

## 1 - Intro

---

---

**<https://redux-toolkit-jobster.netlify.app/landing>**

**Login and press Demo App Button**

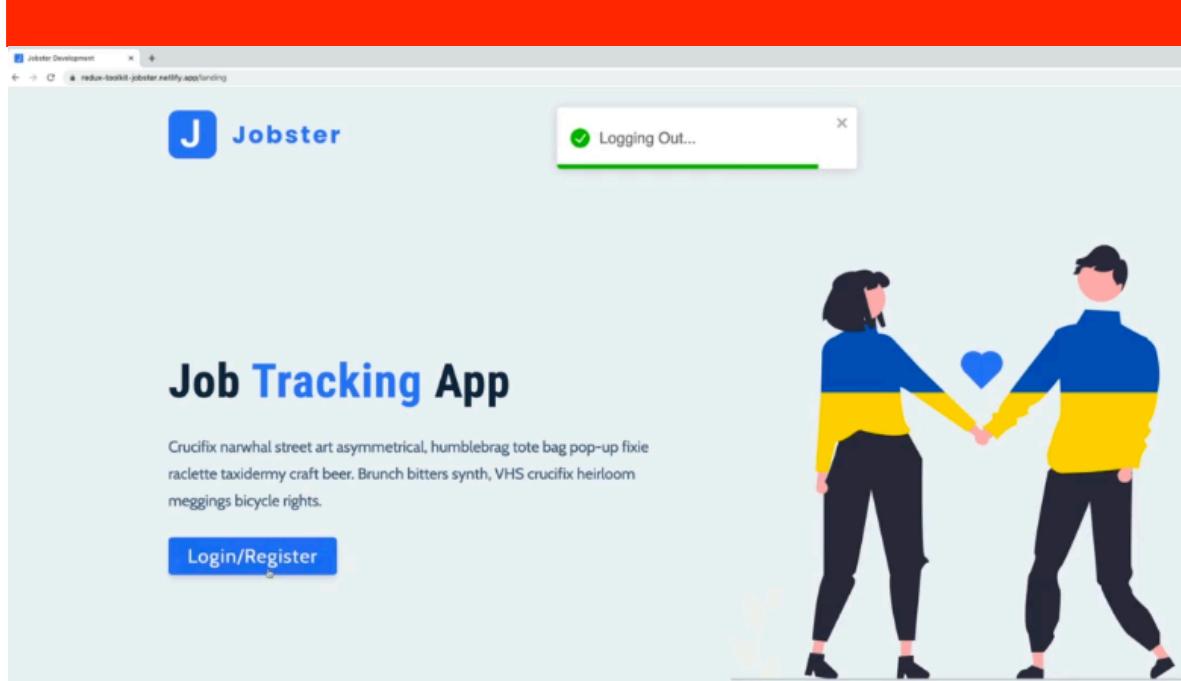
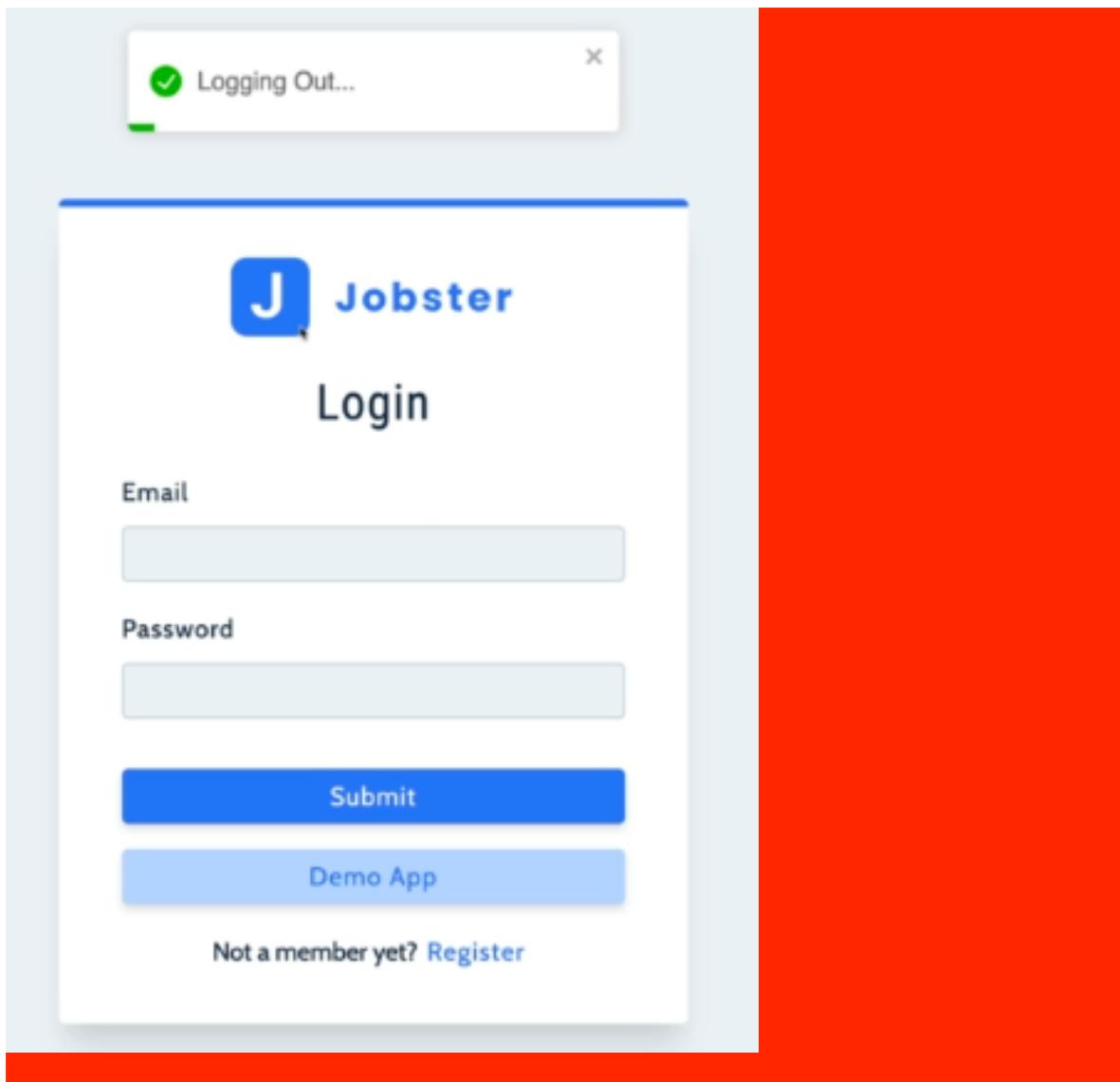
🔗 <https://redux-toolkit-jobster.netlify.app/landing>



# 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 Development

Dashboard

Test User

Stats

All Jobs

Add Job

Profile

24 Pending Applications

27 Interviews Scheduled

24 Jobs Declined

Monthly Applications

Area Chart

Month	Applications
Jul 2021	1
Aug 2021	4
Sep 2021	3
Oct 2021	2
Nov 2021	2
Dec 2021	5

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

The screenshot shows the Jobster application's dashboard. On the left, there is a sidebar with navigation links: Stats, All Jobs (which is selected and highlighted in blue), Add Job, and Profile. The main area is titled "Dashboard" and contains a "Search Form" with fields for Search, Status (set to "all"), Type (set to "all"), Sort (set to "latest"), and a "Clear Filters" button. Below the search form, it says "75 Jobs Found" and lists two job entries:

B	Civil Engineer Bechtelar-Bednar	K	Accounting Assistant III Kunze And Sons
📍 Kiamba	Dec 26th, 2021	📍 Kafr Mandā	Dec 22nd, 2021
💻 Internship	Declined	💻 Remote	Interview

Below the search form, there is a large orange section containing the following text:

## 2 - Setup

---

```
1 # Jobster API
2
3 ##### Starter
4
5 The starter is a copy of jobs-api final project, just with additional data.
6
7 ##### Setup
8
9 - navigate to 06.5-jobster-api/starter
10 - install dependencies
11
12 `` `sh
13 npm install
14 `` `
```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Displays the project structure:
  - 06.5-jobster-api (selected)
  - final
  - starter
    - client
  - controllers
    - auth.js
    - jobs.js
  - db
  - errors
  - middleware
  - models
- Terminal:** Shows the command line history:

```
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
```

```
> 05-JWT-Basics
> 06-jobs-api
✓ 06.5-jobster-api
> final
< 06.5-jobster-api
  > starter
    > client
    < controllers
      JS auth.js
      JS jobs.js
    > db
    > errors
    > middleware
    < models
      JS Job.js
      JS User.js
    > node_modules
    > routes
      .env
      JS app.js
    > package-lock.json
    > package.json
    > Profile
    > README.MD
    > swagger.yaml
  > 07-file-upload
  > 08-send-email
  > 09-stripe-payment
  > 10-e-commerce-api
  > 11-auth-workflow
  > README.MD
```

27 - start the project  
28  
29 ````sh  
30 npm start  
31 ````  
32  
33 - you should see "Server is listening ...." text  
34  
35 ##### Spring Cleaning  
36

PROBLEMS (6) OUTPUT DEBUG CONSOLE TERMINAL

npm audit fix --force

Run `npm audit` for details.

smilga@work-iMac starter % npm start

> 06-jobs-api@1.0.0 start  
> node app.js

Server is listening on port 5000...

```

35 ##### Spring Cleaning
36
37 ##### Remove Swagger
38
39 - delete swagger.yaml file
40 - remove these lines of code
41 - delete swagger.yaml
42   app.js
43
44 ````js
45 const swaggerUI = require('swagger-ui-express');
46 const YAML = require('yamljs');
47 const swaggerDocument = YAML.load('./swagger.yaml');
48
49 app.get('/', (req, res) => {
50   res.send('<h1>Jobs API</h1><a href="/api-docs">Documentation</a>');
51 });
52 app.use('/api-docs', swaggerUI.serve, swaggerUI.setup(swaggerDocument))
53 ````
```

## /starter/app.js

```

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');
8  const rateLimiter = require('express-rate-limit');
9
10 // Swagger
11 const swaggerUI = require('swagger-ui-express');
12 const YAML = require('yamljs');
13 const swaggerDocument = YAML.load('./swagger.yaml');
14
15 const express = require('express');
16 const app = express();
17
18 const connectDB = require('./db/connect');
19 const authenticateUser = require('./middleware/authentication');
20 // routers
21 const authRouter = require('./routes/auth');
22 const jobsRouter = require('./routes/jobs');
23 // error handler
24 const notFoundMiddleware = require('./middleware/not-found');
25 const errorHandlerMiddleware = require('./middleware/error-handler');
26
27 app.set('trust proxy', 1);
28 app.use(
29   rateLimiter({
30     windowMs: 15 * 60 * 1000, // 15 minutes
31     max: 100, // limit each IP to 100 requests per windowMs
32   })
33 );
```

```
34 app.use(express.json());
35 app.use(helmet());
36 app.use(cors());
37 app.use(xss());
38
39 app.get('/', (req, res) => {
40   res.send(`<h1>Jobs API</h1><a href="/api-docs">Documentation</a>`);
41 });
42 app.use('/api-docs', swaggerUI.serve, swaggerUI.setup(swaggerDocument));
43
44 // routes
45 app.use('/api/v1/auth', authRouter);
46 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
47
48 app.use(notFoundMiddleware);
49 app.use(errorHandlerMiddleware);
50
51 const port = process.env.PORT || 5000;
52
53 const start = async () => {
54   try {
55     await connectDB(process.env.MONGO_URI);
56     app.listen(port, () => {
57       console.log(`Server is listening on port ${port}...`);
58     });
59   } catch (error) {
60     console.log(error);
61   }
62 };
63
64 start();
```

## /starter/app.js

```
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 // Swagger
11 const swaggerUI = require('swagger-ui-express');
12 const YAML = require('yamljs');
13 const swaggerDocument = YAML.load('./swagger.yaml');
14
15 const express = require('express');
16 const app = express();
17
18 const connectDB = require('./db/connect');
19 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);
```

```
06.5-jobster-api > starter > js app.js > ...
29 app.use(express.json());
30 app.use(helmet());
31 app.use(cors());
32 app.use(xss());
33
34 // routes
35 app.use('/api/v1/auth', authRouter);
36 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
37
38 app.use(notFoundMiddleware);
39 app.use(errorHandlerMiddleware);
40
41 const port = process.env.PORT || 5000;
42
43 const start = async () => {
```

```
55 ##### Remove API Limiter
56
57 - remove these lines of code
58
59 app.js
60 |
61 ``js
62 const rateLimiter = require('express-rate-limit');
63
64 app.set('trust proxy', 1);
65 app.use(
66   rateLimiter({
67     windowMs: 15 * 60 * 1000, // 15 minutes
68     max: 100, // limit each IP to 100 requests per windowMs
69   })
70 );
71 ``
```

**remove rateLimiter**

```
06.5-jobster-api > starter > JS app.js > ...
1  require('dotenv').config();
2  require('express-async-errors');
3  (req: any, res: any, next: any): void;
4  // extra security packages
5  const helmet = require('helmet');
6  const cors = require('cors');
7  const xss = require('xss-clean');
8  const rateLimiter = require('express-rate-limit');
9
10 const express = require('express');
11 const app = express();
12
13 const connectDB = require('./db/connect');
14 const authenticateUser = require('./middleware/authentication');
```

```
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.set('trust proxy', 1);
22 app.use([
23   rateLimiter({
24     windowMs: 15 * 60 * 1000, // 15 minutes
25     max: 100, // limit each IP to 100 requests per windowMs
26   })
27 ]);
28 app.use(express.json());
29 app.use(helmet());
```

```
73 ##### Remove CORS
74
75 - we don't want external JS apps to access our API (only our
  front-end)
76 - remove these lines of code
77
78 ````js
79 const cors = require('cors');
80 app.use(cors());
81 ````
```

```
06.5-jobster-api > starter > JS app.js > [e] 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 > ...
```

```
10 // error handler
11
12 const notFoundMiddleware = require('./middleware/not-found');
13 const errorHandlerMiddleware = require('./middleware/error-handler');
14
15 app.use(express.json());
16 app.use(helmet());
17 app.use(cors());
18 app.use(xss());
19
20 // routes
21 app.use('/api/v1/auth', authRouter);
22 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
23
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);
```

```
83 ##### Package.json
84
85 - add "dev" script with nodemon
86 - change engines to current version (in my case 16)
87
88 package.json
89
90 ````js
91
92 "scripts": {
93     "start": "node app.js",
94     "dev": "nodemon app.js"
95 },
96
97 "engines": {
98     "node": "16.x"
99 }
100 ````|
101
102 - restart server with "npm run dev"
```

```
06.5-jobster-api > starter > package.json > scripts > dev
```

```
1  {
2    "name": "06-jobs-api",
3    "version": "1.0.0",
4    "description": "",
5    "main": "app.js",
6    ▷ Debug
7    "scripts": {
8      "start": "node app.js",
9      "dev": "nodemon app.js"
10     },
11    "author": "",
12    "license": "ISC",
13    "dependencies": {
14      "bcryptjs": "^2.4.3",
15      "cors": "^2.8.5"
```

**if we want, change it to latest node version**

```
20  },
21  "devDependencies": {
22    "nodemon": "^2.0.9"
23  },
24  "engines": {
25    "node": "16.x"
26  }
27}
28
```

```
{
  "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"
}
}
```

The screenshot shows the MongoDB Cloud interface. On the left, there's a sidebar with a tree view of projects and databases. Under 'DEPLOYMENT', 'Database', and 'PREVIEW', there are several entries like '01-task-manager', '06-JOB-API', and '065-JOBSTER'. Under 'DATA SERVICES', there are 'Triggers', 'Data API', 'Data Federation', and 'SECURITY' sections. The 'SECURITY' section includes 'Database Access', 'Network Access', and 'Advanced' options. The main area is titled '065-JOBSTER' and shows 'DATABASE SIZE: 0B', 'INDEX SIZE: 8KB', and 'TOTAL COLLECTIONS: 1'. A table lists the 'users' collection with 0 documents, 0B size, 2 indexes, and 4KB index size. A 'CREATE COLLECTION' button is at the top right.

