folder
05-JWT-Basics
Yesterday at 9:47 AM
Folder
06-jobs-api
Yesterday at 9:50 AM
Folder
06.5-jobster-api
Yesterday at 9:49 AM
Folder
final
Yesterday at 9:49 AM
Folder
starter
Today at 10:47 AM
Folder
app.js
Yesterday at 2:39 PM
1 KB
client
Yesterday at 9:51 AM
Folder
controllers
Yesterday at 9:47 AM
Folder
db
Yesterday at 9:47 AM
Folder
errors
Yesterday at 9:47 AM
Folder
middleware
Yesterday at 9:47 AM
Folder
mock-data.json
Today at 10:47 AM
14 KB
JSON File
models
Yesterday at 9:47 AM
Folder
node_modules
Yesterday at 1:09 PM
Folder
package-lock.json
Yesterday at 1:09 PM
157 KB
JSON File
package.json
Yesterday at 1:33 PM
746 bytes
JSON File
Procfile
Yesterday at 9:47 AM
16 bytes
Document



14 - Populate Database

==========

````
populate.js

```js
require('dotenv').config();

const mockData = require('./mock-data.json');

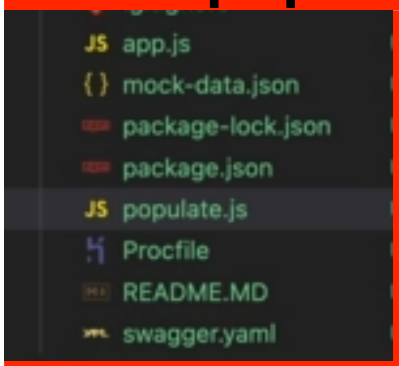
const Job = require('./models/Job');
const connectDB = require('./db/connect');

const start = async () => {
  try {
    await connectDB(process.env.MONGO_URI);

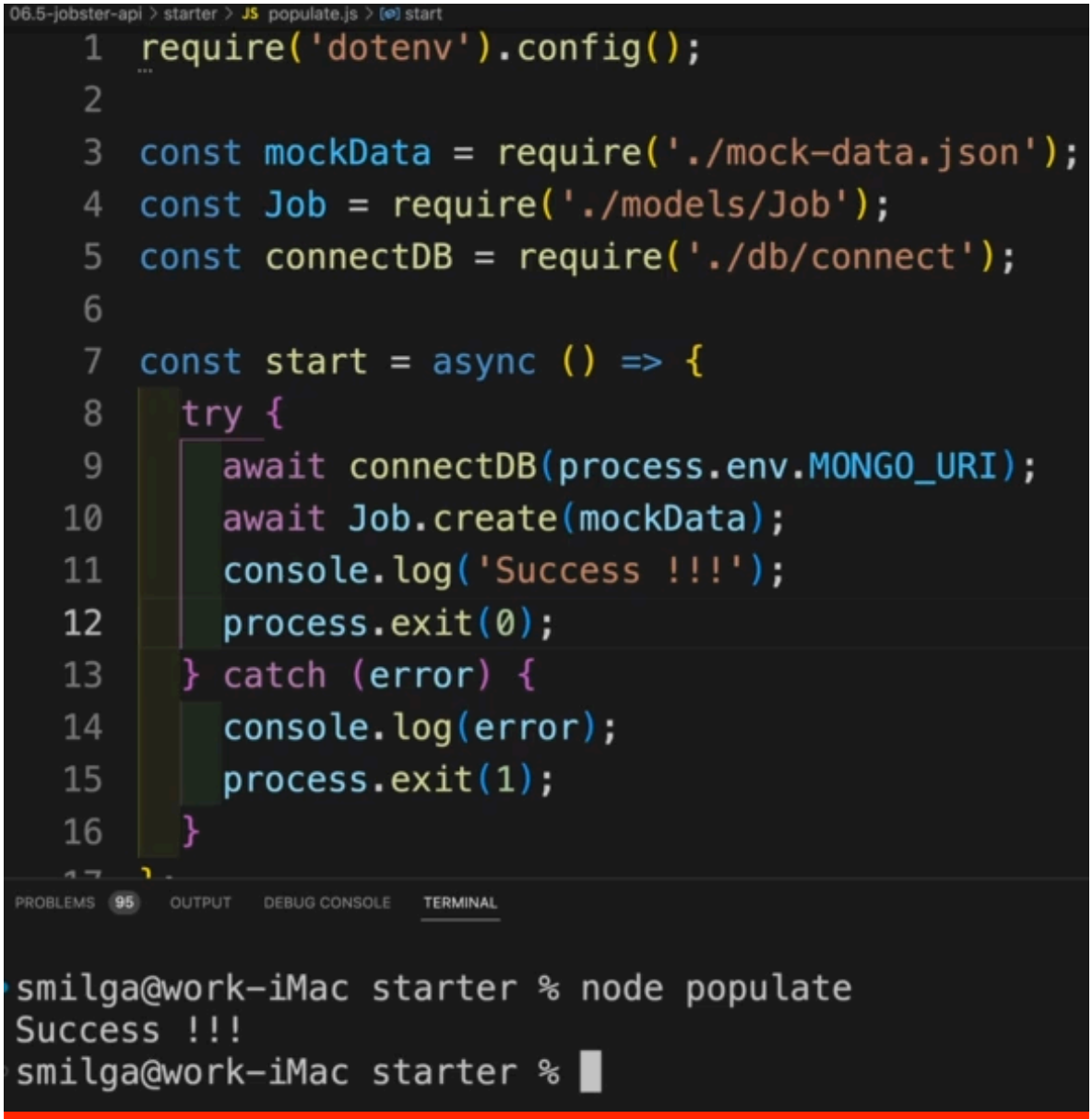
    await Job.create(mockData);
    console.log('Success!!!');
    process.exit(0);
  } catch (error) {
    console.log(error);
    process.exit(1);
  }
};

start();
```
````

create populate.js file in root



stop the server and populate

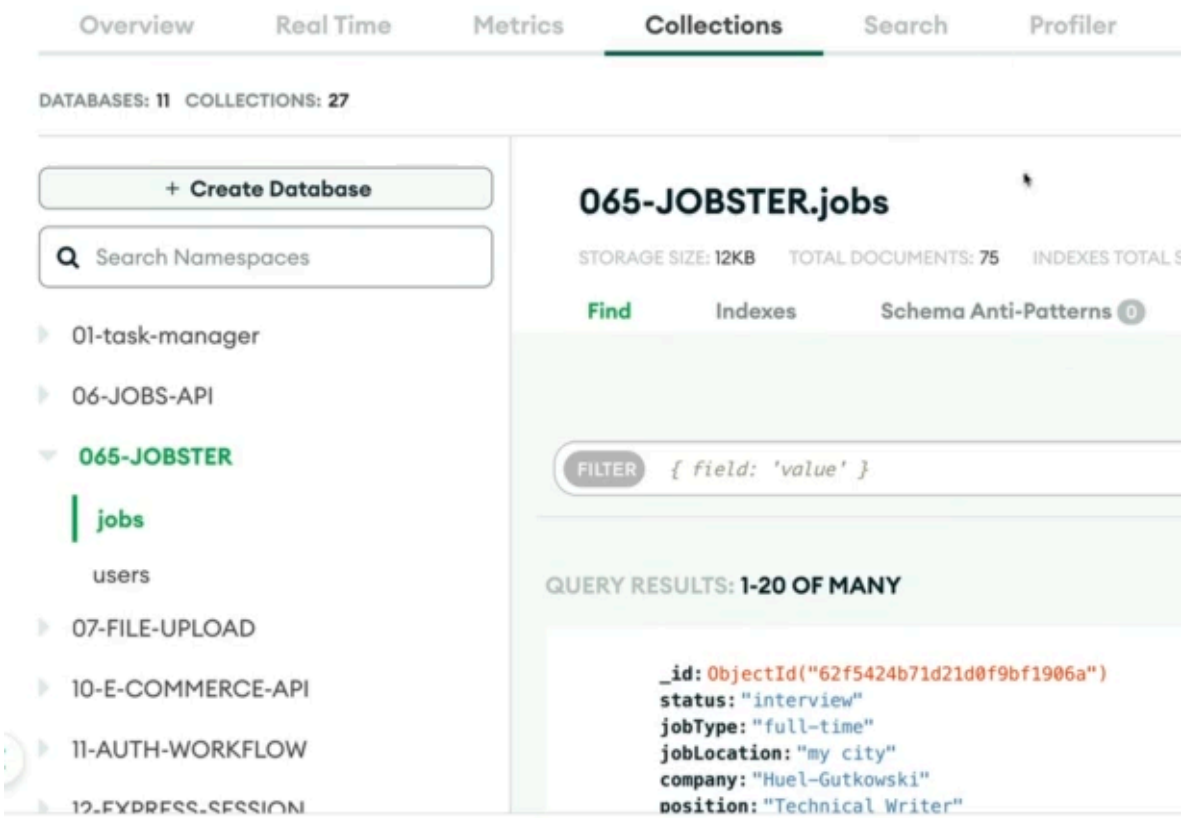


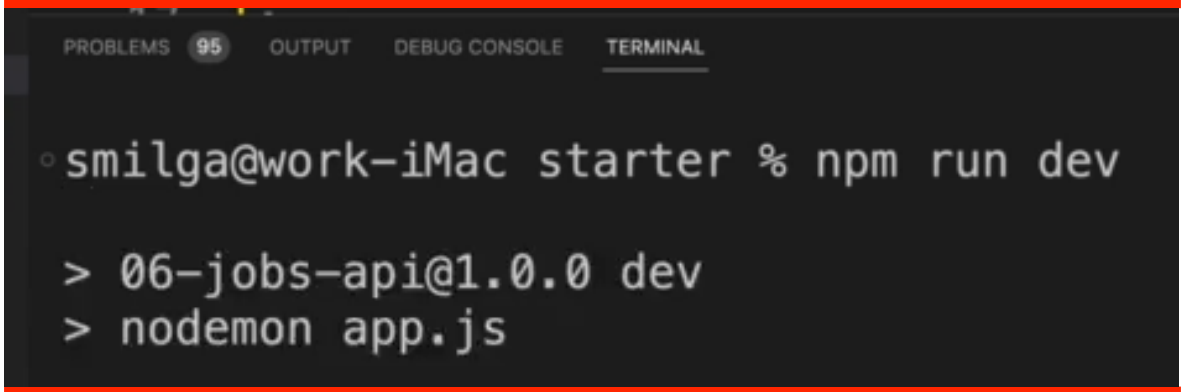
require('dotenv').config();
const mockData = require('./mock-data.json');
const Job = require('./models/Job');
const connectDB = require('./db/connect');

```
const start = async () => {
  try {
    await
connectDB(process.env.MONGO_URI);
    await Job.create(mockData);
    console.log('Success !!!');
    process.exit(0);
  } catch (error) {
    console.log(error);
    process.exit(1);
  }
};

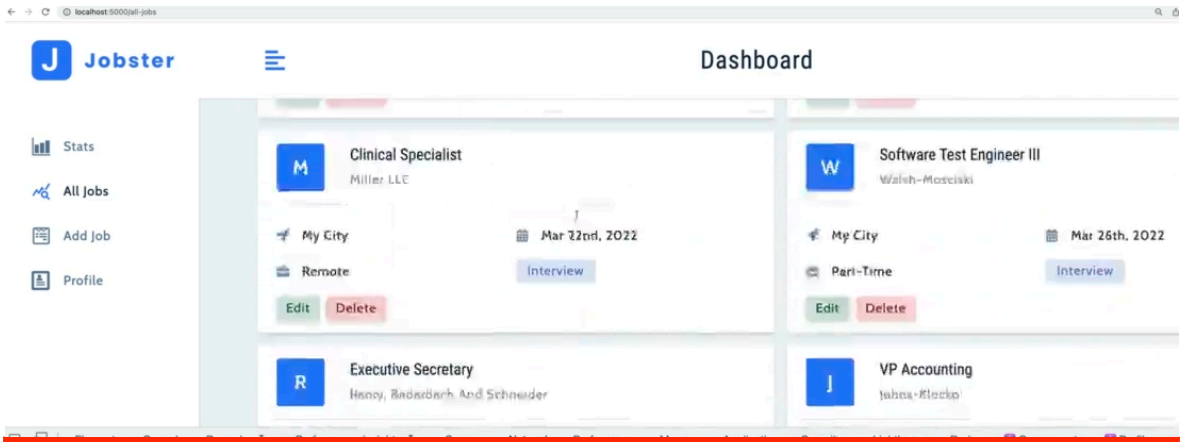
start();
```

go to database and check



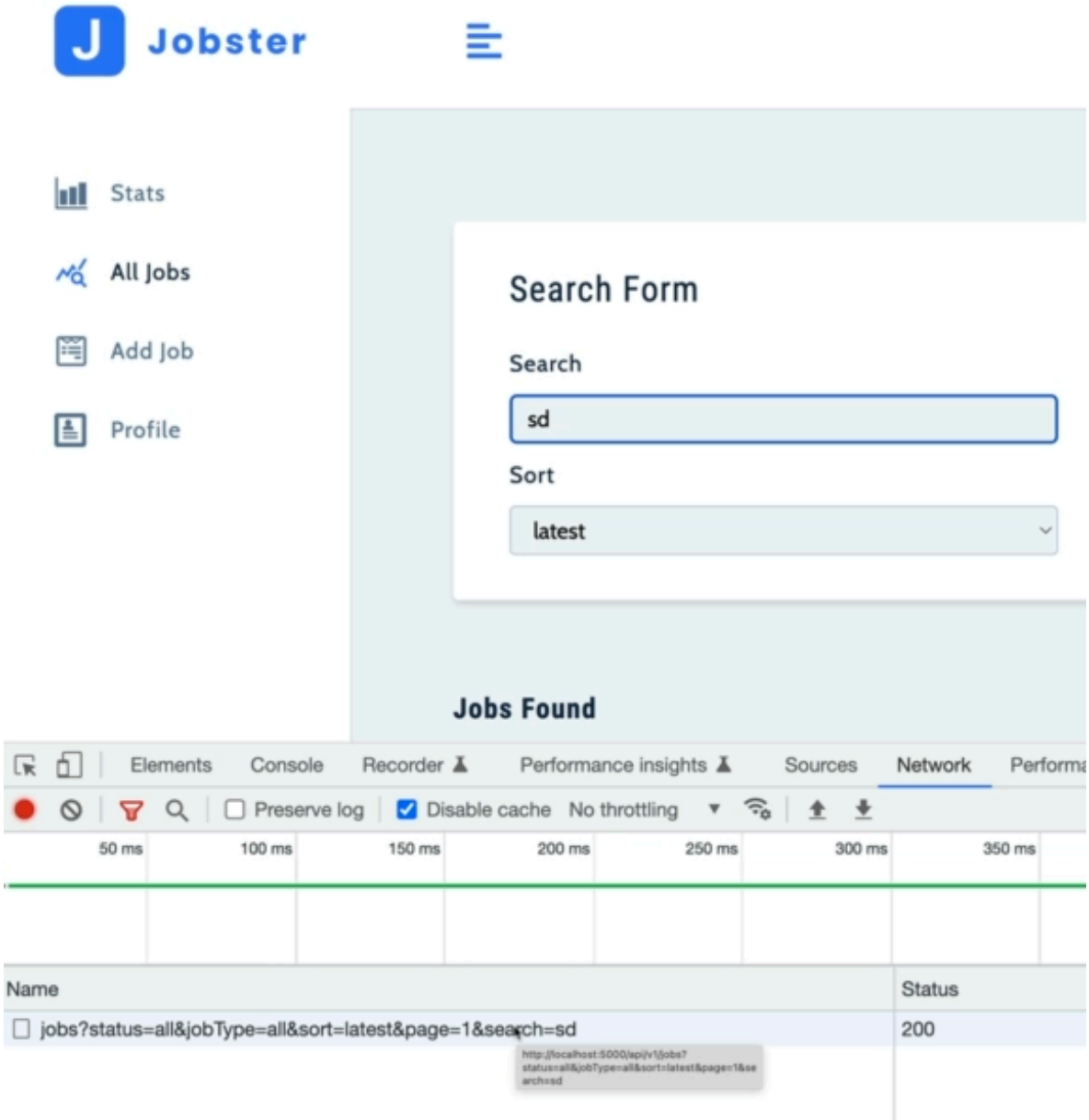


localhost:5000/all-jobs

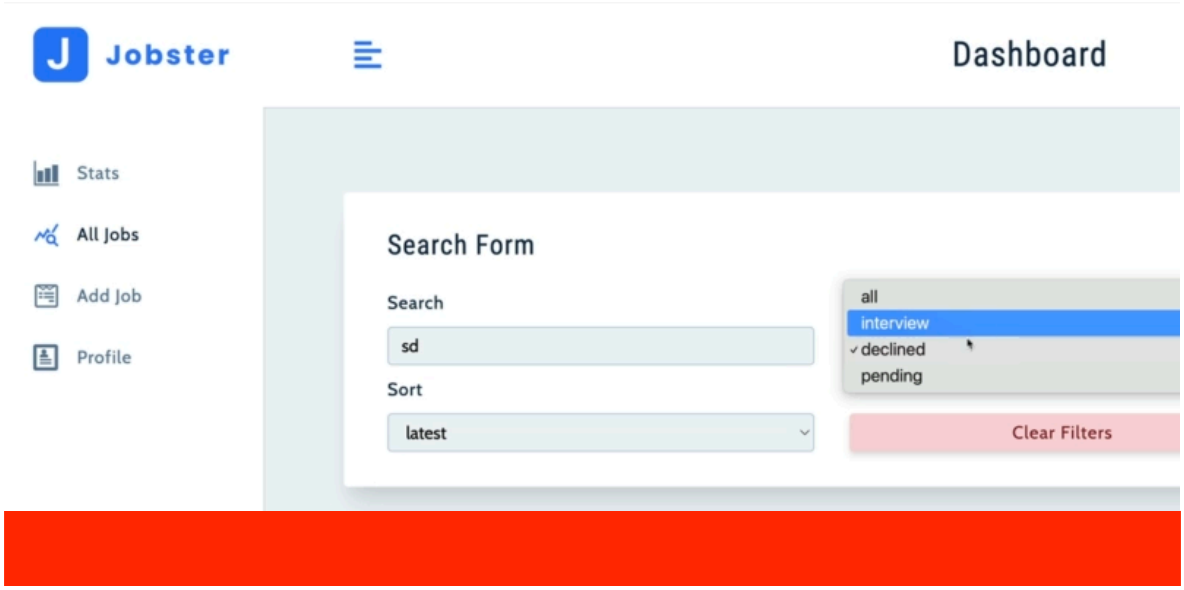


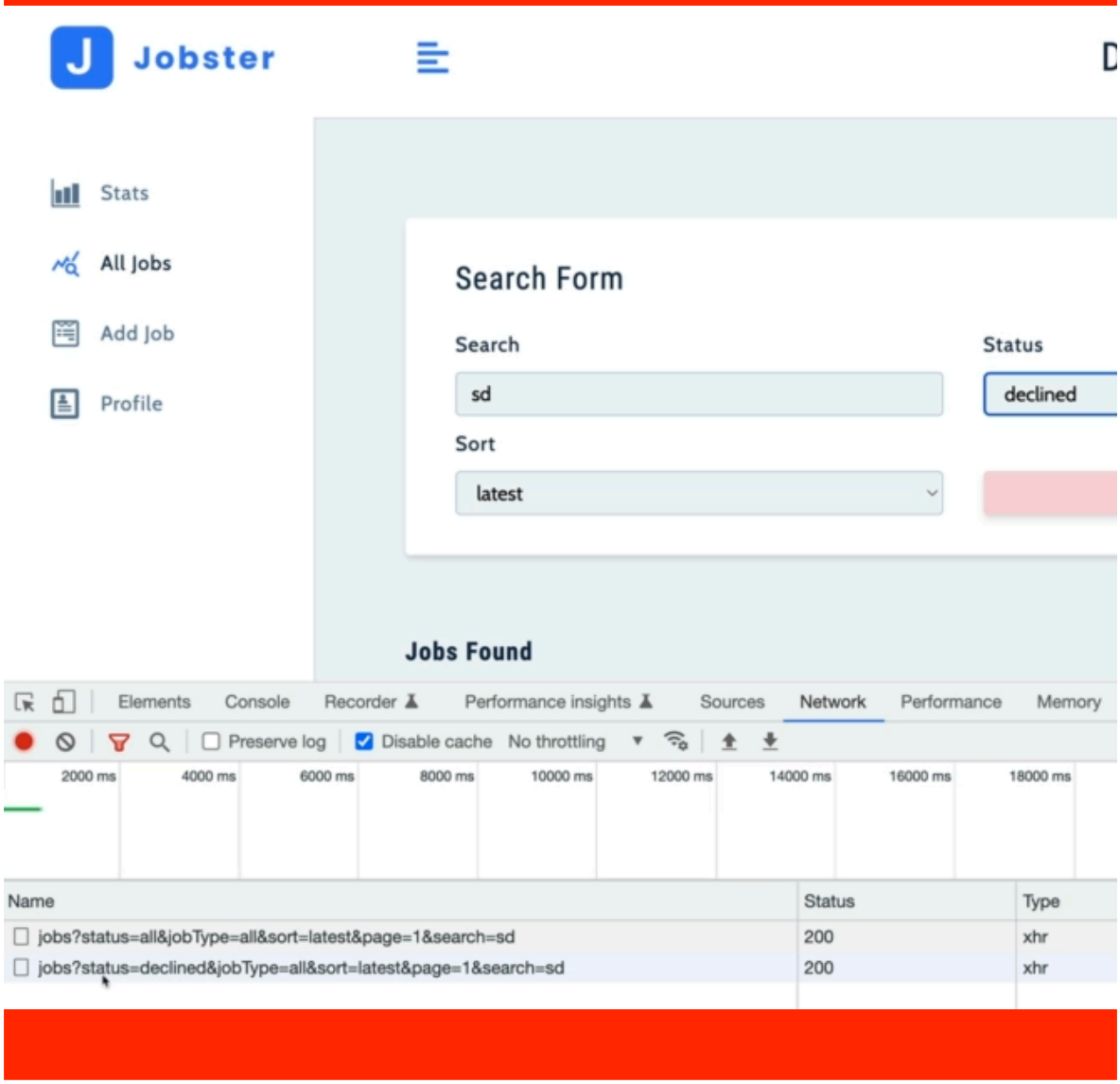
15 - Search Functionality

==========



change in the Status


Jobster
Dashboard
Stats
All Jobs
Add Job
Profile
Search Form
Search
sd
Sort
latest
all
interview
declined
pending
Clear Filters


Jobster
Stats
All Jobs
Add Job
Profile
Search Form
Search
sd
Status
declined
Sort
latest
Jobs Found
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
Preserve log
Disable cache
No throttling
2000 ms
4000 ms
6000 ms
8000 ms
10000 ms
12000 ms
14000 ms
16000 ms
18000 ms
Name
Status
Type
jobs?status=all&jobType=all&sort=latest&page=1&search=sd
200
xhr
jobs?status=declined&jobType=all&sort=latest&page=1&search=sd
200
xhr

```
#### Modify Get All Jobs

- latest mongoose version change

controllers/jobs.js

```js
const getAllJobs = async (req, res) => {
  const { search, status, jobType, sort } = req.query;

  // protected route
  const queryObject = {
    createdBy: req.user.userId,
  };

  if (search) {
    queryObject.position = { $regex: search, $options: 'i' };
  }
  // add stuff based on condition

  if (status && status !== 'all') {
    queryObject.status = status;
  }
  if (jobType && jobType !== 'all') {
    queryObject.jobType = jobType;
  }

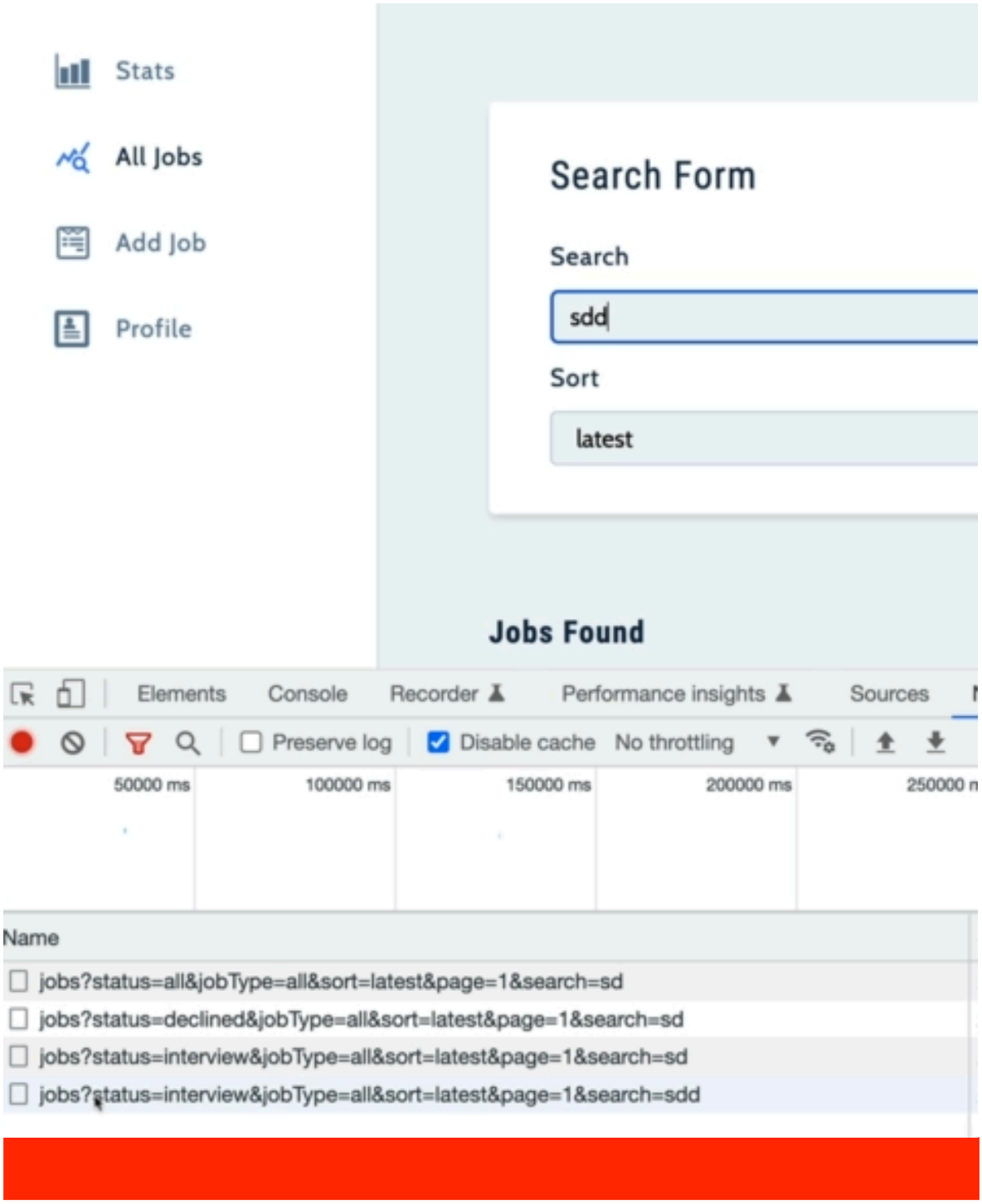
  // NO AWAIT

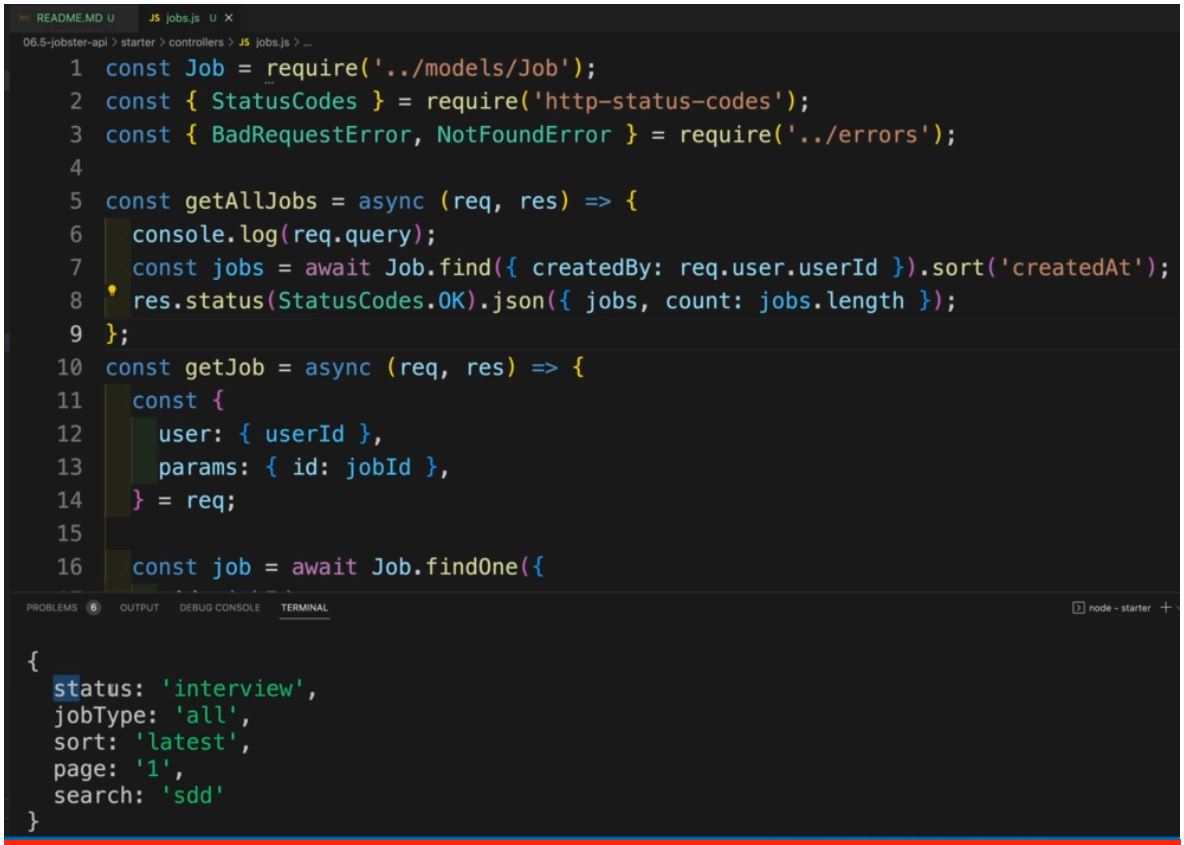
  let result = Job.find(queryObject);

  // chain sort conditions

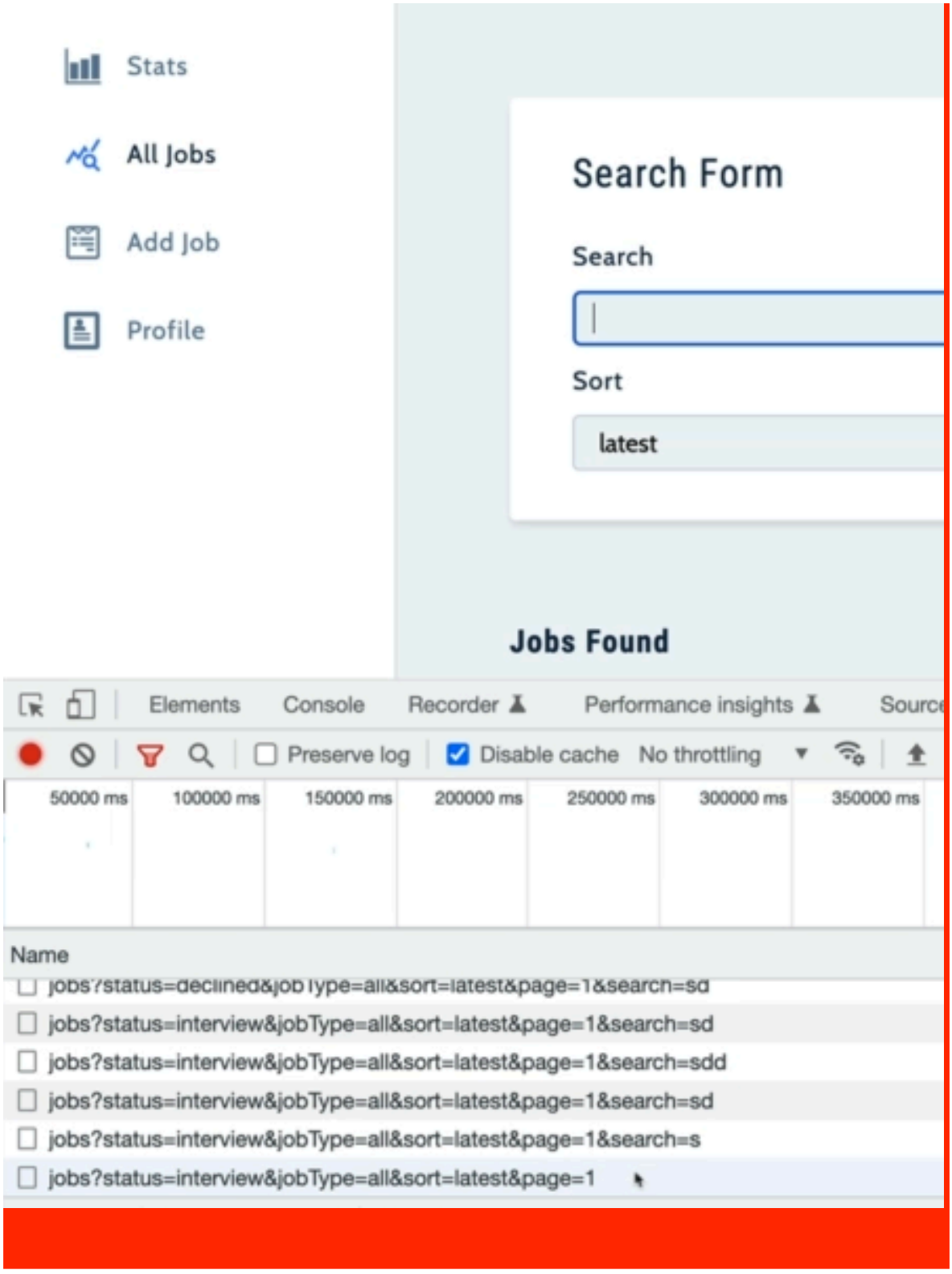
  if (sort === 'latest') {
    result = result.sort('-createdAt');
  }
  if (sort === 'oldest') {
    result = result.sort('createdAt');
  }
  if (sort === 'a-z') {
    result = result.sort('position');
  }
  if (sort === 'z-a') {
    result = result.sort('-position');
  }
```
```