**check the functionality**

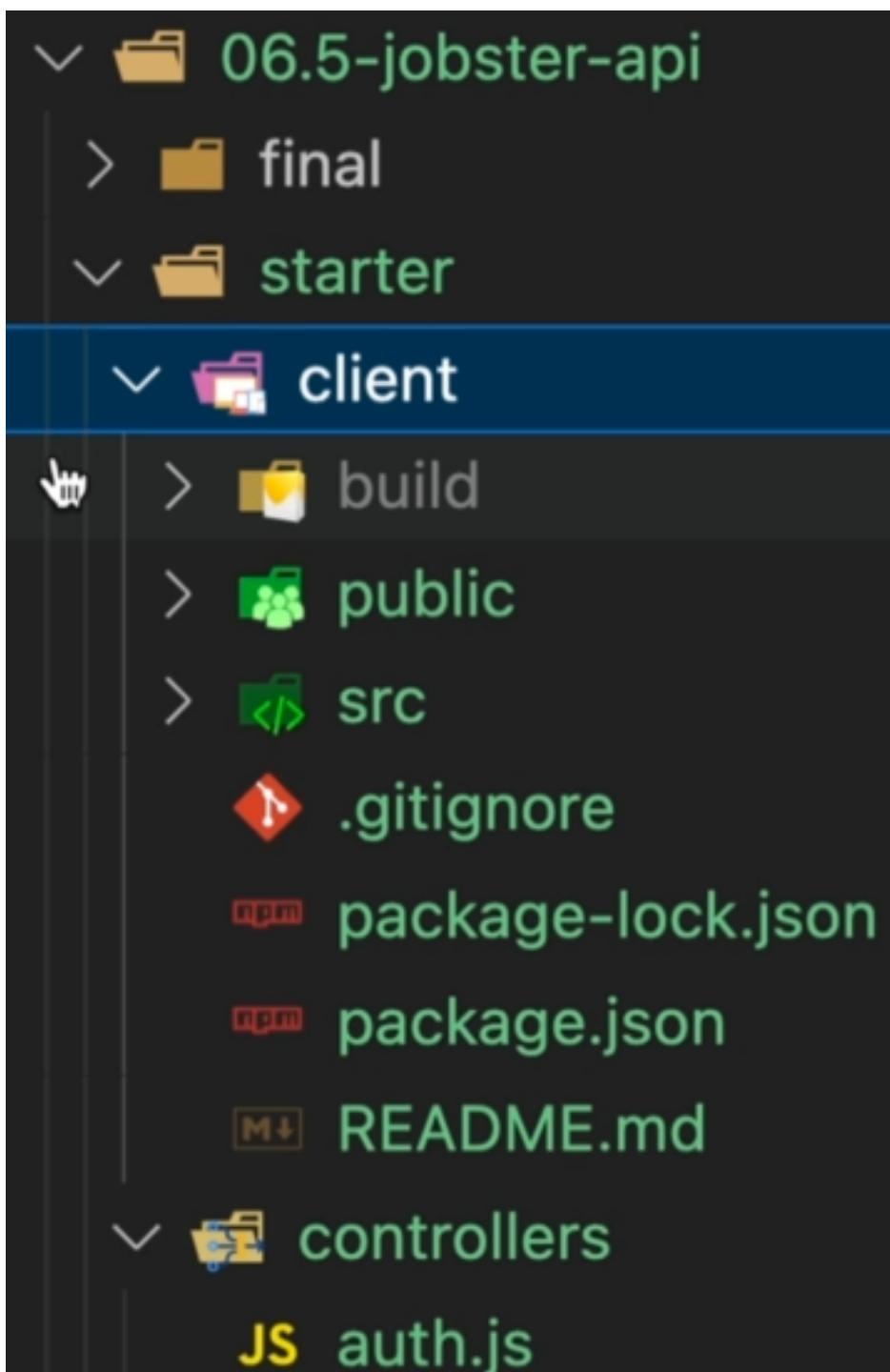


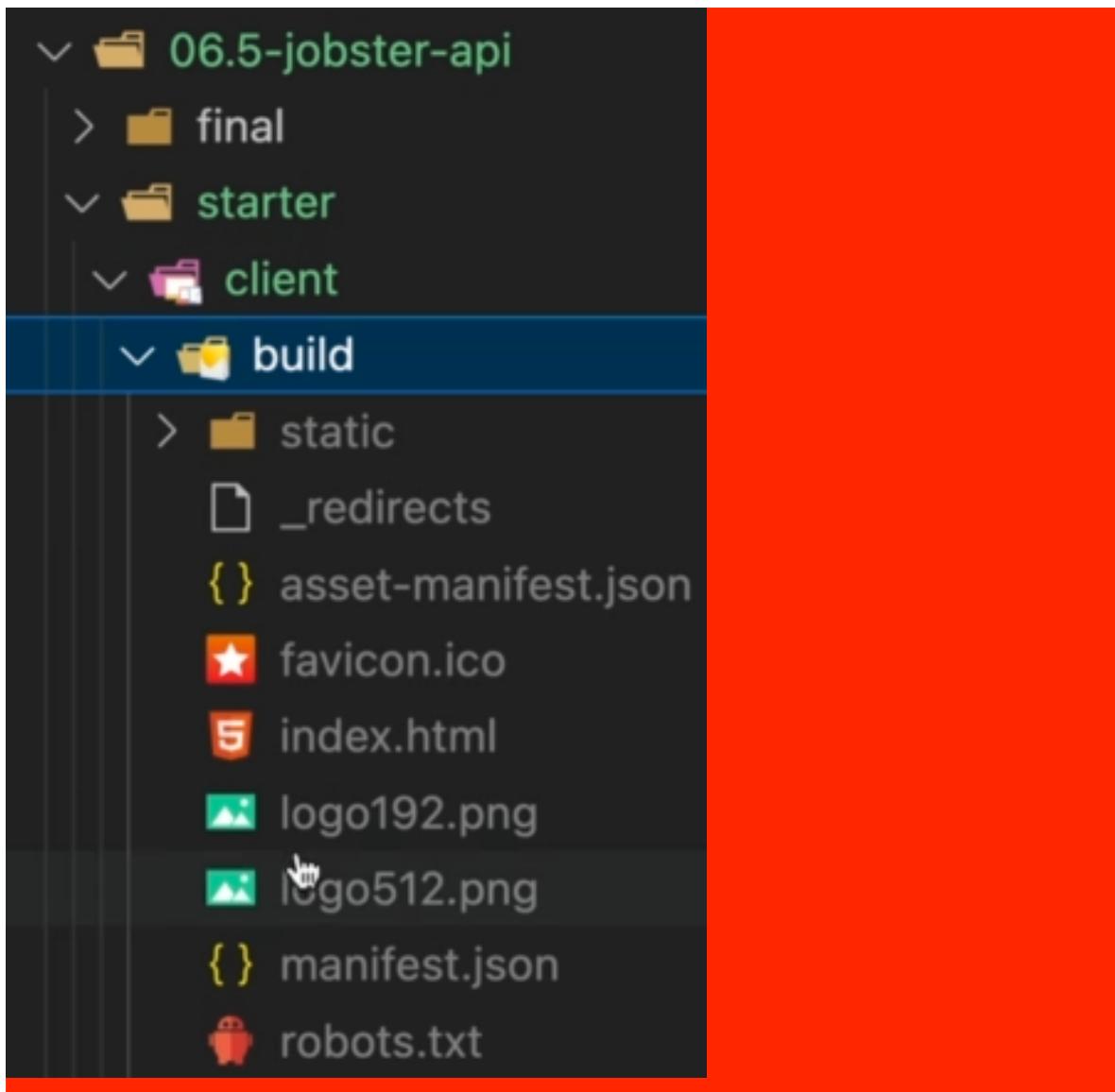
**Route does not exist**

**4 - Client Folder**

---

```
104 ##### Client Folder
105
106 - let's explore client folder
107 - open client folder
108 - it's a react app built with CRA
109 - it's the same as in my React Course (JOBSTER APP),
110   just base url points to our current server (instead of heroku app)
111
112 utils/axios.js
113
114 ````js
115 const customFetch = axios.create({
116   baseURL: '/api/v1',
117 });
118 `````
119
120 - notice the build folder (production ready application)
121 - in CRA we can create build folder by running "npm run build"
122 - that's the one we will use for our front-end
```





OPEN EDITORS

06.5-jobster-api > starter > README.MD > # Jobster API > ##### Package.json > ##### Client Folder

```
102 - restart server with "npm run dev"
103
104 ##### Client Folder
105
106 - let's explore client folder
107 - open client folder
108 - it's a react app built with CRA
109 - it's the same as in my React Course (JOBSTER APP),
110 just base url points to https://jobsterapp.com server (instead of heroku app)
111
112 utils/axios.js
113
114 ``js
115 const customFetch = axios.create({
```

A modal dialog box is displayed in the bottom right corner, asking "Are you sure you want to move 'public' into 'build'?", with options "Move", "Cancel", and "Do not ask me again".

**previous project, we created in public folder**

```
> 📁 controllers
> 📁 db
> 📁 errors
> 📁 middleware
> 📁 models
▽ 📁 public
  JS browser-app.js
  JS edit-task.js
  ★ favicon.ico
  5 index.html
  3 main.css
  3 normalize.css
  5 task.html
> 📁 routes
  🔍 .gitignore
  JS app.js
  📜 package-lock.json
  📜 package.json
```

## 5 - Setup Front-End

---

```
124 ##### Setup Front-End
125
126 - require "path" module
127 - setup express static (as first middleware)
128 | to serve static assets from client/build
129 - so now instead of public folder we are using client/build
130
131 app.js
132
133 ````js
134 const path = require('path');
135
136 app.use(express.static(path.resolve(__dirname, './client/build')));
137
138 // place as first middleware
139 app.use(express.json());
140 app.use(helmet());
141 app.use(cors());
142 app.use(xss());
143
144
145 - serve index.html for all routes (apart from API)
146 - front-end routes pick's it up from there
147
148 ````js
149 app.use('/api/v1/auth', authRouter);
150 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
151
152 // serve index.html
153 app.get('*', (req, res) => {
154   res.sendFile(path.resolve(__dirname, './client/build', 'index.html'))
155 });
156
157 app.use(notFoundMiddleware);
158 app.use(errorHandlerMiddleware);
159
160
161 - navigate to localhost:5000
162 - 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')));**

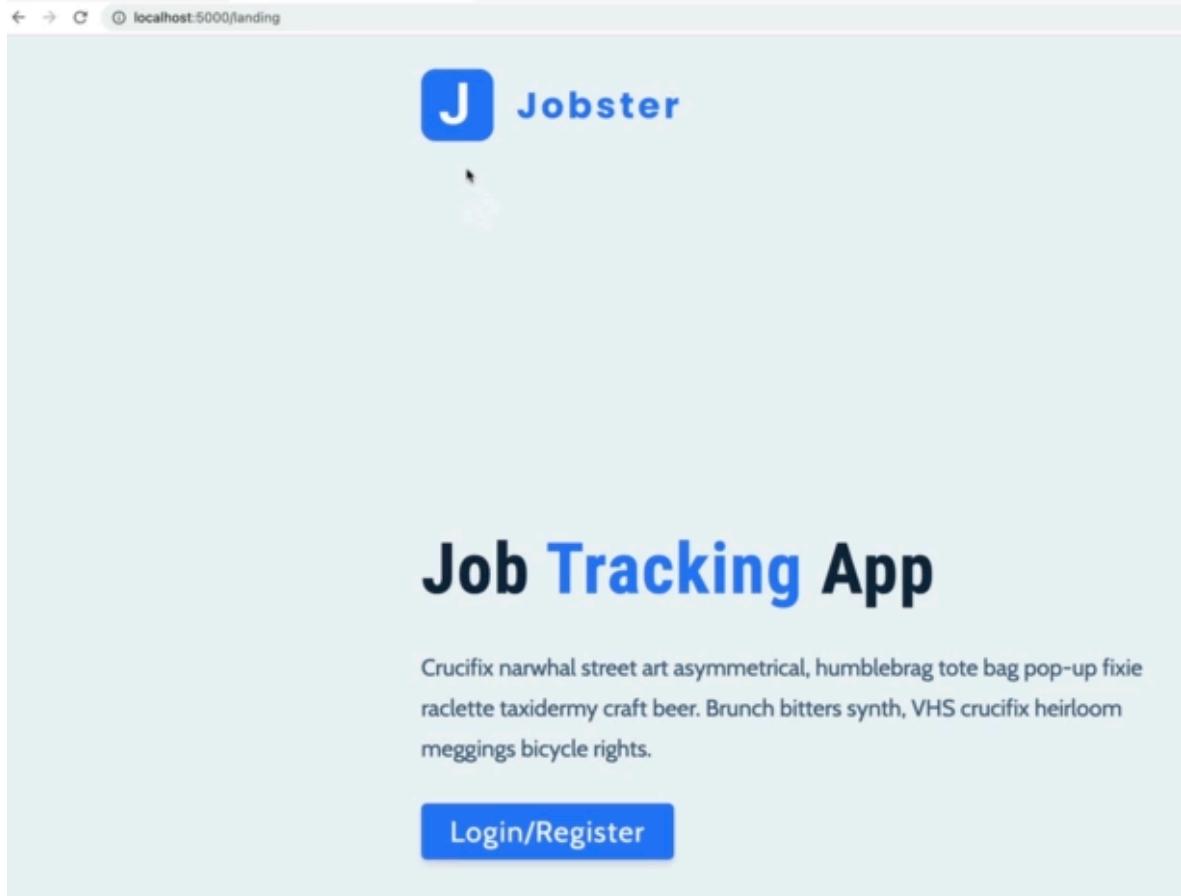
```
06.5-jobster-api > starter > JS app.js > ...
15 const authRouter = require('./routes/auth');
16 const jobsRouter = require('./routes/jobs');
17 // error handler
18 const notFoundMiddleware = require('./middleware/not-found');
19 const errorHandlerMiddleware = require('./middleware/error-handler');
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

The screenshot shows a browser window with two tabs: "Data | Cloud: MongoDB Cloud" and "Jobster Development". The main content area displays the "Job Tracking App" landing page with a blue header, a logo, and a "Login/Register" button. Below the header, there is placeholder text: "Crucifix narwhal street art asymmetrical, humblebrag tote bag pop-up fixie raclette taxidermy craft beer. Brunch bitters synth, VHS crucifix heirloom meggings bicycle rights." At the bottom of the page is a "Login/Register" button. The browser's developer tools are open, specifically the "Application" tab under "Storage". The "LocalStorage" section is expanded, showing an entry for "http://localhost:5000". Other sections like "Manifest", "Service Workers", and "Session Storage" are also visible.

## 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();
```

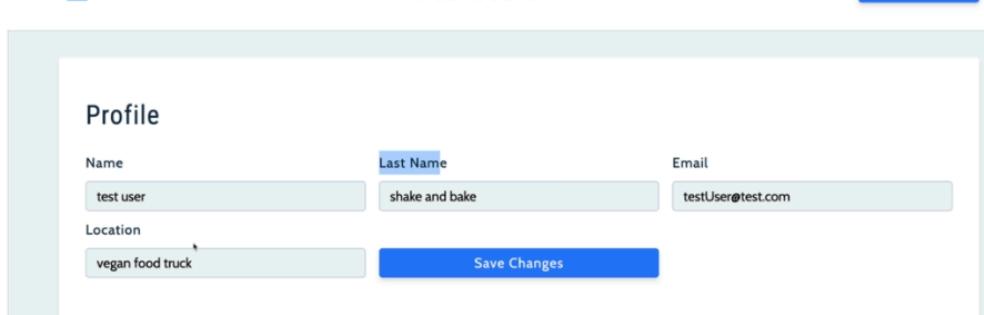
## 6 - Modify User Model

---

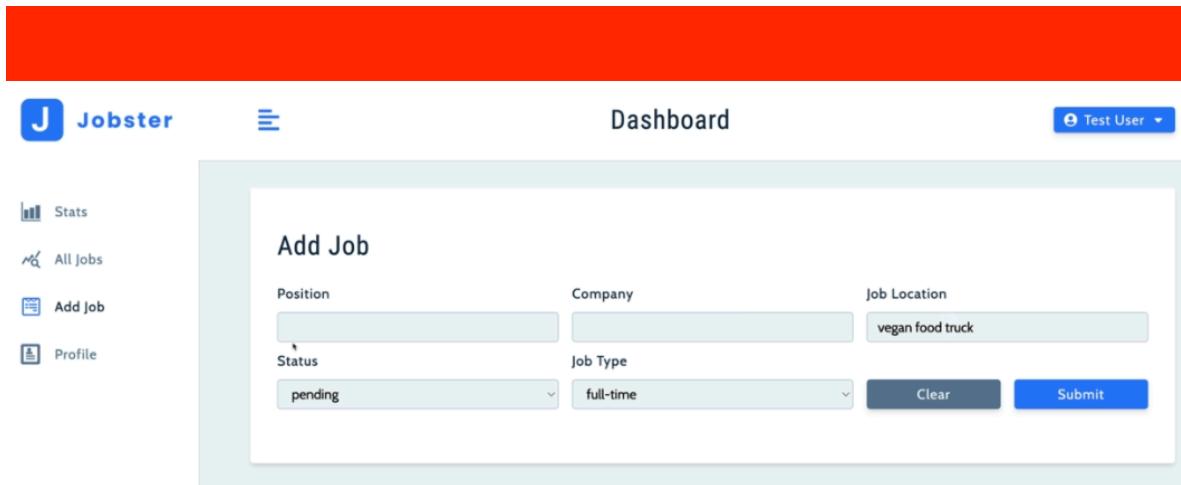
```

163
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**



The screenshot shows the Jobster application's dashboard with a sidebar on the left containing links for Stats, All Jobs, Add Job, and Profile. The main area is titled "Profile" and contains three input fields: "Name" (test user), "Last Name" (shake and bake), and "Email" (testUser@test.com). Below these is a "Location" field containing "vegan food truck". A blue "Save Changes" button is at the bottom right of the form.