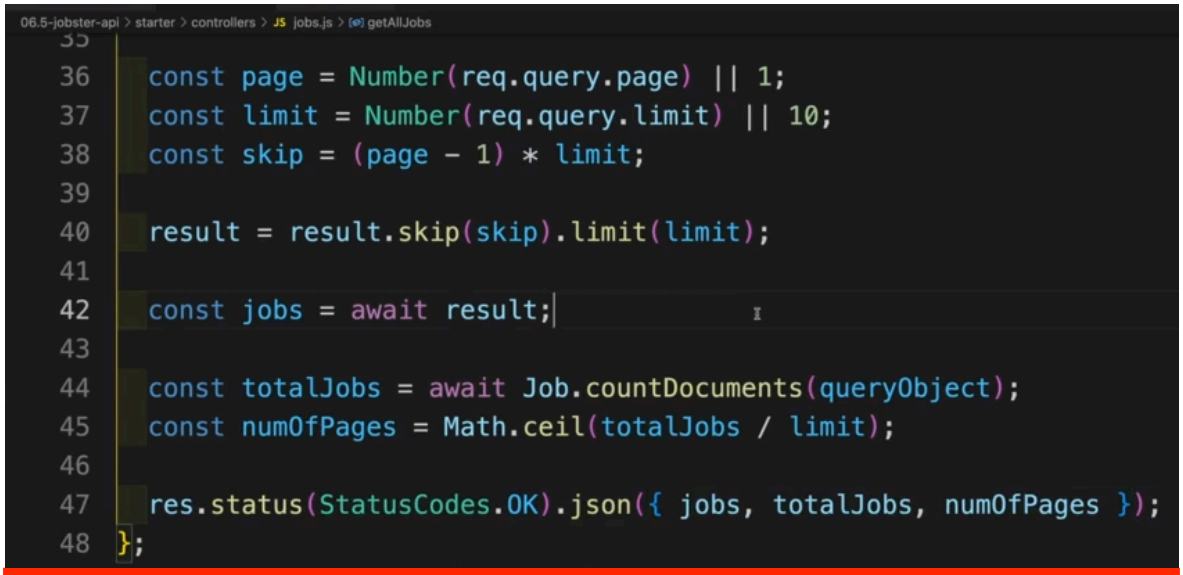
```
417    // setup pagination
418    const page = Number(req.query.page) || 1;
419    const limit = Number(req.query.limit) || 10;
420    const skip = (page - 1) * limit;
421
422    result = result.skip(skip).limit(limit);
423
424    const jobs = await result;
425
426    const totalJobs = await Job.countDocuments(queryObject);
427    const numOfPages = Math.ceil(totalJobs / limit);
428
429    res.status(StatusCodes.OK).json({ jobs, totalJobs, numOfPages });
430  };
431  ```
```


Stats
All Jobs
Add Job
Profile
Search Form
Search
sdd
Sort
latest
Jobs Found
Elements
Console
Recorder
Performance insights
Sources
Preserve log
Disable cache
No throttling
50000 ms
100000 ms
150000 ms
200000 ms
Name
jobs?status=all&jobType=all&sort=latest&page=1&search=sd
jobs?status=declined&jobType=all&sort=latest&page=1&search=sd
jobs?status=interview&jobType=all&sort=latest&page=1&search=sd
jobs?status=interview&jobType=all&sort=latest&page=1&search=sdd



if search is empty


Stats
All Jobs
Add Job
Profile
Search Form
Search
Sort
latest
Jobs Found
Elements
Console
Recorder
Performance insights
Preserve log
Disable cache
No throttling
50000 ms
100000 ms
150000 ms
200000 ms
250000 ms
300000 ms
350000 ms
Name
jobs?status=declined&jobType=all&sort=latest&page=1&search=sd
jobs?status=interview&jobType=all&sort=latest&page=1&search=sd
jobs?status=interview&jobType=all&sort=latest&page=1&search=sdd
jobs?status=interview&jobType=all&sort=latest&page=1&search=sd
jobs?status=interview&jobType=all&sort=latest&page=1&search=s
jobs?status=interview&jobType=all&sort=latest&page=1




```
const getAllJobs = async (req, res) => {
  const { search, status, jobType, sort }
= req.query;

  const queryObject = {
    createdBy: req.user.userId,
  };

  if (search) {
    queryObject.position = { $regex:
search, $options: 'i' };
  }
  if (status && status !== 'all') {
    queryObject.status = status;
  }
  if (jobType && jobType !== 'all') {
```

```
    queryObject.jobType = jobType;
  }
  let result = Job.find(queryObject);

  if (sort === 'latest') {
    result = result.sort('-createdAt');
  }
  if (sort === 'oldest') {
    result = result.sort('createdAt');
  }
  if (sort === 'a-z') {
    result = result.sort('position');
  }
  if (sort === 'z-a') {
    result = result.sort('-position');
  }

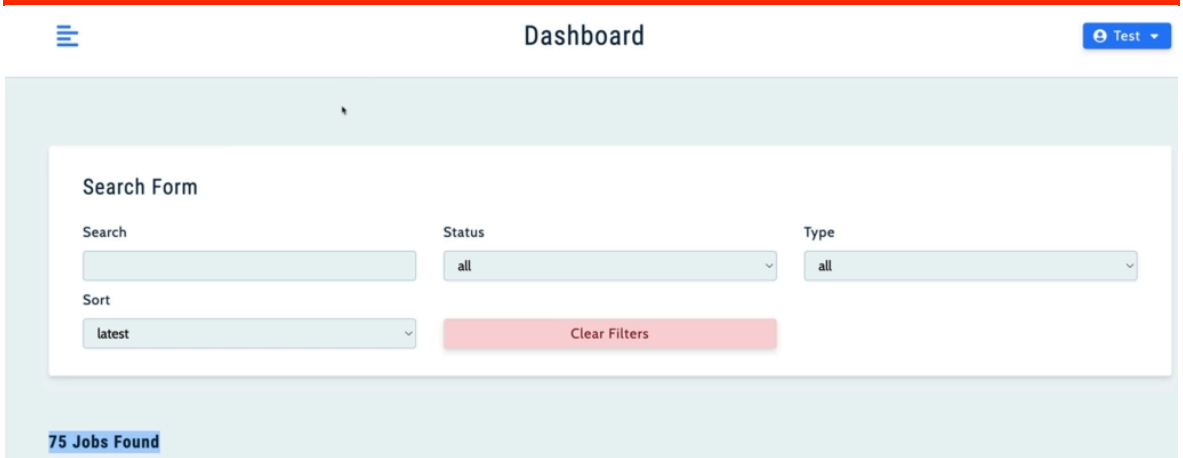
  const page = Number(req.query.page)
|| 1;
  const limit = Number(req.query.limit)
|| 10;
  const skip = (page - 1) * limit;

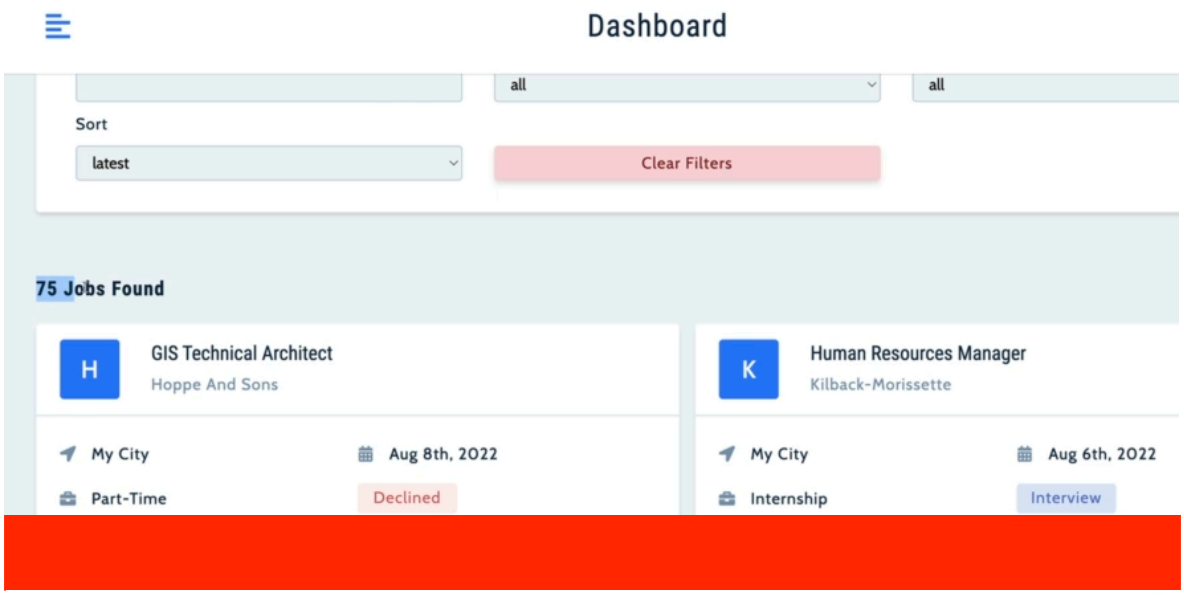
  result = result.skip(skip).limit(limit);
```

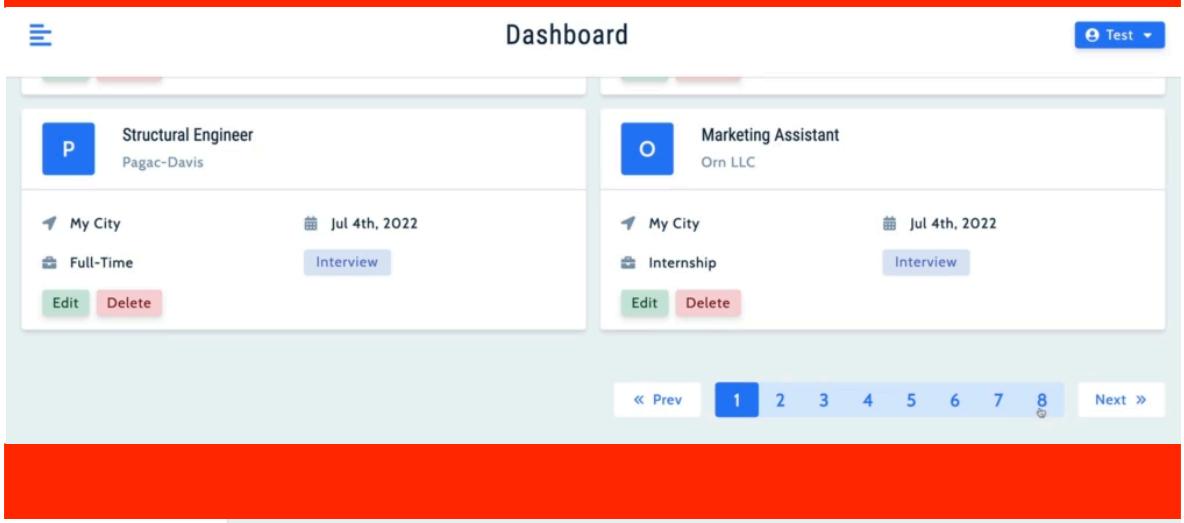
```
  const jobs = await result;

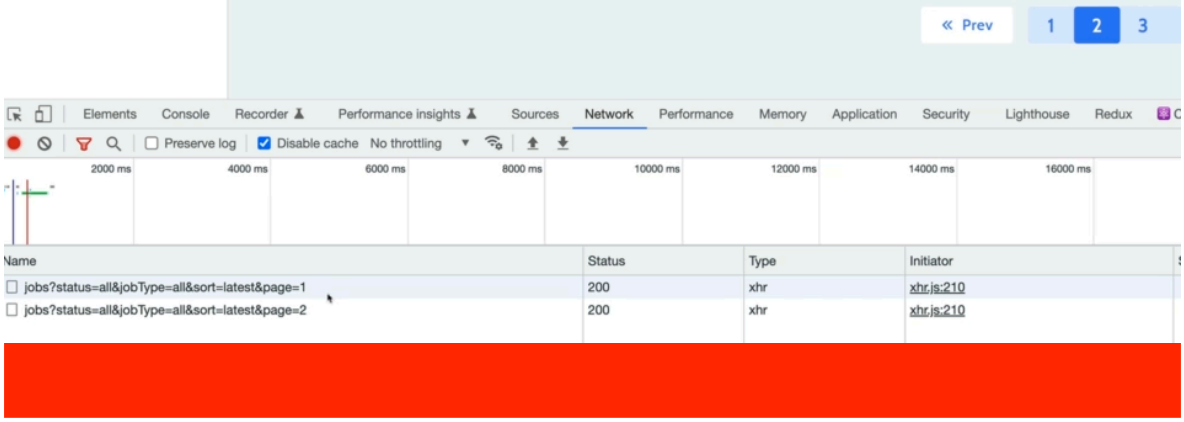
  const totalJobs = await
Job.countDocuments(queryObject);
  const numOfPages =
Math.ceil(totalJobs / limit);

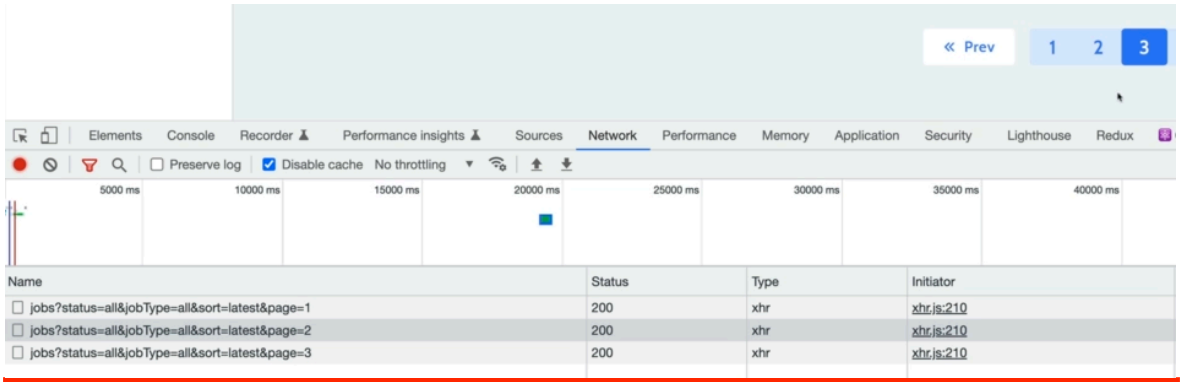

res.status(StatusCodes.OK).json({ jobs
, totalJobs, numOfPages });
};
```




Dashboard
all
all
Sort
latest
Clear Filters
75 Jobs Found
H
GIS Technical Architect
Hoppe And Sons
My City
Aug 8th, 2022
Part-Time
Declined
K
Human Resources Manager
Kilback-Morissette
My City
Aug 6th, 2022
Internship
Interview


Dashboard
Test
P
Structural Engineer
Pagac-Davis
My City
Jul 4th, 2022
Full-Time
Interview
Edit
Delete
O
Marketing Assistant
Orn LLC
My City
Jul 4th, 2022
Internship
Interview
Edit
Delete
« Prev
1
2
3
4
5
6
7
8
Next »

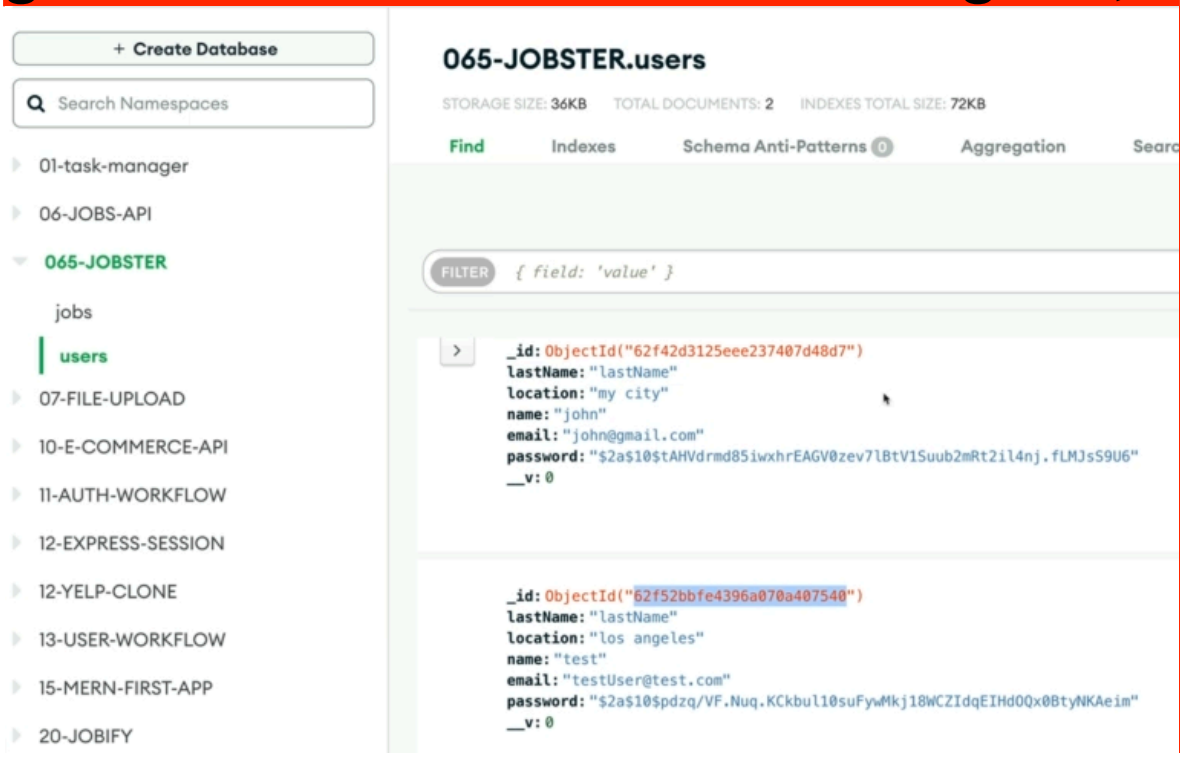

« Prev
1
2
3
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Redux
Preserve log
Disable cache
No throttling
2000 ms
4000 ms
6000 ms
8000 ms
10000 ms
12000 ms
14000 ms
16000 ms
Name
Status
Type
Initiator
jobs?status=all&jobType=all&sort=latest&page=1
200
xhr
xhr.js:210
jobs?status=all&jobType=all&sort=latest&page=2
200
xhr
xhr.js:210



20 - Check for test user in auth Middleware

==========

test user should not do any CRUD operations, so
get the test userId from the mongodb,



middleware/authentication.js

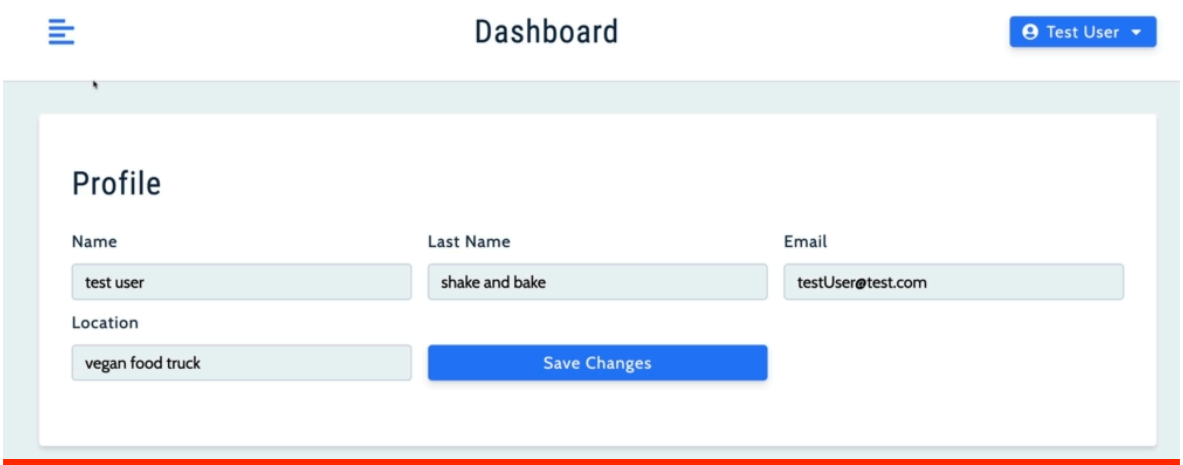
==========

```
const User = require('../models/User');
const jwt = require('jsonwebtoken');
const { UnauthenticatedError } =
require('../errors');

const auth = async (req, res, next) => {
  // check header
  const authHeader =
req.headers.authorization;
  if (!authHeader || !
authHeader.startsWith('Bearer')) {
    throw new
UnauthenticatedError('Authentication
invalid');
  }
  const token = authHeader.split(' ')[1];
```

```
  try {
    const payload = jwt.verify(token,
process.env.JWT_SECRET);
    // attach the user to the job routes
    const testUser = payload.userId ===
'62f801d0510a7c1ed2312d52';
    req.user = { userId: payload.userId,
testUser };
    next();
  } catch (error) {
    throw new
UnauthenticatedError('Authentication
invalid');
  }
};

module.exports = auth;
```




```
443  - create testingUser in middleware
444
445  middleware/testUser
446
447  ```js
448  const { BadRequestError } = require('../errors');
449
450  const testUser = (req, res, next) => {
451    if (req.user.testUser) {
452      throw new BadRequestError('Test User. Read Only!');
453    }
454    next();
455  };
456
457  module.exports = testUser;
458  ```
```

middleware/testUser

==========

```
const { BadRequestError } = require('../errors');

const testUser = (req, res, next) => {
  if (req.user.testUser) {
    throw new BadRequestError('Test
```

User. Read Only');
  }
  next();
};

module.exports = testUser;

```
- add to auth routes (updateUser)

```js
const express = require('express');
const router = express.Router();
const authenticateUser = require('../middleware/authentication');
const testUser = require('../middleware/testUser');
const { register, login, updateUser } = require('../controllers/auth')
  ;

router.post('/register', register);
router.post('/login', login);
router.patch('/updateUser', authenticateUser, testUser, updateUser);

module.exports = router;
```
```

routes/auth.js
==========
const express = require('express');
const router = express.Router();
const authenticateUser = require('../
middleware/authentication');
const testUser = require('../middleware/
testUser');

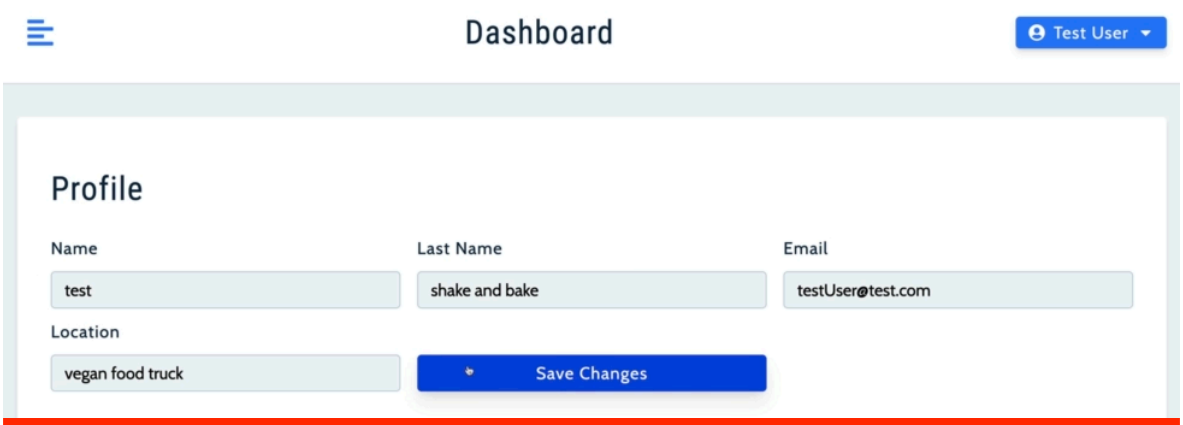
const { register, login, updateUser } =

```
require('../controllers/auth');

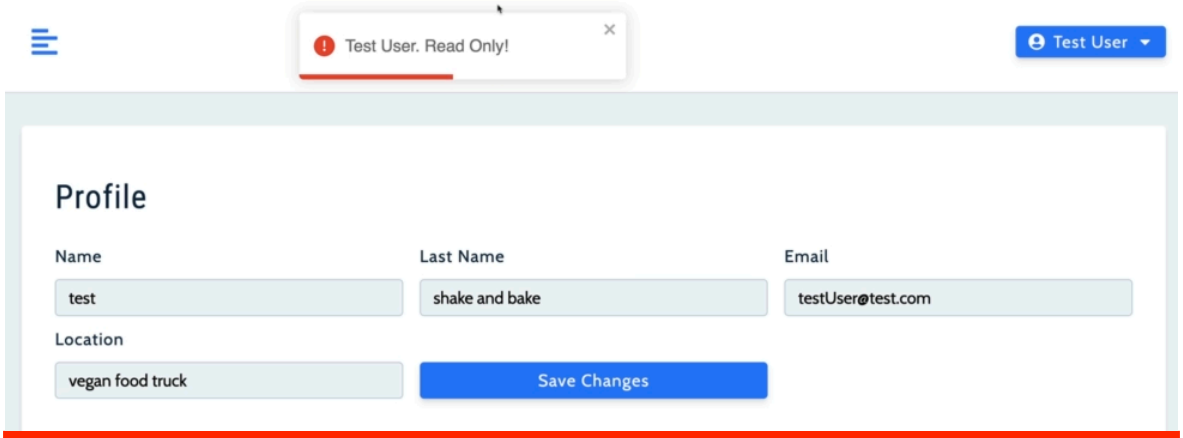
router.post('/register', register);
router.post('/login',  login);
router.patch('/updateUser',
authenticateUser, testUser,
updateUser);

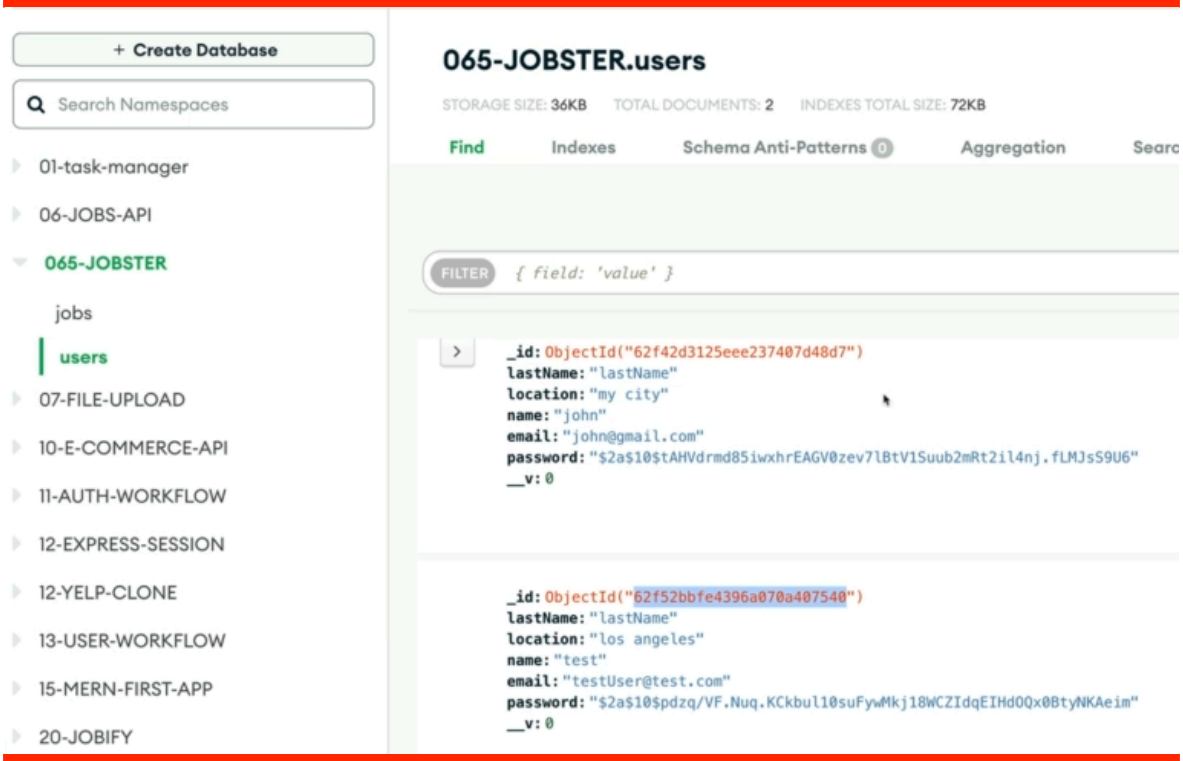
module.exports = router;
```

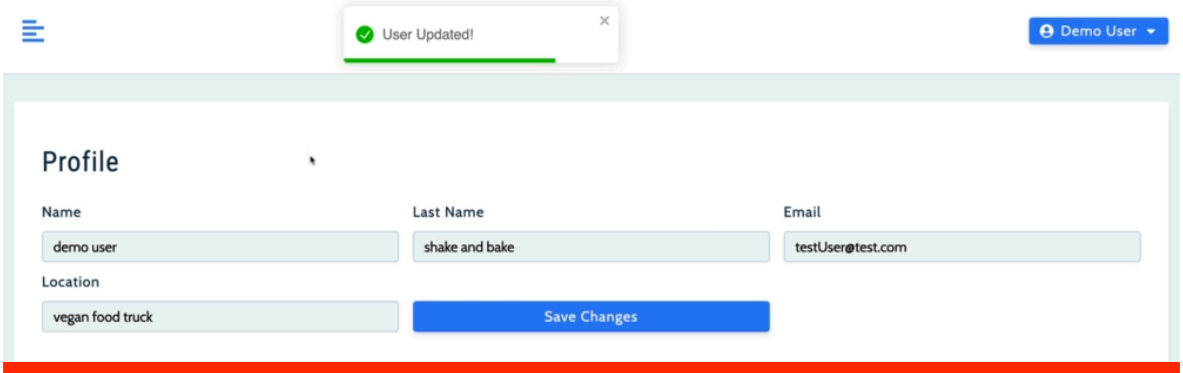
if we change the Name (test user to test) in the Profile page



we get back alert

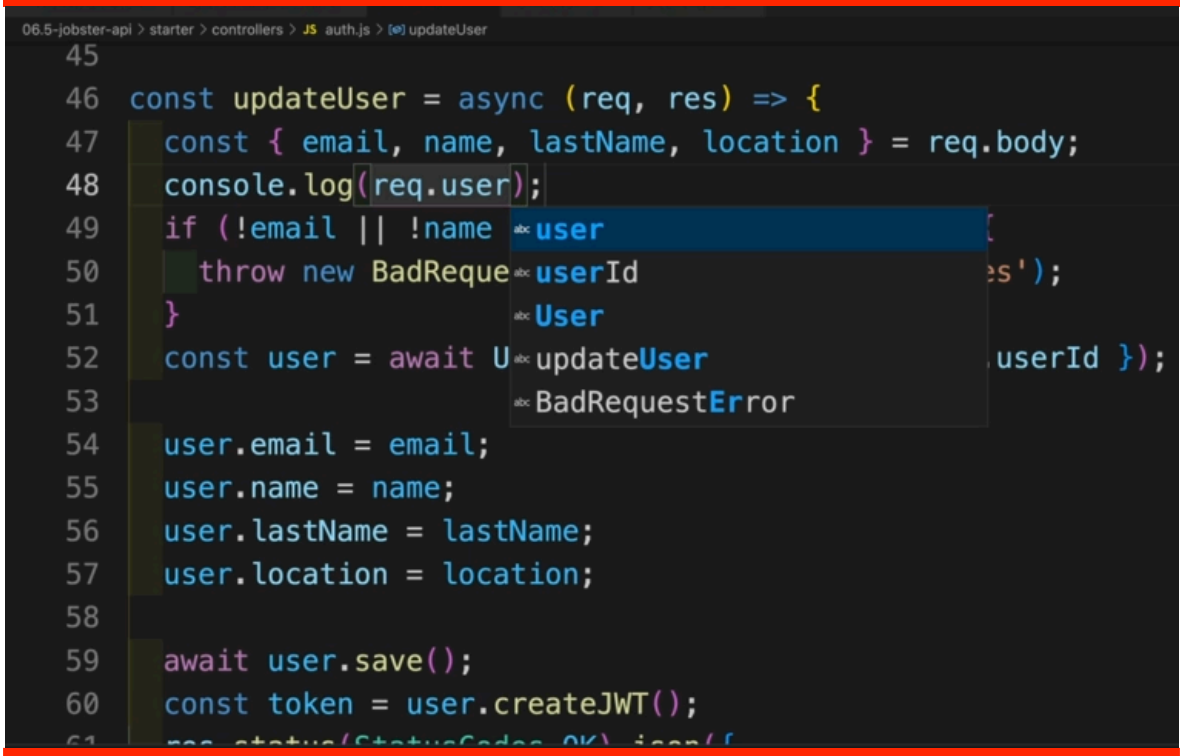






/controllers/auth.js

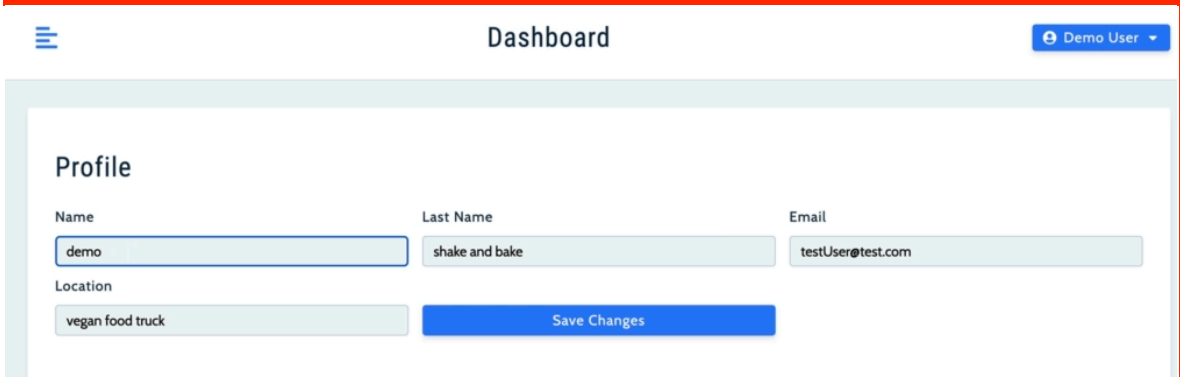
==========

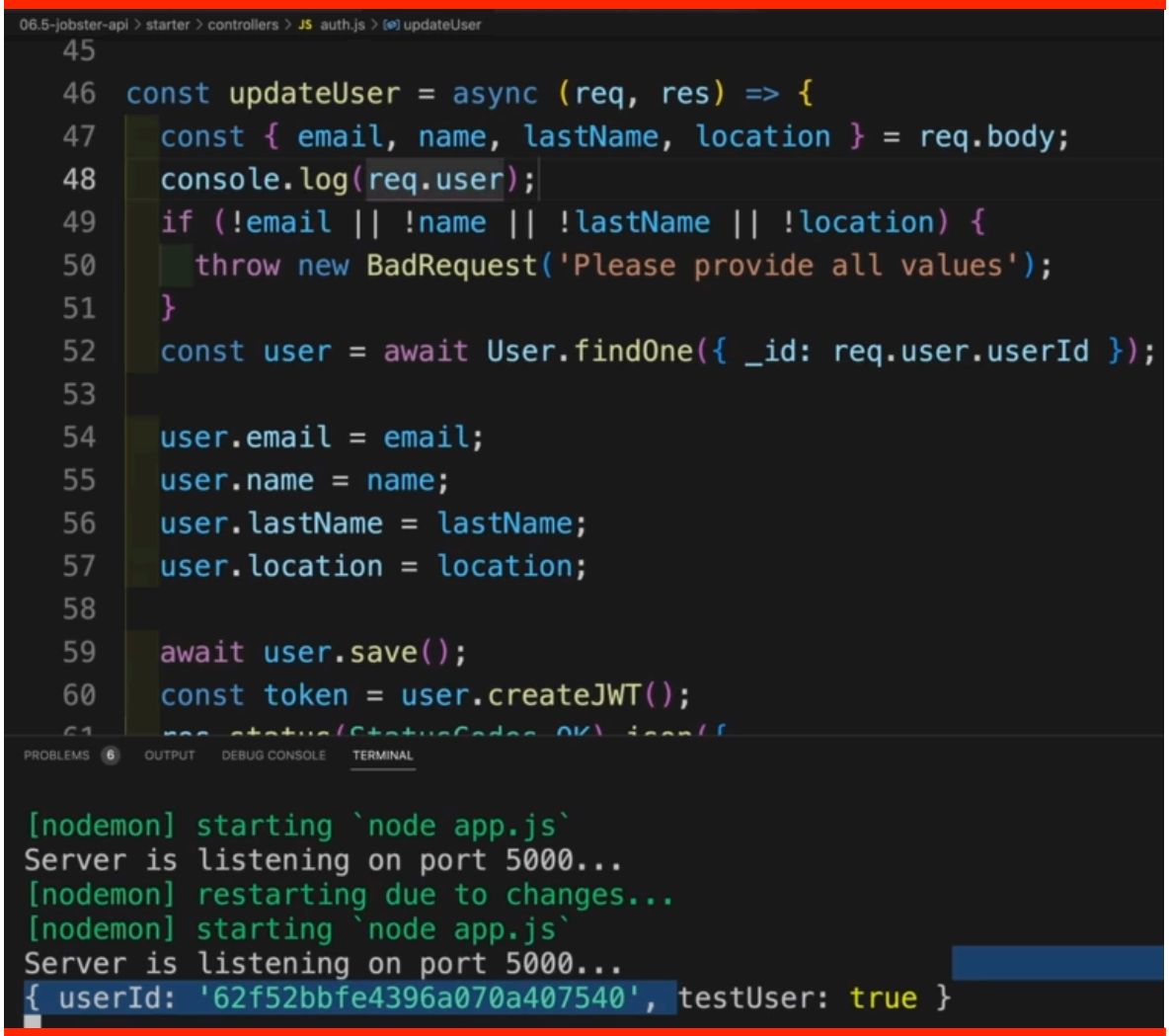



```
const updateUser = async (req, res) =>
{
  const { email, name, lastName,
location } = req.body;
  console.log(req.user);
  if (!email || !name || !lastName || !
location) {
    throw new BadRequest('Please
provide all values');
  }
  const user = await User.findOne({ _id:
req.user.userId });
```

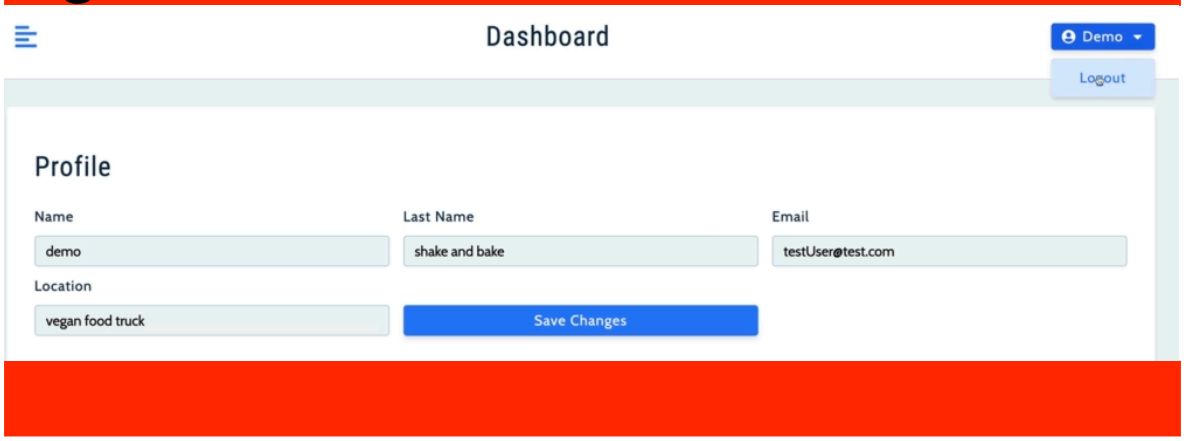
```
user.email = email;
user.name = name;
user.lastName = lastName;
user.location = location;

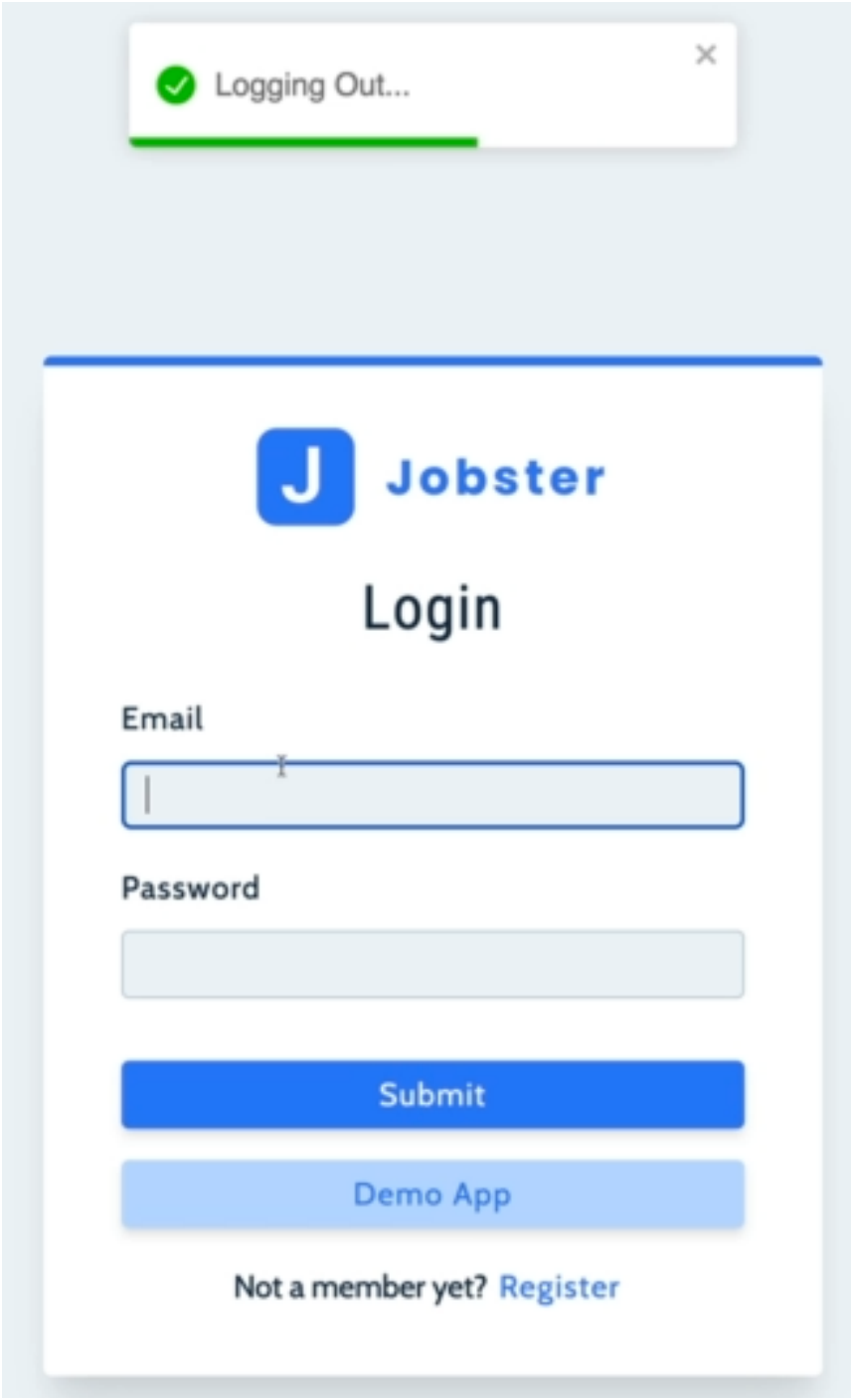
await user.save();
const token = user.createJWT();
res.status(StatusCodes.OK).json({
  user: {
    email: user.email,
    lastName: user.lastName,
    location: user.location,
    name: user.name,
    token,
  },
 });
};
```

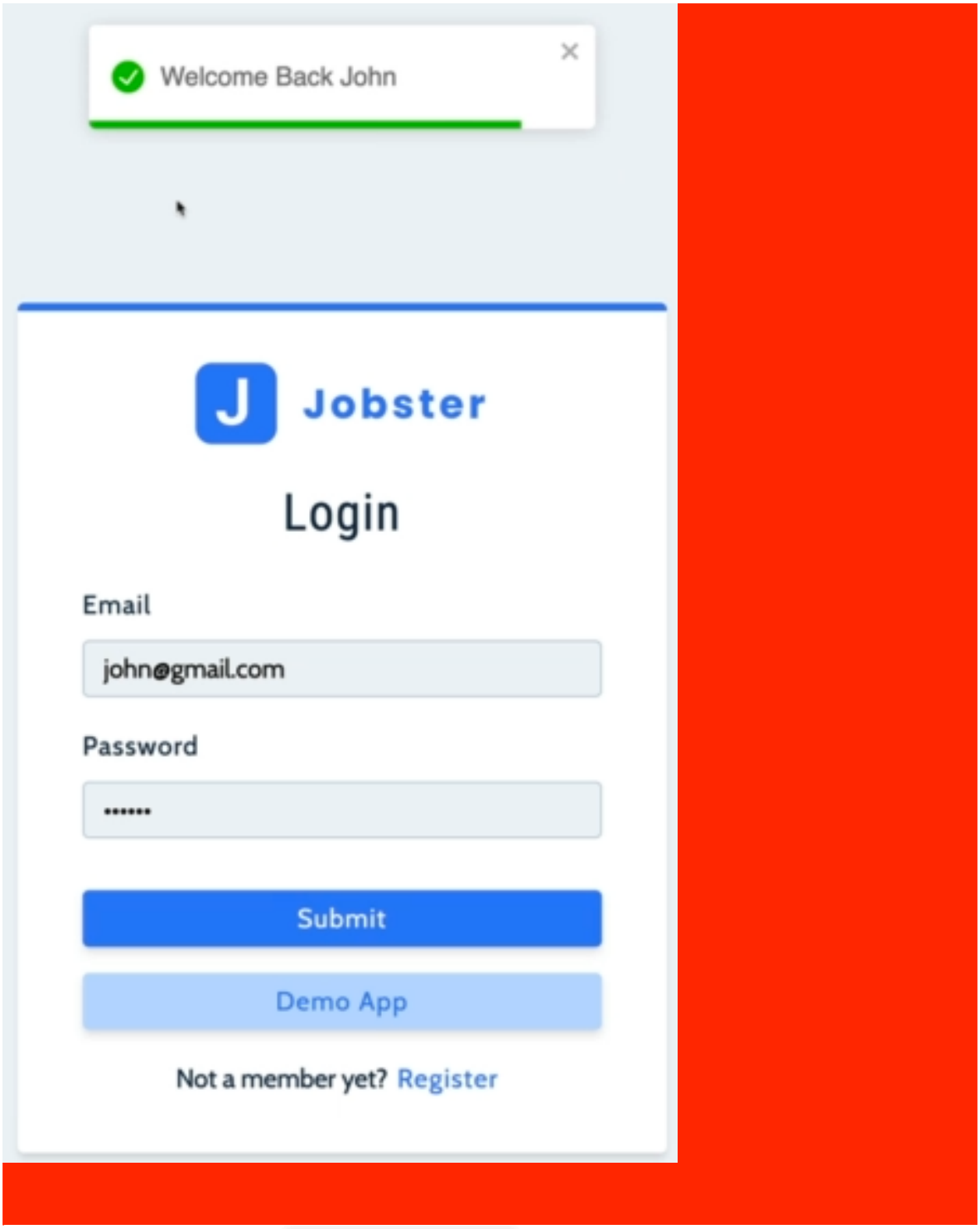


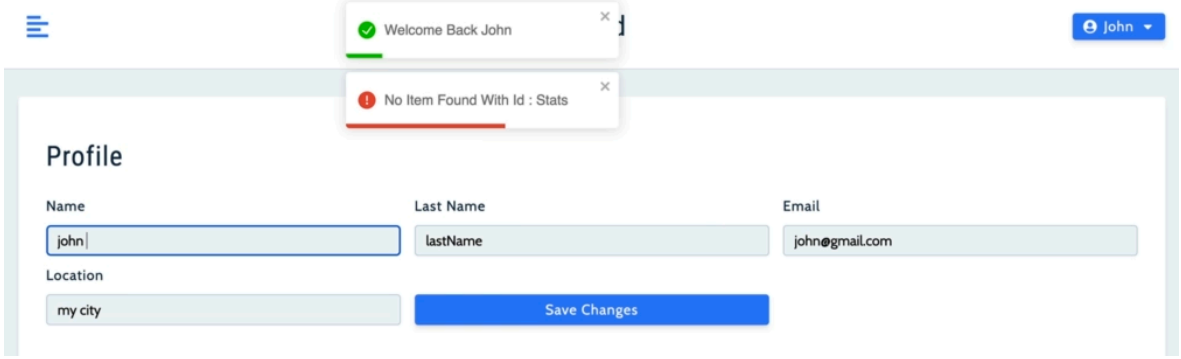

06.5-jobster-api > starter > controllers > JS auth.js > updateUser
45
46 const updateUser = async (req, res) => {
47 const { email, name, lastName, location } = req.body;
48 console.log(req.user);
49 if (!email || !name || !lastName || !location) {
50 throw new BadRequest('Please provide all values');
51 }
52 const user = await User.findOne({ _id: req.user.userId });
53
54 user.email = email;
55 user.name = name;
56 user.lastName = lastName;
57 user.location = location;
58
59 await user.save();
60 const token = user.createJWT();
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL
[nodemon] starting `node app.js`
Server is listening on port 5000...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Server is listening on port 5000...
{ userId: '62f52bbfe4396a070a407540', testUser: true }

log out

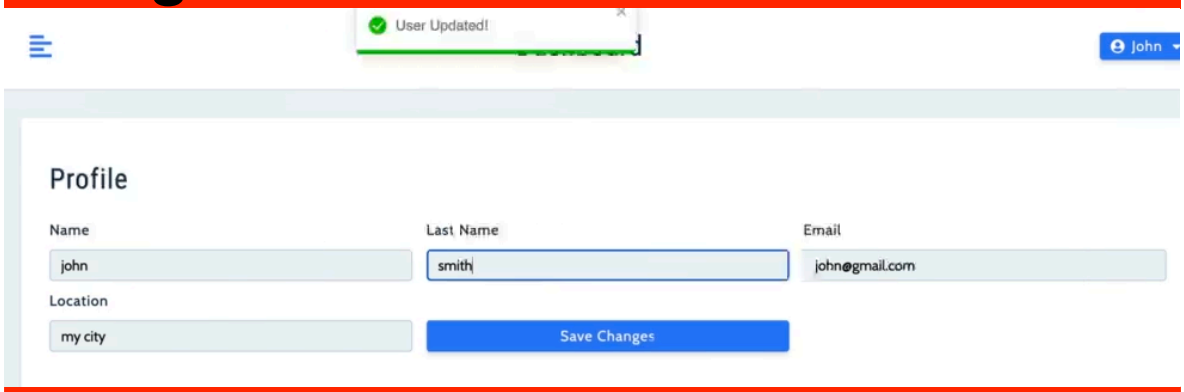

Dashboard
Demo
Logout
Profile
Name
demo
Last Name
shake and bake
Email
testUser@test.com
Location
vegan food truck
Save Changes

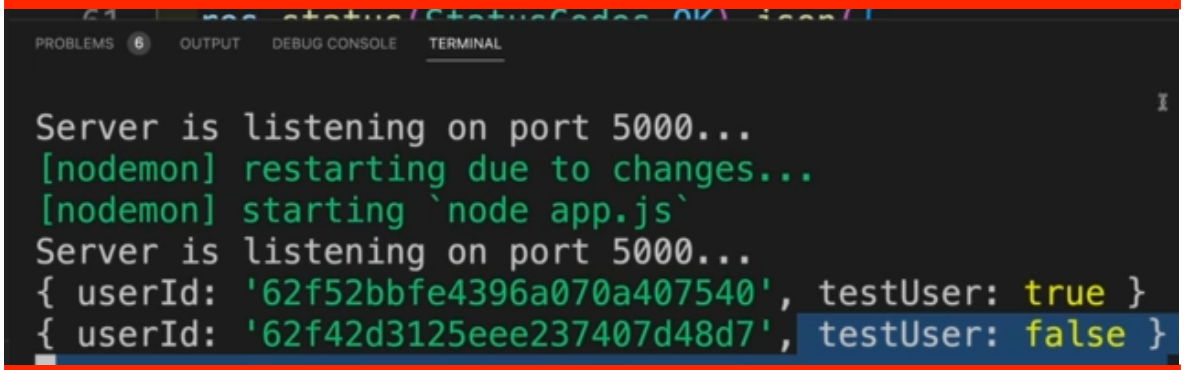

Logging Out...
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register

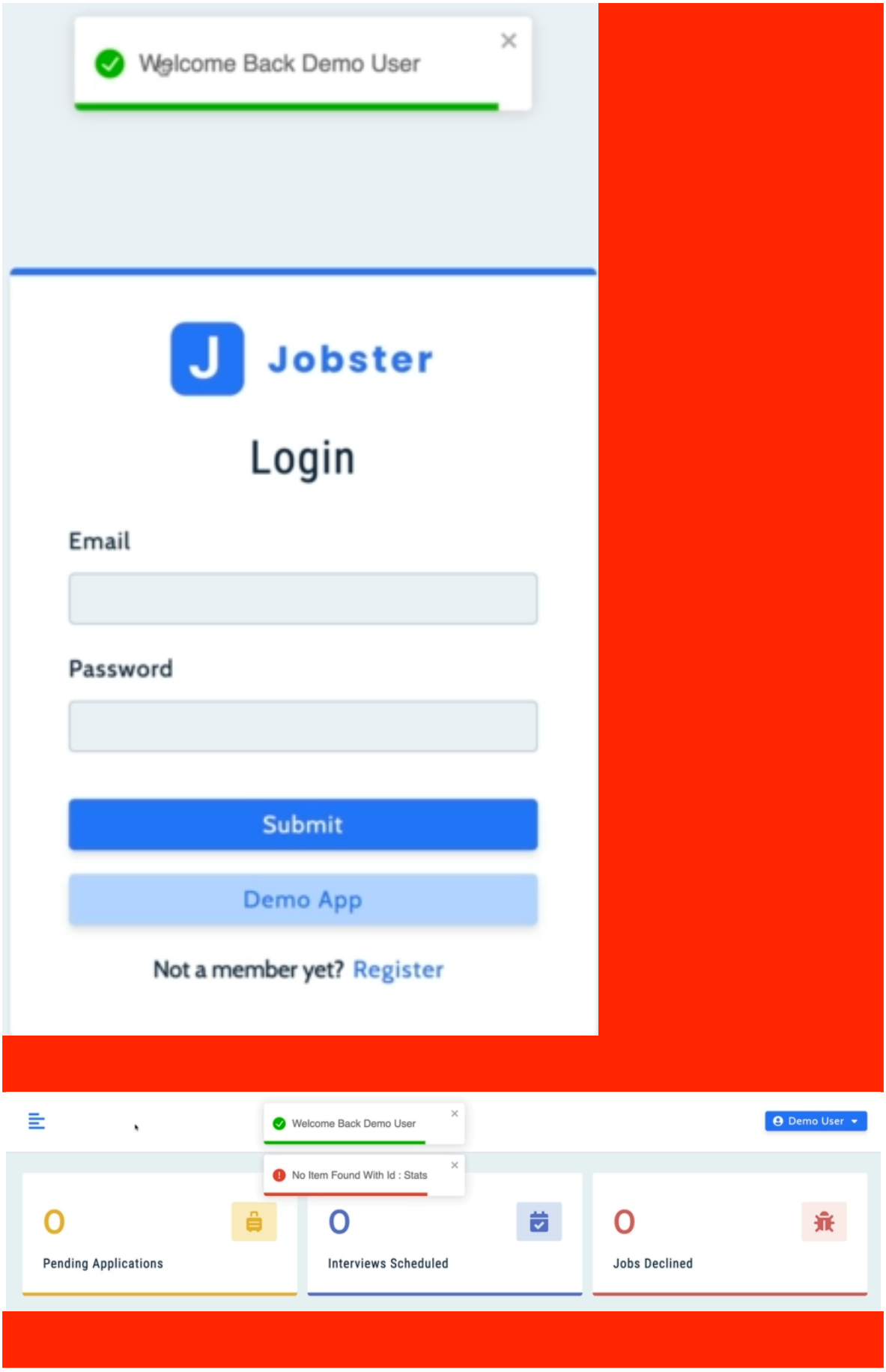

Welcome Back John
Jobster
Login
Email
john@gmail.com
Password
Submit
Demo App
Not a member yet? Register


Welcome Back John
No Item Found With Id : Stats
John
Profile
Name
john
Last Name
lastName
Email
john@gmail.com
Location
my city
Save Changes

change last name






Welcome Back Demo User
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register

Welcome Back Demo User
No Item Found With Id : Stats
Demo User
0
Pending Applications
0
Interviews Scheduled
0
Jobs Declined

21 - Restrict CRUD to Test User

==========

```
- add to job routes (createJob, updateJob, deleteJob)

routes/jobs.js

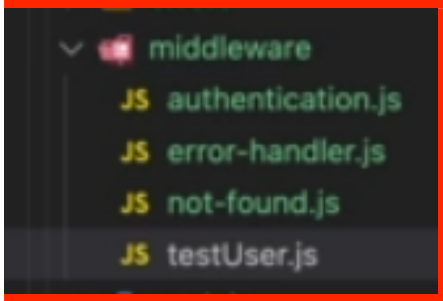
```js
const express = require('express');

const router = express.Router();
const {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
} = require('../controllers/jobs');
const testUser = require('../middleware/testUser');

router.route('/').post(testUser, createJob).get(getAllJobs);
router.route('/stats').get(showStats);
router
  .route('/:id')
  .get(getJob)
  .delete(testUser, deleteJob)
  .patch(testUser, updateJob);

module.exports = router;
```
```

create testUser.js file under middleware folder



middleware/testUser.js

==========

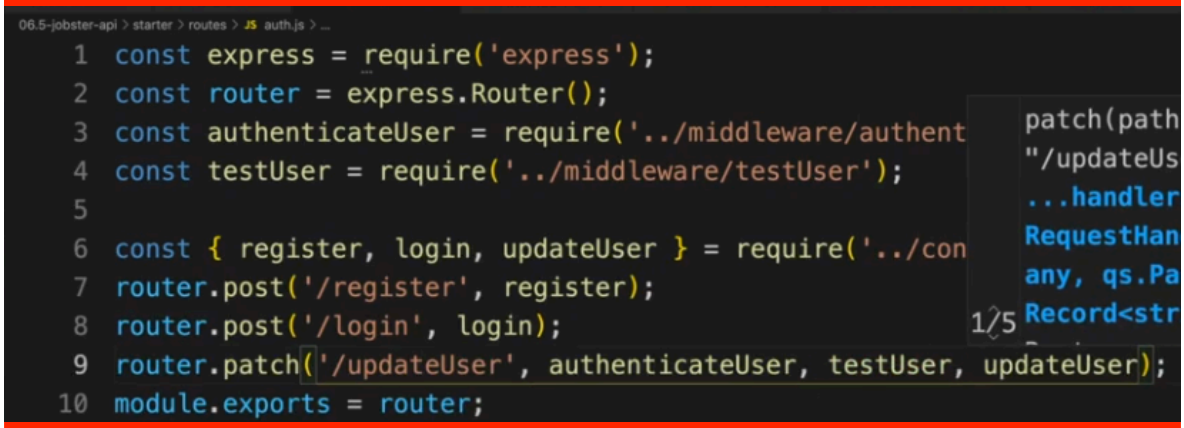
```
const { BadRequestError } = require('../errors');

const testUser = (req, res, next) => {
  if (req.user.testUser) {
    throw new BadRequestError('Test User. Read Only');
  }
  next();
};

module.exports = testUser;
```

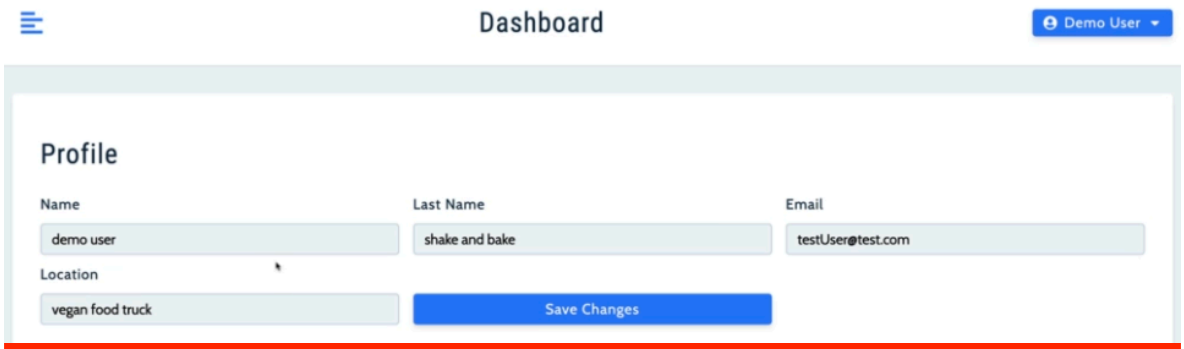
/routes/auth.js

==========

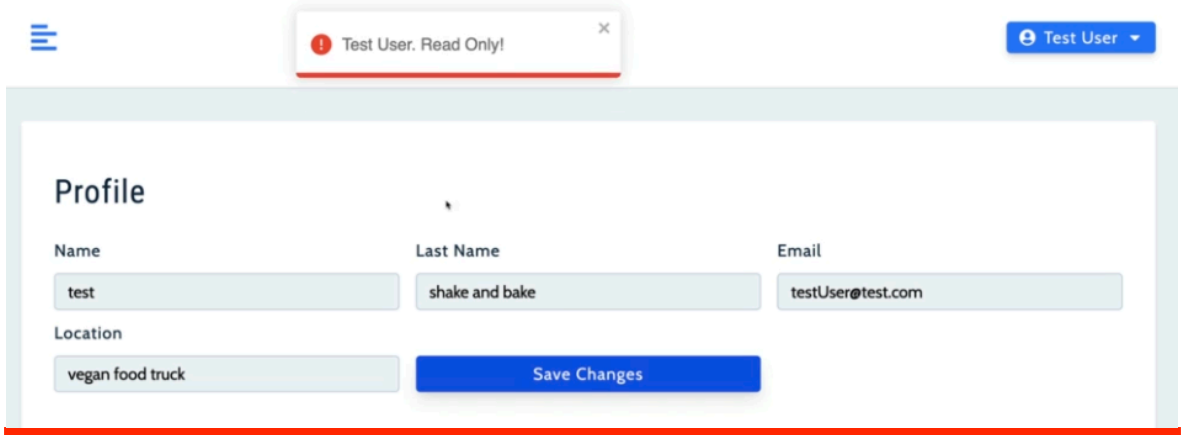



```
const express = require('express');
const router = express.Router();
const authenticateUser = require('../
middleware/authentication');
const testUser = require('../middleware/
testUser');

const { register, login, updateUser } =
require('../controllers/auth');
router.post('/register',  register);
router.post('/login',  login);
router.patch('/updateUser',
authenticateUser, testUser,
updateUser);
module.exports = router;
```



change the name and send request



/routes/jobs.js

==========

```
const testUser = require('../middleware/testUser');
```

06.5-jobster-api > starter > routes > JS jobs.js > ...

```
1  const express = require('express')
2  const testUser = require('../middleware/testUser');
3
4  const router = express.Router()
5  const {
6    createJob,
7    deleteJob,
8    getAllJobs,
9    updateJob,
10   getJob,
11 } = require('../controllers/jobs')
12
13 router.route('/').post(createJob).get(getAllJobs)
14
15 router.route('/:id').get(getJob).delete(deleteJob).patch(updateJob)
```

router.route('/').post(testUser, createJob).get(getAllJobs);

router
.route('/:id')
.get(getJob)
.delete(testUser, deleteJob)
.patch(testUser, updateJob);

06.5-jobster-api > starter > routes > JS jobs.js > ...

```
5  const {
6    createJob,
7    deleteJob,
8    getAllJobs,
9    updateJob,
10   getJob,
11 } = require('../controllers/jobs')
12
13 router.route('/').post(testUser,createJob).get(getAllJobs)
14
15 router.route('/:id').get(getJob).delete(testUser,deleteJob).patch(testUser,updateJob)
16
17 module.exports = router
```

patch(...handlers: RequestHandler<{ id: string; }, any, any, qs.ParsedQs, Record<string, any>>[]

1/6 IRoute<"/:id">

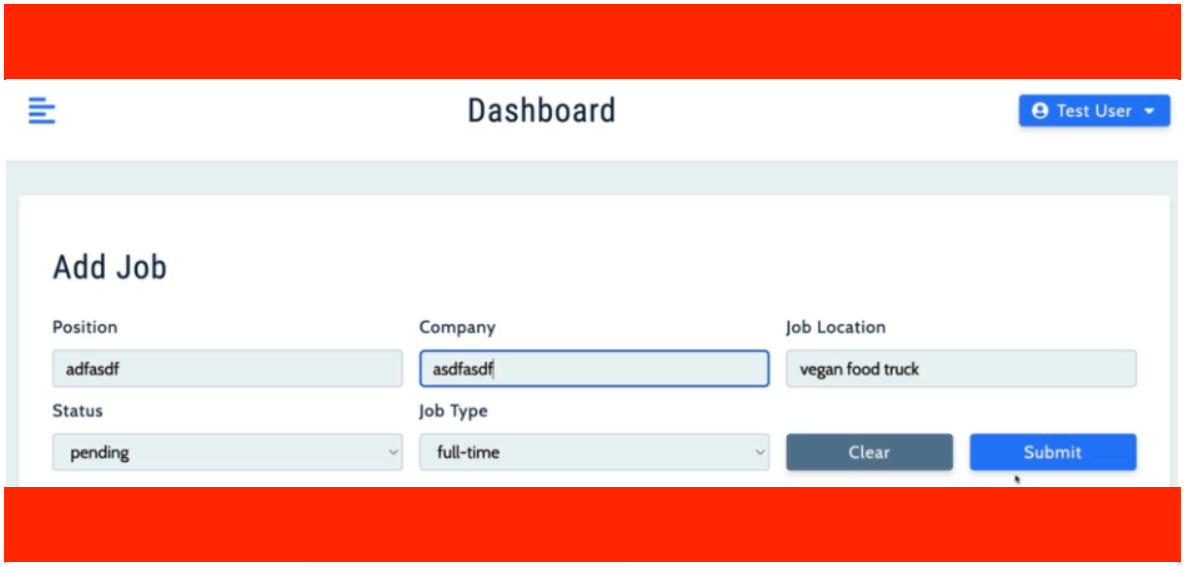
```
const express = require('express');
const testUser = require('../middleware/
testUser');

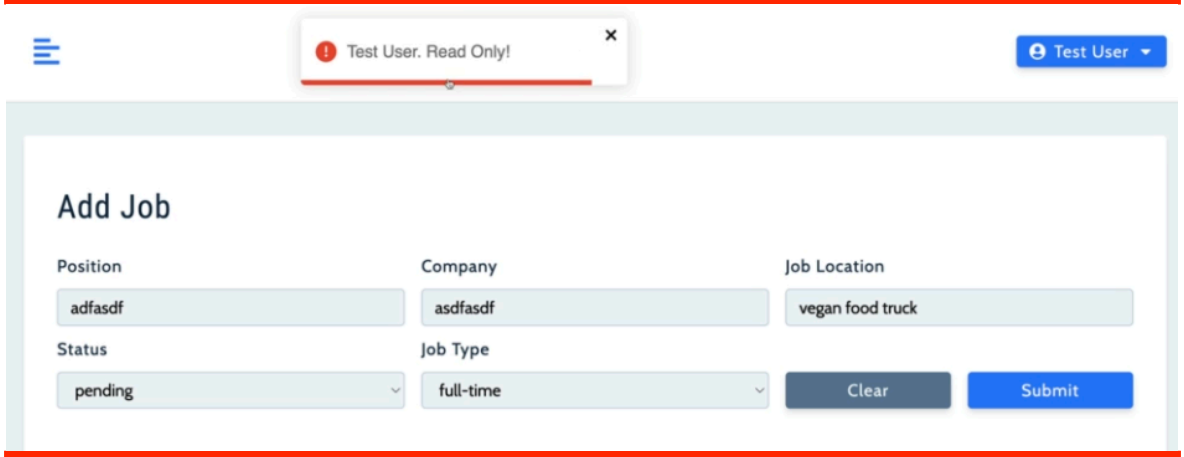
const router = express.Router();
const {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
} = require('../controllers/jobs');

router.route('/').post(testUser,
createJob).get(getAllJobs);

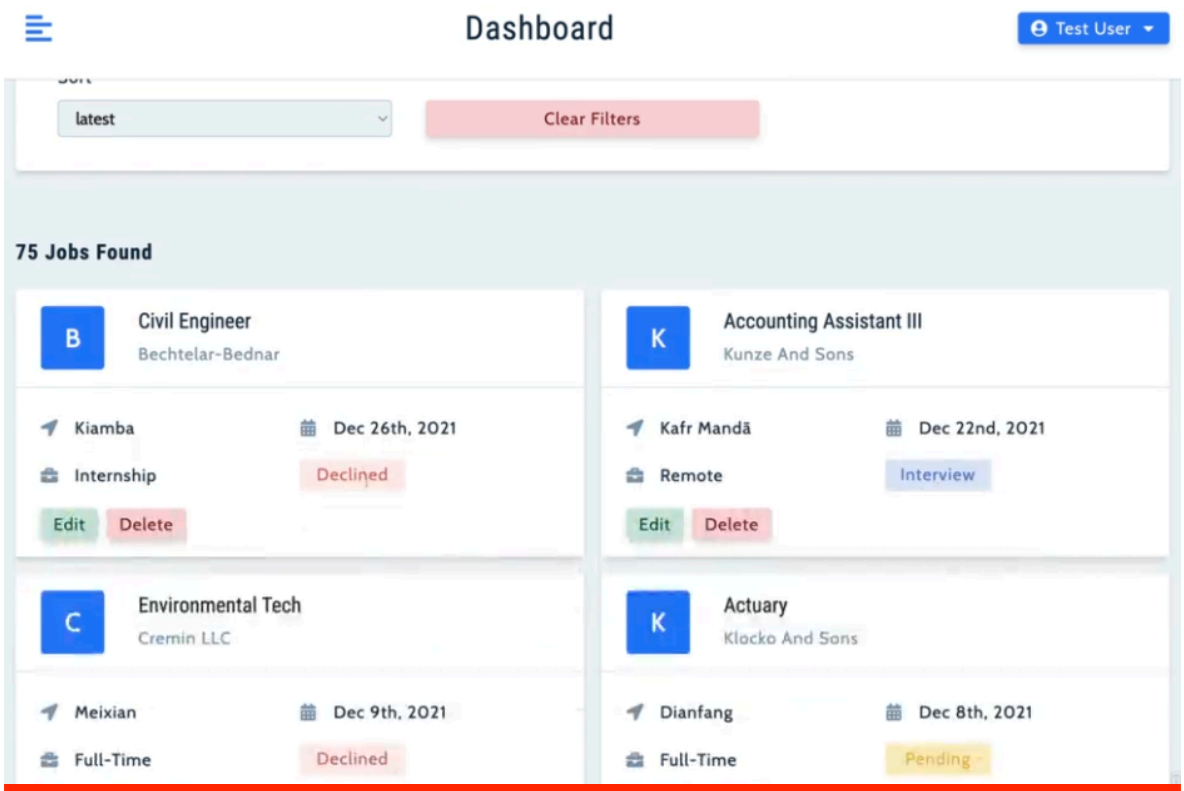
router
  .route('/:id')
  .get(getJob)
  .delete(testUser, deleteJob)
  .patch(testUser, updateJob);

module.exports = router;
```