**/starter/models/User.js**

---

---

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

```
06.5-jobster-api > starter > models > js User.js > ⑥ UserSchema > ⑦ location > ⑧ default  
17     'Please provide a valid email',  
18   ],  
19   unique: true,  
20 },  
21 password: {  
22   type: String,  
23   required: [true, 'Please provide password'],  
24   minlength: 6,  
25 },  
26 lastName: {  
27   type: String,  
28   trim: true,  
29   maxlength: 20,  
30   default: 'lastName',  
31 },  
32 location: {  
33   type: String,  
34   trim: true,  
35   maxlength: 20,  
36   default: 'my city',  
37 },  
38 });
```

```
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 =
```

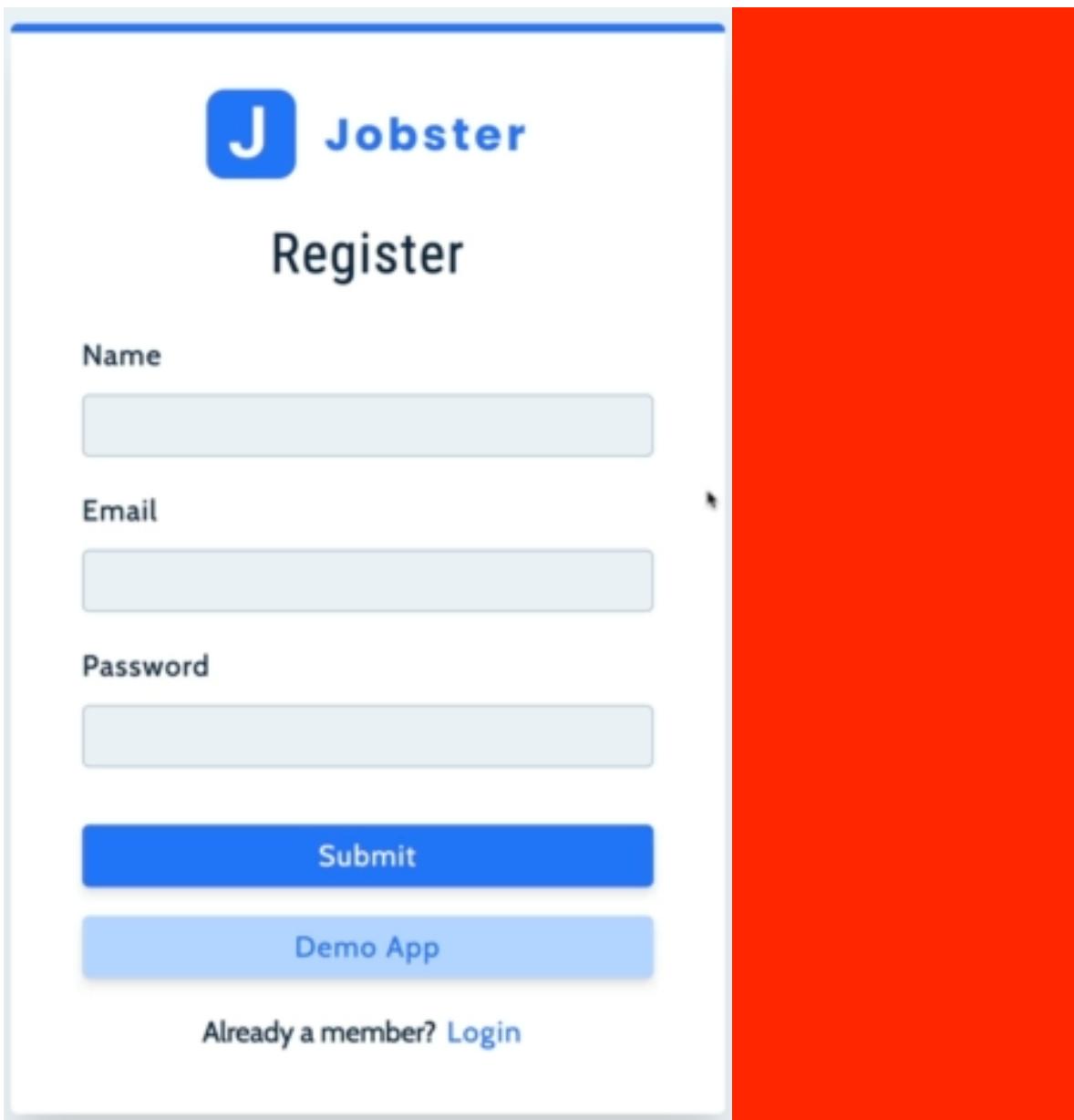
```
function () {
  return jwt.sign(
    { userId: this._id, name: this.name },
    process.env.JWT_SECRET,
    {
      expiresIn:
process.env.JWT_LIFETIME,
    }
  );
};

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

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

## 7 - Modify Register and Login

```
186 ##### Modify Response in Register and Login
187
188 - change the response structure in
189 | register and login controllers (just keep old StatusCodes)
190
191 controllers/auth.js
192
193 ````js
194 res.status(StatusCodes.CREATED).json({
195   user: {
196     email: user.email,
197     lastName: user.lastName,
198     location: user.location,
199     name: user.name,
200     token,
201   },
202 });
203 ````
```



/starter/controllers/auth.js

---

```
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,
  },
});
});
```

```
5-jobster-api > starter > controllers > JS auth.js > [o] register
  2  const { StatusCodes } = require('http-status-codes');
  3  const { BadRequestError, UnauthenticatedError } = require('../errors');
  4
  5  const register = async (req, res) => {
  6    const user = await User.create({ ...req.body });
  7    const token = user.createJWT();
  8    res
  9      .status(StatusCodes.CREATED)
10      .json({
11        user: {
12          email: user.email,
13          lastName: user.lastName,
14          location: user.location,
15          name: user.name,
16          token,
17        },
18      });
19  };
20
21  const login = async (req, res) => {
22    const { email, password } = req.body;
23
24    if (!email || !password) {
25      throw new BadRequestError('Please provide email and password');
```

**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,
  },
});
```

```
06.5-jobster-api > starter > controllers > JS auth.js > [e] login
25   const user = await User.findOne({ email });
26   if (!user) {
27     throw new UnauthenticatedError('Invalid Credentials');
28   }
29   const isPasswordCorrect = await user.comparePassword(password);
30   if (!isPasswordCorrect) {
31     throw new UnauthenticatedError('Invalid Credentials');
32   }
33   // compare password
34   const token = user.createJWT();
35   res.status(StatusCodes.OK).json({
36     user: {
37       email: user.email,
38       lastName: user.lastName,
39       location: user.location,
40       name: user.name,
41       token,
42     },
43   });
44 };
45
46 module.exports = {
47   register,
48   login,
```

**localhost:5000/register**

 Hello There John ×

 **Jobster**

## Register

Name

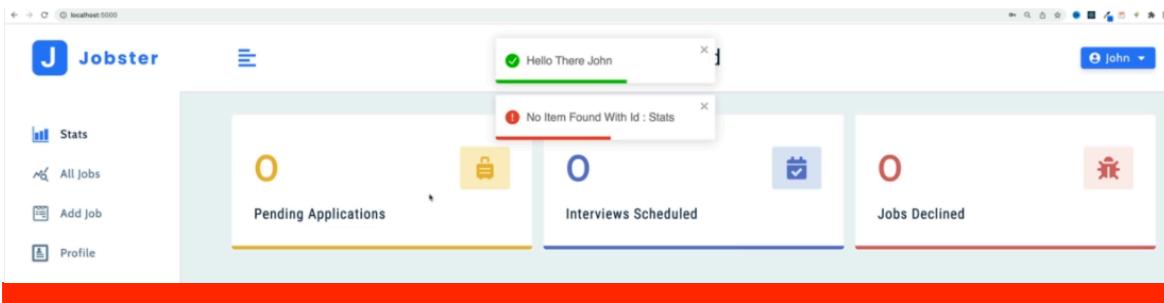
Email

Password

**Submit**

**Demo App**

Already a member? [Login](#)



the server should respond as per front end requirement

`_id: ObjectId("62f42d3125eee237407d48d7")  
lastName: "lastName"  
location: "my city"  
name: "john"  
email: "john@gmail.com"  
password: "$2a$10$tAHVdrmd85iwxhrEAGV0zev7lBtV1Suub2mRt2l4nj.fLMJsS9U6"  
__v: 0`

logout

 Logging Out... ×

---

 **Jobster**

## Login

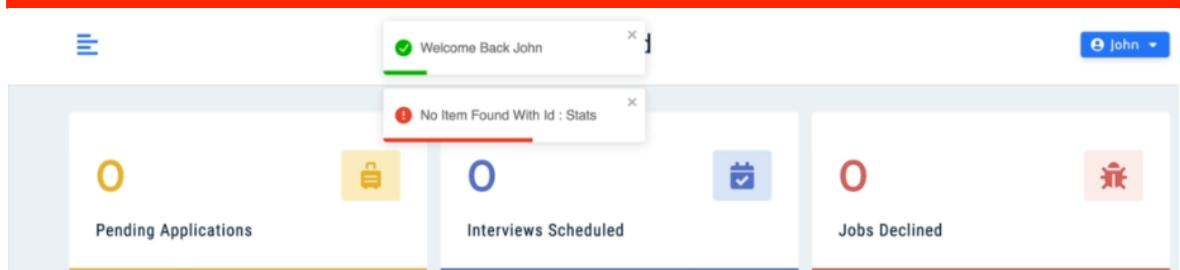
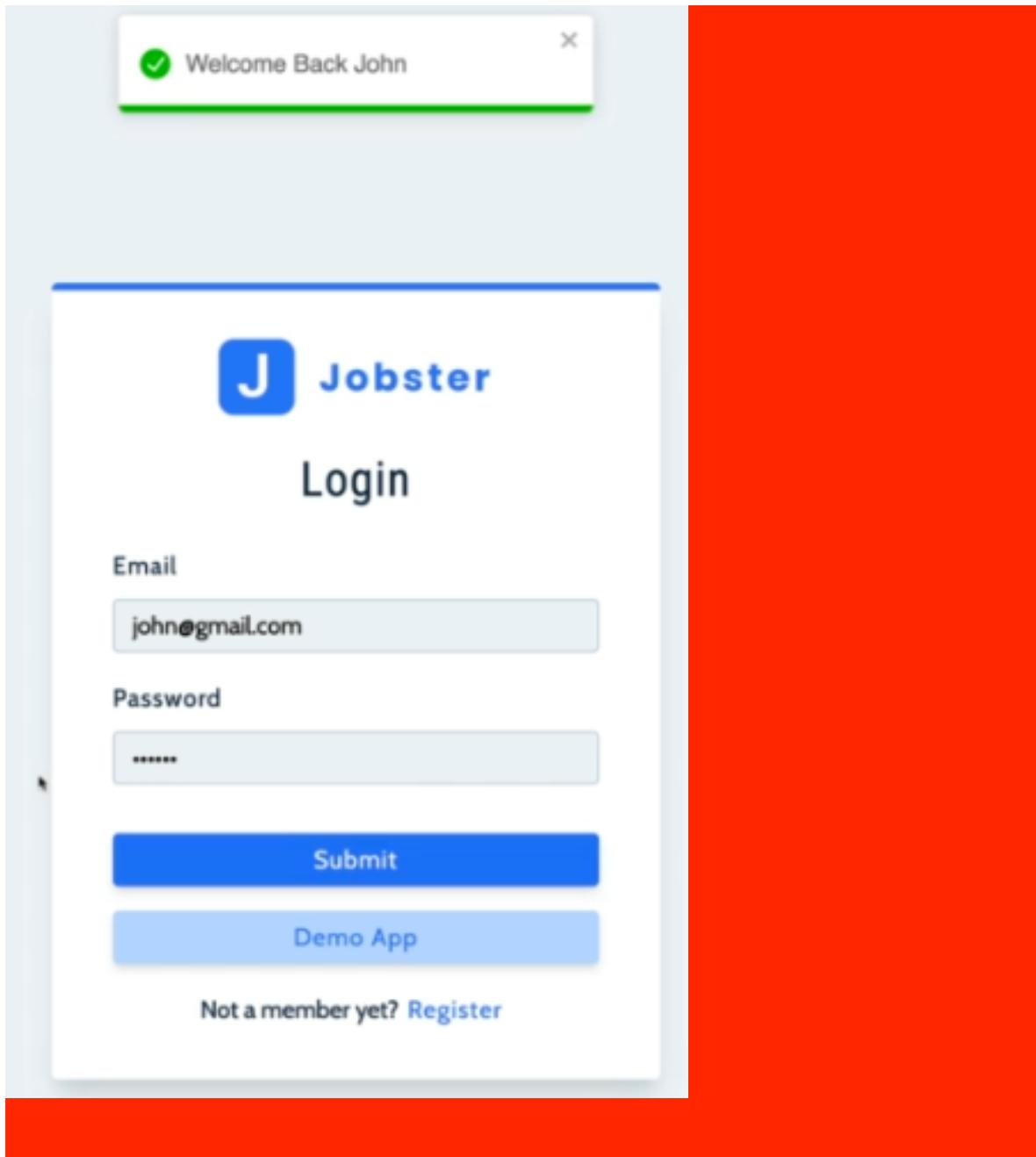
Email

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).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,  
  },  
});  
};
```

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

## 8 - Create Test User

---

```

207 ##### Create Test User
208 - test front-end request
209 - in postman or front-end
210 - make sure email and password are the same (or change the front-end)
211
212 ````js
213 {
214   "name": "demo user",
215   "email": "testUser@test.com",
216   "password": "secret"
217 }
218 ```
219
220
221 - navigate to client/src/pages/Register.js
222 - make sure email and password match your test user
223
224 ````js
225 <button
226   type='button'
227   className='btn btn-block btn-hipster'
228   disabled={isLoading}
229   onClick={() =>
230     dispatch(loginUser({ email: 'testUser@test.com', password: 'secret' }))
231   }
232 ````
```

## when we click Demo App

The screenshot shows a browser window with a login form. The URL is `localhost:5000/register`. The form has two fields: 'Email' and 'Password'. A red error message box is displayed above the fields, containing the text 'Invalid Credentials'. Below the fields are two blue buttons: 'Submit' and 'Demo App'. At the bottom of the form, there is a link 'Not a member yet? [Register](#)'. The browser's developer tools are open at the bottom, specifically the Network tab. The Network tab shows a table of network requests. One row is highlighted in red, corresponding to the failed login attempt. The row details are:

Name	Status	Type	Initiator
login	401	xhr	xhr.js:210

The screenshot shows a browser window with a login form. The form has fields for 'Email' and 'Password', a 'Submit' button, and a 'Demo App' link. An error message 'Invalid Credentials' is displayed above the form. Below the browser is the Chrome DevTools Network tab, which shows a failed request to `/api/v1/auth/login` with a status code of 401 Unauthorized.

Not a member yet? [Register](#)

Name

Request URL: `http://localhost:5000/api/v1/auth/login`

Request Method: `POST`

Status Code: `401 Unauthorized`

Remote Address: `[::1]:5000`

Referrer Policy: `strict-origin-when-cross-origin`

View source

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

Screenshot of the Chrome DevTools Network tab showing a successful login request.

**Network Tab Headers:**

- Elements
- Console
- Recorder
- Performance insights
- Sources
- Network

**Request Details:**

- Name:** login
- Headers:** (not shown)
- Payload:** (selected)
- Preview:** view source
- Response:** (not shown)
- Initiator:** (not shown)
- Timing:** (not shown)

**Request Payload:**

```
{email: "testUser@test.com", password: "secret"}  
email: "testUser@test.com"  
password: "secret"
```

**Statistics:** 1 / 2 requests | 831 B / 528 kB transferred

**Demo App**

Hello There Demo User

demo user

Email

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' }

```
_id: ObjectId("62f52bbfe4396a070a407540")
lastName: "lastName"
location: "my city"
name: "john"
email: "john@gmail.com"
password: "$2a$10$tAHVdrmd85iwxhrEAGV0zev7lBtV1Suub2mRt2il4nj.fLMJs59U6"
__v: 0
```

```
_id: ObjectId("62f52bbfe4396a070a407540")
lastName: "lastName"
location: "my city"
name: "demo user"
email: "testUser@test.com"
password: "$2a$10$qxqH8a0CoZPGfsSdKYXNWe7cuxhr0zMZfWDtecnpS0BxGywi/LaEa"
__v: 0
```

**Jobster**

Welcome Back Demo User

Login

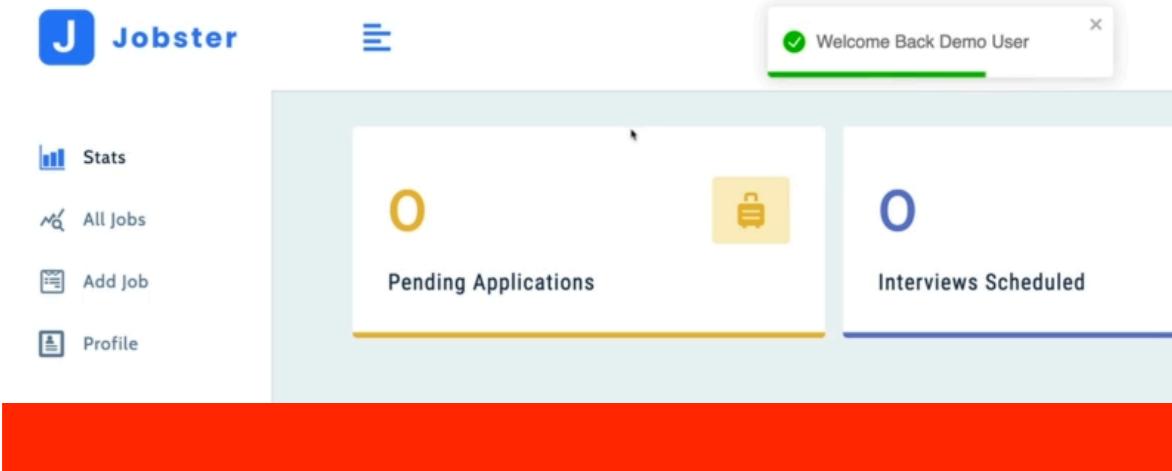
Email

Password

Submit

Demo App

Not a member yet? [Register](#)



```
234 ##### Update User Functionality
235
236 - import authenticateUser middleware
237 - setup updateUser route (protected route)
238
239 routes/auth.js
240
241 ````js
242 const express = require('express');
243 const router = express.Router();
244 const authenticateUser = require('../middleware/authentication');
245 const { register, login, updateUser } = require('../controllers/auth');
246 router.post('/register', register);
247 router.post('/login', login);
248 router.patch('/updateUser', authenticateUser, updateUser);
249 |
250 module.exports = router;
251 ````

252
253 - create updateUser controller
254 - setup export
255
256 controllers/auth.js
257
258 ````js
259 const updateUser = async (req, res) => {
260   console.log(req.user);
261   console.log(req.body);
262 };
263 ````
```

## Profile Page

**Jobster**

**Dashboard**

## Profile

Name: demo user      Last Name: lastName      Email: testUser@test.com

Location: my city

**Save Changes**

localhost:5000/profile

This screenshot shows the 'Profile' page of the Jobster application. The 'Name' field contains 'demo user', the 'Last Name' field contains 'lastName', and the 'Email' field contains 'testUser@test.com'. The 'Location' field contains 'my city'. A blue 'Save Changes' button is visible at the bottom right. The browser address bar shows 'localhost:5000/profile'.

**Jobster**

## Profile

Name:      Last Name: lastName      Email: testUser@test.com

Location: my city

**Save Changes**

localhost:5000/profile

This screenshot shows the 'Profile' page of the Jobster application. The 'Name' field is empty. A red alert box in the top right corner says 'Please Fill Out All Fields'. The 'Last Name' field contains 'lastName', the 'Email' field contains 'testUser@test.com', and the 'Location' field contains 'my city'. A blue 'Save Changes' button is visible at the bottom right. The browser address bar shows 'localhost:5000/profile'.

**Jobster**

**Dashboard**

**Demo User**

## Profile

Name: john      Last Name: lastName      Email: testUser@test.com

Location: my city

**Save Changes**

localhost:5000/profile

This screenshot shows the 'Profile' page of the Jobster application. The 'Name' field contains 'john', the 'Last Name' field contains 'lastName', and the 'Email' field contains 'testUser@test.com'. The 'Location' field contains 'my city'. A blue 'Save Changes' button is visible at the bottom right. Below the page, a network monitoring tool is open, showing a timeline of requests. One request named 'updateUser' is highlighted in red, and a tooltip says 'Route does not exist'. The browser address bar shows 'localhost:5000/profile'.

The screenshot shows the Network tab in Chrome DevTools. A single request named "updateUser" is listed. The Headers section shows:

- Request URL: http://localhost:5000/api/v1/auth/updateUser
- Request Method: PATCH
- Status Code: 404 Not Found
- Remote Address: [::1]:5000
- Referrer Policy: strict-origin-when-cross-origin

The Payload section shows the request body:

```
name: "john", email: "testUser@test.com", lastName: "lastName", location: "my city"  
email: "testUser@test.com"  
lastName: "lastName"  
location: "my city"  
name: "john"
```

**we are getting user from middleware authentication.js file**

```
06.5-jobster-api > starter > middleware > JS authentication.js > (e) auth > (e) payload
 3 const { UnauthenticatedError } = require('../errors')
 4
 5 const auth = async (req, res, next) => {
 6   // check header
 7   const authHeader = req.headers.authorization
 8   if (!authHeader || !authHeader.startsWith('Bearer')) {
 9     throw new UnauthenticatedError('Authentication invalid')
10   }
11   const token = authHeader.split(' ')[1]
12
13   try {
14     const payload = jwt.verify(token, process.env.JWT_SECRET)
15     // attach the user to the job routes
16     req.user = { userId: payload.userId, name: payload.name }
17     next()
18   } catch (error) {
19     throw new UnauthenticatedError('Authentication invalid')
20   }
21 }
22
23 module.exports = auth
```

## /controllers/auth.js

```
06.5-jobster-api > starter > controllers > JS auth.js > (e) <unknown>

  38   lastName: user.lastName,
  39   location: user.location,
  40   name: user.name,
  41   token,
  42 },
  43 );
  44 };
  45
  46 const updateUser = async (req, res) => {
  47   console.log(req.user);
  48   console.log(req.body);
  49 };
  50
  51 module.exports = [
  52   register,
  53   login,
  54   updateUser,
  55 ];
```

## /starter/routes/auth.js

```
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 { register, login, updateUser } = require('../controllers/auth');
  5 router.post('/register', register);
  6 router.post('/login', login);
  7 router.patch('/updateUser', authenticateUser, updateUser);
  8 module.exports = router;
```

**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;
```

Jobster

Dashboard

Profile

Name: john

Last Name: lastName

Email: testUser@test.com

Location: my city

Please Wait...

Network

Elements Console Recorder Performance Insights Sources Network Performance Memory Application Security Lighthouse Redux Components

Disable cache No throttling

Filter: All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party

20000 ms 40000 ms 60000 ms 80000 ms 100000 ms 120000 ms 140000 ms 160000 ms 180000 ms 200000 ms 220000 ms 240000 ms 260000 ms 280000 ms 300000 ms 320000 ms 34

Name	Headers	Payload	Preview	Response	Initiator	Timing
updateUser						
updateUser		Request Payload	view source			

```
name: "john", email: "testUser@test.com", lastName: "lastName", location: "my city"
email: "testUser@test.com"
lastName: "lastName"
location: "my city"
name: "john"
```

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for README.MD, auth.js (active), auth.js, and auth.js. Below the tabs, the file content is displayed:

```
1 const express = require('express');
2 const router = express.Router();
3 const authenticateUser = require('../middleware/authentication');
4 const { register, login, updateUser } = require('../controllers/auth');
5 router.post('/register', register);
6 router.post('/login', login);
7 router.patch('/updateUser', authenticateUser, updateUser);
8 module.exports = router;
9
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL
[nodemon] starting `node app.js`
Server is listening on port 5000...
{ userId: '62f52bbfe4396a070d407540', name: 'demo user' }
{
  name: 'john',
  email: 'testUser@test.com',
  lastName: 'lastName',
  location: 'my city'
}
```

## 10 - Update User (Complete)

---

```
265 - complete updateUser
266
267 controllers/auth.js
268
269 ````js
270 const updateUser = async (req, res) => {
271   const { email, name, lastName, location } = req.body;
272   if (!email || !name || !lastName || !location) {
273     throw new BadRequestError('Please provide all values');
274   }
275   const user = await User.findOne({ _id: req.user.userId });
276
277   user.email = email;
278   user.name = name;
279   user.lastName = lastName;
280   user.location = location;
281
282   await user.save();
283
284   const token = user.createJWT();
285
286   res.status(StatusCodes.OK).json({
287     user: {
288       email: user.email,
289       lastName: user.lastName,
290       location: user.location,
291       name: user.name,
292       token,
293     },
294   });
295 };
296 ````
```

## /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 > [e] login

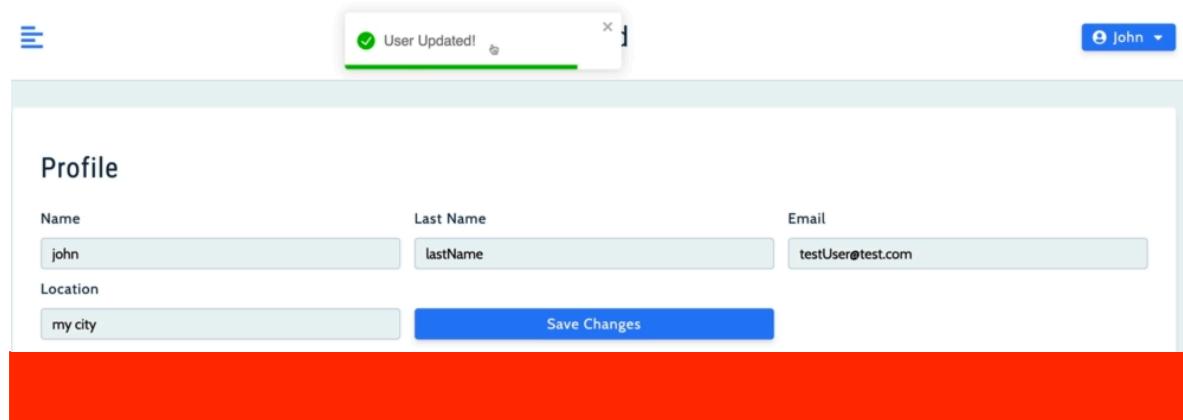
46 const updateUser = async (req, res) => {
47   const { email, name, lastName, location } = req.body;
48   if (!email || !name || !lastName || !location) {
49     throw new BadRequest('Please provide all values');
50   }
51   const user = await User.findOne({ _id: req.user.userId });
52
53   user.email = 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 > [e] 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   };

```



The screenshot shows the Network tab in Chrome DevTools. A single request is listed: a POST to '/api/users/updateUser'. The Headers tab shows a Content-Type of 'application/json'. The Response tab shows a JSON object representing a user:

```
[{"user": {"email": "testUser@test.com", "lastName": "lastName", "location": "my city", "name": "john", "token": "eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoiZXh0dXNlcjkzQGdtYWlsLmNvbSIsImVzZXJuYW1lIjoidGxhYmxlIn0.eyJleHAiOjE2MjUyNjUwOTMsImV4cCI6MTIyNTI2NTk5MywiZnJvbSI6Im15IiwibmFtZSI6ImdpbmEifQ.WBfCgHvDyPQWzqfKuXzrJLcRzg"}]
```

## /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, because of user.save( )**

```
06.5-jobster-api > starter > controllers > JS auth.js > [e] updateUser
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();
61     res.status(StatusCodes.OK).json({
62       user: {
63         email: user.email,
64         lastName: user.lastName,
65         location: user.location,
66         name: user.name,
67         token,
68       },
69     });
70   }.
```

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

Hello There Peter

J Jobster

## Register

Name

Email

Password

Submit

Hello There Peter

Peter

Logout

0 Pending Applications

0 Interviews Scheduled

0 Jobs Declined

**log out and log in one more time**

 Logging Out... X

 Jobster

## Login

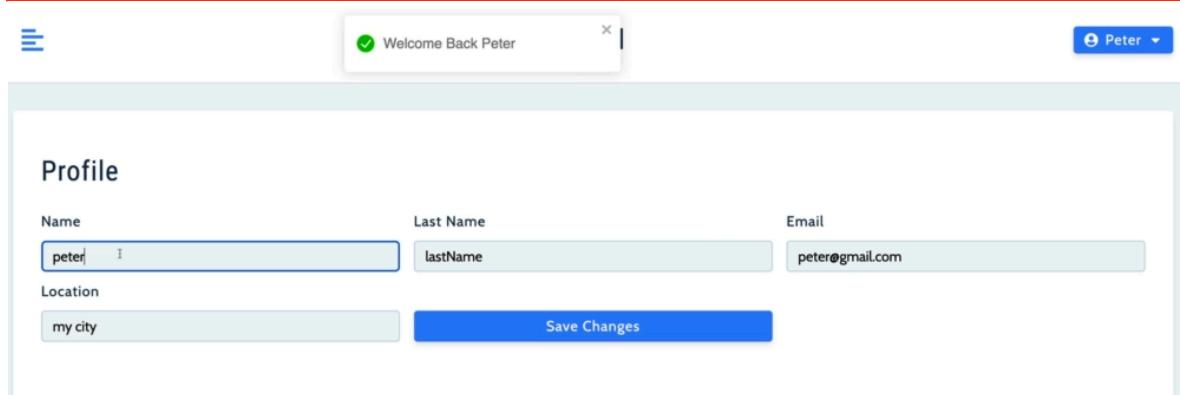
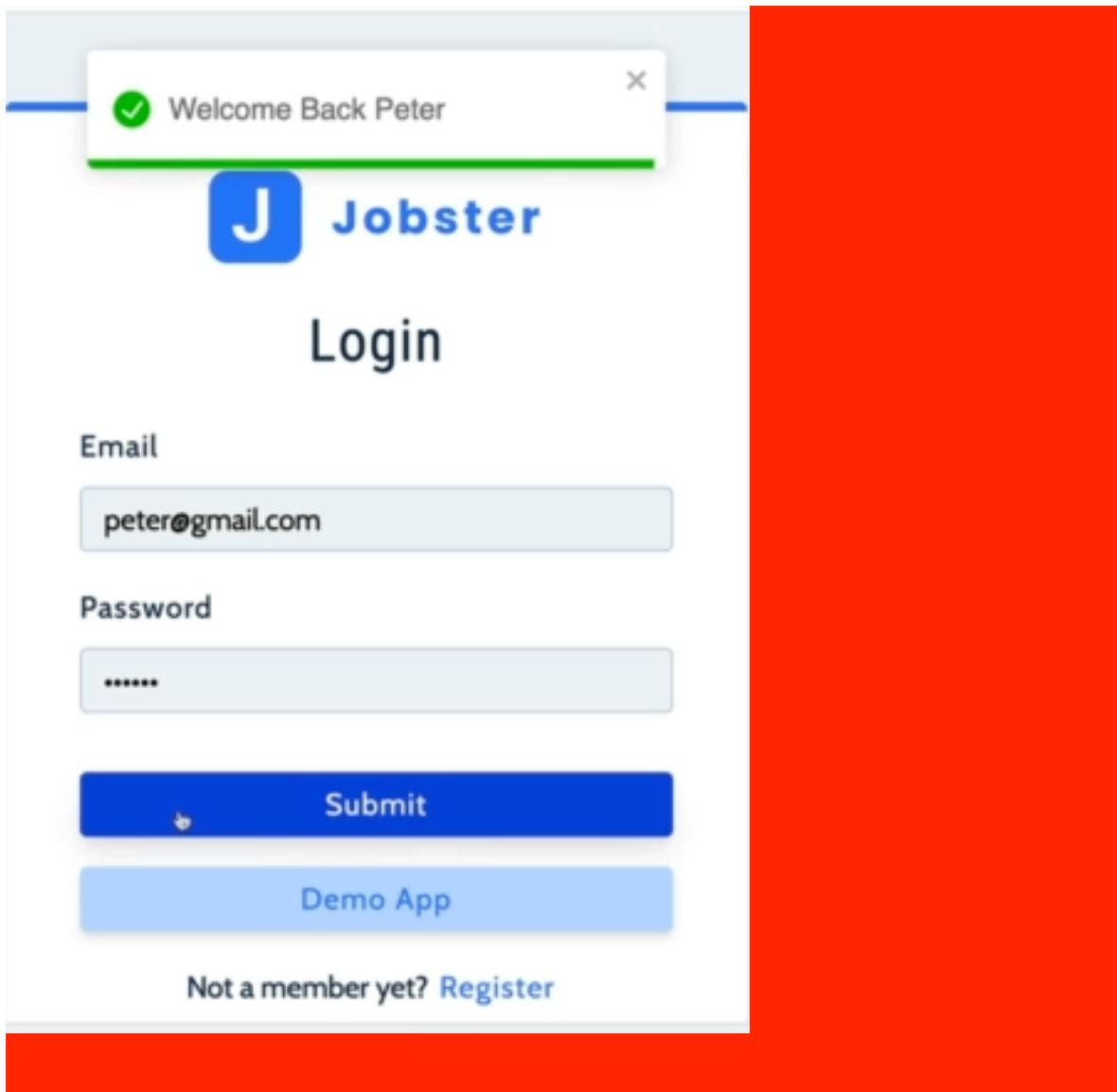
Email

Password

**Submit**

**Demo App**

Not a member yet? [Register](#)



**if we change name to john**

The screenshot shows a web application interface with a light gray header bar. On the left is a blue vertical sidebar icon. In the top right corner, there is a user dropdown menu labeled "John". A green notification box in the top center says "User Updated!". Below it, the word "Profile" is displayed in bold black text. The main content area has three input fields: "Name" (with value "john"), "Last Name" (with placeholder "lastName"), and "Email" (with value "peter@gmail.com"). Below these is a "Location" field containing "my city". A blue "Save Changes" button is positioned to the right of the location field. The background of the main content area is white. At the bottom of the page, there is a large red banner with the word "logout" in white. The URL "localhost:5000/landing" is visible in the browser's address bar. The overall theme includes a blue logo with a stylized letter "J" and the word "Jobster" in blue text.

User Updated!

Profile

Name Last Name Email

john lastName peter@gmail.com

Location my city Save Changes

Logout

localhost:5000/landing

J Jobster

User Updated!

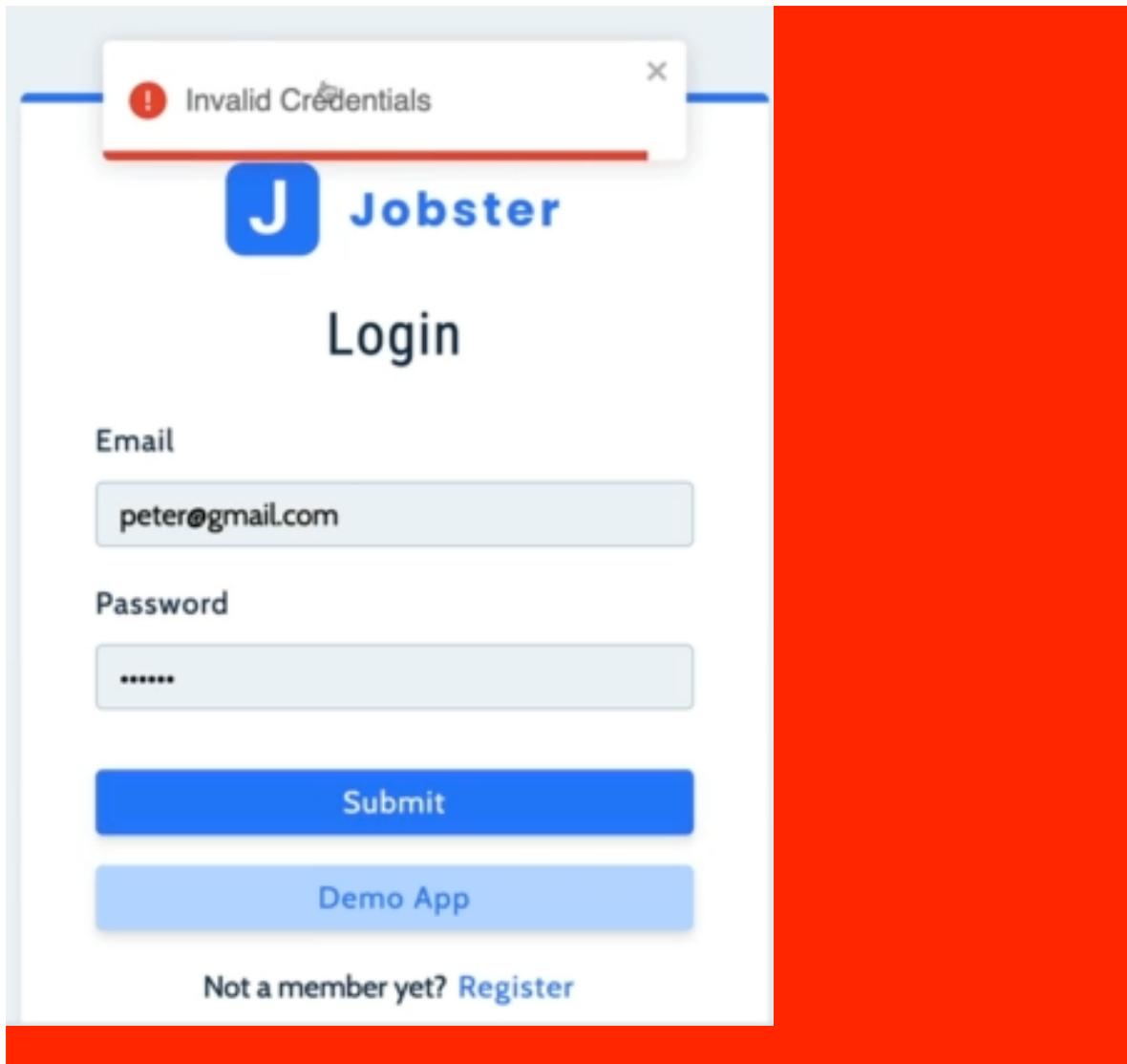
Logging Out...

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

we can not login no more



The image shows a screenshot of a web application's login page. At the top, there is a red alert box with a white border containing a red exclamation mark icon and the text "Invalid Credentials". Below the alert, the logo "Jobster" is displayed, featuring a blue square with a white letter "J" followed by the word "Jobster" in blue. The main title "Login" is centered above the input fields. There are two input fields: "Email" with the value "peter@gmail.com" and "Password" with several dots indicating the password. Below the inputs are two buttons: a blue "Submit" button and a light blue "Demo App" button. At the bottom of the form, there is a link "Not a member yet? Register".

! Invalid Credentials

J Jobster

Login

Email

peter@gmail.com

Password

.....

Submit

Demo App

Not a member yet? [Register](#)

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

```
06.5-jobster-api > starter > models > js User.js > ...
37     default: 'my city',
38   },
39 });
40
41 UserSchema.pre('save', async function () {
42   const salt = await bcrypt.genSalt(10);
43   this.password = await bcrypt.hash(this.password, salt);
44 });
45
46 UserSchema.methods.createJWT = function () {
47   return jwt.sign(
48     { userId: this._id, name: this.name },
49     process.env.JWT_SECRET,
50     {
51       expiresIn: process.env.JWT_LIFETIME,
52     }
53   );
54 }

297
298 - Why New Token?
299
300 #### GOTCHA
301
302 - this.modifiedPaths();
303
304 ``js
305 UserSchema.pre('save', async function () {
306   if (!this.isModified('password')) return;
307   const salt = await bcrypt.genSalt(10);
308   this.password = await bcrypt.hash(this.password, salt);
309 });
310 ``
```

## /models/User.js

---

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

```
06.5-jobster-api > starter > models > JS User.js > UserSchema.pre('save') callback
 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     }
 54   );
 55 }
```

Hello There David

J Jobster

## Register

Name

Email

Password

Submit

☰

Hello There David

David

### Profile

Name	Last Name	Email
david	lastName	david@gmail.com

Location

Save Changes

change name to peter

Dashboard

Profile

Name: peter      Last Name: lastName      Email: david@gmail.com

Location: my city

Please Wait...

File: README.MD    JS User.js    JS auth.js    JS jobs.js

06.5-jobster-api > starter > models > JS 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     }
54   );
55 }
```

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' ]
```

```
i5-jobster-api > starter > models > js User.js > UserSchema.pre('save') callback
 37   default: 'my city',
 38 },
 39 });
 40
 41 UserSchema.pre('save', async function () {
 42   if (!this.isModified('password')) return;
 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     }
 54   );
 55 }
```

```
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: [  
        /^(([^\u00d7)(\n]\u00b2,;:\s@"]+(\u00d7[\u00d7)(\n]\u00b2,;:  
\s@"]+)*|(.+"))@((\u00d7[0-9]{1,3}\u00d7[0-9]  
{1,3}\u00d7[0-9]{1,3}\u00d7[0-9]{1,3}\u00d7)|(([a-zA-  
Z\u00d70-9]+\u00d7)+[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.comparePassword = async function
(canditatePassword) {
  const isMatch = await
bcrypt.compare(canditatePassword,
this.password);
  return isMatch;
};


```

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

remove all users to check

**065-JOBSTER.users**

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

FILTER { field: 'value' }

QUERY RESULTS: 1-3 OF 3

```
_id: ObjectId("62f42d3125eee237407d48d7")
lastName: "smith"
location: "my city"
name: "john "
email: "john@gmail.com"
password: "$2a$10$64ViwjGFhDPomDhkVHb1uSFaqzBY1nx7JbB2gAw9fTGJ12ZuwI2a"
__v: 0
```

```
_id: ObjectId("62f801d0510a7c1ed2312d52")
name: "demo "
email: "testUser@test.com"
password: "$2a$10$jsH3023c9nXNBRET2qsV0rqUcoercl3BMmCFDqsIdSXpgA03fJtC"
lastName: "shake and bake"
location: "vegan food truck"
```

coding addict Access Manager Billing

Testing Deployment Database Data Lake PREVIEW DATA SERVICES Triggers Data API Data Federation

Atlas App Services Charts

+ Create Database

Search Namespaces

01-task-manager 06-JOB-API 065-JOBSTER

jobs users

**065-JOBSTER.users**

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 3 INDEXES TO

Find Indexes Schema Anti-Patterns

FILTER { field: 'value' }

QUERY RESULTS: 0

log out and register one more time

 Logging Out...

**Jobster**

## Login

Email

Password

**Submit**

**Demo App**

Not a member yet? [Register](#)

# Register

Name

Email

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

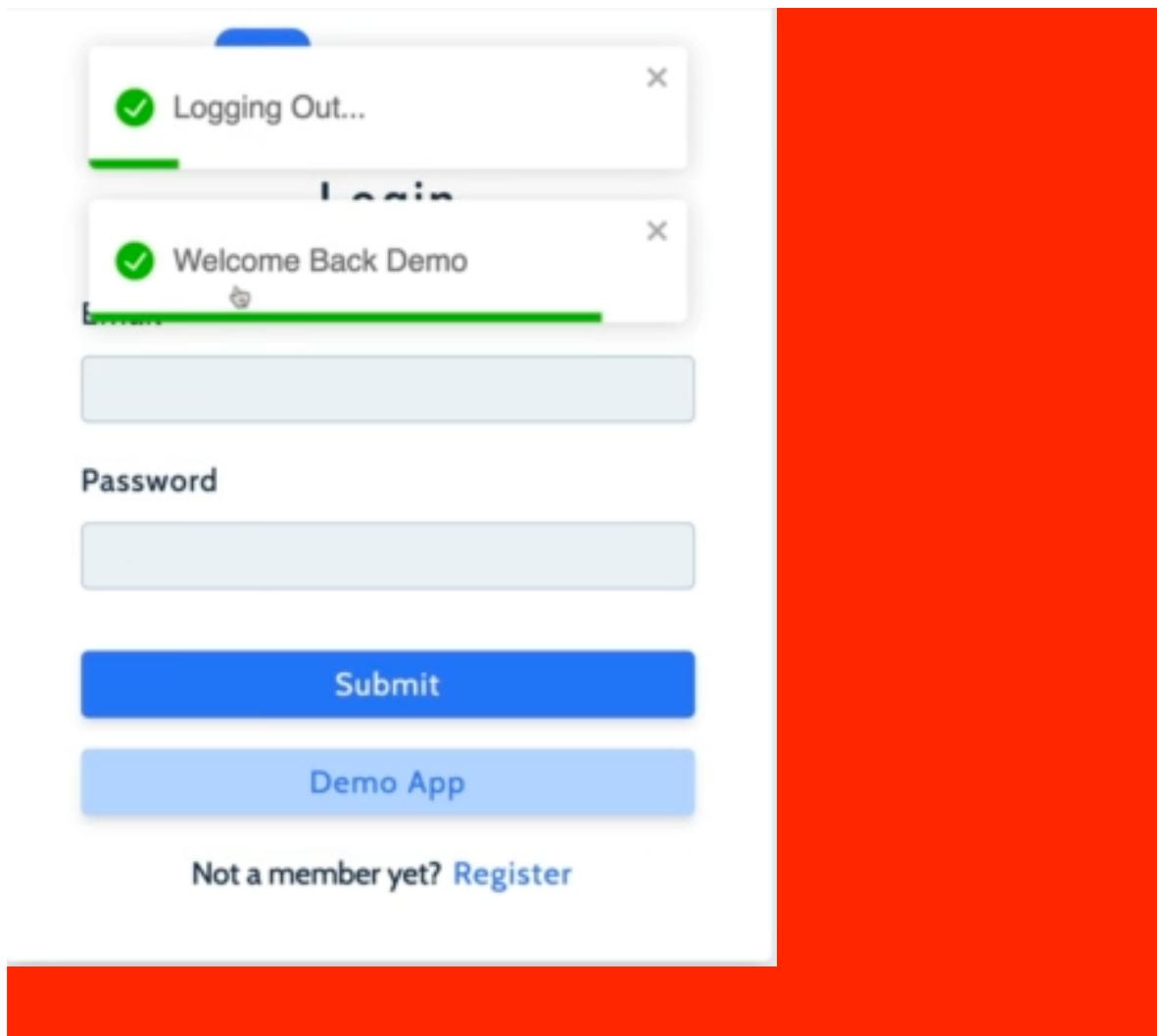
0 Jobs Declined

# change name to Demo

A screenshot of a web application interface. At the top right, there is a user dropdown menu labeled "Demo" with a dropdown arrow, and a "Logout" button. A success message box titled "User Updated!" with a green checkmark icon is displayed. Below the message, the word "Profile" is visible. The main content area contains a form with fields for Name (value: "demo"), Last Name (value: "lastName"), and Email (value: "testUser@test.com"). There is also a Location field containing "my city". A blue "Save Changes" button is located below the form fields.

log out and try Demo App

A screenshot of a login page. At the top, a success message box titled "User Updated!" with a green checkmark icon is shown. Below it, another message box titled "Logging Out..." with a green checkmark icon is displayed. The main content area contains a "Password" input field, a "Submit" button, and a "Demo App" button. At the bottom, there is a link "Not a member yet? Register".



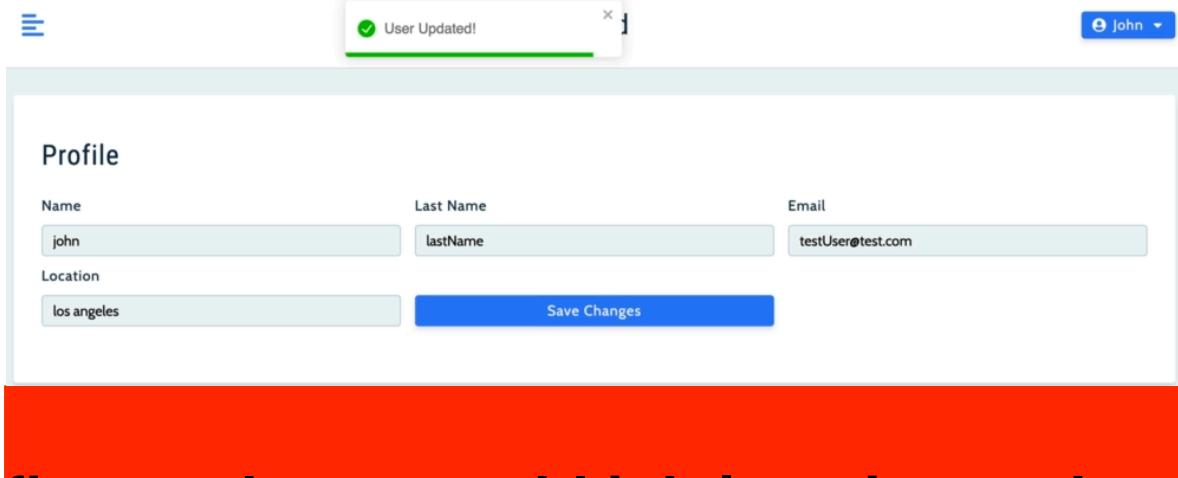
A screenshot of a dashboard for a job search application. It features three main statistics: "Pending Applications" (0), "Interviews Scheduled" (0), and "Jobs Declined" (0). Each statistic is accompanied by a small icon: a briefcase for applications, a calendar for interviews, and a person icon for declined jobs. The dashboard has a light gray background with horizontal lines separating the sections.

## 12 - Complete Jobs CRUD Functionality

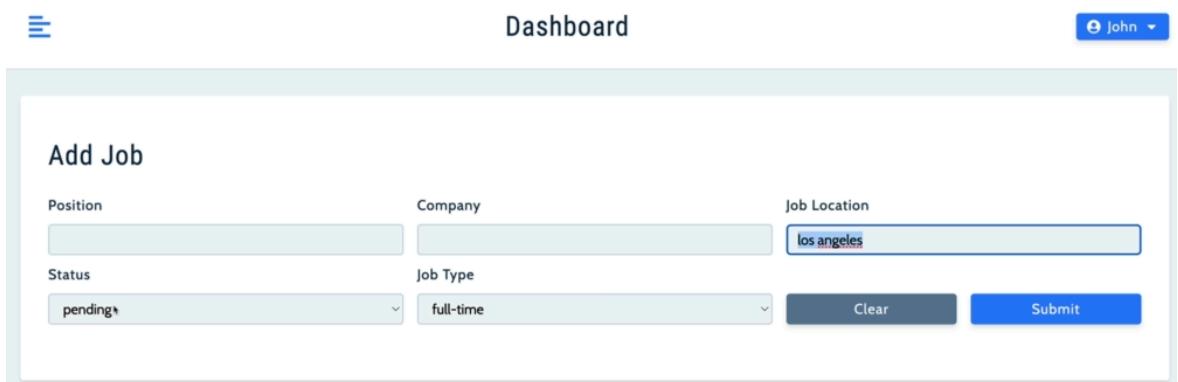
---

---

=  
**change the location and save**



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



```
312 ##### Modify Job Model
313
314 - add following properties
315
316 models/Job.js
317
318 ````js
319 jobType: {
320     type: String,
321     enum: ['full-time', 'part-time', 'remote', 'internship'],
322     default: 'full-time',
323 },
324 jobLocation: {
325     type: String,
326     default: 'my city',
327     required: true,
328 },
329 ````
```

The screenshot shows a user interface for adding a new job. At the top, there's a header with a menu icon, the word "Dashboard", and a user profile for "John". Below the header is a form titled "Add Job". The form has several input fields: "Position" (empty), "Company" (empty), "Job Location" (set to "los angeles"), "Status" (set to "pending"), and "Job Type" (a dropdown menu with four options: "full-time" (selected), "part-time", "remote", and "internship"). There are also "Clear" and "Submit" buttons at the bottom of the form.

/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 > (e) 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	Company	Job Location
		los angeles

Status

pending

Job Type

full-time

Clear

Submit

Dashboard

John

### Add Job

Position: front-end Company: google Job Location: los angeles

Status: pending Job Type: full-time

Job Created

### Add Job

Position: Company: Job Location: los angeles

Status: pending Job Type: full-time

Elements Console Recorder Performance insights Sources Network Per

Preserve log Disable cache No throttling

Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc

50000 ms	100000 ms	150000 ms	200000 ms	250000 ms	300000 ms	350000 ms
----------	-----------	-----------	-----------	-----------	-----------	-----------

Name	Headers	Payload	Preview	Response	Initiator	Timing
<input type="checkbox"/> updateUser	<b>General</b> Request URL: http://localhost:5000/api/v1/jobs Request Method: POST Status Code: 201 Created Remote Address: [::1]:5000 Referrer Policy: no-referrer					
<input type="checkbox"/> updateUser						
<input type="checkbox"/> jobs						

Network tab showing the "jobs" request details:

Name	Headers	Payload	Preview	Response	Initiator	Timing
updateUser						
updateUser						
jobs						

**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"
```

Network tab showing the "jobs" response details:

Name	Headers	Payload	Preview	Response	Initiator	Timing
updateUser						
updateUser						
jobs						

```
1 {"job": {"status": "pending", "jobType": "full-time", "jobLocation": "los angeles", "_id": "62f533a1183fb40c928a58cc", "position": "front-end", "company": "google", "createdBy": "62f52bbfe4396a07e", "updatedBy": "62f52bbfe4396a07e", "createdAt": "2022-08-11T14:33:33.000Z", "updatedAt": "2022-08-11T14:33:33.000Z"}}
```

## All Jobs

Jobster Dashboard - All Jobs

Stats

All Jobs

Add Job

Profile

Search Form

Search:

Status:

Type:

Sort:

Clear Filters

Jobster Dashboard - Job Found

Stats

All Jobs

Add Job

Profile

Job Found

G Front-End Google

Los Angeles

Full-Time

Aug 11th, 2022

Pending

Edit Delete

**Name**

- updateUser
- updateUser
- jobs
- jobs?status=all&jobType=all&s...

**Headers** Payload Preview Response Initiator Timing

**General**

Request URL: http://localhost:5000/api/v1/jobs?status=all&jobType=all&sort=latest&page=1  
Request Method: GET  
Status Code: 200 OK  
Remote Address: [::1]:5000  
Referrer Policy: no-referrer

**Response Headers**

View source

**Name**

- updateUser
- updateUser
- jobs
- jobs?status=all&jobType=all&s...

**Headers** Payload Preview Response Initiator Timing

1 [{"job": {"status": "pending", "jobType": "full-time", "jobLocation": "los angeles", "\_id": "62f533a1183fb40c928a58cc", "position": "front-end", "company": "google", "createdBy": "62f52bbfe4396af", "updatedBy": "62f52bbfe4396af", "createdAt": "2022-08-11T10:00:00.000Z", "updatedAt": "2022-08-11T10:00:00.000Z"}]}

## we will add another job

**Add Job**

Position Company Job Location  
los angeles

Status Job Type  
pending full-time

Clear Submit

**Dashboard**

**Jobs Found**

Job Title	Company	Location	Status	Action
Front-End	Google	Los Angeles	Pending	Edit Delete
Back-End	Netflix	Los Angeles	Pending	Edit Delete

## Edit Back-End job and change to Front-End

Dashboard

Edit Job

Position	Company	Job Location
front-end	netflix	los angeles
Status	Job Type	
pending	full-time	<button>Clear</button> <button>Submit</button>

Job Modified...