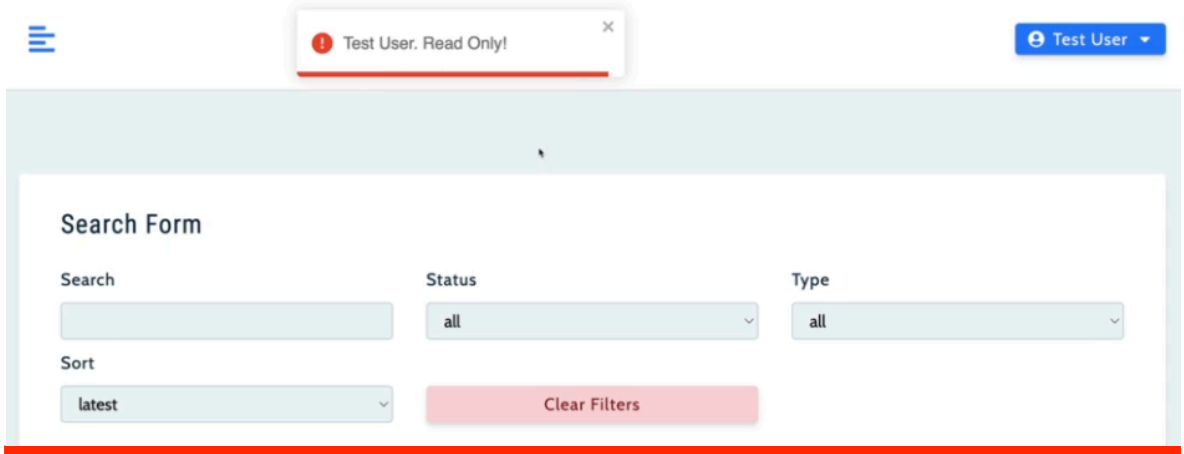




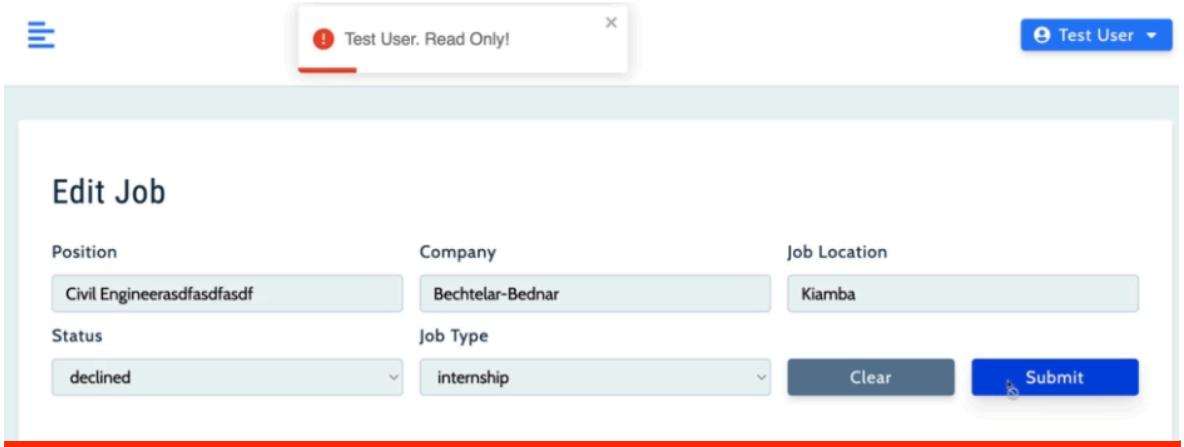
it shows only 75 jobs



if we try to delete



same goes to editing job



22 - API Limiter

==========

to avoid spam our application

```
#### API Limiter

routes/auth.js

```js
const express = require('express');
const router = express.Router();
const authenticateUser = require('../middleware/authentication');
const testUser = require('../middleware/testUser');
const { register, login, updateUser } = require('../controllers/auth')
  ;

const rateLimiter = require('express-rate-limit');
const apiLimiter = rateLimiter({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 10,
  message: {
    msg: 'Too many requests from this IP, please try again after 15
      minutes',
  },
});

router.post('/register', apiLimiter, register);
router.post('/login', apiLimiter, login);
router.patch('/updateUser', authenticateUser, testUser, updateUser);

module.exports = router;
```
```

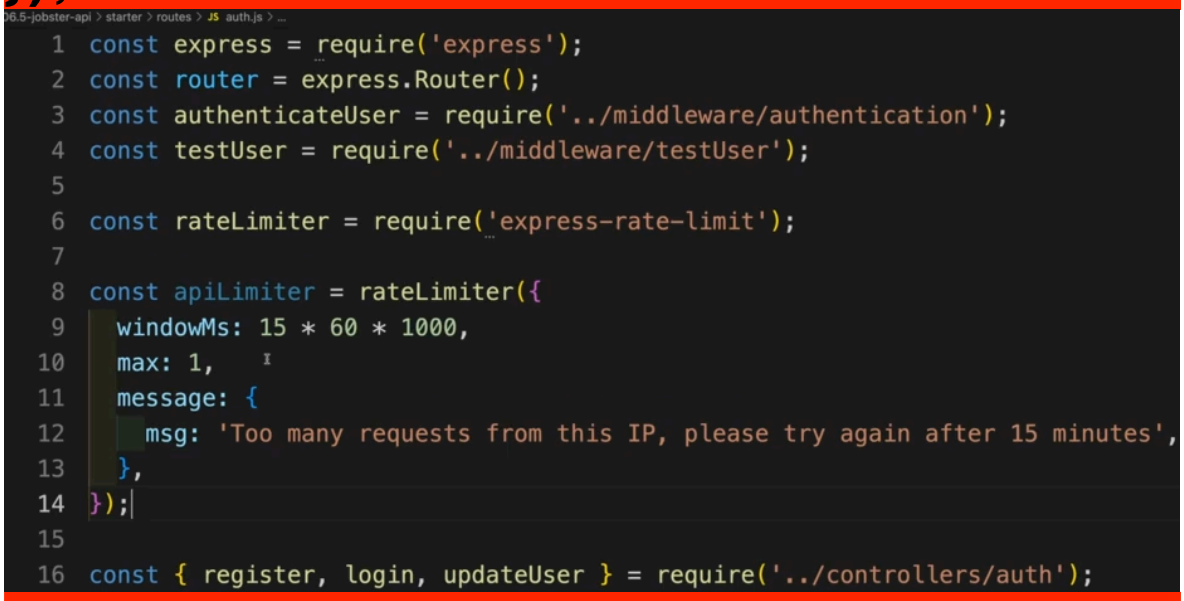
/routes/auth.js

==========

check the max: 1, (we restricted to 1)

```
const rateLimiter = require('express-rate-limit');

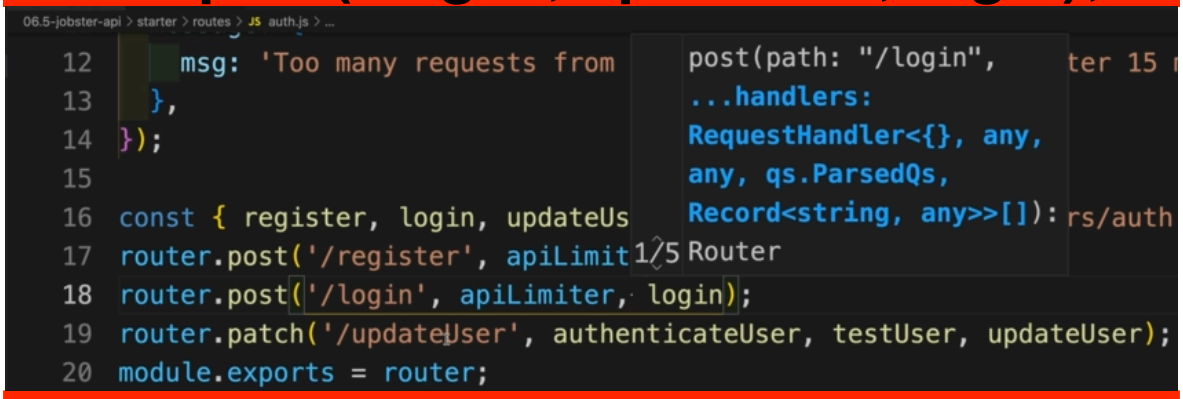
const apiLimiter = rateLimiter({
  windowMs: 15 * 60 * 1000,
  max: 1,
  message: {
    msg: 'Too many requests from this IP, please try again after 15 minutes',
  },
});
```



```
06.5-jobster-api > starter > routes > JS auth.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const authenticateUser = require('../middleware/authentication');
4  const testUser = require('../middleware/testUser');
5
6  const rateLimiter = require('express-rate-limit');
7
8  const apiLimiter = rateLimiter({
9    windowMs: 15 * 60 * 1000,
10   max: 1,
11   message: {
12     msg: 'Too many requests from this IP, please try again after 15 minutes',
13   },
14 });
15
16 const { register, login, updateUser } = require('../controllers/auth');
```

```
router.post('/register', apiLimiter, register);
```

router.post('/login', apiLimiter, login);



/routes/auth.js

==========

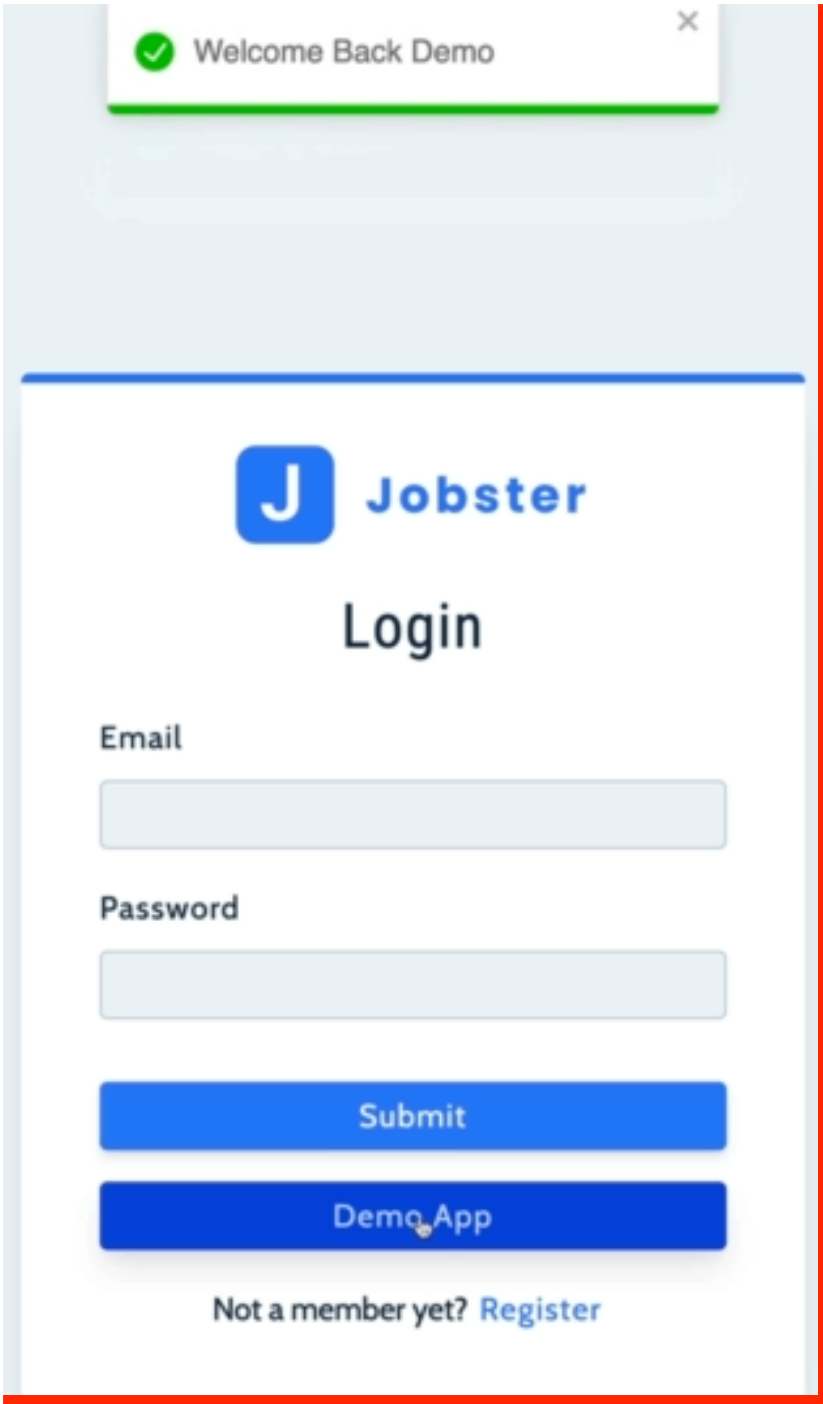
```
const express = require('express');
const router = express.Router();
const authenticateUser = require('../
middleware/authentication');
const testUser = require('../middleware/
testUser');

const rateLimiter = require('express-
rate-limit');

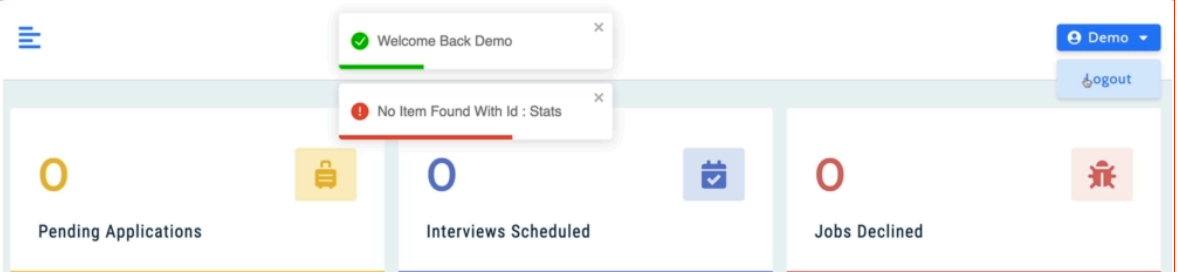
const apiLimiter = rateLimiter({
  windowMs: 15 * 60 * 1000,
  max: 10,
  message: {
    msg: 'Too many requests from this
```

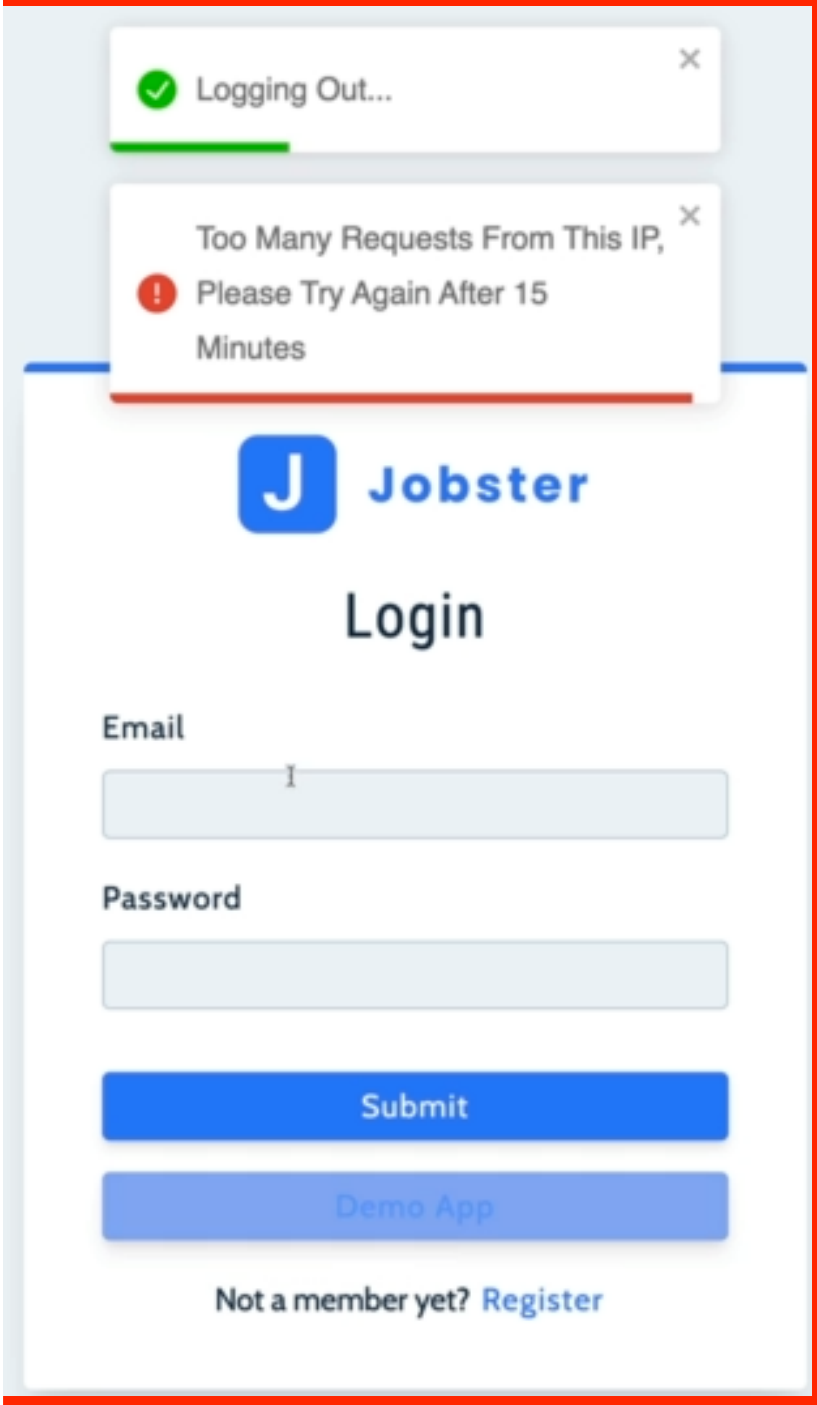
```
IP, please try again after 15 minutes',
  },
});

const { register, login, updateUser } =
require('../controllers/auth');
router.post('/register', apiLimiter,
register);
router.post('/login', apiLimiter, login);
router.patch('/updateUser',
authenticateUser, testUser,
updateUser);
module.exports = router;
```


Welcome Back Demo
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register

logout


Welcome Back Demo
No Item Found With Id : Stats
Demo
Logout
0
Pending Applications
0
Interviews Scheduled
0
Jobs Declined

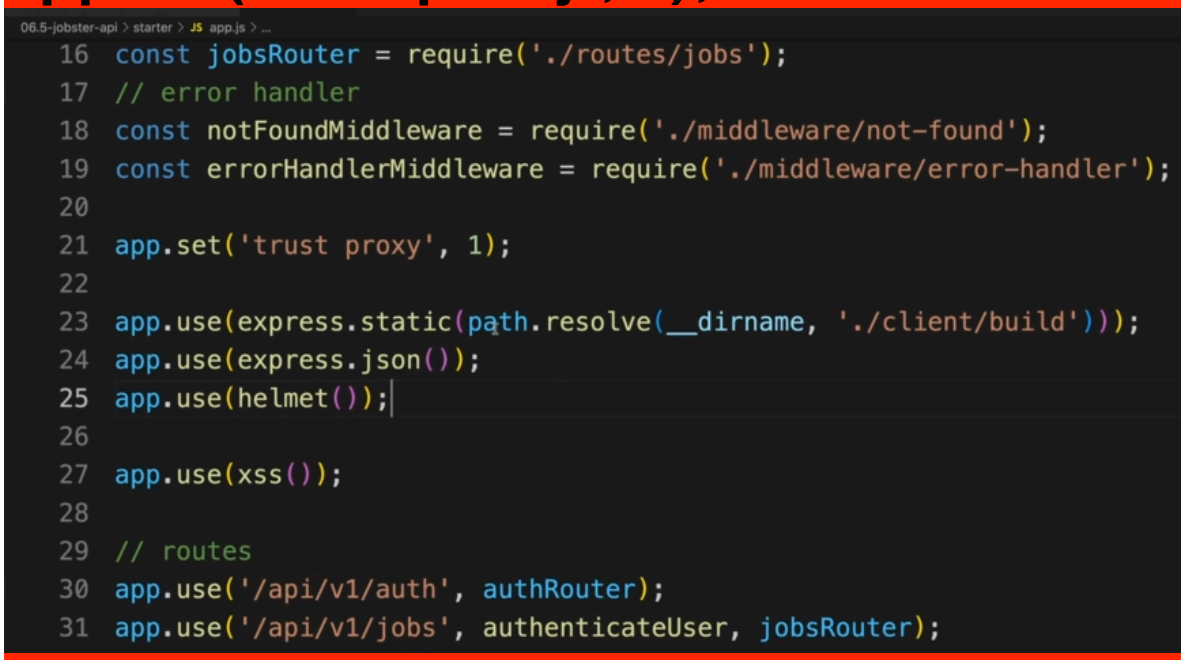



````
app.js

```js
app.set('trust proxy', 1);

app.use(express.static(path.resolve(__dirname, './client/build')));
```
````

app.js
======
app.set('trust proxy', 1);



app.js
======
require('dotenv').config();
require('express-async-errors');

const path = require('path');
// extra security packages
const helmet = require('helmet');
const xss = require('xss-clean');

const express = require('express');
const app = express();

```
const connectDB = require('./db/
connect');
const authenticateUser = require('./
middleware/authentication');
// routers
const authRouter = require('./routes/
auth');
const jobsRouter = require('./routes/
jobs');
// error handler
const notFoundMiddleware = require('./
middleware/not-found');
const errorHandlerMiddleware =
require('./middleware/error-handler');

app.set('trust proxy', 1);

app.use(express.static(path.resolve(__
dirname, './client/build')));
app.use(express.json());
app.use(helmet());

app.use(xss());
```

```
// routes
app.use('/api/v1/auth', authRouter);
app.use('/api/v1/jobs',
authenticateUser, jobsRouter);

app.get('*', (req, res) => {
  res.sendFile(path.resolve(__dirname,
'./client/build', 'index.html'));
});

app.use(notFoundMiddleware);
app.use(errorHandlerMiddleware);

const port = process.env.PORT || 5000;

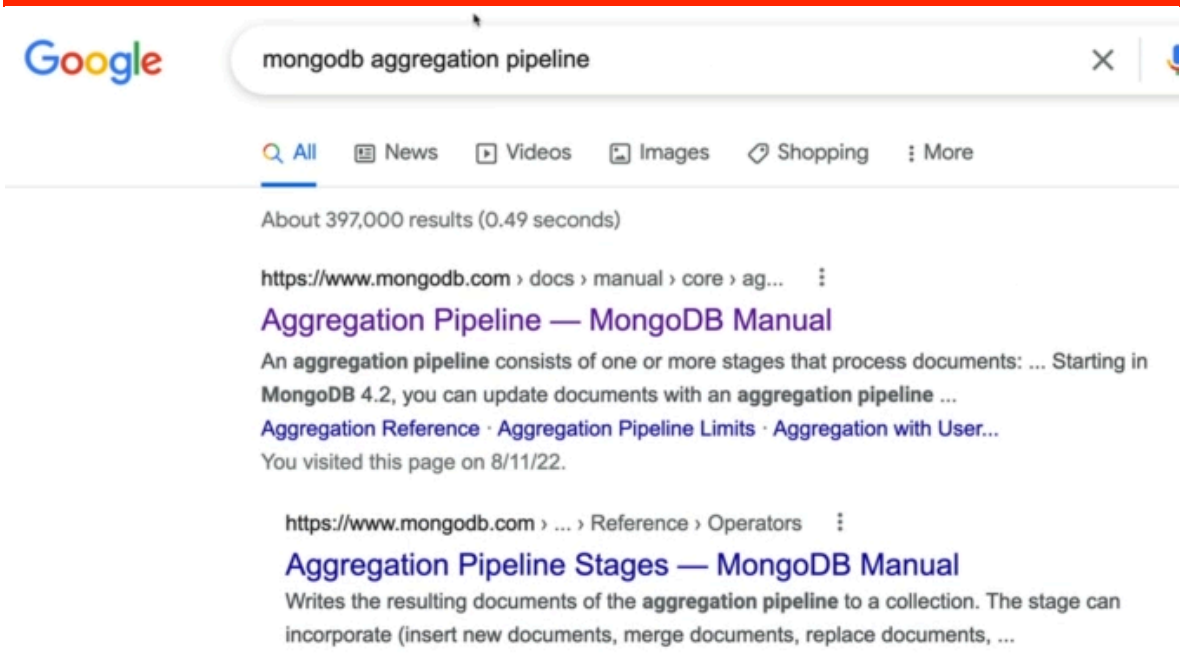
const start = async () => {
  try {
    await
connectDB(process.env.MONGO_URI);
    app.listen(port, () =>
      console.log(`Server is listening on
port ${port}...`)
    );
```

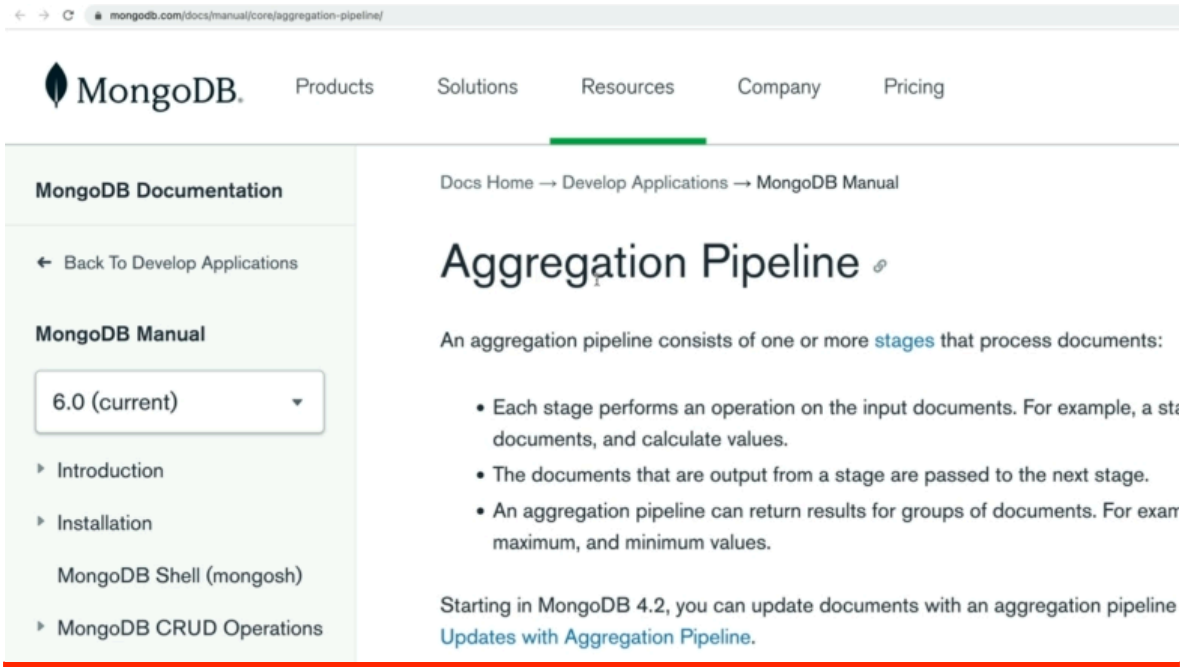
```
  } catch (error) {
    console.log(error);
  }
};

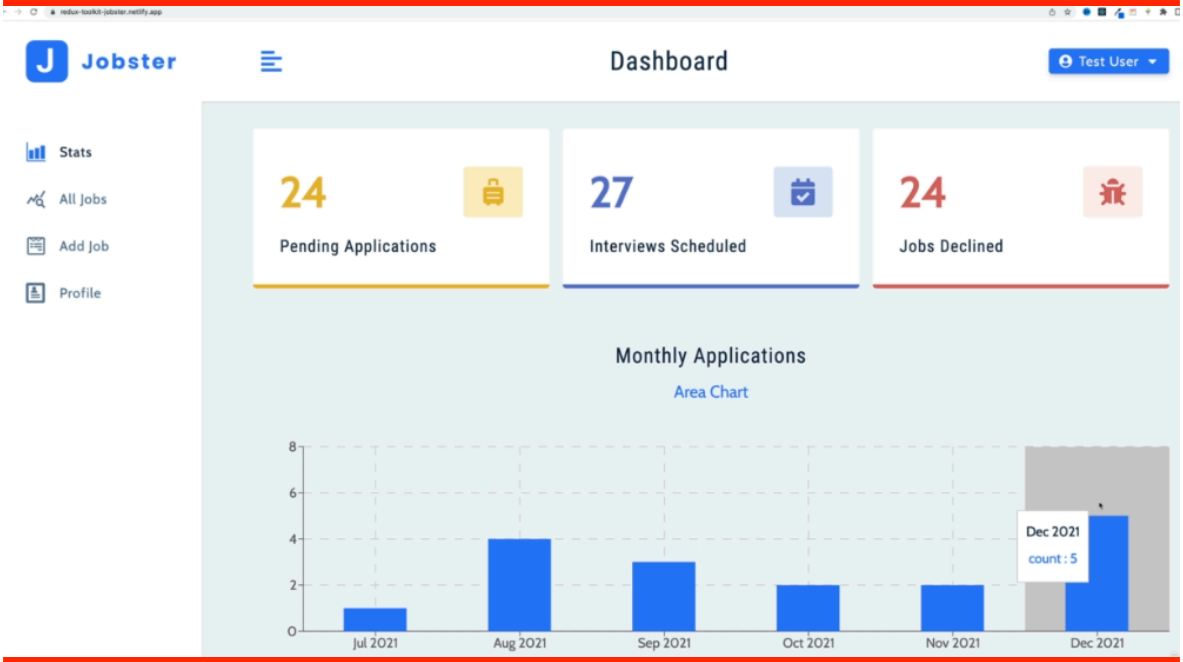
start();
```

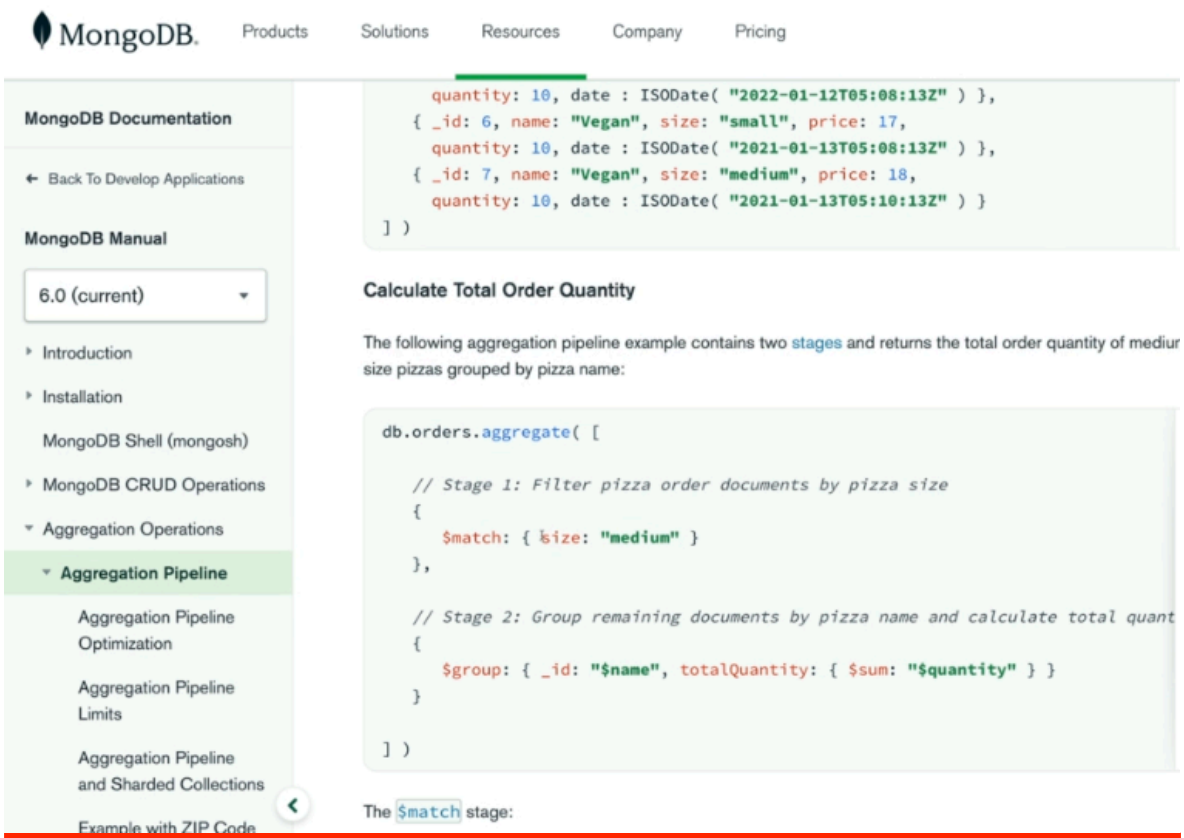
23 - Stats

==========



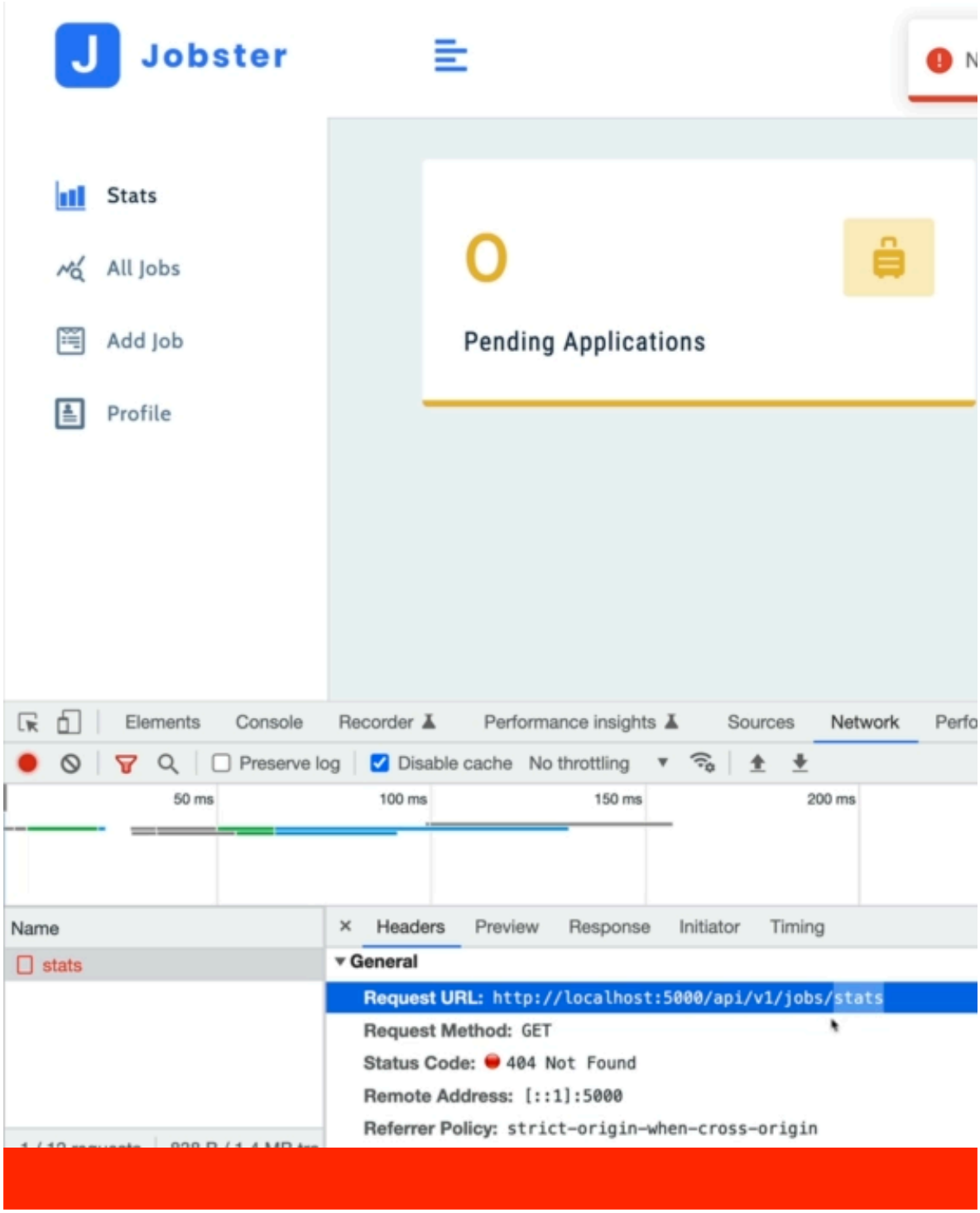

mongodb.com/docs/manual/core/aggregation-pipeline/
MongoDB
Products
Solutions
Resources
Company
Pricing
MongoDB Documentation
Back To Develop Applications
MongoDB Manual
6.0 (current)
Introduction
Installation
MongoDB Shell (mongosh)
MongoDB CRUD Operations
Docs Home → Develop Applications → MongoDB Manual
Aggregation Pipeline
An aggregation pipeline consists of one or more stages that process documents:
Each stage performs an operation on the input documents. For example, a sta
documents, and calculate values.
The documents that are output from a stage are passed to the next stage.
An aggregation pipeline can return results for groups of documents. For exam
maximum, and minimum values.
Starting in MongoDB 4.2, you can update documents with an aggregation pipeline
Updates with Aggregation Pipeline.


Jobster
Dashboard
Test User
Stats
All Jobs
Add Job
Profile
24
Pending Applications
27
Interviews Scheduled
24
Jobs Declined
Monthly Applications
Area Chart
8
6
4
2
0
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021
Dec 2021
count : 5



24 - Show Stats Controller

==========


Jobster
Stats
All Jobs
Add Job
Profile
0
Pending Applications
Elements
Console
Recorder
Performance insights
Sources
Network
Preserve log
Disable cache
No throttling
50 ms
100 ms
150 ms
200 ms
Name
stats
Headers
Preview
Response
Initiator
Timing
General
Request URL: http://localhost:5000/api/v1/jobs/stats
Request Method: GET
Status Code: 404 Not Found
Remote Address: [::1]:5000
Referrer Policy: strict-origin-when-cross-origin

````
#### Setup Stats Route

controllers/jobs

```js
const showStats = (req, res) => {
  res
    .status(StatusCodes.OK)
    .json({ defaultStats: {}, monthlyApplications: [] });
};

module.exports = {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
};
```

routes/jobs.js

```js
const express = require('express');

const router = express.Router();
const {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
} = require('../controllers/jobs');

router.route('/').post(createJob).get(getAllJobs);
router.route('/stats').get(showStats);
router.route('/:id').get(getJob).delete(deleteJob).patch(updateJob);

module.exports = router;
```
````

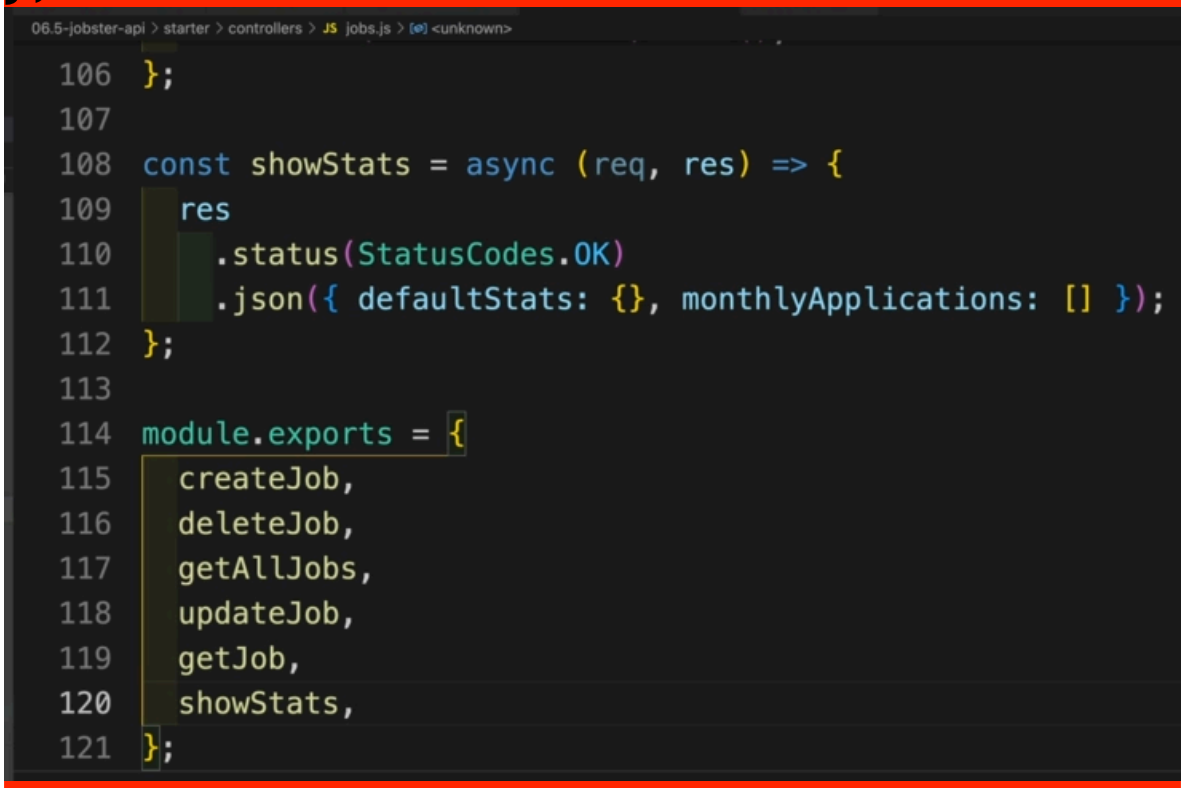
controllers/jobs.js

==========

```
const showStats = async (req, res) => {
    res.status(StatusCodes.OK)
    .json({ defaultStats: { },
monthlyApplications: [ ] });
```

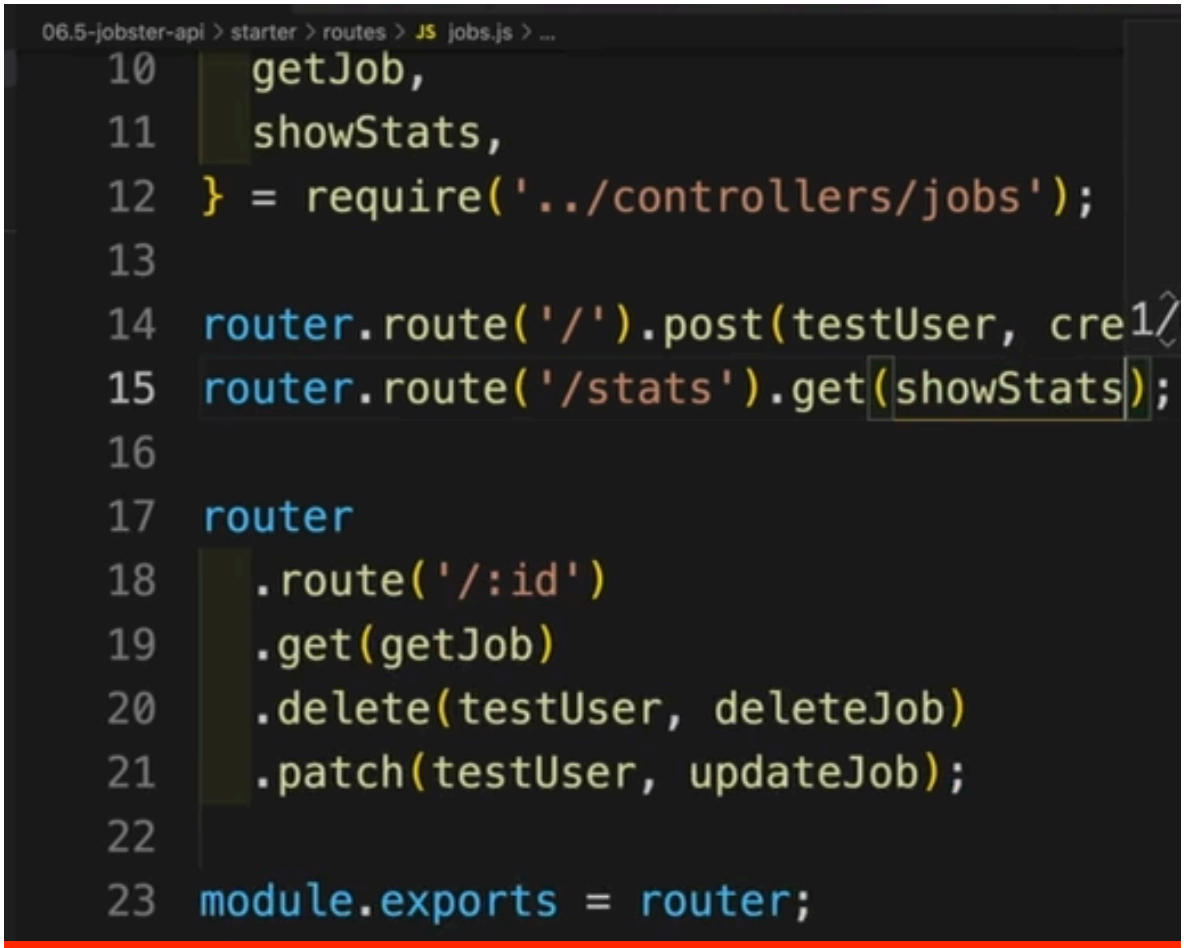
```
};

module.exports = {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
};
```



routes/jobs.js

==========




```
const express = require('express');
const testUser = require('../middleware/testUser');

const router = express.Router();
const {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
} = require('../controllers/jobs');

router.route('/').post(testUser, createJob).get(getAllJobs);
router.route('/stats').get(showStats);

router
  .route('/:id')
  .get(getJob)
  .delete(testUser, deleteJob)
  .patch(testUser, updateJob);

module.exports = router;
```

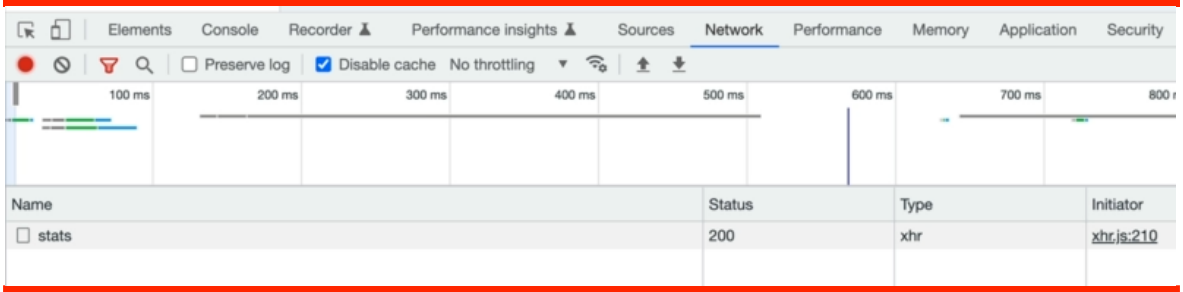
```
const express = require('express');
const testUser = require('../middleware/
testUser');

const router = express.Router();
const {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
} = require('../controllers/jobs');

router.route('/').post(testUser,
createJob).get(getAllJobs);
router.route('/stats').get(showStats);

router
  .route('/:id')
  .get(getJob)
  .delete(testUser, deleteJob)
  .patch(testUser, updateJob);
```

module.exports = router;



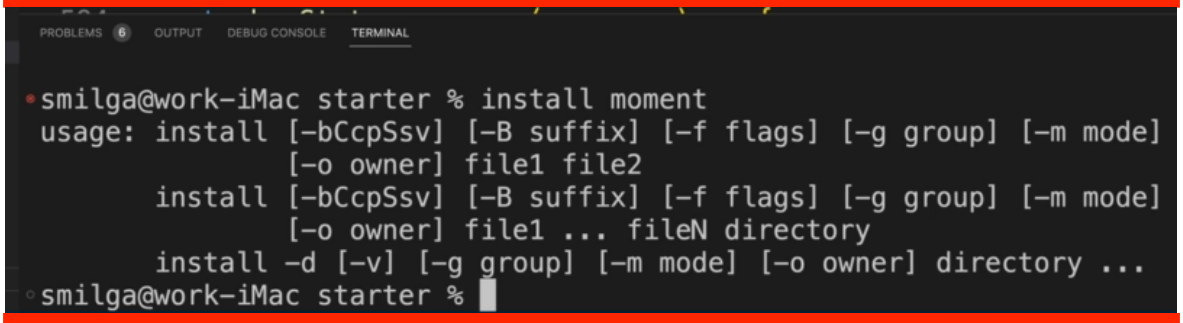
25 - Setup Status Aggregation Pipeline

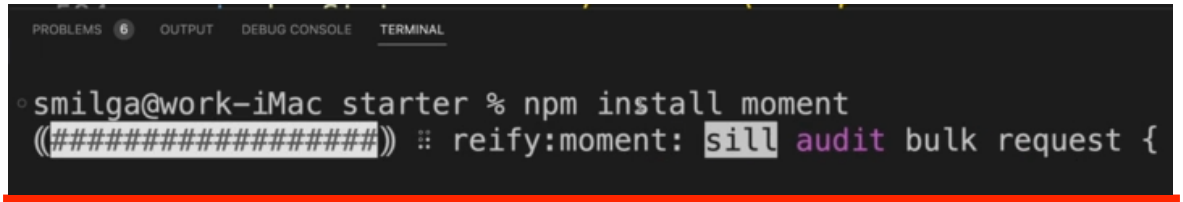
==========

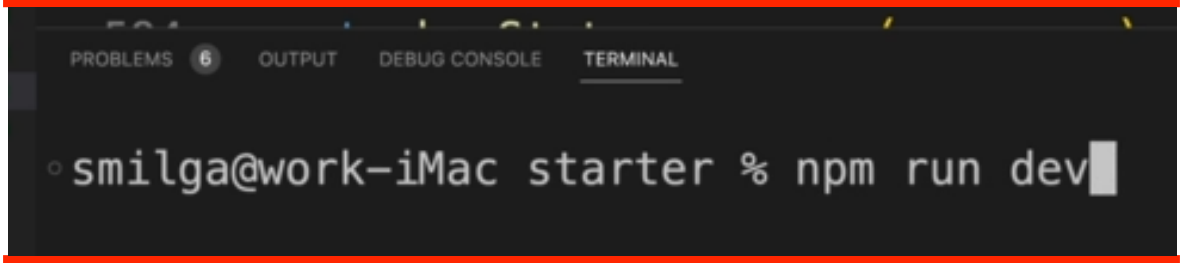
- npm install moment
- import mongoose and moment

controllers/jobs

```js
const mongoose = require('mongoose');
const moment = require('moment');
```





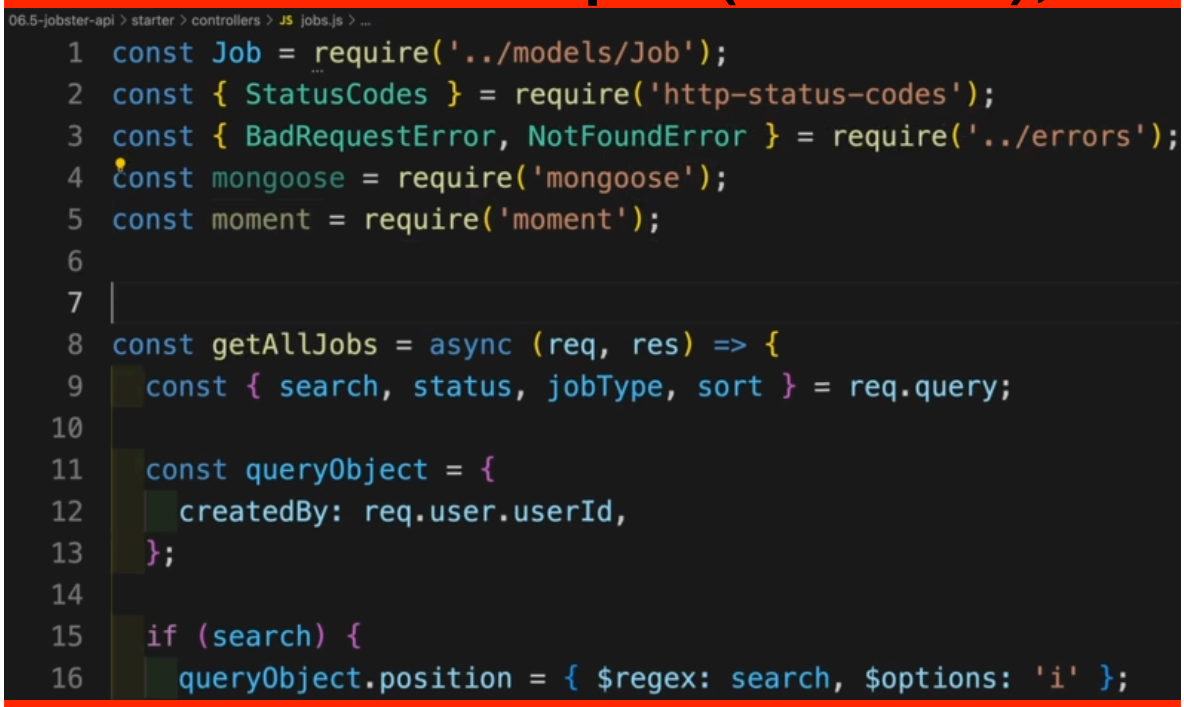


controllers/jobs.js

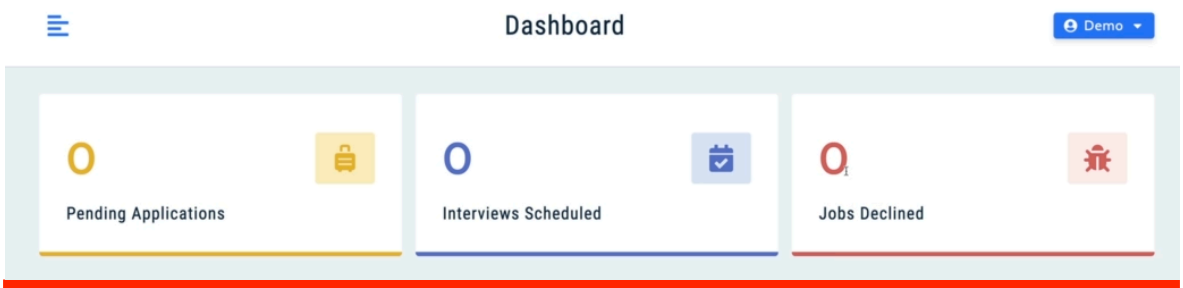
==========

const mongoose = require('mongoose');

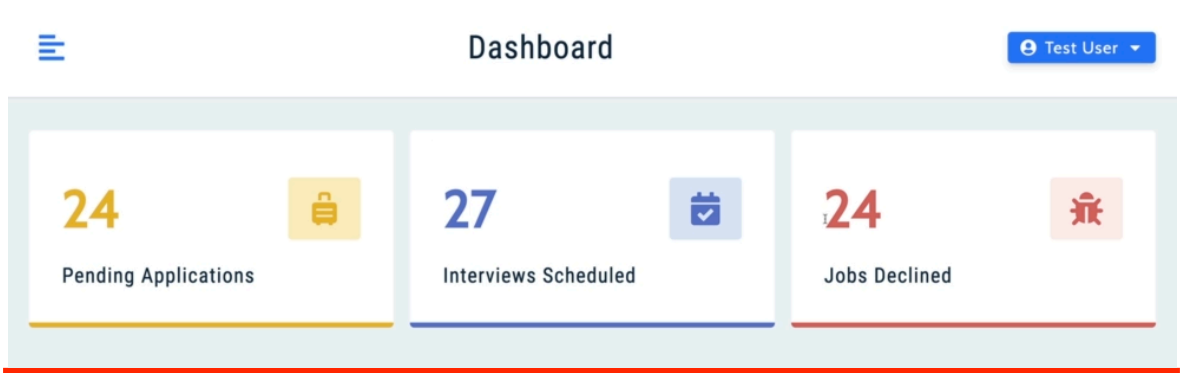
const moment = require('moment');



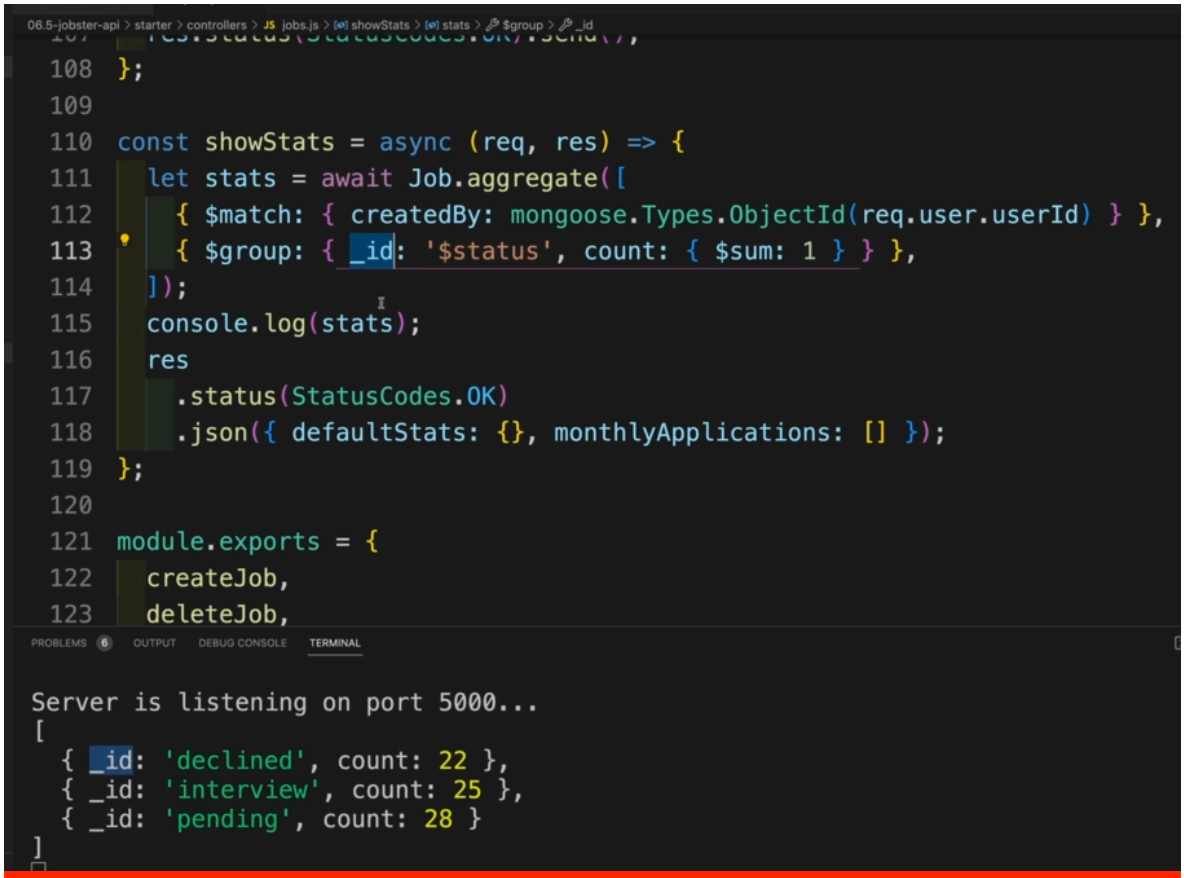
if user id is Demo



if user id is Test User, based on the user status, we have to count using { $group: { _id: '$status', count: { $sum: 1 } } }




```
const showStats = async (req, res) => {
  let stats = await Job.aggregate([
    { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
    { $group: { _id: '$status', count: { $sum: 1 } } },
  ]);
console.log(stats);
```

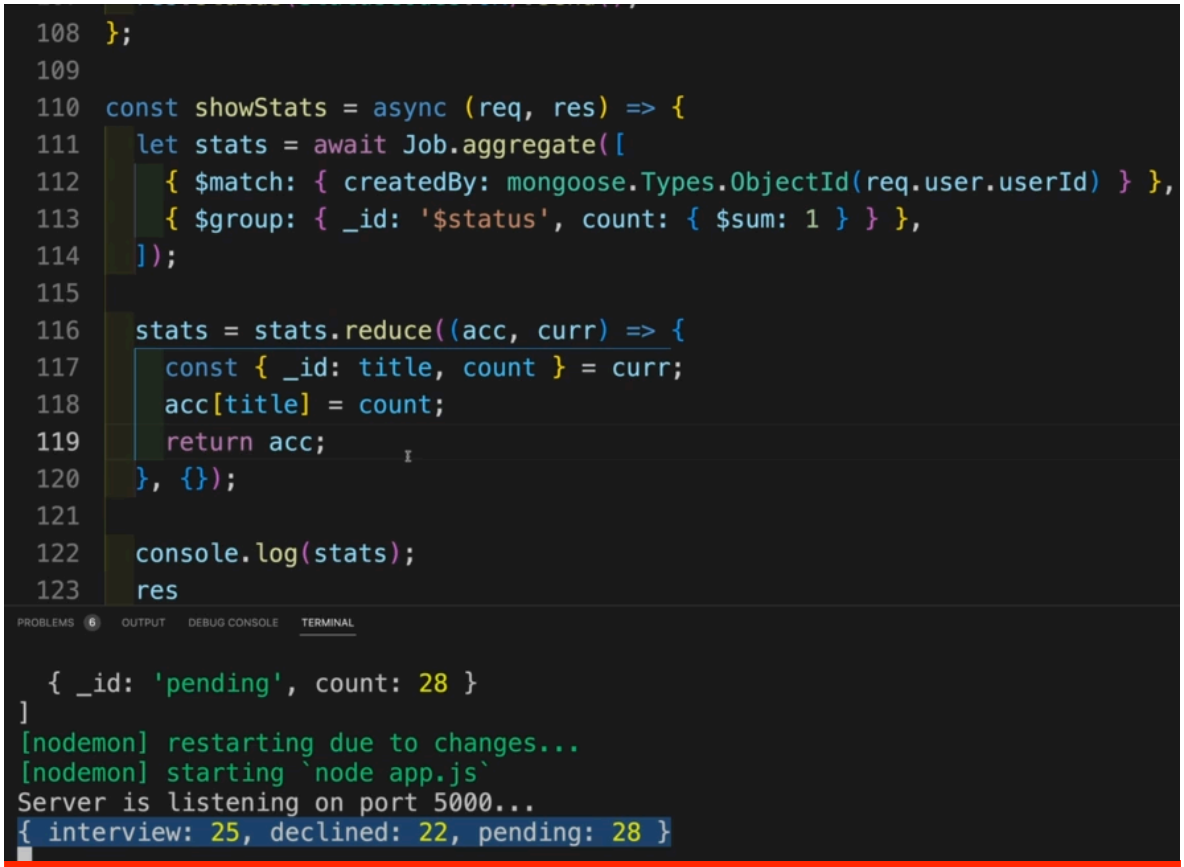


26 - Refactor Status Data

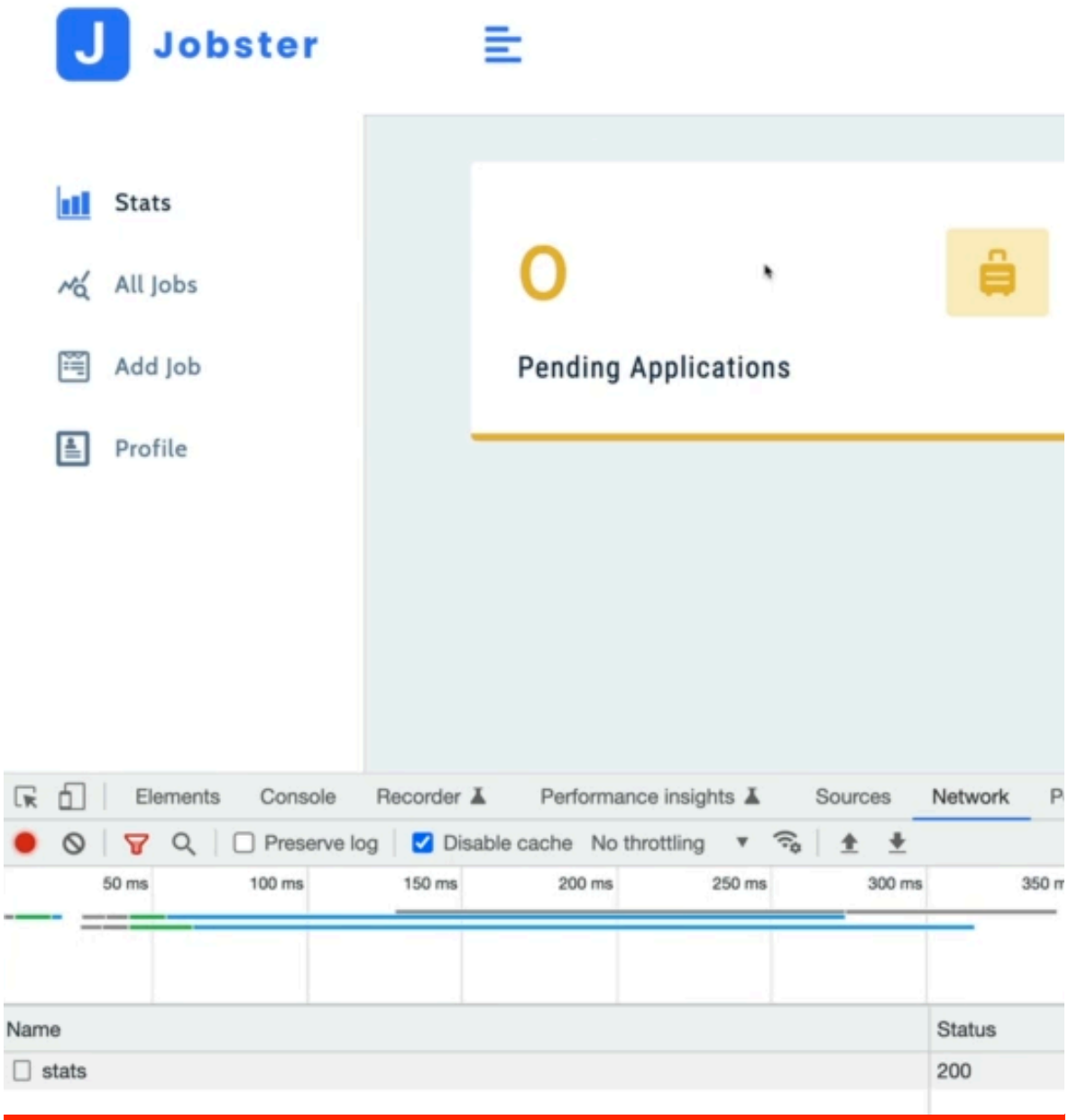
==========

```
110  const showStats = async (req, res) => {
111    let stats = await Job.aggregate([
112      { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
113      { $group: { _id: '$status', count: { $sum: 1 } } },
114    ]);
115
116    stats = stats.reduce((acc, curr) => {
117      const { _id: title, count } = curr;
118      acc[title] = count;
119      return acc;
120    }, {});
121
122    const defaultStats = {
123      pending: stats.pending || 0,
124      interview: stats.interview || 0,
125      declined: stats.declined || 0,
126    };
127
128    let monthlyApplications = await Job.aggregate([
129      { $match: { createdBy: mongoose.Types.ObjectId(req.user.userId) } },
130      {
131        $group: {
132          _id: { year: { $year: '$createdAt' }, month: { $month: '$createdAt'
133          count: { $sum: 1 },
134        },
135      },
136      { $sort: { '_id.year': -1, '_id.month': -1 } },
137      { $limit: 6 },
138    ]);
139
140    monthlyApplications = monthlyApplications
141      .map((item) => {
142        const {
143          _id: { year, month },
144          count,
145        } = item;
146        const date = moment()
147          .month(month - 1)
148          .year(year)
149          .format('MMM Y');
150        return { date, count };
151      })
152      .reverse();
153
154    res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications });
155  };
```