Add Job

Position	Company	Job Location
		los angeles
Status	Job Type	
pending	full-time	<button>Clear</button> <button>Submit</button>

Name

	Headers	Payload	Preview	Response	Initiator	Timing
updateUser	General					
updateUser	Request URL: http://localhost:5000/api/v1/jobs/62f533fe183fb40c928a58cf					
jobs	Request Method: PATCH					
jobs?status=all&jobType=all&s...	Status Code: 200 OK					
jobs	Remote Address: [::1]:5000					
jobs?status=all&jobType=all&s...	Referrer Policy: no-referrer					
62f533fe183fb40c928a58cf	Response Headers					

Dashboard

Jobs Found

G	Front-End	Google	N	Front-End	Netflix
Los Angeles	Aug 11th, 2022	Pending	Los Angeles	Aug 11th, 2022	Pending
Full-Time			Full-Time		
<button>Edit</button>	<button>Delete</button>		<button>Edit</button>	<button>Delete</button>	

Press Delete to delete the changed record



Dashboard

[Stats](#)[All Jobs](#)[Add Job](#)[Profile](#)**Job Found**

Front-End

Google

[Los Angeles](#)[Aug 11th, 2022](#)[Full-Time](#)

Pending

[Edit](#)[Delete](#)

Name	Headers	Preview	Response	Initiator	Timing
<a href="#">62f533a1183fb40c928a58cc</a> <a href="#">jobs?status=all&amp;jobType=all&amp;s...</a>	<b>General</b> Request URL: http://localhost:5000/api/v1/jobs/62f533a1183fb40c928a58cc Request Method: DELETE Status Code: 200 OK Remote Address: [::1]:5000 Referrer Policy: no-referrer  <b>Response Headers</b> Connection: keep-alive				

## press All Jobs



Dashboard

[Stats](#)[All Jobs](#)[Add Job](#)[Profile](#)

Search

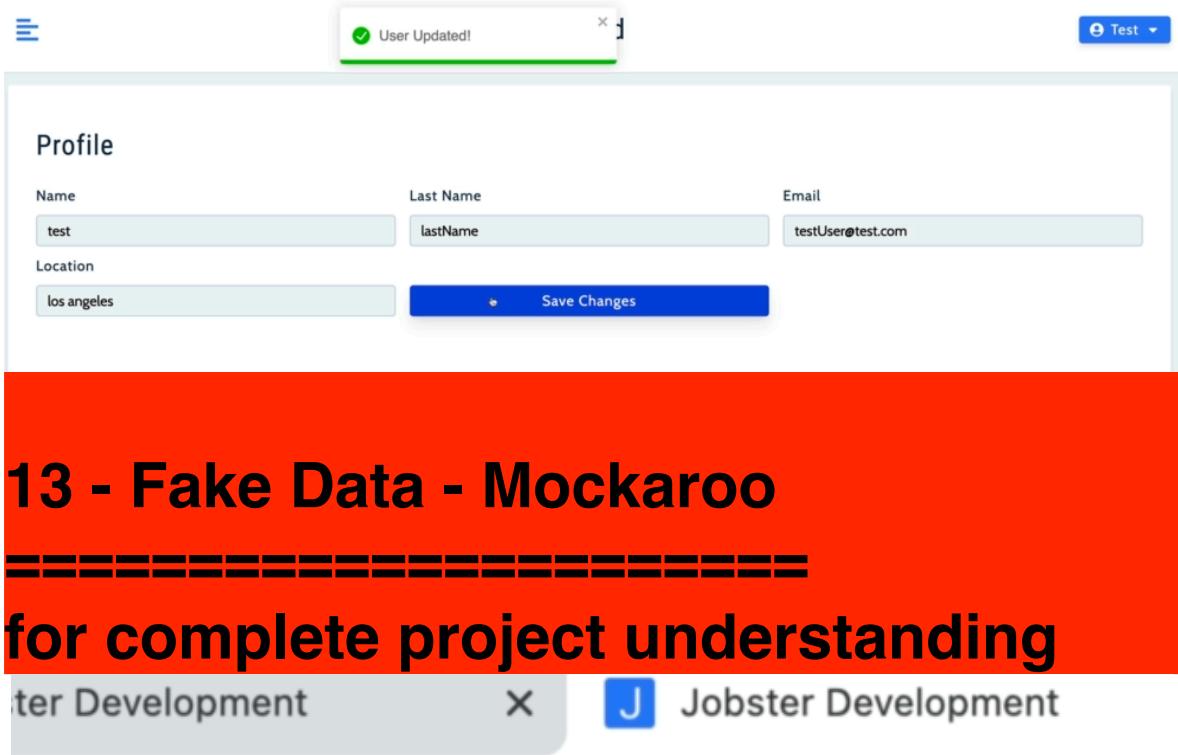
Status

Type

Sort

[Clear Filters](#)

No jobs to display...



## 13 - Fake Data - Mockaroo

---

### for complete project understanding

Jobster Development      X      J Jobster Development

C 🔒 redux-toolkit-jobster.netlify.app/all-jobs

<https://www.mockaroo.com>

```
333 #### Setup Mock Data
334
335 - [Mockaroo] (https://www.mockaroo.com/)
336 - create mock-data.json (root)
337 - provide test user id
```

 mockaroo

SCHEMAS DATASETS MOCK APIs SCENARIOS P

Looking to generate **fake data** based on your **production schema**?

Need some mock data to test your app? Mockaroo lets you generate up to 1,000 rows of realistic test data.

Need more data? Plans start at just \$60/year. Mockaroo is also available as a docker image that you can run on your own infrastructure.

Field Name	Type	Options
id	Row Number	blank: 0 % $\Sigma$ X
first_name	First Name	blank: 0 % $\Sigma$ X
last_name	Last Name	blank: 0 % $\Sigma$ X
email	Email Address	blank: 0 % $\Sigma$ X
gender	Gender	blank: 0 % $\Sigma$ X
ip_address	IP Address v4	blank: 0 % $\Sigma$ X

ADD ANOTHER FIELD

**we have to create same as job model**

```
06.5-jobster-api > starter > models > JS Job.js > [x] JobSchema > [x] jobType
 2
 3 const JobSchema = new mongoose.Schema(
 4   {
 5     company: {
 6       type: String,
 7       required: [true, 'Please provide company name'],
 8       maxlength: 50,
 9     },
10     position: {
11       type: String,
12       required: [true, 'Please provide position'],
13       maxlength: 100,
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: [
```

The screenshot shows the Mockaroo website interface. At the top, there's a navigation bar with links for SCHEMAS, DATASETS, MOCK APIs, SCENARIOS, and PROJECTS. A search bar on the right contains the text "fake". Below the navigation, a section titled "Choose a Type" has a button labeled "All (1)" which is highlighted in green, indicating it's selected. Other buttons include Advanced (0), Basic (0), Car (0), Commerce (0), Construction (0), Crypto (0), Health (0), IT (0), Location (0), Nature (0), and Person (0). To the left, there's a sidebar with a message about mock data testing and a "Field Name" dropdown containing the value "company". The main content area displays a list titled "Fake Company Name" with three items: "Morar Group", "Stark-Glover", and "Sawayn and Sons".

Field Name Type Options

company	Fake Company Name	CSV	blank: 0 %	$\Sigma$	X
---------	-------------------	-----	------------	----------	---

**ADD ANOTHER FIELD**

# Rows: 997 Format: CSV

File Encoding: Unix (LF) ▾ Include:  header  BOM

Cassandra CQL

Firebase

InfluxDB

Custom

Excel

XML

DBUnit XML

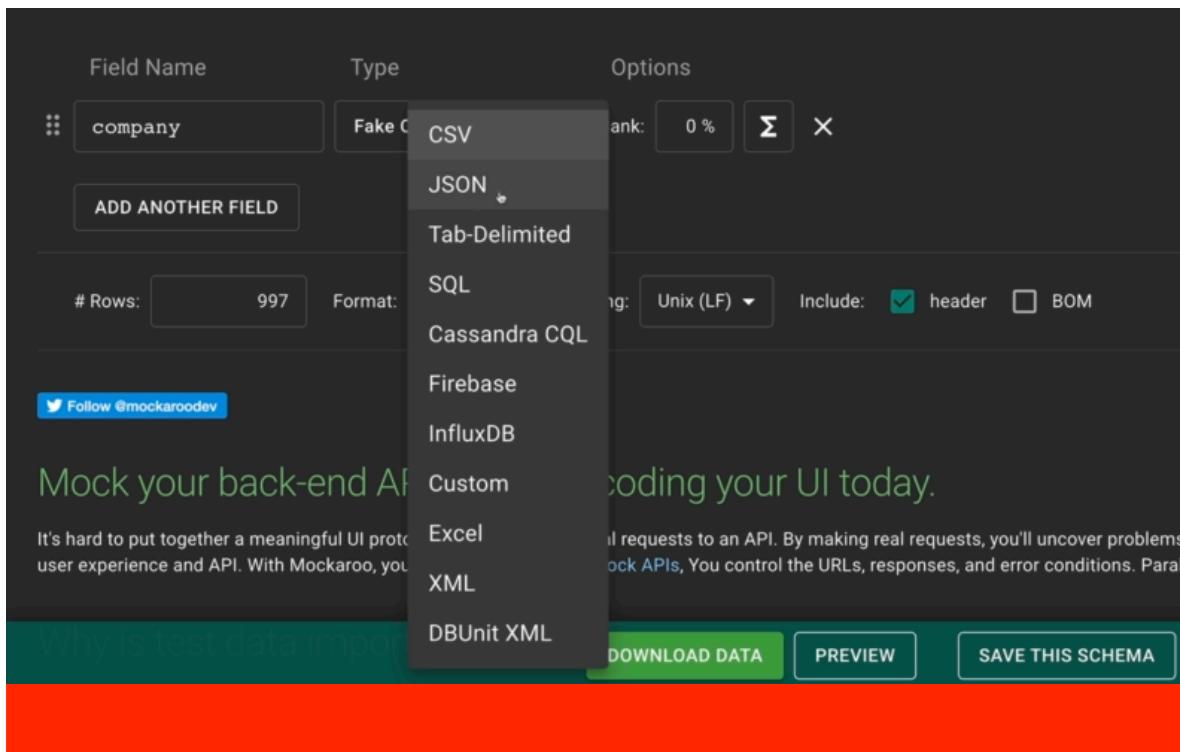
[Follow @mockarodev](#)

Mock your back-end API in minutes without writing code. Start coding your UI today.

It's hard to put together a meaningful UI prototype that matches both your user experience and API. With Mockaroo, you can generate test data for your API in seconds. Just make a few requests to an API. By making real requests, you'll uncover problems with your API design. Mockaroo lets you mock APIs. You control the URLs, responses, and error conditions. Parallel requests let you test multiple endpoints at once.

Why is test data important?

DOWNLOAD DATA PREVIEW SAVE THIS SCHEMA



Field Name Type Options

company	Fake Company Name	JSON	blank: 0 %	$\Sigma$	X
---------	-------------------	------	------------	----------	---

**ADD ANOTHER FIELD**

# Rows: 997 Format: JSON  array  include null values

Hint: Use "." in column names to generate nested json objects, brackets to generate arrays. [More information...](#)

# generating array of objects

## 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 Title

Design Engineer  
General Manager  
Help Desk Technician

Field Name

- company
- position

Field Name	Type	Options
company	Fake Company Name	blank: 0 % $\Sigma$ X
position	Job Title	blank: 0 % $\Sigma$ X

**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$10$tAHVdrmd85iwxhrEAGV0zev7lBtV1Suub2mRt2i14nj.fLMJsS9U6"
__v: 0
```

```
_id: ObjectId("62f52bbfe4396a070a407540")
lastName: "lastName"
location: "my city"
name: "demo user"
email: "testUser@test.com"
password: "$2a$10$qxqH8a0CoZPGfs5dKYXNle7cuxhr0zMZfwDtecnpS0BxGywi/LaEa"
__v: 0
```

Field Name	Type	Options
company	Fake Company Name	blank: 0 % $\Sigma$ X
position	Job Title	blank: 0 % $\Sigma$ X
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

SC

## Choose a Type

All (1)

Advanced (0)

Basic (0)

Car (0)

Comm

### Datetime

07/04/2013

4.7.2013

04-Jul-2013

### Field Name



company



position



status



status



createdBy



createdAt

Need some mock data to test your app? Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats.

Need more data? Plans start at just \$60/year. Mockaroo is also available as a docker image.

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	
createdAt	Datetime	08/11/2021 to 08/11/2022	format: m/d/yyyy

[ADD ANOTHER FIELD](#)

Field Name	Type	Options	
company	Fake Company Name	blank: 0 %	d/m/yyyy
position	Job Title	blank: 0 %	dd/mm/yyyy
status	Custom List	interview, declined, pending	d.m.yyyy
status	Custom List	full-time, part-time, remote, internship	dd.mm.yyyy
createdBy	Custom List	62f52bbfe4396a070a407540	dd-mm-yyyy
createdAt	Datetime	08/11/2021 to 08/11/2022	yyyy/mm/dd

[ADD ANOTHER FIELD](#)

**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 % <span style="border: 1px solid black; padding: 2px;">Σ</span> <span style="color: red;">X</span>
position	Job Title	blank: 0 % <span style="border: 1px solid black; padding: 2px;">Σ</span> <span style="color: red;">X</span>
status	Custom List	interview, declined, pending
jobType	Custom List	full-time, part-time, remote, internship
createdBy	Custom List	62f52bbfe4396a070a407540
createdAt	Datetime	08/11/2021 <span style="border: 1px solid black; padding: 2px;">CALENDAR</span> to 08/11/2022 <span style="border: 1px solid black; padding: 2px;">CALENDAR</span> format: ISO 8601 (UTC) <span style="border: 1px solid black; padding: 2px;">▼</span> blank: 0 % <span style="border: 1px solid black; padding: 2px;">Σ</span> <span style="color: red;">X</span>

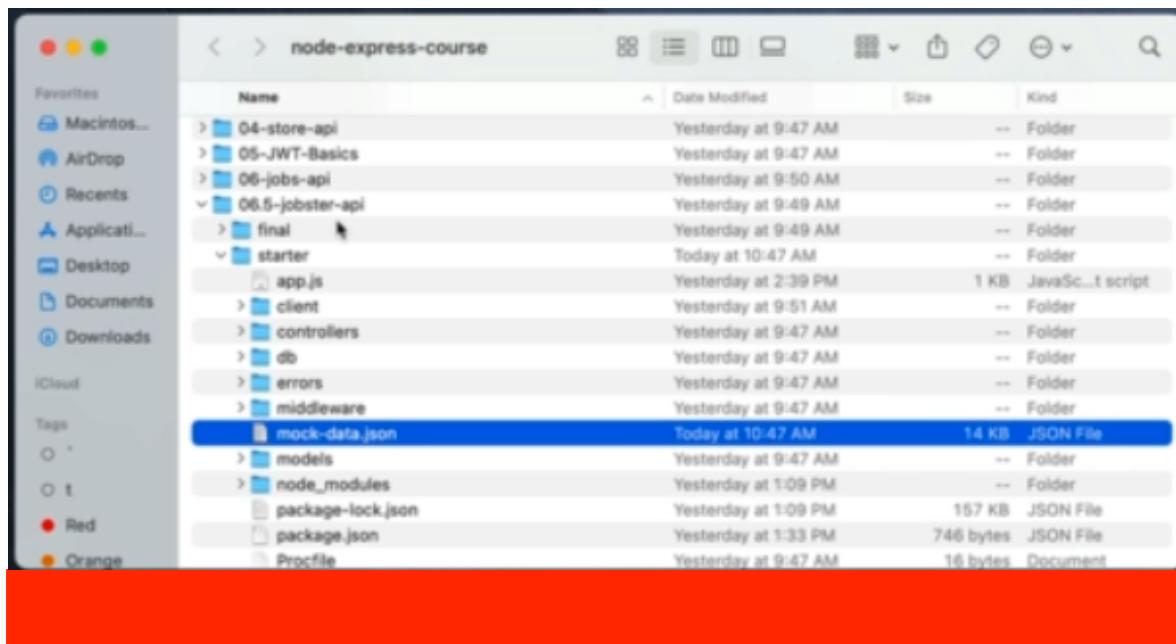
[ADD ANOTHER FIELD](#)

# Rows:  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
    ├ package.json
    └ README.md
    └ controllers
        ├ auth.js
        └ jobs.js
    └ db
    └ errors
    └ middleware
        ├ authentication.js
        ├ error-handler.js
        └ not-found.js
    └ models
        ├ Job.js
        └ User.js
    └ node_modules
    └ routes
        ├ auth.js
        └ jobs.js
    └ .env
    └ .gitignore
    └ app.js
    └ mock-data.json
    └ package-lock.json
    └ package.json
    └ Procfile
    └ README.MD
    └ swagger.yaml
```

2 {"company": "Ha",  
"jobType": "rem",  
"createdAt": "2",  
"id": 2  
3 {"company": "Ro",  
"jobType": "par",  
"createdAt": "2",  
"id": 3  
4 {"company": "Ho",  
"jobType": "par",  
"createdAt": "2",  
"id": 4  
5 {"company": "Ru",  
"jobType": "par",  
"createdAt": "2",  
"id": 5  
6 {"company": "Vo",  
"jobType": "int",  
"createdAt": "2",  
"id": 6  
7 {"company": "Ni",  
"jobType": "int",  
"createdAt": "2",  
"id": 7}

## 14 - Populate Database

---

```
343  populate.js
344
345  ```js
346  require('dotenv').config();
347
348  const mockData = require('./mock-data.json');
349
350  const Job = require('./models/Job');
351  const connectDB = require('./db/connect');
352
353  const start = async () => {
354    try {
355      await connectDB(process.env.MONGO_URI);
356
357      await Job.create(mockData);
358      console.log('Success!!!!');
359      process.exit(0);
360    } catch (error) {
361      console.log(error);
362      process.exit(1);
363    }
364  };
365
366  start();
367  ````
```

## create populate.js file in root

```
JS app.js
{} mock-data.json
-- package-lock.json
-- package.json
JS populate.js
H Procfile
-- README.MD
-- swagger.yaml
```

## stop the server and populate

```
06.5-jobster-api > starter > JS populate.js > ⏪ start
 1  require('dotenv').config();
 2
 3  const mockData = require('./mock-data.json');
 4  const Job = require('./models/Job');
 5  const connectDB = require('./db/connect');
 6
 7  const start = async () => {
 8    try {
 9      await connectDB(process.env.MONGO_URI);
10      await Job.create(mockData);
11      console.log('Success !!!');
12      process.exit(0);
13    } catch (error) {
14      console.log(error);
15      process.exit(1);
16    }

```

PROBLEMS 95 OUTPUT DEBUG CONSOLE TERMINAL

```
smilga@work-iMac starter % node populate
Success !!!
smilga@work-iMac starter % █
```