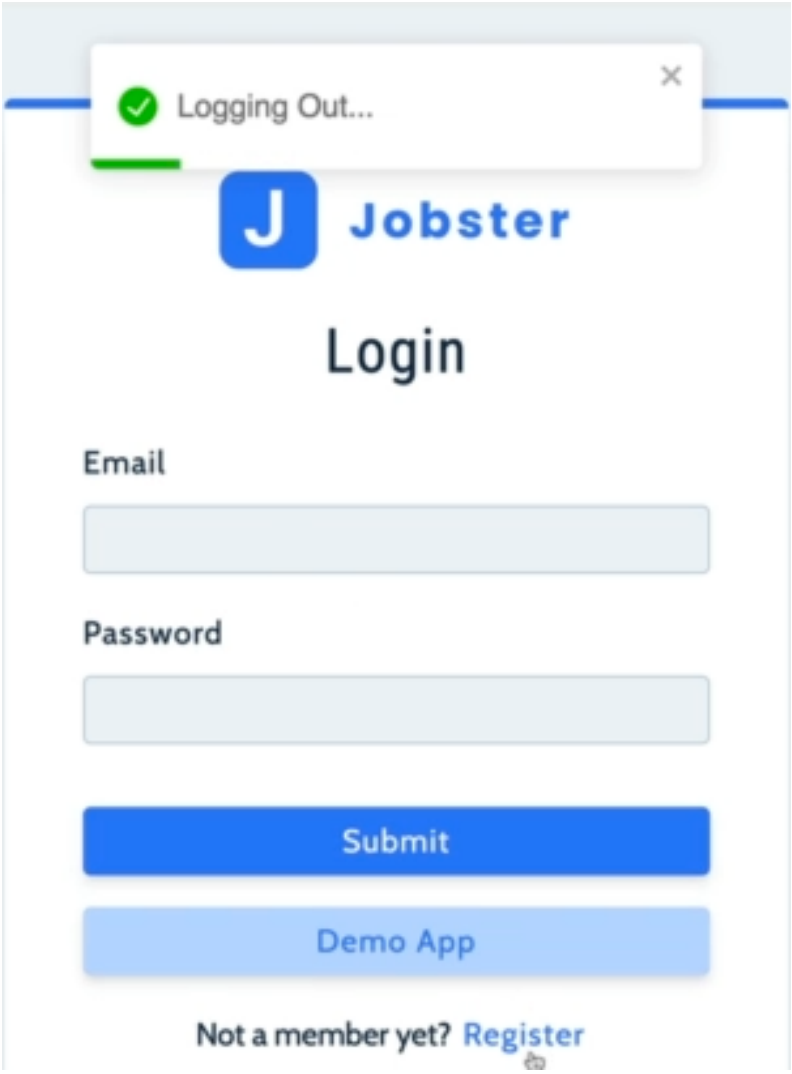
```
stats = stats.reduce((acc, curr) => {
  const { _id: title, count } = curr;
  acc[title] = count;
  return acc;
}, {});
```

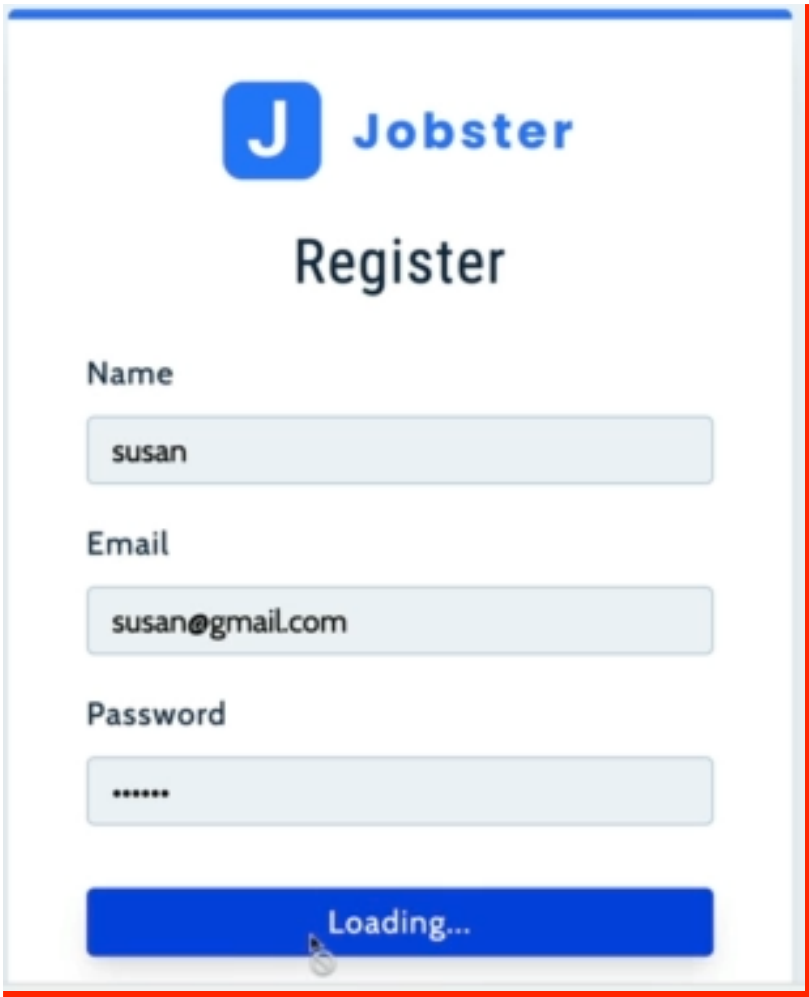


send request in the browser to see the above result

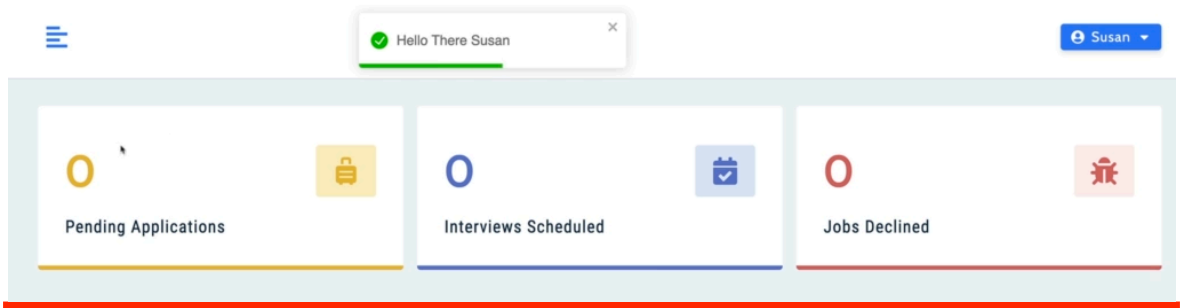


logout and register new user

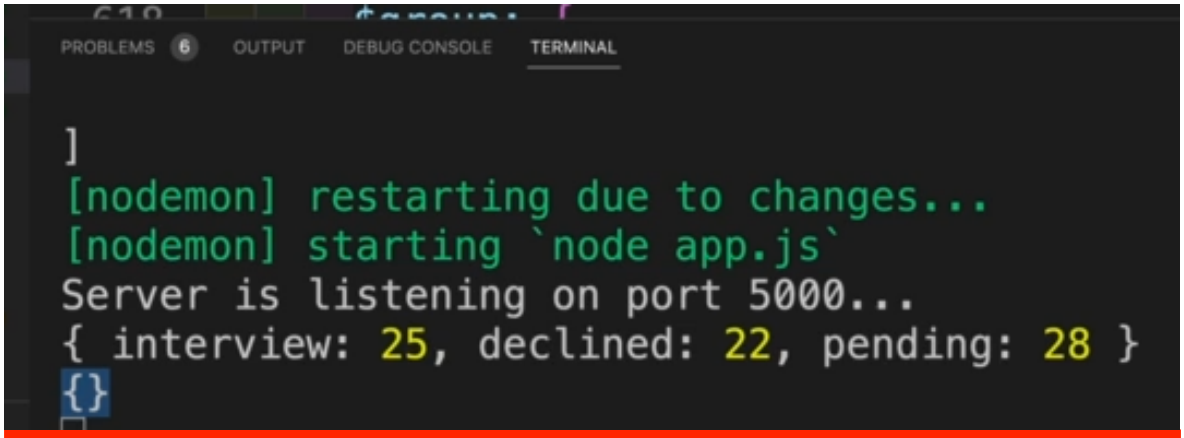

Logging Out...
Jobster
Login
Email
Password
Submit
Demo App
Not a member yet? Register



we can see all are 0.



if we see the output in terminal, it is { } empty object

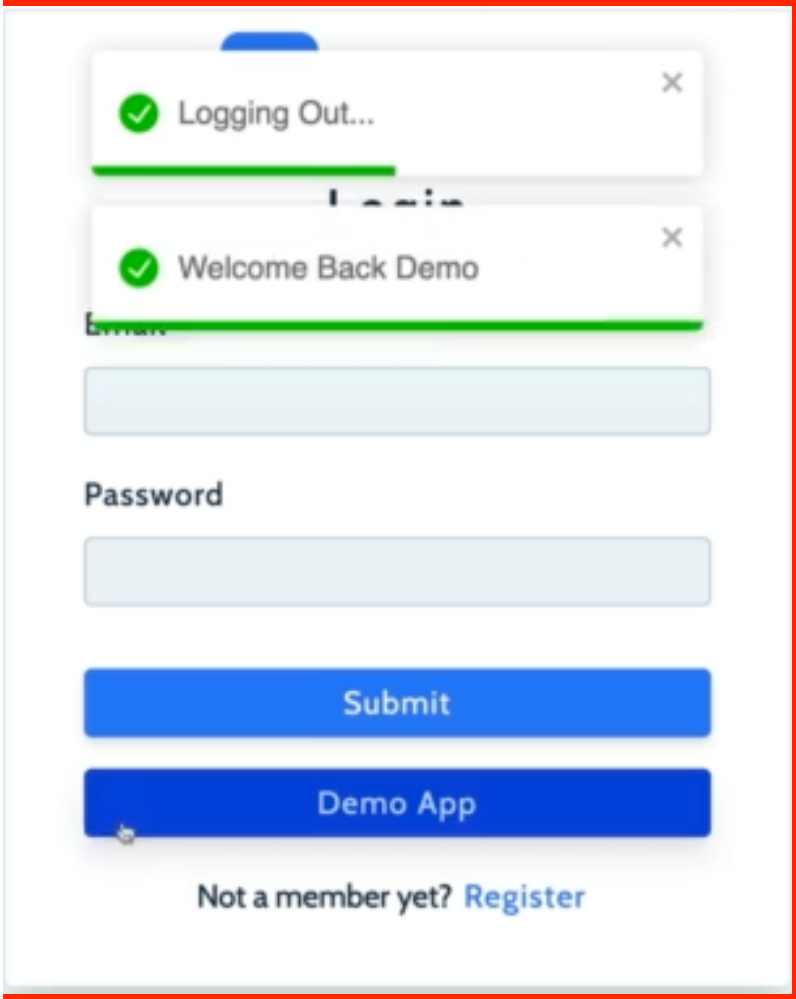


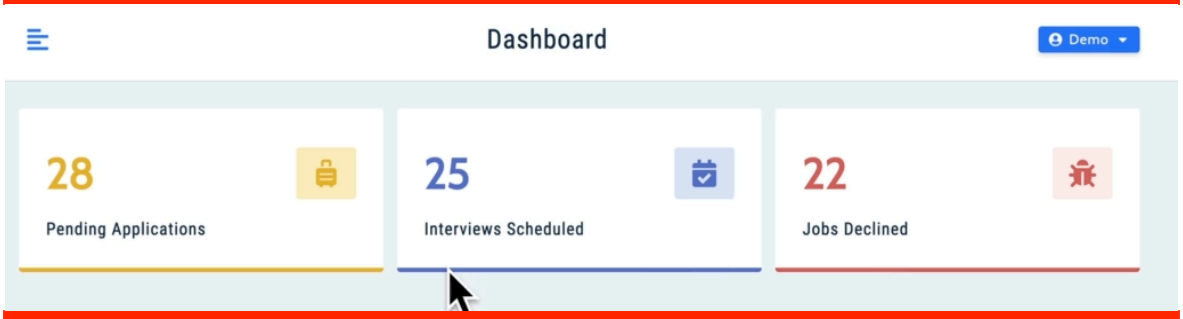

```
const defaultStats = {
    pending: stats.pending ll 0,
    interview: stats.interview ll 0,
    declined: stats.declined ll 0,
  };

res.status(StatusCodes.OK).json({ defa
ultStats, monthlyApplications: [] });
```

log out from susan user and login as

demo



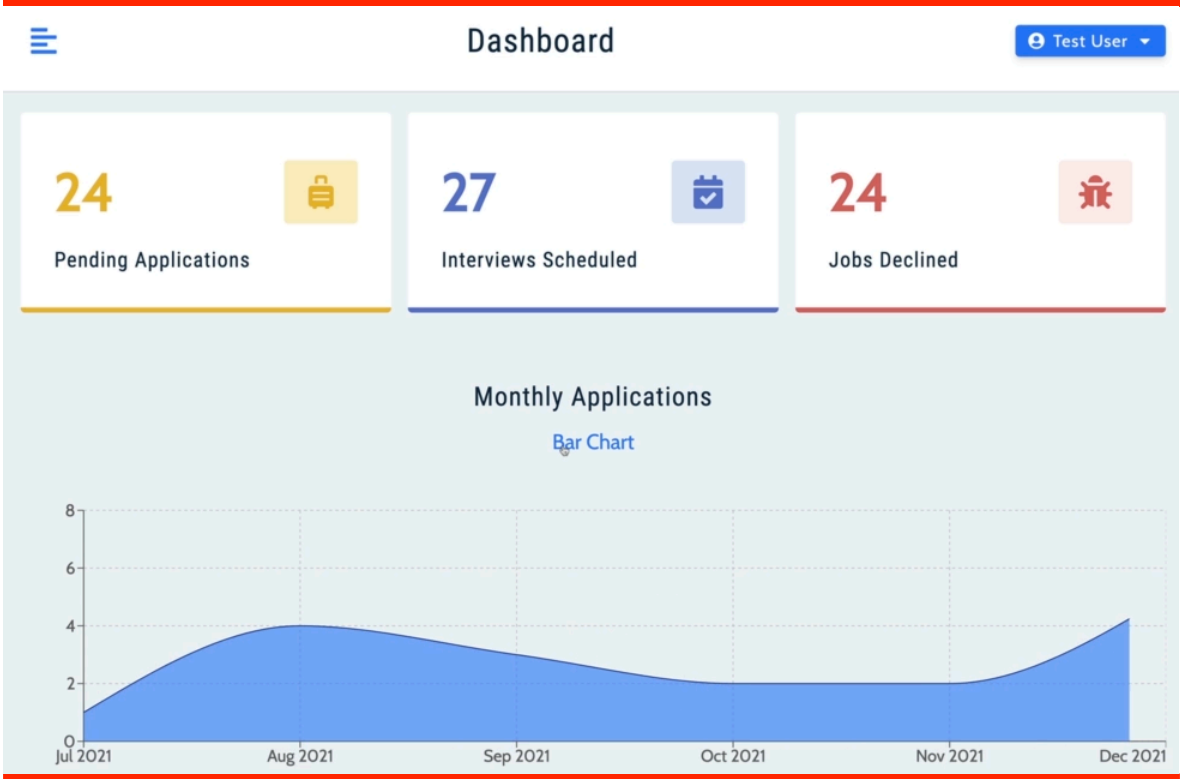


27 - Setup Monthly Applications Aggregation Pipeline

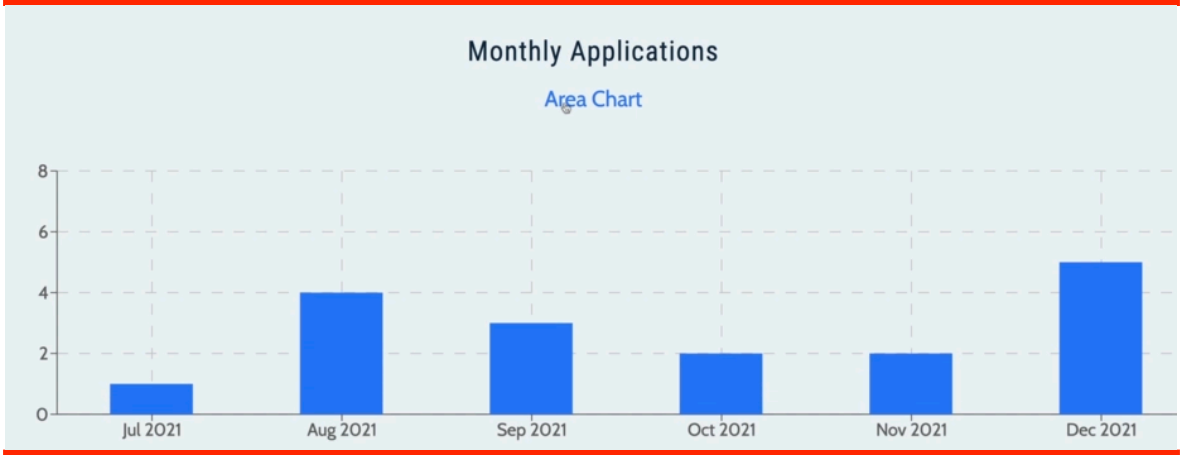
==========

==========

Bar chart

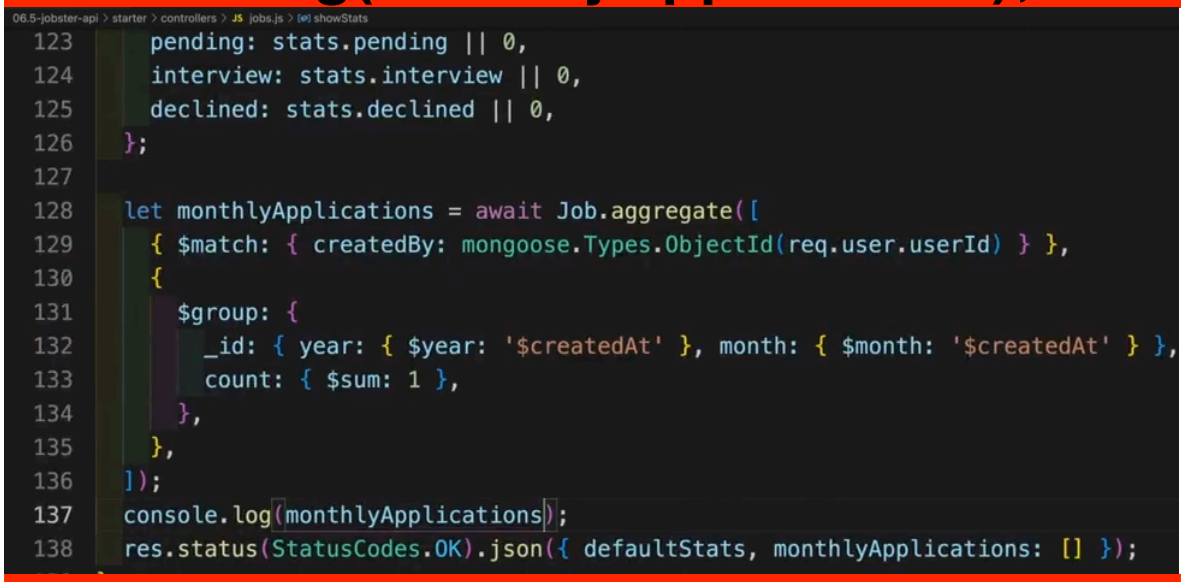


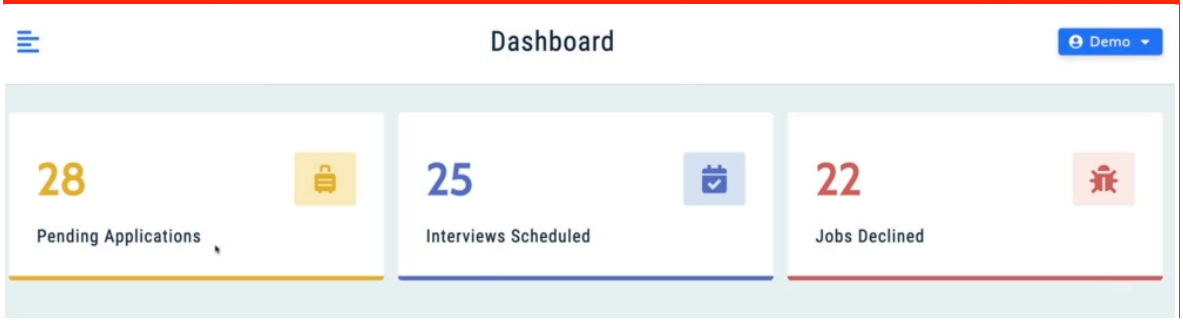
Area Chart

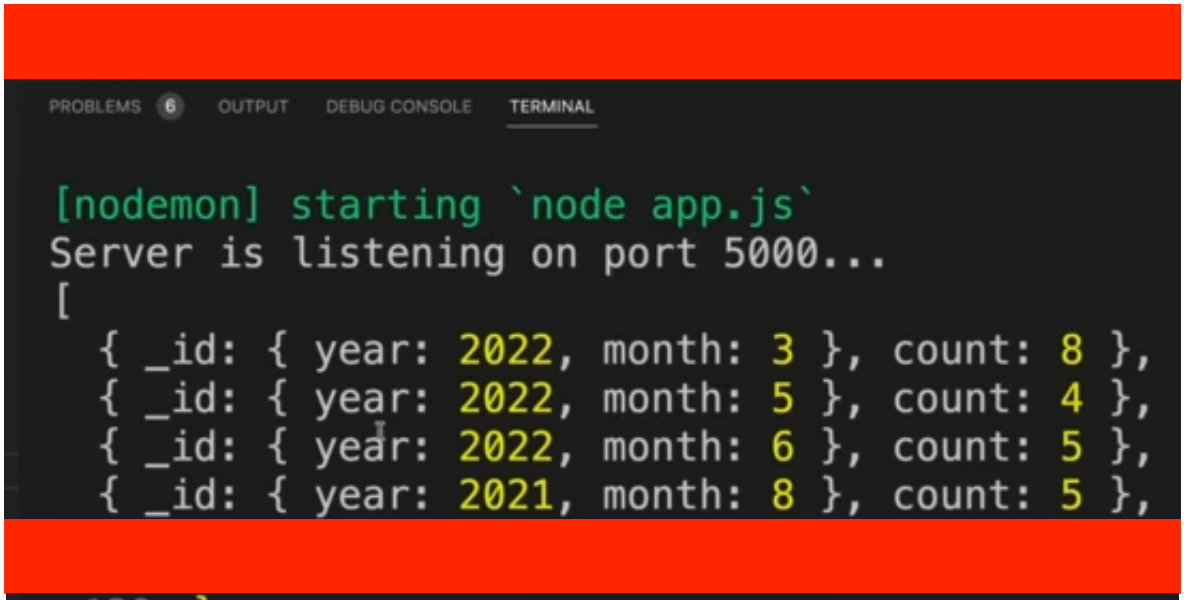



```
let monthlyApplications = await
Job.aggregate([
   { $match: { createdBy:
```

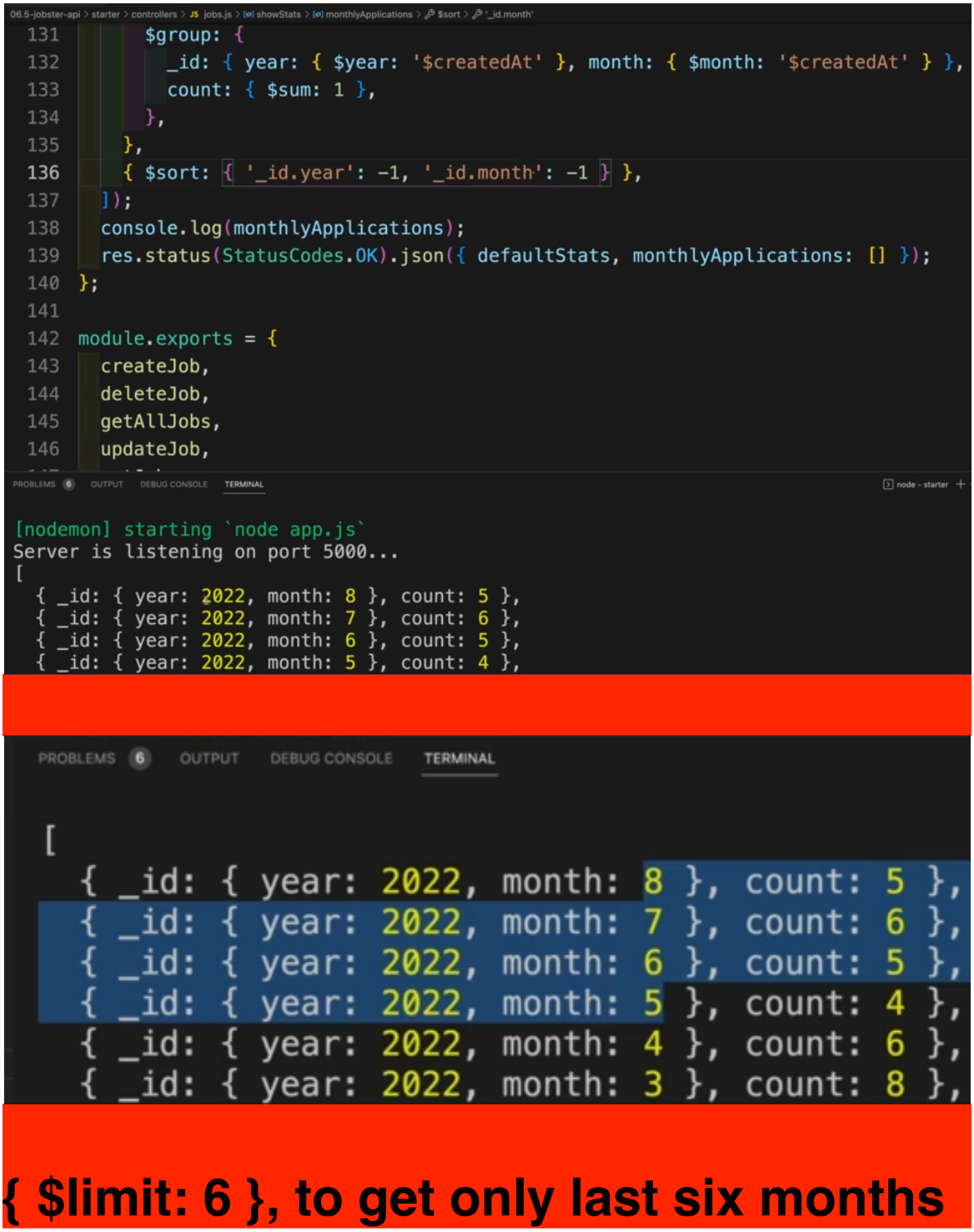
```
mongoose.Types.ObjectId(req.user.userId) } },
    {
      $group: {
        _id: { year: { $year: '$createdAt' }, month: { $month: '$createdAt' } },
        count: { $sum: 1 },
      },
    },
  ]);
console.log(monthlyApplications);
```



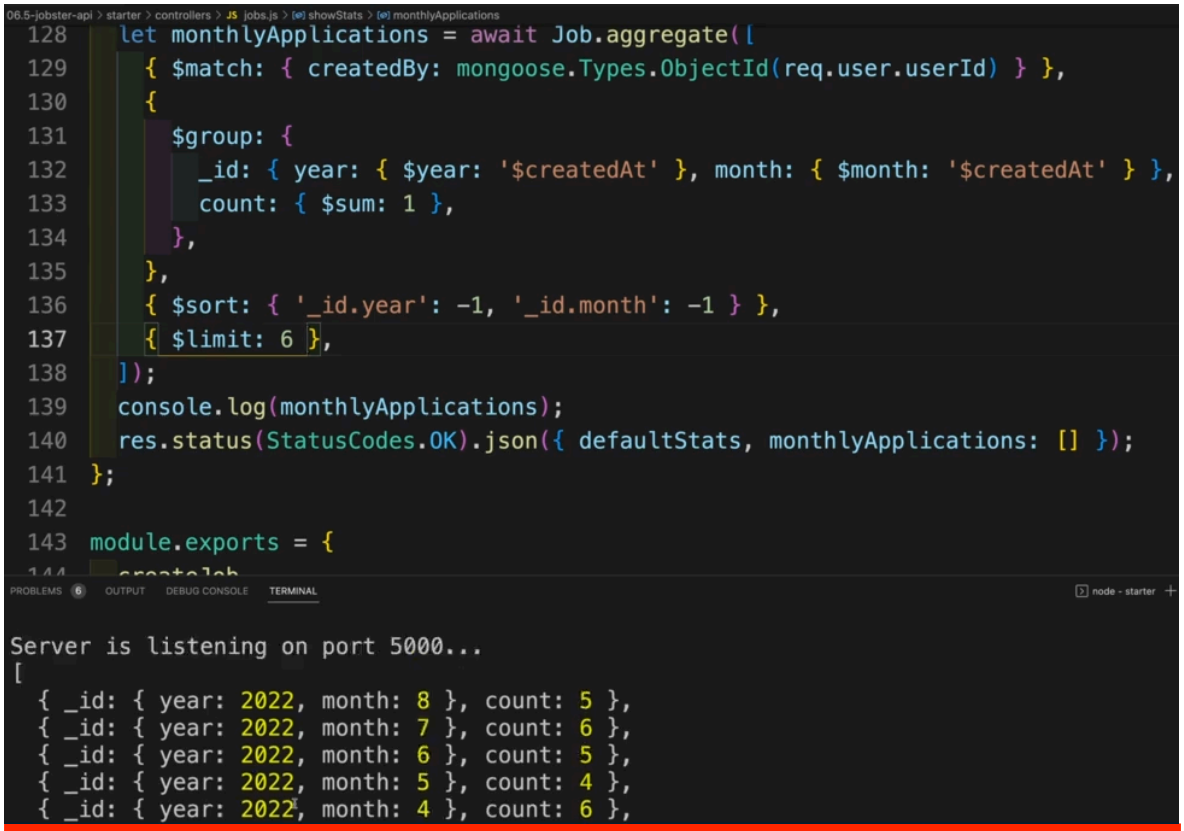




to sort for last six months, from higher to low level
{ $sort: { '_id.year': -1, '_id.month': -1 } },



{ $limit: 6 }, to get only last six months



28 - Refactor Monthly Applications Data

==========

==

we have to show last months as first one

momentjs.com

==========

moment.js

moment.min.js 18.1k gz

moment-with-locales.js

moment-with-locales.min.js 73.6k gz

```
npm install moment --save   # npm
yarn add moment             # Yarn
Install-Package Moment.js   # NuGet
spm install moment --save   # spm
meteor add momentjs:moment  # meteor
bower install moment --save # bower (deprec
```

Format Dates

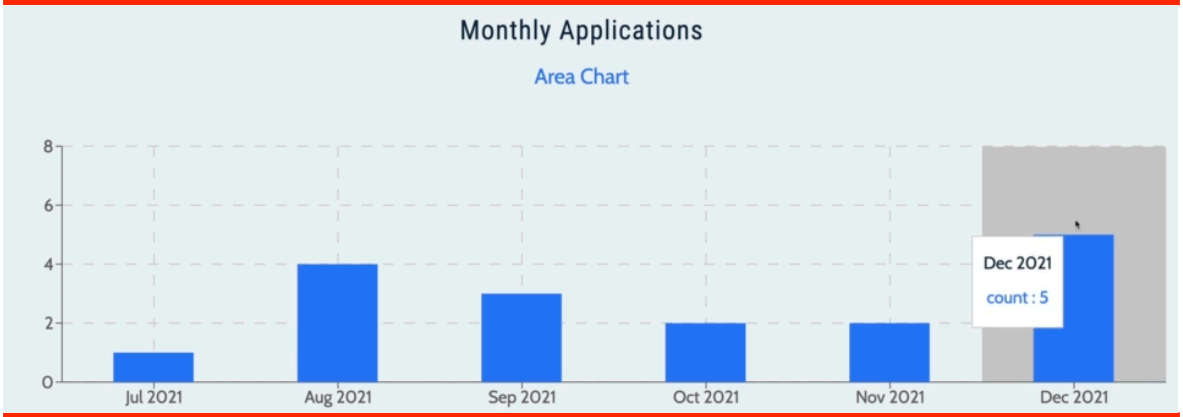
```
moment().format('MMMM Do YYYY, h:mm:ss a'); // August 14th 2022, 11:37:00 am
moment().format('dddd');                    // Sunday
moment().format("MMM Do YY");               // Aug 14th 22
moment().format('YYYY [escaped] YYYY');     // 2022 escaped 2022
moment().format();                          // 2022-08-14T11:37:00-07:00
```

Relative Time

```
moment("20111031", "YYYYMMDD").fromNow(); // 11 years ago
moment("20120620", "YYYYMMDD").fromNow(); // 10 years ago
moment().startOf('day').fromNow();        // 12 hours ago
moment().endOf('day').fromNow();          // in 12 hours
moment().startOf('hour').fromNow();       // 37 minutes ago
```