**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

Overview Real Time Metrics Collections Search Profiler

DATABASES: 11 COLLECTIONS: 27

+ Create Database

Search Namespaces

- 01-task-manager
- 06-JOBSTER-API
- 065-JOBSTER
  - jobs
  - users
- 07-FILE-UPLOAD
- 10-E-COMMERCE-API
- 11-AUTH-WORKFLOW
- 12-EXPRESS-SESSION

## 065-JOBSTER.jobs

STORAGE SIZE: 12KB TOTAL DOCUMENTS: 75 INDEXES TOTAL \$

Find Indexes Schema Anti-Patterns 0

FILTER { field: 'value' }

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId("62f5424b71d21d0f9bf1906a")
status: "interview"
jobType: "full-time"
jobLocation: "my city"
company: "Huel-Gutkowski"
position: "Technical Writer"
```

PROBLEMS 95 OUTPUT DEBUG CONSOLE TERMINAL

```
smilga@work-iMac starter % npm run dev
> 06-jobs-api@1.0.0 dev
> nodemon app.js
```

# localhost:5000/all-jobs

localhost:5000/all-jobs

**Jobster**

**Dashboard**

The dashboard displays four job listings:

- Clinical Specialist** at **Miller LLC**. Status: **Interview**. Date: **Mar 22nd, 2022**. Location: **My City**. Type: **Remote**. Actions: **Edit**, **Delete**.
- Software Test Engineer III** at **Walsh-Marcinski**. Status: **Interview**. Date: **Mar 26th, 2022**. Location: **My City**. Type: **Part-Time**. Actions: **Edit**, **Delete**.
- Executive Secretary** at **Haney, Badenoch, And Schneider**. Status: **Interview**. Date: **Mar 26th, 2022**. Location: **My City**. Type: **Remote**. Actions: **Edit**, **Delete**.
- VP Accounting** at **Jahna-Robles**. Status: **Interview**. Date: **Mar 26th, 2022**. Location: **My City**. Type: **Full-Time**. Actions: **Edit**, **Delete**.

## 15 - Search Functionality

---



Jobster



Stats

All Jobs

Add Job

Profile

## Search Form

Search

sd

Sort

latest

## Jobs Found

		Elements	Console	Recorder	Performance insights	Sources	Network	Performance
		<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/> Preserve log	<input checked="" type="checkbox"/> Disable cache	No throttling	<input type="button"/>
50 ms	100 ms	150 ms	200 ms	250 ms	300 ms	350 ms		
Name							Status	
<input type="checkbox"/> jobs?status=all&jobType=all&sort=latest&page=1&search=sd							200	

change in the Status

**Jobster**

**Dashboard**

The screenshot shows the Jobster dashboard with a sidebar on the left containing links for Stats, All Jobs, Add Job, and Profile. The main area features a search form with a search input containing "sd", a sort dropdown set to "latest", and a status filter dropdown showing "interview" selected. A red bar is at the bottom.

**Search Form**

Search: sd

Sort: latest

Status: interview

Clear Filters

**Jobster**

**Dashboard**

The screenshot shows the Jobster dashboard with a sidebar on the left containing links for Stats, All Jobs, Add Job, and Profile. The main area features a search form with a search input containing "sd", a sort dropdown set to "latest", and a status filter dropdown showing "declined" selected. A red bar is at the bottom.

**Search Form**

Search: sd

Sort: latest

Status: declined

**Jobs Found**

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

Network tab in the browser developer tools showing a request to "jobs?status=declined&jobType=all&sort=latest&page=1&search=sd" with a status of 200 and type xhr.

```
369 ##### Modify Get All Jobs
370
371 - latest mongoose version change
372
373 controllers/jobs.js
374
375 ````js
376 const getAllJobs = async (req, res) => {
377   const { search, status, jobType, sort } = req.query;
378
379   // protected route
380   const queryObject = {
381     createdBy: req.user.userId,
382   };
383
384   if (search) {
385     queryObject.position = { $regex: search, $options: 'i' };
386   }
387   // add stuff based on condition
388
389   if (status && status !== 'all') {
390     queryObject.status = status;
391   }
392   if (jobType && jobType !== 'all') {
393     queryObject.jobType = jobType;
394   }
395
396   // NO AWAIT
397
398   let result = Job.find(queryObject);
399
400   // chain sort conditions
401
402   if (sort === 'latest') {
403     result = result.sort('-createdAt');
404   }
405   if (sort === 'oldest') {
406     result = result.sort('createdAt');
407   }
408   if (sort === 'a-z') {
409     result = result.sort('position');
410   }
411   if (sort === 'z-a') {
412     result = result.sort('-position');
413   }
```

```
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 numPages = Math.ceil(totalJobs / limit);
428
429 res.status(StatusCodes.OK).json({ jobs, totalJobs, numPages });
430 };
431 ````
```

```
06.5-jobster-api > starter > controllers > JS jobs.js > [edit] getJob
1 const Job = require('../models/Job')
2 const { StatusCodes } = require('http-status-codes')
3 const { BadRequestError, NotFoundError } = require('../errors')
4
5 const getAllJobs = async (req, res) => {
6   console.log(req.query);
7   const jobs = await Job.find({ createdBy: req.user.userId }).sort('createdAt')
8   res.status(StatusCodes.OK).json({ jobs, count: jobs.length })
9 }
10 const getJob = async (req, res) => {
11   const {
12     user: { userId },
13     params: { id: jobId },
14   } = req
15
16   const job = await Job.findOne({
```

 Stats

 All Jobs

 Add Job

 Profile

## Search Form

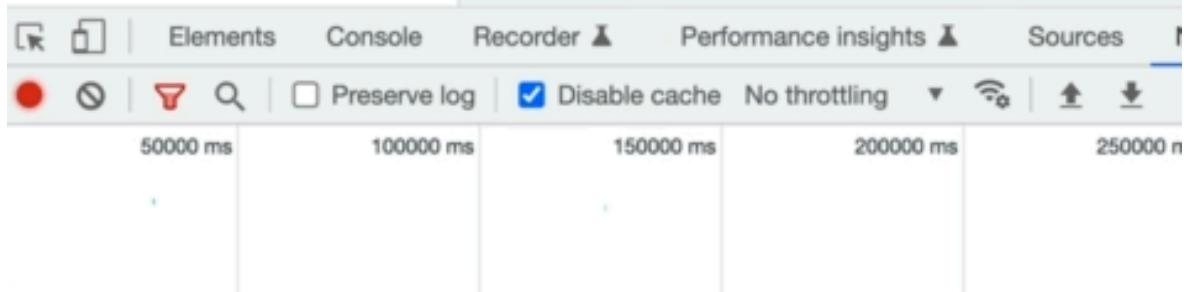
Search

sdd

Sort

latest

## Jobs Found



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

```
1 const Job = require('../models/Job');
2 const { StatusCodes } = require('http-status-codes');
3 const { BadRequestError, NotFoundError } = require('../errors');
4
5 const getAllJobs = async (req, res) => {
6   console.log(req.query);
7   const jobs = await Job.find({ createdBy: req.user.userId }).sort('createdAt');
8   res.status(StatusCodes.OK).json({ jobs, count: jobs.length });
9 }
10 const getJob = async (req, res) => {
11   const {
12     user: { userId },
13     params: { id: jobId },
14   } = req;
15
16   const job = await Job.findOne({
17     _id: jobId,
18     createdBy: userId,
19   });
20
21   if (!job) {
22     throw new NotFoundError(`No job found with ${jobId}`);
23   }
24
25   res.status(StatusCodes.OK).json(job);
26 }
27
28 const search = {
29   status: 'interview',
30   jobType: 'all',
31   sort: 'latest',
32   page: '1',
33   search: 'sdd'
34 }
```

**if search is empty**

 Stats

 All Jobs

 Add Job

 Profile

## Search Form

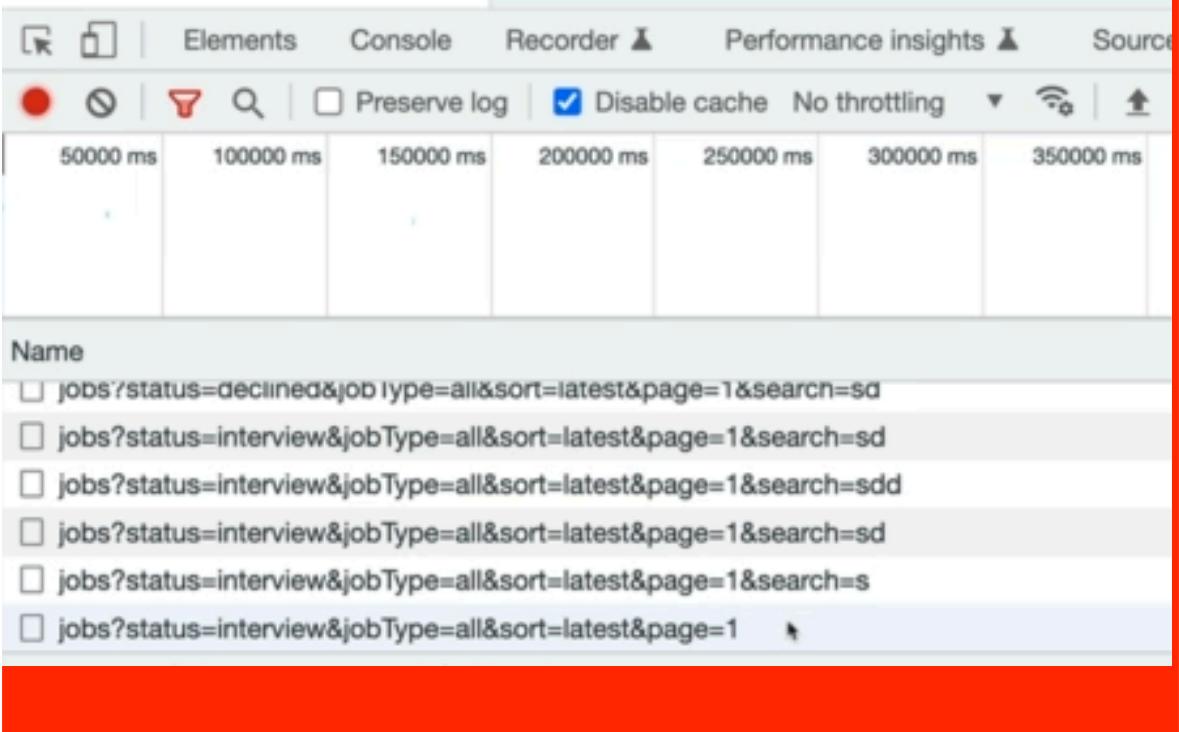
Search

|

Sort

latest

## Jobs Found



The screenshot shows a browser developer tools interface with the Network tab selected. The timeline at the top indicates a total duration of 350,000 ms. Below the timeline, a list of network requests is shown, each with a checkbox and a URL starting with "jobs?status=declined&jobType=all&sort=latest&page=1&search=sd". The last item in the list is partially cut off.

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

```
06.5-jobster-api > starter > controllers > JS jobs.js > [edit] getAllJobs
55
56  const page = Number(req.query.page) || 1;
57  const limit = Number(req.query.limit) || 10;
58  const skip = (page - 1) * limit;
59
60  result = result.skip(skip).limit(limit);
61
62  const jobs = await result;
63
64  const totalJobs = await Job.countDocuments(queryObject);
65  const numOfPages = Math.ceil(totalJobs / limit);
66
67  res.status(StatusCodes.OK).json({ jobs, totalJobs, numOfPages });
68};
```

```
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') {
```

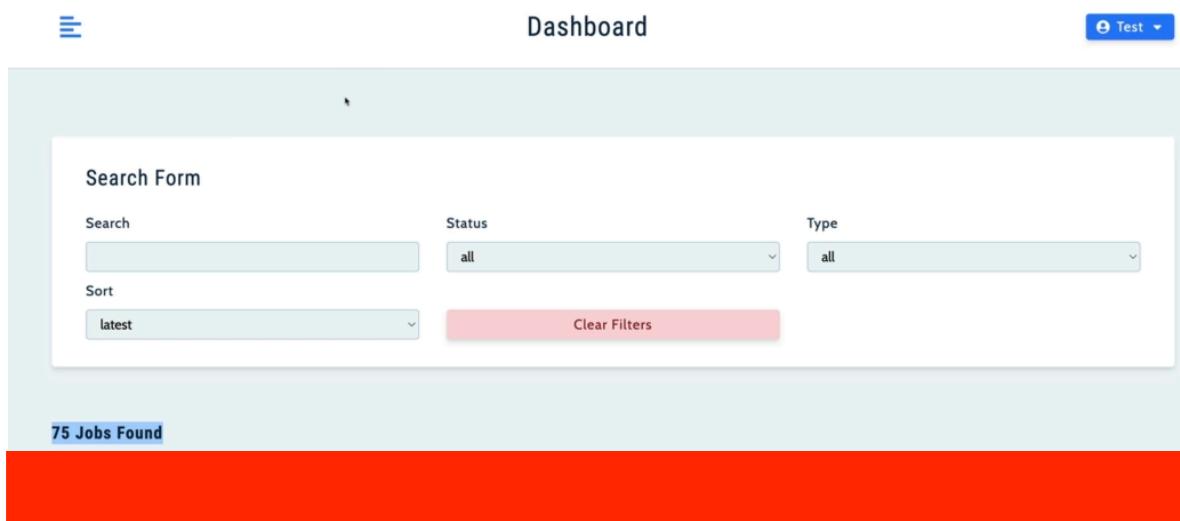
```
queryObject.jobType = jobType;
}
let result = Job.find(queryObject);

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 });  
};
```



The screenshot shows a MongoDB Compass interface. At the top, there's a navigation bar with a menu icon, the word "Dashboard", and a "Test" button. Below the navigation is a search form titled "Search Form". It contains fields for "Search" (with an input field), "Status" (set to "all"), "Type" (set to "all"), and "Sort" (set to "latest"). There's also a "Clear Filters" button. Below the search form, a message "75 Jobs Found" is displayed above a large red horizontal bar.

## Dashboard

Sort: latest | Clear Filters

75 Jobs Found

Job Title	Company	Location	Date	Status
GIS Technical Architect	Hoppe And Sons	My City	Aug 8th, 2022	Declined
Human Resources Manager	Kilback-Morissette	My City	Aug 6th, 2022	Interview

## Dashboard

Test

Job Title	Company	Location	Date	Status
Structural Engineer	Pagac-Davis	My City	Jul 4th, 2022	Interview
Marketing Assistant	Orn LLC	My City	Jul 4th, 2022	Interview

« Prev 1 2 3 4 5 6 7 8 Next »

« Prev 1 2 3 Next »

Elements Console Recorder Performance insights Sources Network Performance Memory Application Security Lighthouse Redux C

Preserve log Disable cache No throttling

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,**

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
jobs?status=all&jobType=all&sort=latest&page=3	200	xhr	xhr.js:210

```
433 ##### Make Test User Read-Only
434
435 middleware/authentication.js
436
437 ````js
438 const payload = jwt.verify(token, process.env.JWT_SECRET);
439 const testUser = payload.userId === '62eff8bcd9af70b4155349d';
440 req.user = { userId: payload.userId, testUser };
441 ````
```

## 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;
```



## Profile

Name	Last Name	Email
<input type="text" value="test user"/>	<input type="text" value="shake and bake"/>	<input type="text" value="testUser@test.com"/>
Location	<input type="text" value="vegan food truck"/> <input type="button" value="Save Changes"/>	

```
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;
```

```
460 - add to auth routes (updateUser)
461
462 ````js
463 const express = require('express');
464 const router = express.Router();
465 const authenticateUser = require('../middleware/authentication');
466 const testUser = require('../middleware/testUser');
467 const { register, login, updateUser } = require('../controllers/auth')
468 ;
469 router.post('/register', register);
470 router.post('/login', login);
471 router.patch('/updateUser', authenticateUser, testUser, updateUser);
472
473 module.exports = router;
474 ````
```

## routes/auth.js

---

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

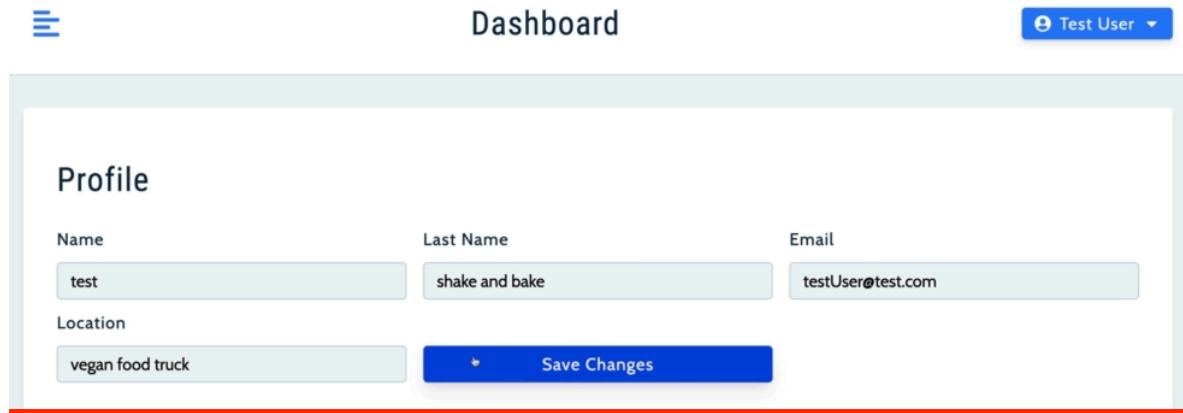
```
const { register, login, updateUser } =
```

```
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**



The screenshot shows a web application interface. At the top, there's a navigation bar with a menu icon, the text "Dashboard", and a user dropdown set to "Test User". Below the navigation is a "Profile" section. Inside the profile section, there are four input fields: "Name" (containing "test"), "Last Name" (containing "shake and bake"), "Email" (containing "testUser@test.com"), and "Location" (containing "vegan food truck"). At the bottom of the profile section is a blue "Save Changes" button.

**we get back alert**

Test User. Read Only!

## Profile

Name	Last Name	Email
test	shake and bake	testUser@test.com
Location	vegan food truck	
<input type="button" value="Save Changes"/>		

+ Create Database

Search Namespaces

- 01-task-manager
- 06-JOB-API
- 065-JOBSTER**
  - jobs
  - 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: 36KB TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 72KB

Find	Indexes	Schema Anti-Patterns	Aggregation	Search
<input type="text" value="FILTER { field: 'value' }"/>				
<pre>_id: ObjectId("62f42d3125eee237407d48d7") lastName: "lastName" location: "my city" name: "john" email: "john@gmail.com" password: "\$2a\$10\$tAHVdrmd85iwxhrEAGV0zev7lBtV1Suub2mRt2iI4nj.fLMJs59U6" __v: 0</pre>				
<pre>_id: ObjectId("62f52bbfe4396a070a407540") lastName: "lastName" location: "los angeles" name: "test" email: "testUser@test.com" password: "\$2a\$10\$pdzq/VF.Nuq.KCkbui10suFywMkj18WCZIdqEIHdOQx0BtyNKAeim" __v: 0</pre>				

User Updated!

## Profile

Name	Last Name	Email
demo user	shake and bake	testUser@test.com
Location	vegan food truck	
<input type="button" value="Save Changes"/>		

/controllers/auth.js

```
06.5-jobster-api > starter > controllers > JS auth.js > [e] 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 BadRequestError('Please provide all values');
51   }
52   const user = await User.findOne({ _id: req.user.userId });
53   if (user) {
54     user.email = email;
55     user.name = name;
56     user.lastName = lastName;
57     user.location = location;
58     await user.save();
59     const token = user.createJWT();
60     res.status(StatusCodes.OK).json({ token });
61   }
}
```

```
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,
  },
});
};
```

The screenshot shows a web application interface. At the top, there's a navigation bar with a menu icon, the text "Dashboard", and a user dropdown labeled "Demo User". Below the navigation is a light blue header bar with the word "Profile". The main content area has a white background and contains a form for editing a user's profile. The form has four input fields: "Name" (value "demo"), "Last Name" (value "shake and bake"), "Email" (value "testUser@test.com"), and "Location" (value "vegan food truck"). A blue "Save Changes" button is positioned at the bottom right of the form.