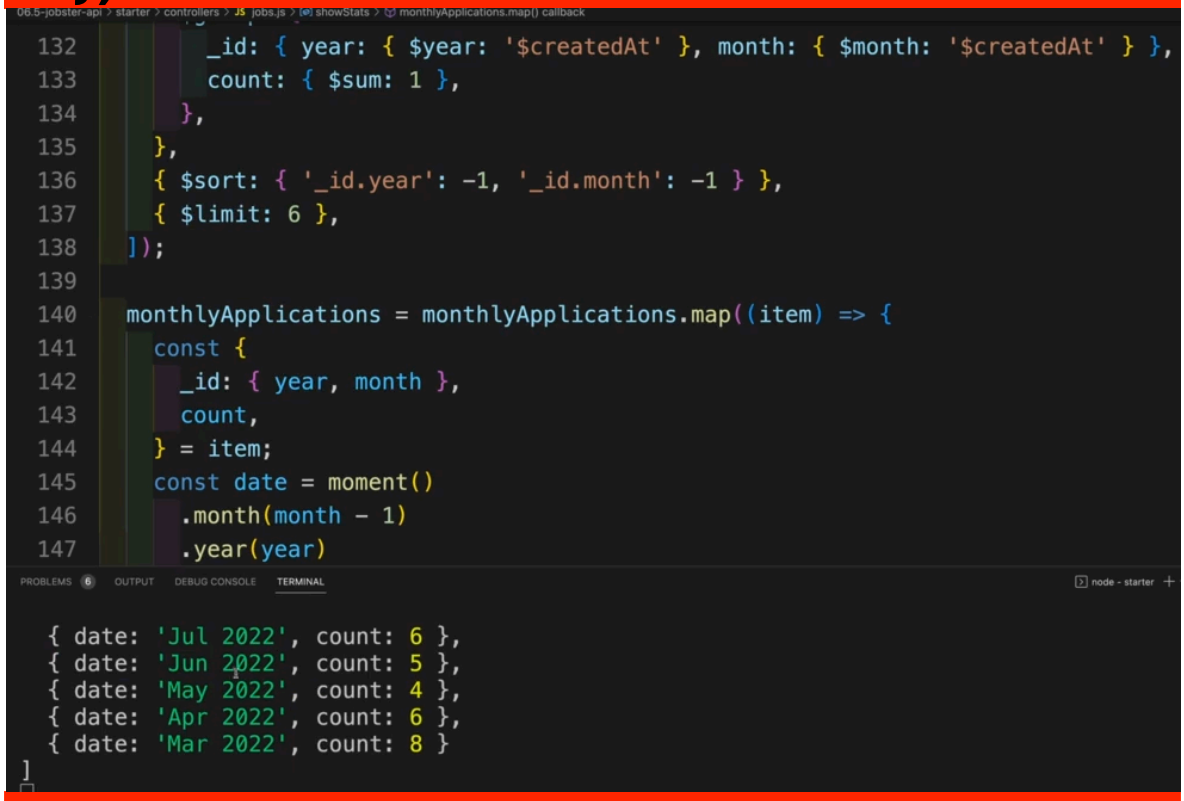
Calendar Time

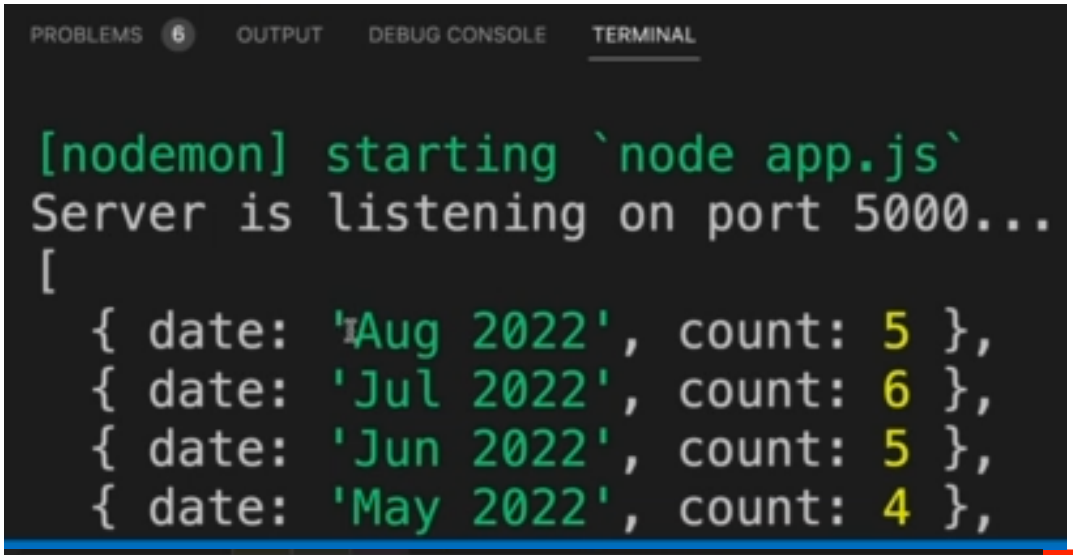
```
moment().subtract(10, 'days').calendar(); // 08/04/2022
moment().subtract(6, 'days').calendar();  // Last Monday at 11:37 AM
moment().subtract(3, 'days').calendar();  // Last Thursday at 11:37 AM
moment().subtract(1, 'days').calendar();  // Yesterday at 11:37 AM
```

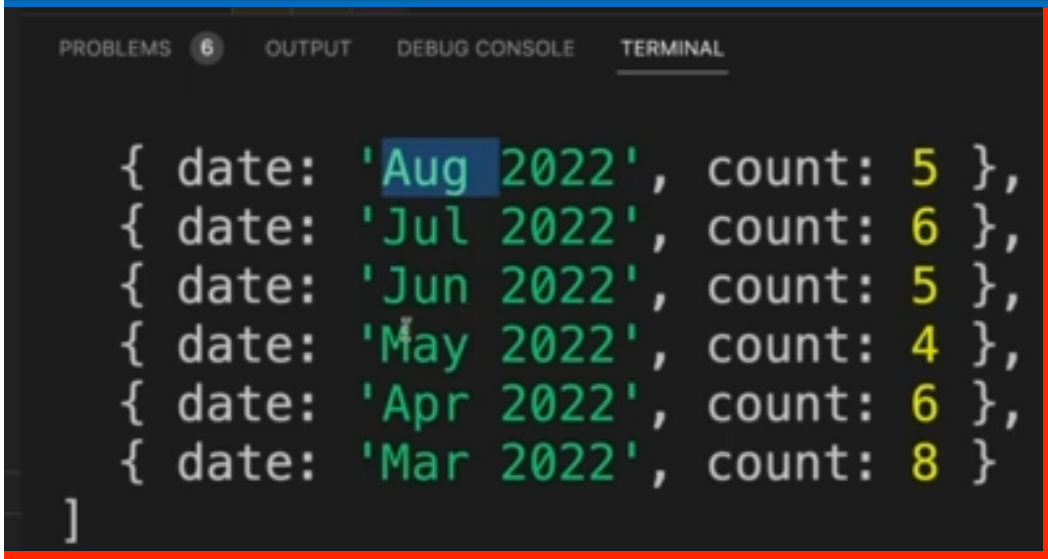


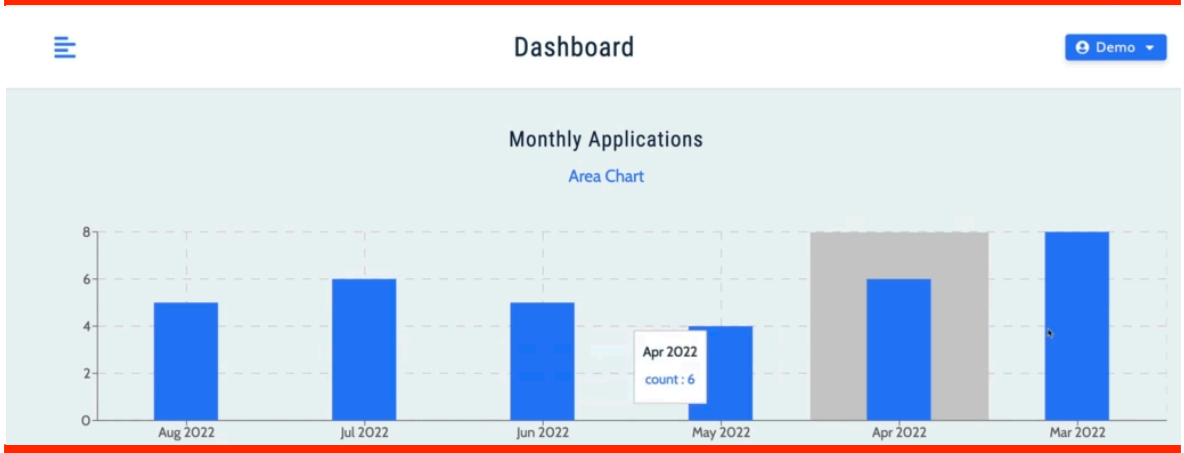
monthlyApplications =

```
monthlyApplications
  .map((item) => {
    const {
      _id: { year, month },
      count,
    } = item;
    const date = moment()
      .month(month - 1)
      .year(year)
      .format('MMM Y');
    return { date, count };
  })
```

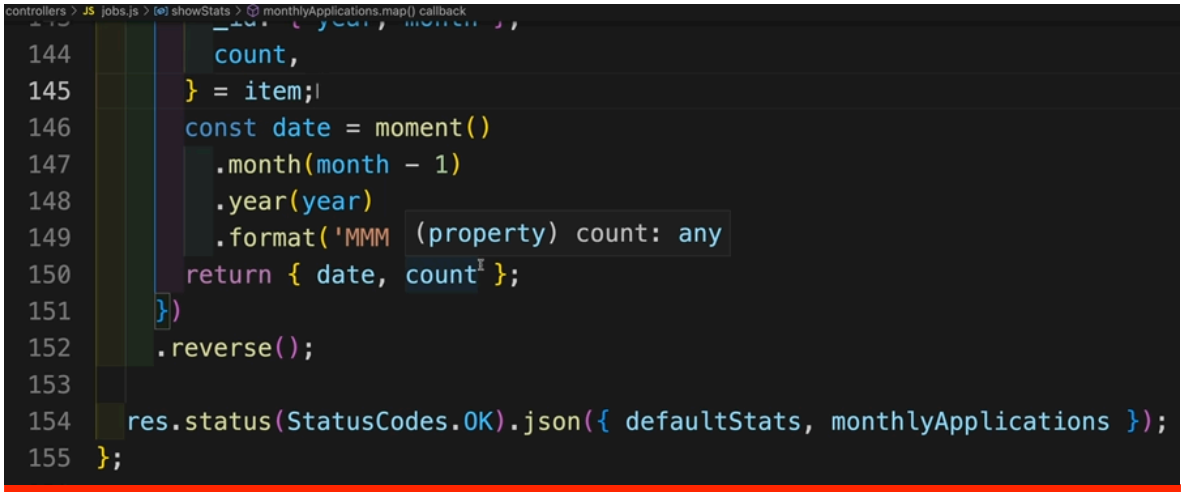



PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
[nodemon] starting `node app.js`
Server is listening on port 5000...
[
{ date: 'Aug 2022', count: 5 },
{ date: 'Jul 2022', count: 6 },
{ date: 'Jun 2022', count: 5 },
{ date: 'May 2022', count: 4 },


PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
{ date: 'Aug 2022', count: 5 },
{ date: 'Jul 2022', count: 6 },
{ date: 'Jun 2022', count: 5 },
{ date: 'May 2022', count: 4 },
{ date: 'Apr 2022', count: 6 },
{ date: 'Mar 2022', count: 8 }
]


Dashboard
Demo
Monthly Applications
Area Chart
8
6
4
2
0
Apr 2022
count : 6
Aug 2022
Jul 2022
Jun 2022
May 2022
Apr 2022
Mar 2022

.reverse();



controllers/jobs.js

==========

```
const Job = require('../models/Job');
const { StatusCodes } = require('http-status-codes');
const { BadRequestError, NotFoundError } = require('../errors');
const mongoose = require('mongoose');
const moment = require('moment');

const getAllJobs = async (req, res) => {
  const { search, status, jobType, sort } = req.query;

  const queryObject = {
```

```
    createdBy: req.user.userId,
  };

  if (search) {
    queryObject.position = { $regex:
search, $options: 'i' };
  }
  if (status && status !== 'all') {
    queryObject.status = status;
  }
  if (jobType && jobType !== 'all') {
    queryObject.jobType = jobType;
  }
  let result = Job.find(queryObject);

  if (sort === 'latest') {
    result = result.sort('-createdAt');
  }
  if (sort === 'oldest') {
    result = result.sort('createdAt');
  }
  if (sort === 'a-z') {
    result = result.sort('position');
  }
```

```
  if (sort === 'z-a') {
    result = result.sort('-position');
  }

  const page = Number(req.query.page)
ll 1;
  const limit = Number(req.query.limit)
ll 10;
  const skip = (page - 1) * limit;

  result = result.skip(skip).limit(limit);

  const jobs = await result;

  const totalJobs = await
Job.countDocuments(queryObject);
  const numOfPages =
Math.ceil(totalJobs / limit);


res.status(StatusCodes.OK).json({ jobs
, totalJobs, numOfPages });
};
const getJob = async (req, res) => {
```

```
  const {
    user: { userId },
    params: { id: jobId },
  } = req;

  const job = await Job.findOne({
    _id: jobId,
    createdBy: userId,
  });
  if (!job) {
    throw new NotFoundError(`No job
with id ${jobId}`);
  }

res.status(StatusCodes.OK).json({ job }
);
};

const createJob = async (req, res) => {
  req.body.createdBy = req.user.userId;
  const job = await
Job.create(req.body);

res.status(StatusCodes.CREATED).jso
```

```
n({ job });
};

const updateJob = async (req, res) => {
  const {
    body: { company, position },
    user: { userId },
    params: { id: jobId },
  } = req;

  if (company === '' || position === '') {
    throw new
BadRequestError('Company or
Position fields cannot be empty');
  }
  const job = await
Job.findByIdAndUpdate(
    { _id: jobId, createdBy: userId },
    req.body,
    { new: true, runValidators: true }
  );
  if (!job) {
    throw new NotFoundError(`No job
with id ${jobId}`);
```

```
  }

res.status(StatusCodes.OK).json({ job }
);
};

const deleteJob = async (req, res) => {
  const {
    user: { userId },
    params: { id: jobId },
  } = req;

  const job = await
Job.findByIdAndRemove({
    _id: jobId,
    createdBy: userId,
  });
  if (!job) {
    throw new NotFoundError(`No job
with id ${jobId}`);
  }
  res.status(StatusCodes.OK).send();
};
```

```
const showStats = async (req, res) => {
  let stats = await Job.aggregate([
    { $match: { createdBy:
mongoose.Types.ObjectId(req.user.use
rId) } },
    { $group: { _id: '$status', count:
{ $sum: 1 } } },
  ]);

  stats = stats.reduce((acc, curr) => {
    const { _id: title, count } = curr;
    acc[title] = count;
    return acc;
  }, {});

  const defaultStats = {
    pending: stats.pending || 0,
    interview: stats.interview || 0,
    declined: stats.declined || 0,
  };

  let monthlyApplications = await
Job.aggregate([
    { $match: { createdBy:
```

```
mongoose.Types.ObjectId(req.user.use
rId) } },
    {
      $group: {
        _id: { year: { $year: '$createdAt' },
month: { $month: '$createdAt' } },
        count: { $sum: 1 },
      },
    },
    { $sort: { '_id.year': -1, '_id.month':
-1 } },
    { $limit: 6 },
  ]);

  monthlyApplications =
monthlyApplications
    .map((item) => {
      const {
        _id: { year, month },
        count,
      } = item;
      const date = moment()
        .month(month - 1)
        .year(year)
```

```
        .format('MMM Y');
      return { date, count };
    })
    .reverse();

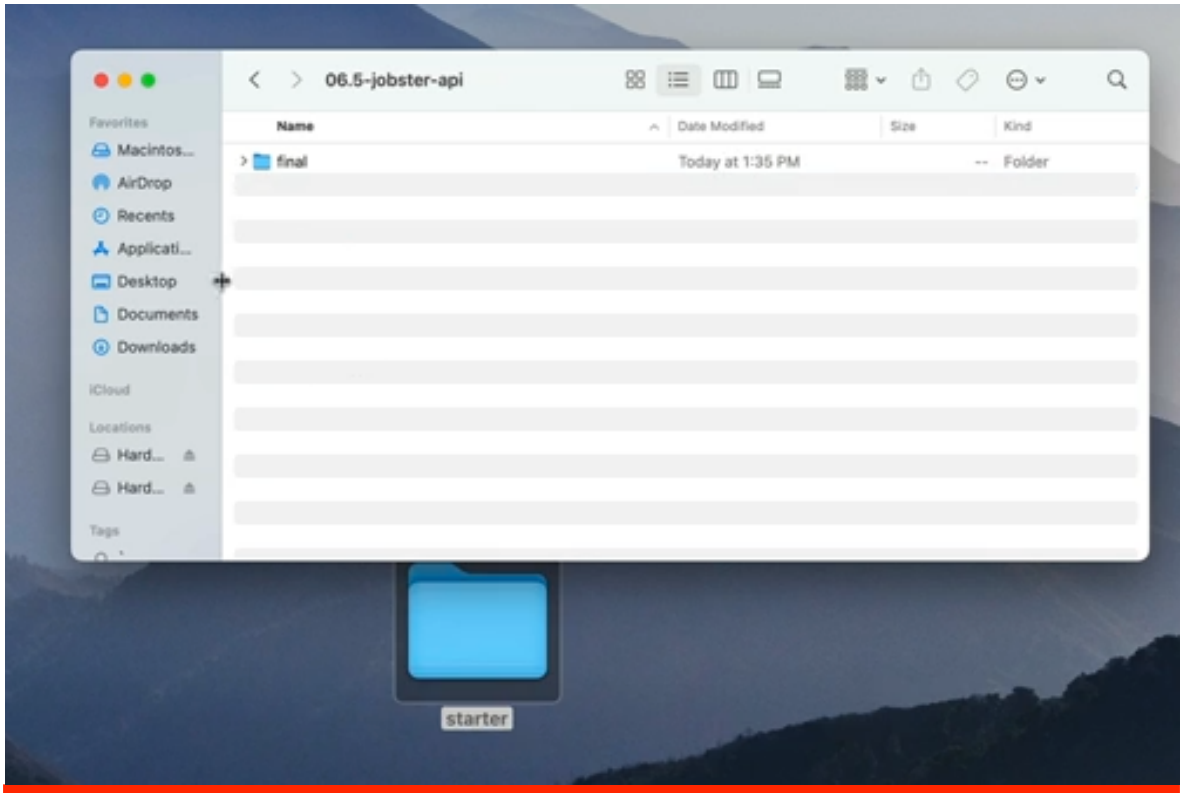

res.status(StatusCodes.OK).json({ defa
ultStats, monthlyApplications });
};

module.exports = {
  createJob,
  deleteJob,
  getAllJobs,
  updateJob,
  getJob,
  showStats,
};
```

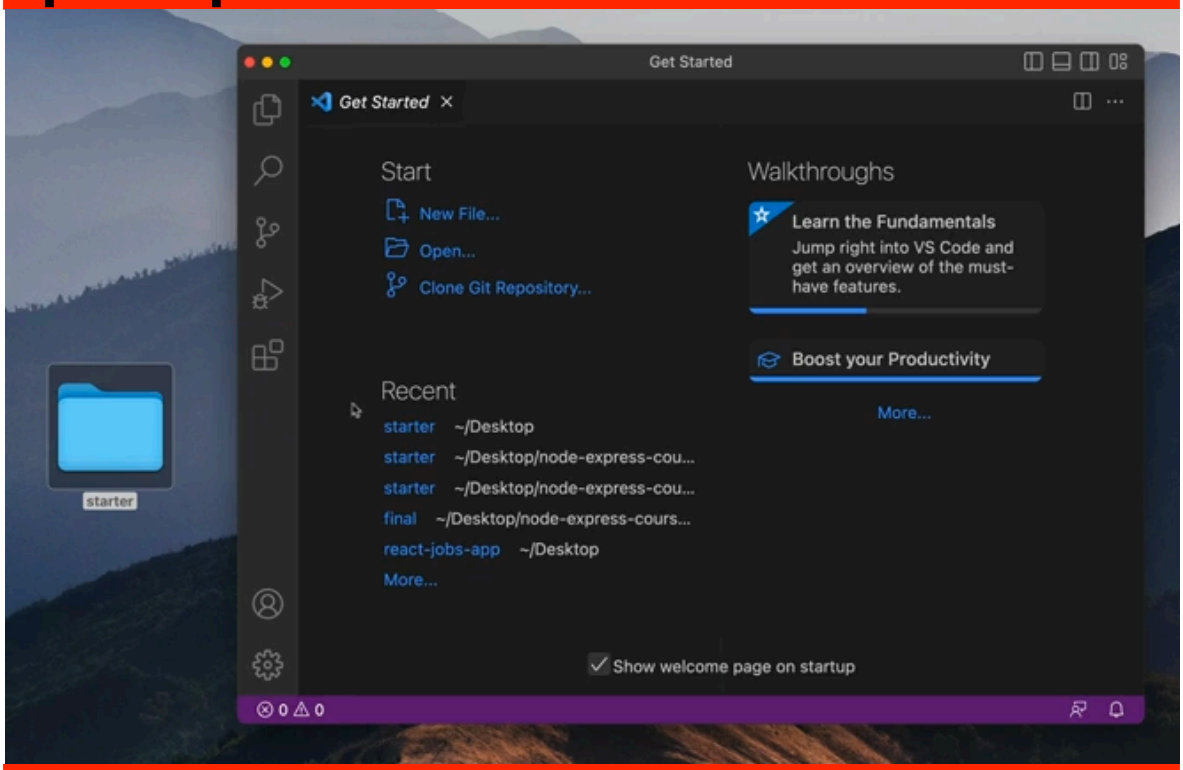
20 - Deployment

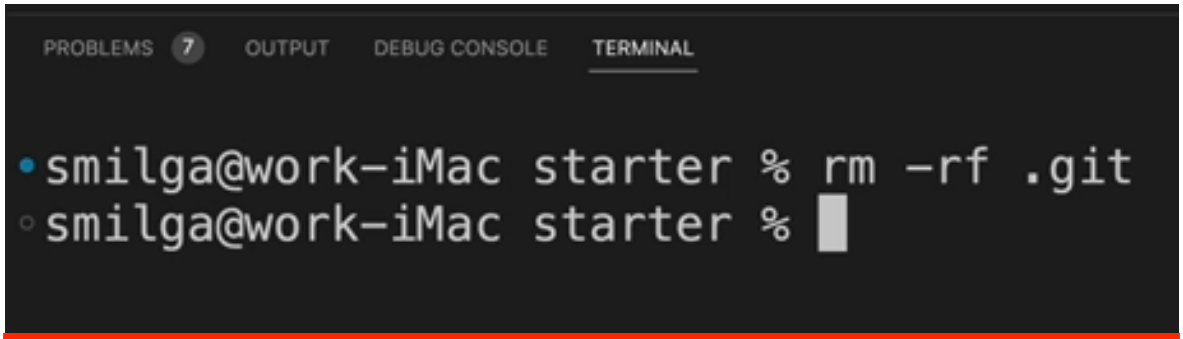
==========

move this application to Desktop



open up in the vs code

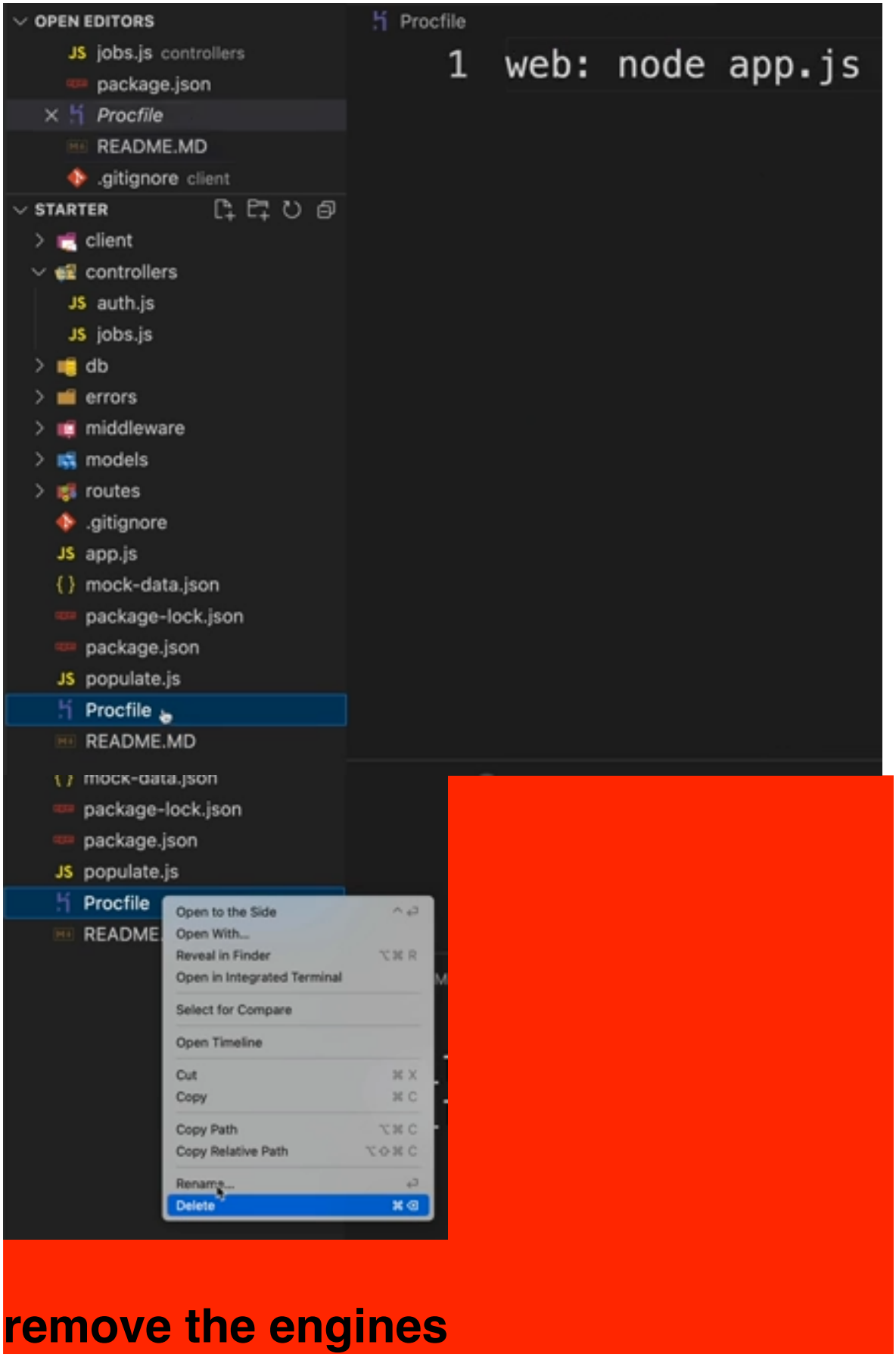




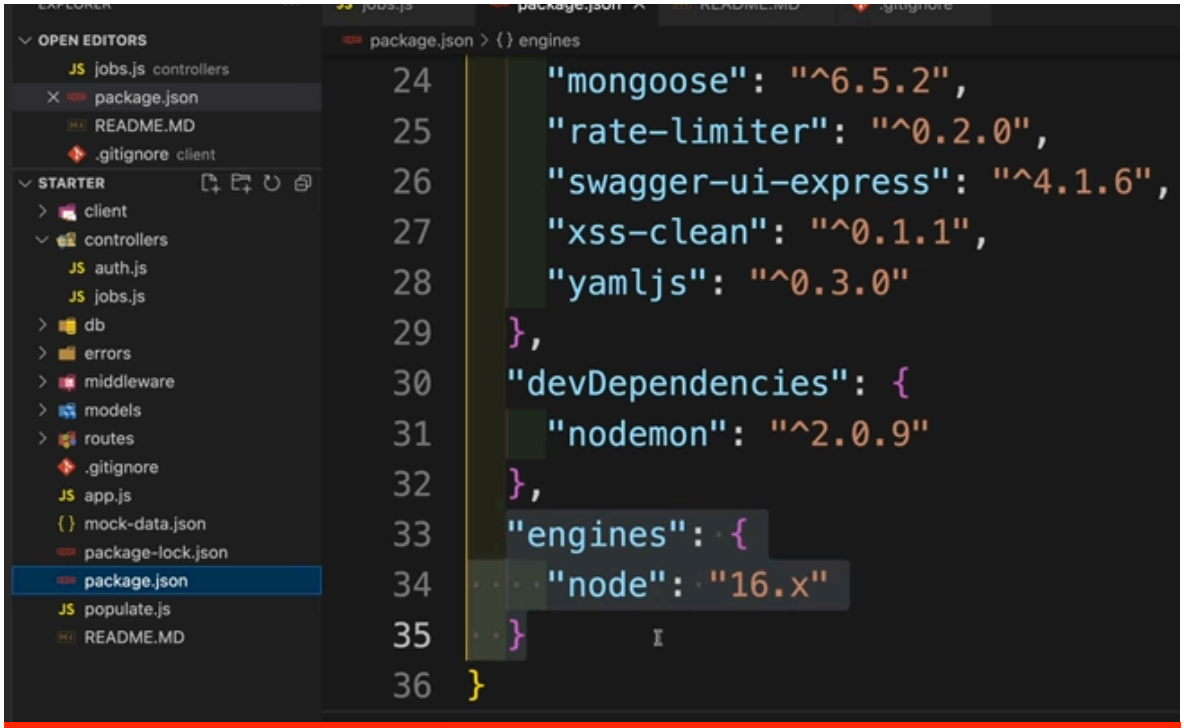

```
- remove Procfile
- remove engines from package.json

```json
"engines": {
    "node": "16.x"
  }
```

- fix build folder (remove /build from client/.gitignore)
- setup new github repo
- deploy to render
```



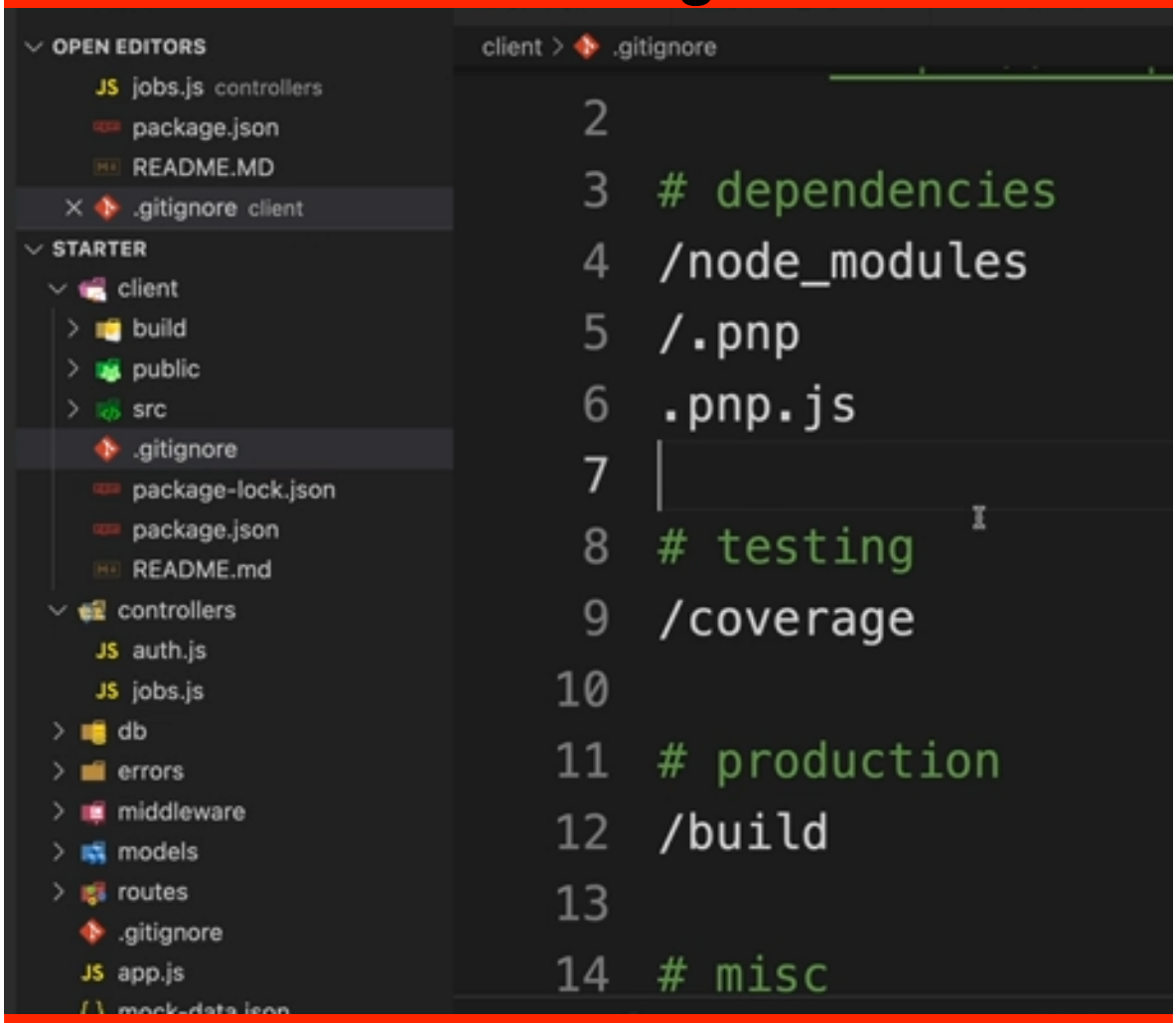
remove the engines



in this project, we have front end React project,

so we have to create as one repo, server and front end

in the client we have .gitignore,
here we have ignored to push /build file
so client will be missing in server

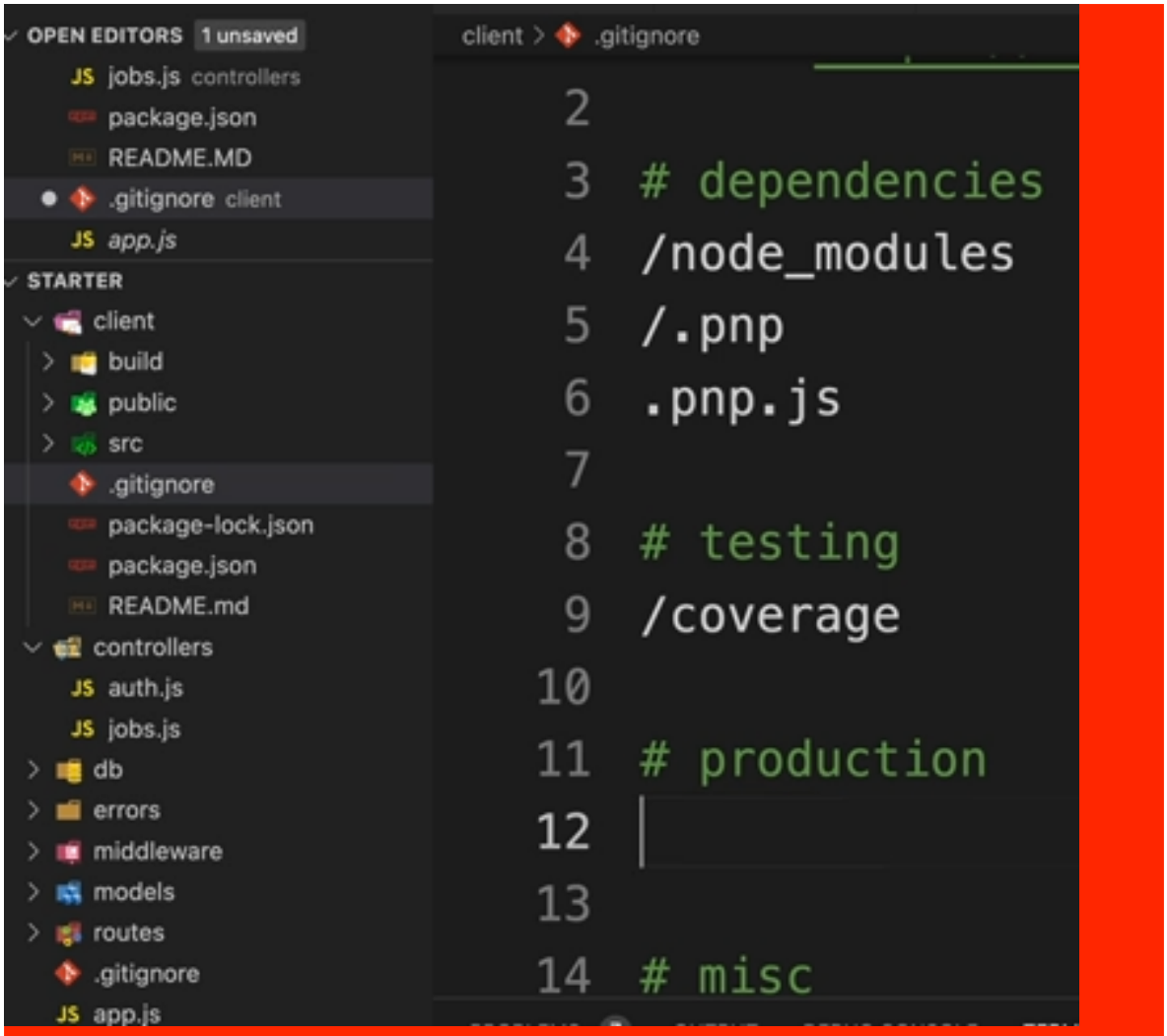


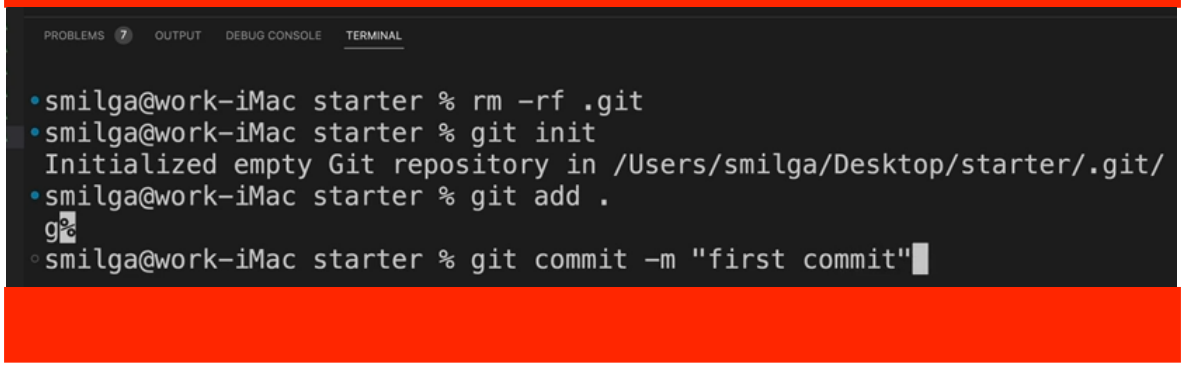
to avoid that we have to remove this rule from gitignore or
more complex, one would be setting public folder or

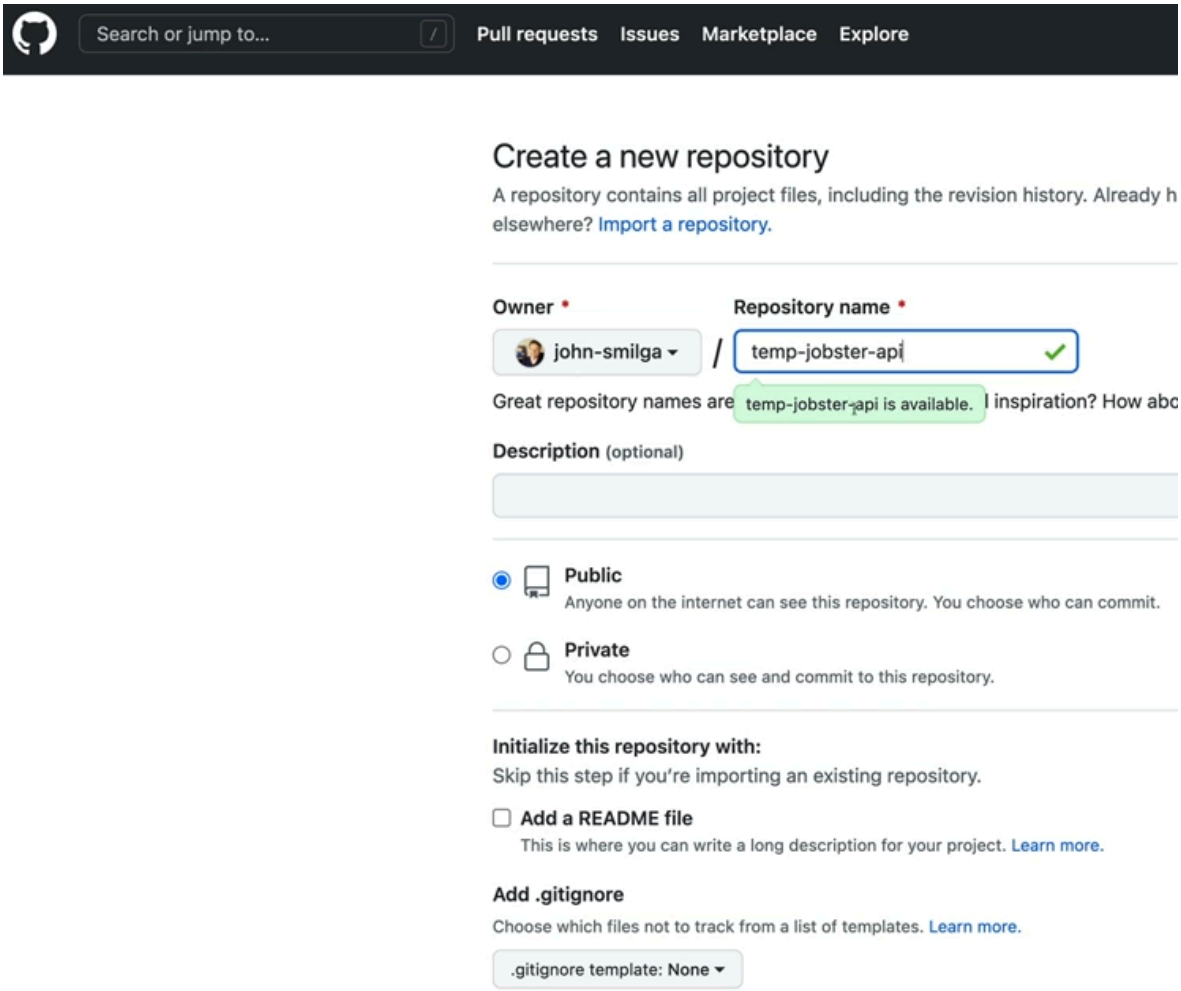
whatever folder you want here on the server and then
copy and paste the contents of the build one

if we do the second option, we have to change this logic also

so we will go with first option
remove /build from gitignore


OPEN EDITORS 1 unsaved
jobs.js controllers
package.json
README.MD
.gitignore client
app.js
STARTER
client
build
public
src
.gitignore
package-lock.json
package.json
README.md
controllers
auth.js
jobs.js
db
errors
middleware
models
routes
.gitignore
app.js
client > .gitignore
dependencies
/node_modules
/.pnp
.pnp.js
testing
/coverage
production
misc

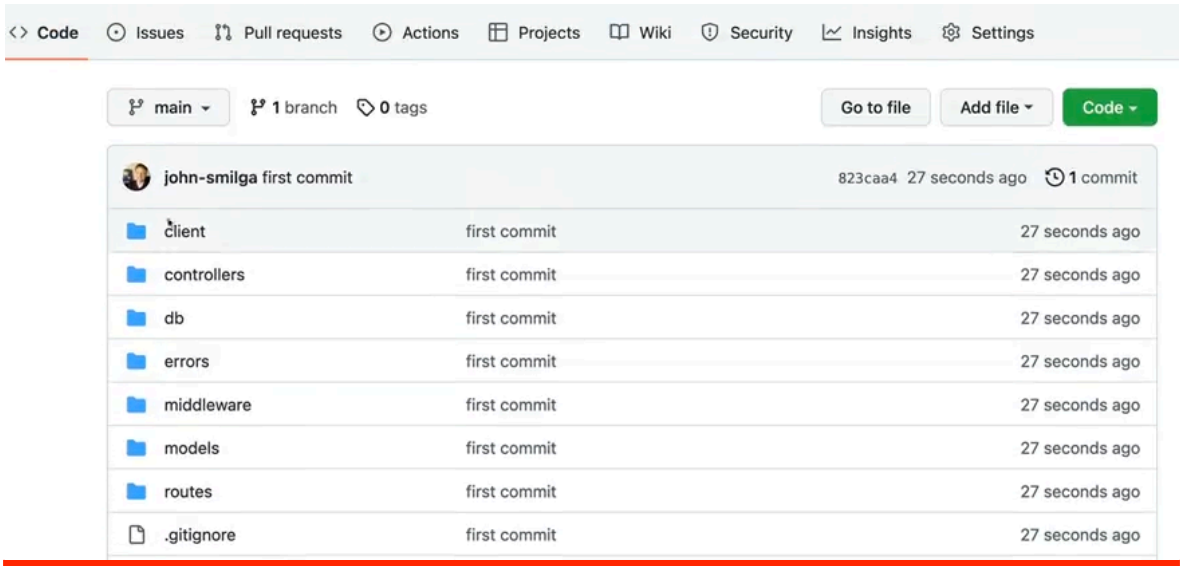

PROBLEMS 7
OUTPUT
DEBUG CONSOLE
TERMINAL
smilga@work-iMac starter % rm -rf .git
smilga@work-iMac starter % git init
Initialized empty Git repository in /Users/smilga/Desktop/starter/.git/
smilga@work-iMac starter % git add .
g%
smilga@work-iMac starter % git commit -m "first commit"



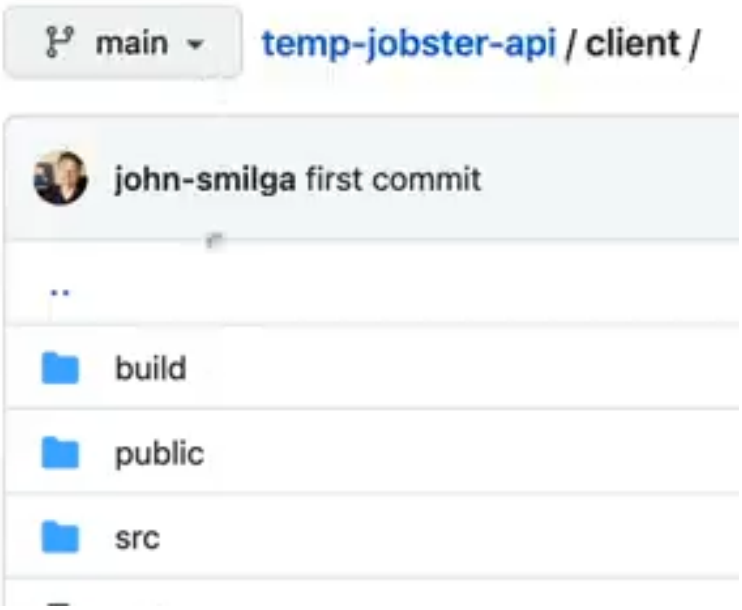
...or push an existing repository from the command line

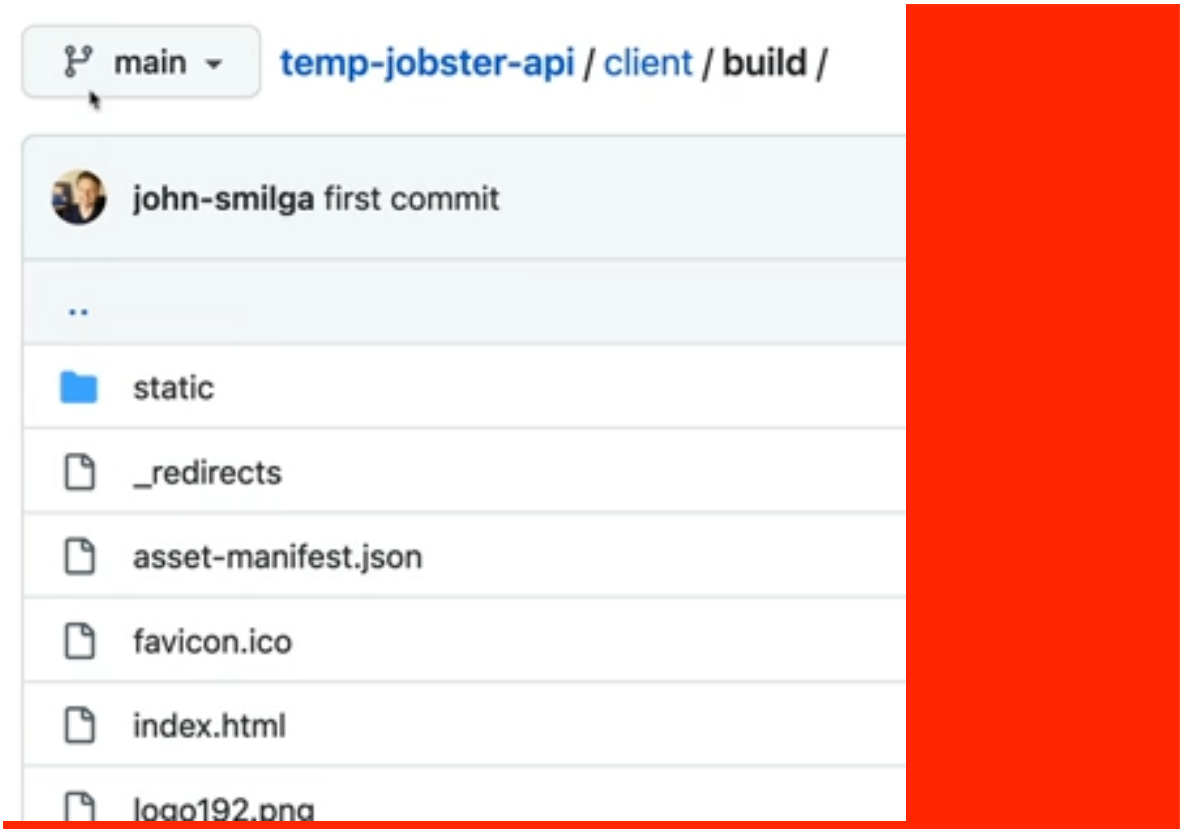
```
git remote add origin git@github.com:john-smilga/temp-jobster-api.git
git branch -M main
git push -u origin main
```

go and check in the GitHub

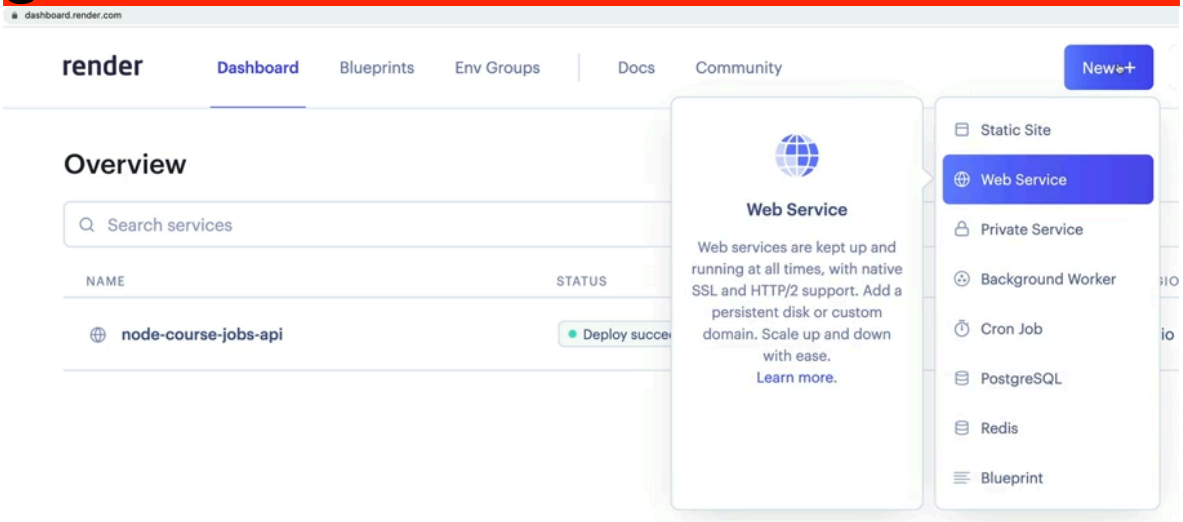


including /client/build files

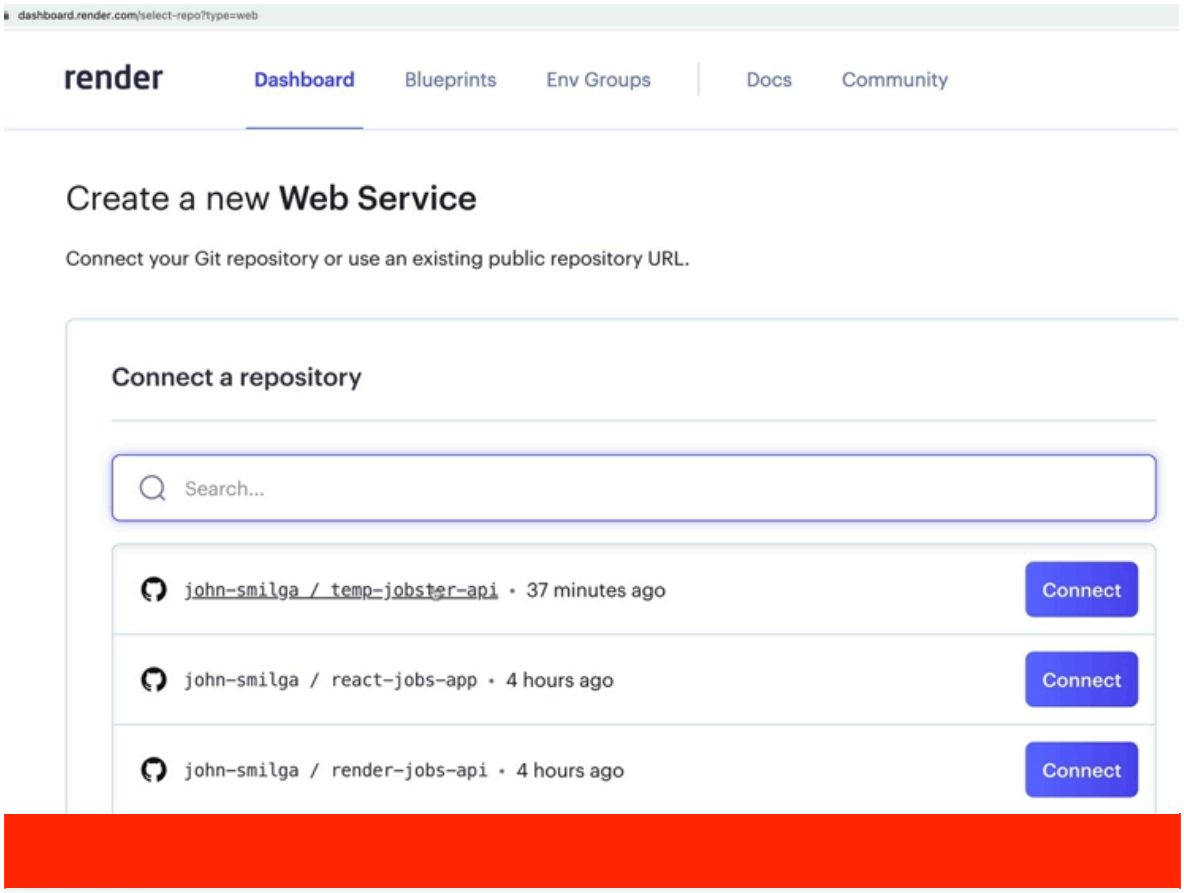


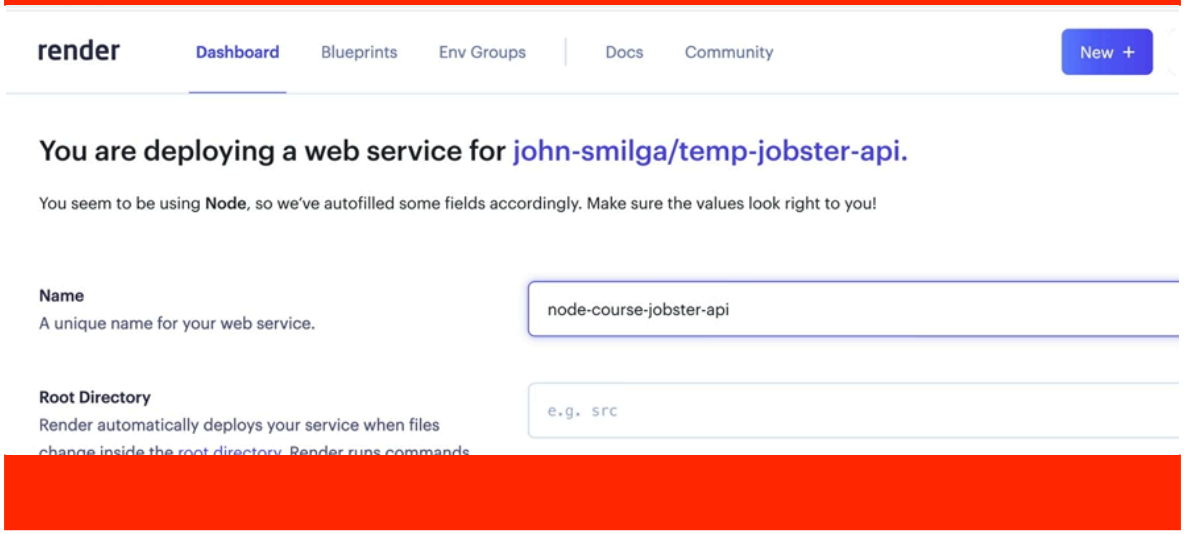


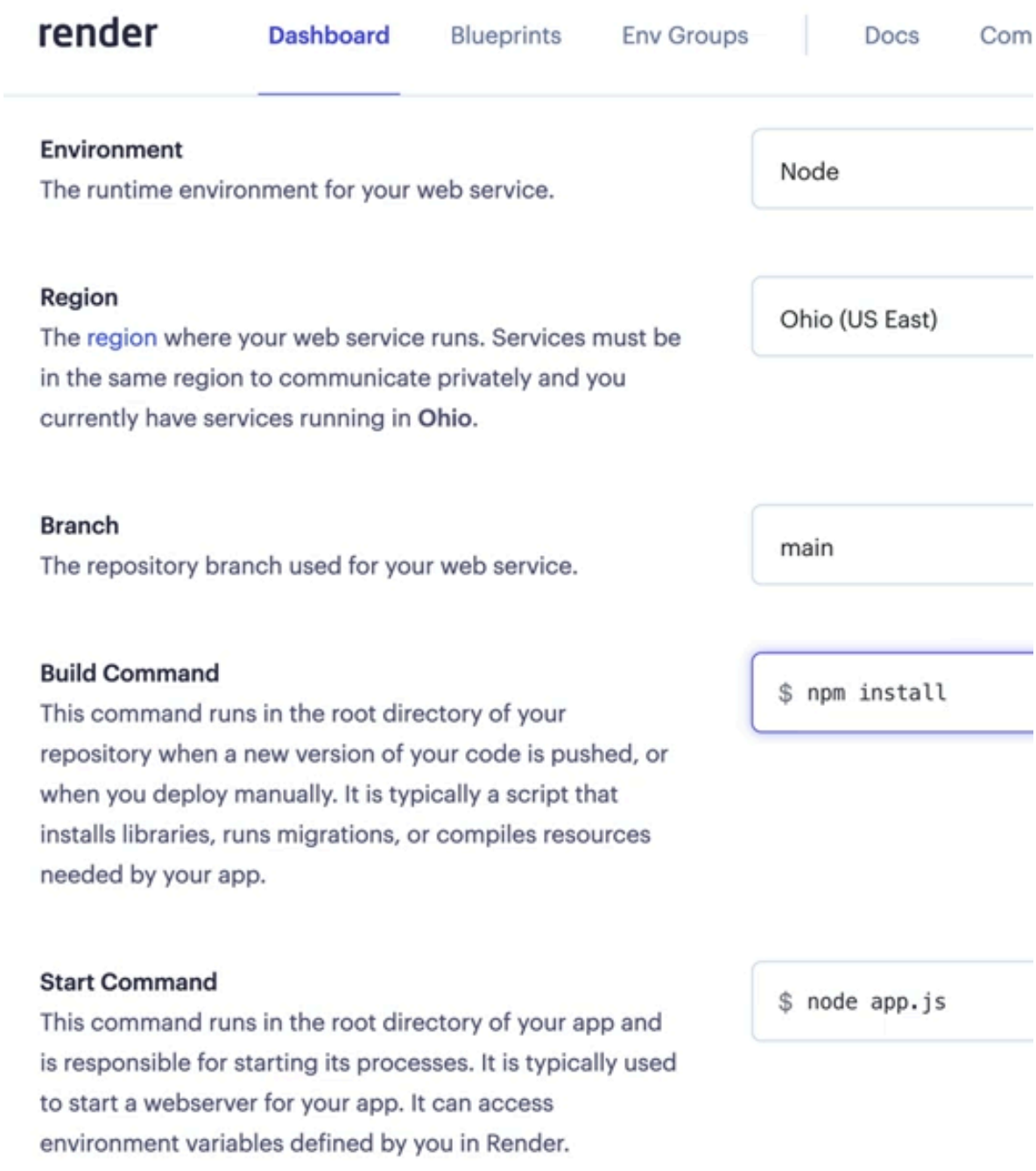
go to the render



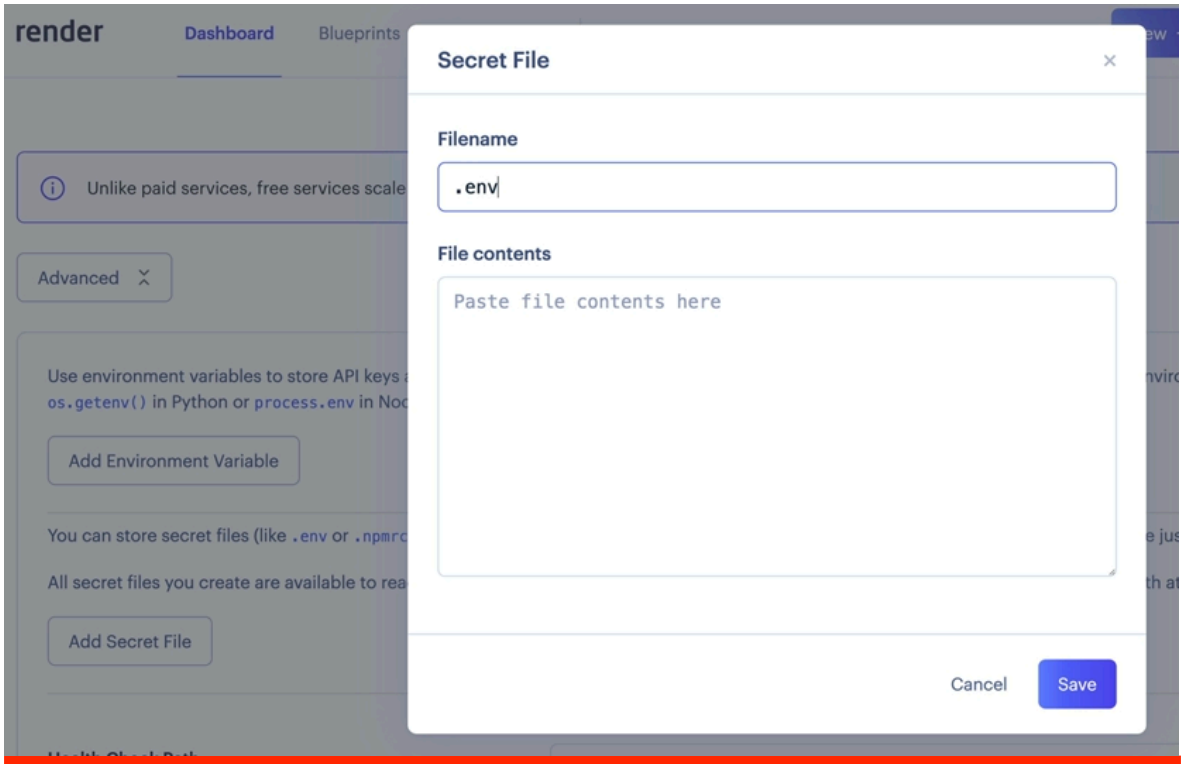
check our project and press connect

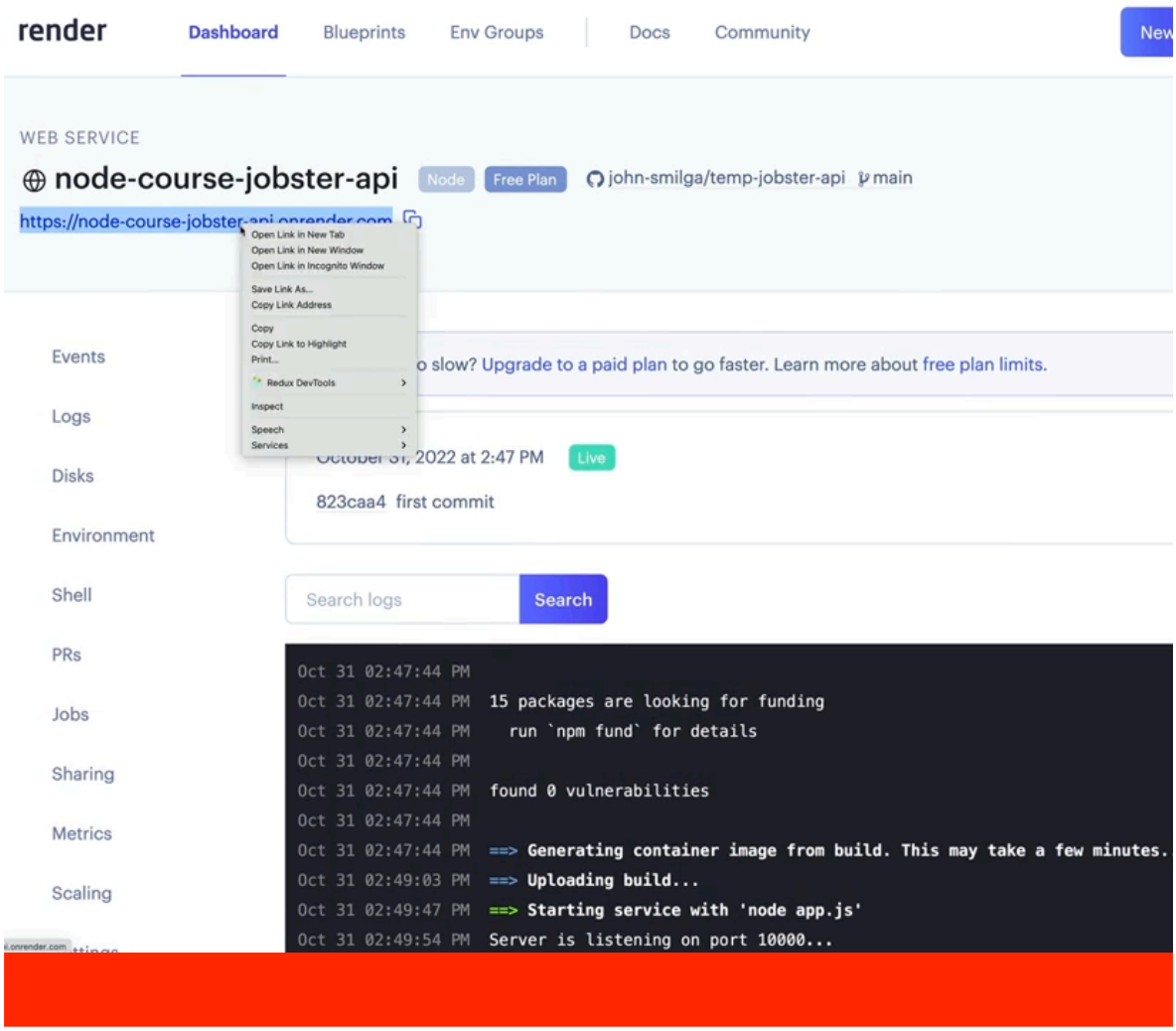

dashboard.render.com/select-repo?type=web
render
Dashboard
Blueprints
Env Groups
Docs
Community
Create a new Web Service
Connect your Git repository or use an existing public repository URL.
Connect a repository
Search...
john-smilga / temp-jobster-api • 37 minutes ago
Connect
john-smilga / react-jobs-app • 4 hours ago
Connect
john-smilga / render-jobs-api • 4 hours ago
Connect


render
Dashboard
Blueprints
Env Groups
Docs
Community
New +
You are deploying a web service for john-smilga/temp-jobster-api.
You seem to be using Node, so we've autofilled some fields accordingly. Make sure the values look right to you!
Name
A unique name for your web service.
node-course-jobster-api
Root Directory
Render automatically deploys your service when files
e.g. src



here we will try advanced option with Add Secret File
copy the contents of .env (all secret variables)


render
Dashboard
Blueprints
Secret File
Filename
.env
File contents
Paste file contents here
Unlike paid services, free services scale
Advanced
Use environment variables to store API keys
os.getenv() in Python or process.env in Noc
Add Environment Variable
You can store secret files (like .env or .npmrc
All secret files you create are available to rea
Add Secret File
Cancel
Save


render
Dashboard
Blueprints
Env Groups
Docs
Community
New
WEB SERVICE
node-course-jobster-api
Node
Free Plan
john-smilga/temp-jobster-api
main
https://node-course-jobster-api.onrender.com
Open Link in New Tab
Open Link in New Window
Open Link in Incognito Window
Save Link As...
Copy Link Address
Copy
Copy Link to Highlight
Print...
Redux DevTools
Inspect
Speech
Services
Events
Logs
Disks
Environment
Shell
PRs
Jobs
Sharing
Metrics
Scaling
o slow? Upgrade to a paid plan to go faster. Learn more about free plan limits.
2022 at 2:47 PM
Live
823caa4 first commit
Search logs
Search
Oct 31 02:47:44 PM
Oct 31 02:47:44 PM 15 packages are looking for funding
Oct 31 02:47:44 PM run `npm fund` for details
Oct 31 02:47:44 PM
Oct 31 02:47:44 PM found 0 vulnerabilities
Oct 31 02:47:44 PM
Oct 31 02:47:44 PM ==> Generating container image from build. This may take a few minutes.
Oct 31 02:49:03 PM ==> Uploading build...
Oct 31 02:49:47 PM ==> Starting service with 'node app.js'
Oct 31 02:49:54 PM Server is listening on port 10000...

check on the browser