```
06.5-jobster-api > starter > controllers > JS auth.js > [e] 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();
61   res.status(StatusCode.OK).json({
```

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

The screenshot shows a web application interface. At the top, there's a navigation bar with a 'Demo' dropdown and a 'Logout' button. Below the navigation, the word 'Dashboard' is displayed. The main content area is titled 'Profile'. It contains four input fields: 'Name' (value: 'demo'), 'Last Name' (value: 'shake and bake'), 'Email' (value: 'testUser@test.com'), and 'Location' (value: 'vegan food truck'). A blue 'Save Changes' button is positioned below these fields.



Logging Out...



Jobster

## Login

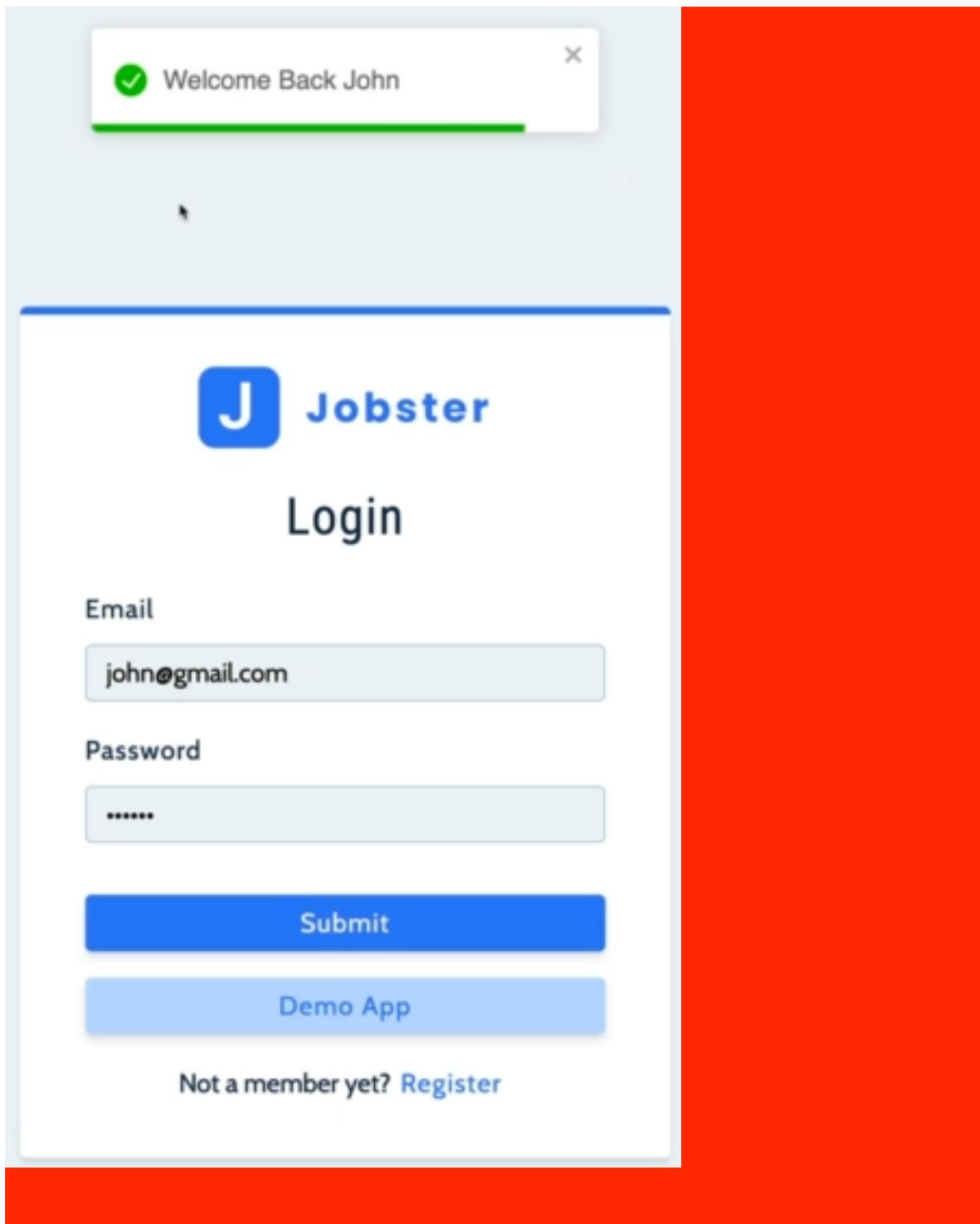
Email

Password

**Submit**

**Demo App**

Not a member yet? [Register](#)



≡

Welcome Back John

No Item Found With Id : Stats

## Profile

Name

Last Name

Email

Location

[Save Changes](#)

john

This is a screenshot of a user's profile page. It shows a navigation menu icon on the left. At the top, there is a green success message "Welcome Back John" and a red error message "No Item Found With Id : Stats". The main section is titled "Profile". It has four input fields: "Name" (containing "john"), "Last Name" (containing "lastName"), "Email" (containing "john@gmail.com"), and "Location" (containing "my city"). Below these fields is a blue "Save Changes" button. In the top right corner, there is a user profile icon with the name "john".

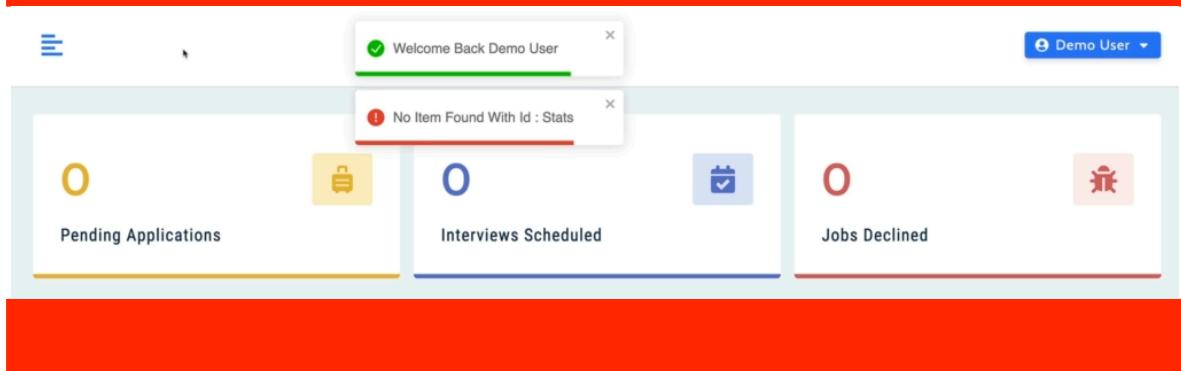
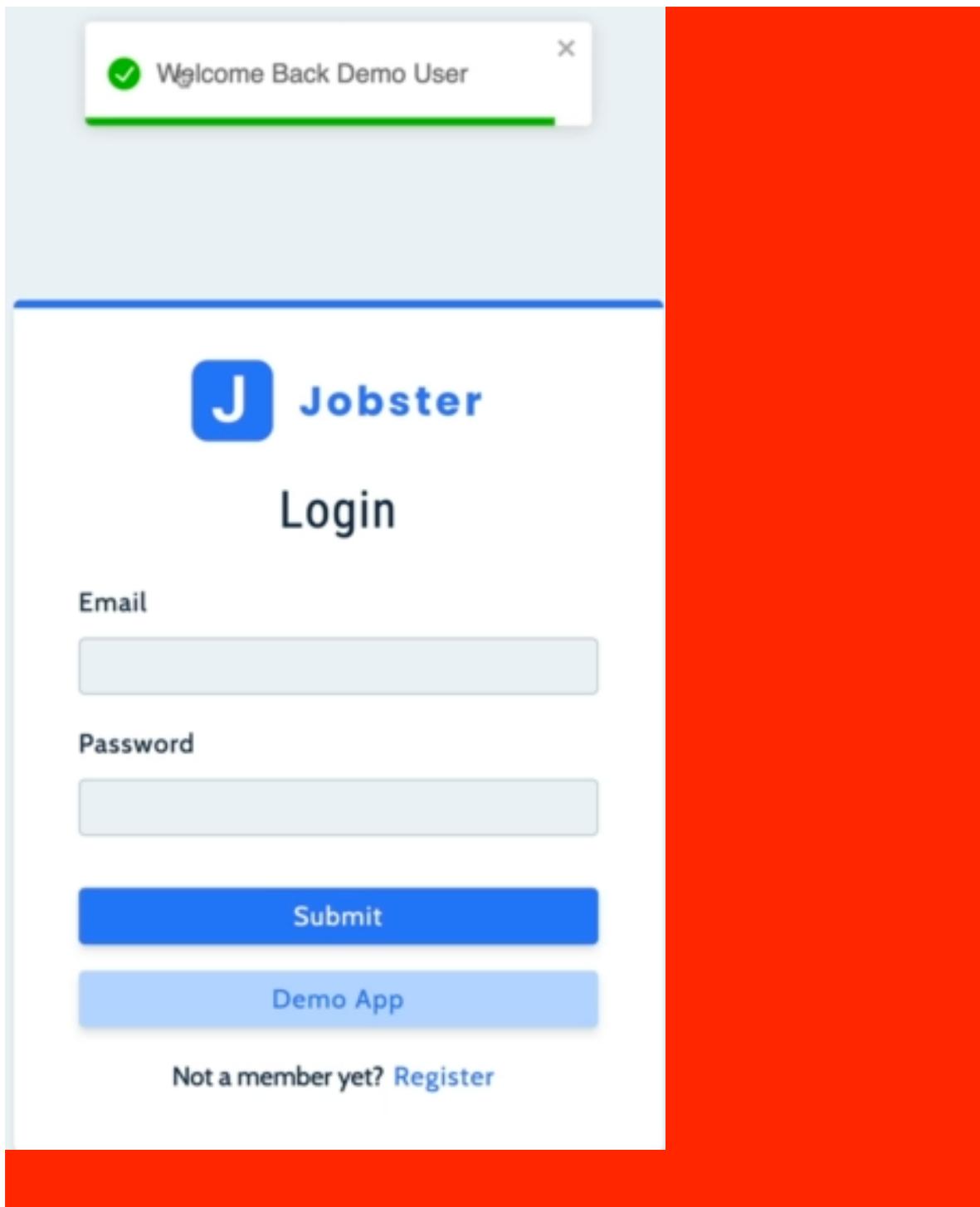
# change last name

User Updated!

Name: john | Last Name: smith | Email: john@gmail.com

Location: my city | Save Changes

```
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 }
{ userId: '62f42d3125eee237407d48d7', testUser: false }
```



## 21 - Restrict CRUD to Test User

---

```
475  - add to job routes (createJob, updateJob, deleteJob)
476
477
478  routes/jobs.js
479
480  ````js
481  const express = require('express');
482
483  const router = express.Router();
484  const {
485    createJob,
486    deleteJob,
487    getAllJobs,
488    updateJob,
489    getJob,
490    showStats,
491  } = require('../controllers/jobs');
492  const testUser = require('../middleware/testUser');
493
494  router.route('/').post(testUser, createJob).get(getAllJobs);
495  router.route('/stats').get(showStats);
496  router
497    .route('/:id')
498    .get(getJob)
499    .delete(testUser, deleteJob)
500    .patch(testUser, updateJob);
501
502  module.exports = router;
503  ````
```

**create testUser.js file under middleware folder**



```
06.5-jobster-api > starter > middleware > JS testUser.js > ...
1 const { BadRequestError } = require('../errors');
2
3 const testUser = (req, res, next) => {
4   if (req.user.testUser) {
5     throw new BadRequestError('Test User. Read Only');
6   }
7   next();
8 };
9
10 module.exports = testUser;
```

## 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**

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

patch(path  
"/updateUs  
...handler  
RequestHan  
any, qs.Pa  
1/5 Record<str

```
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;
```

Dashboard

Demo User

Profile

Name	Last Name	Email
demo user	shake and bake	testUser@test.com

Location

vegan food truck

Save Changes

change the name and send request

Test User. Read Only!

Test User

Profile

Name	Last Name	Email
test	shake and bake	testUser@test.com

Location

vegan food truck

Save Changes

/routes/jobs.js

=====

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

```
06.5-jobster-api > starter > routes > js jobs.js > ...
1 const 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;
```



## Dashboard

Test User ▾

### Add Job

Position	Company	Job Location
adfasdf	asdfasdf	vegan food truck

Status Job Type

pending full-time

Clear Submit



Test User. Read Only! X

## Dashboard

Test User ▾

### Add Job

Position	Company	Job Location
adfasdf	asdfasdf	vegan food truck

Status Job Type

pending full-time

Clear Submit



**it shows only 75 jobs**

**Dashboard**

Test User

Sort: latest | Clear Filters

**75 Jobs Found**

<b>B</b>	Civil Engineer Bechtelar-Bednar	K	Accounting Assistant III Kunze And Sons
👉 Kiamba	📅 Dec 26th, 2021	👉 Kafr Mandā	📅 Dec 22nd, 2021
💻 Internship	Declined	💻 Remote	Interview
<button>Edit</button>	<button>Delete</button>	<button>Edit</button>	<button>Delete</button>
<b>C</b>	Environmental Tech Cremin LLC	K	Actuary Klocko And Sons
👉 Meixian	📅 Dec 9th, 2021	👉 Dianfang	📅 Dec 8th, 2021
💻 Full-Time	Declined	💻 Full-Time	Pending
<button>Edit</button>	<button>Delete</button>	<button>Edit</button>	<button>Delete</button>

**if we try to delete**

Test User. Read Only!

**Search Form**

Search | Status: all | Type: all

Sort: latest | Clear Filters

**same goes to editing job**

Test User. Read Only!

Edit Job

Position	Company	Job Location
Civil Engineerasdfasdf	Bechtelar-Bednar	Kiamba
Status	Job Type	
declined	internship	<input type="button" value="Clear"/> <input type="button" value="Submit"/>

## 22 - API Limiter

---

to avoid spam our application

```
505  ##### API Limiter
506
507  routes/auth.js
508
509  ````js
510  const express = require('express');
511  const router = express.Router();
512  const authenticateUser = require('../middleware/authentication');
513  const testUser = require('../middleware/testUser');
514  const { register, login, updateUser } = require('../controllers/auth')
      ;
515
516  const rateLimiter = require('express-rate-limit');
517  const apiLimiter = rateLimiter({
518    windowMs: 15 * 60 * 1000, // 15 minutes
519    max: 10,
520    message: {
521      msg: 'Too many requests from this IP, please try again after 15
      minutes',
522    },
523  });
524
525  router.post('/register', apiLimiter, register);
526  router.post('/login', apiLimiter, login);
527  router.patch('/updateUser', authenticateUser, testUser, updateUser);
528
529  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);
```

```
06.5-jobster-api > starter > routes > JS auth.js > ...
12 |     msg: 'Too many requests from
13 |     },
14 | );
15 |
16 const { register, login, updateUser
17 router.post('/register', apiLimit1/5 Router
18 router.post('/login', apiLimiter, login);
19 router.patch('/updateUser', authenticateUser, testUser, updateUser);
20 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 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

The image shows a login form for the Jobster application. At the top, there is a green header bar with the text "Welcome Back Demo". Below this is a large blue header with the "Jobster" logo (a blue square with a white "J") and the word "Login". The main form has two input fields: "Email" and "Password", each with a light gray placeholder box. Below the inputs are two blue buttons: "Submit" and "Demo App". At the bottom of the form is a link "Not a member yet? Register".

Jobster

Login

Email

Password

Submit

Demo App

Not a member yet? [Register](#)

logout

The image shows a dashboard for the Jobster application. At the top, there is a green header bar with the text "Welcome Back Demo" and a "Logout" button. Below this is a message box stating "No Item Found With Id : Stats". The dashboard features three main sections: "Pending Applications" (0), "Interviews Scheduled" (0), and "Jobs Declined" (0). Each section has a corresponding icon: a yellow briefcase for applications, a blue calendar for interviews, and a red star for declined jobs.

Welcome Back Demo

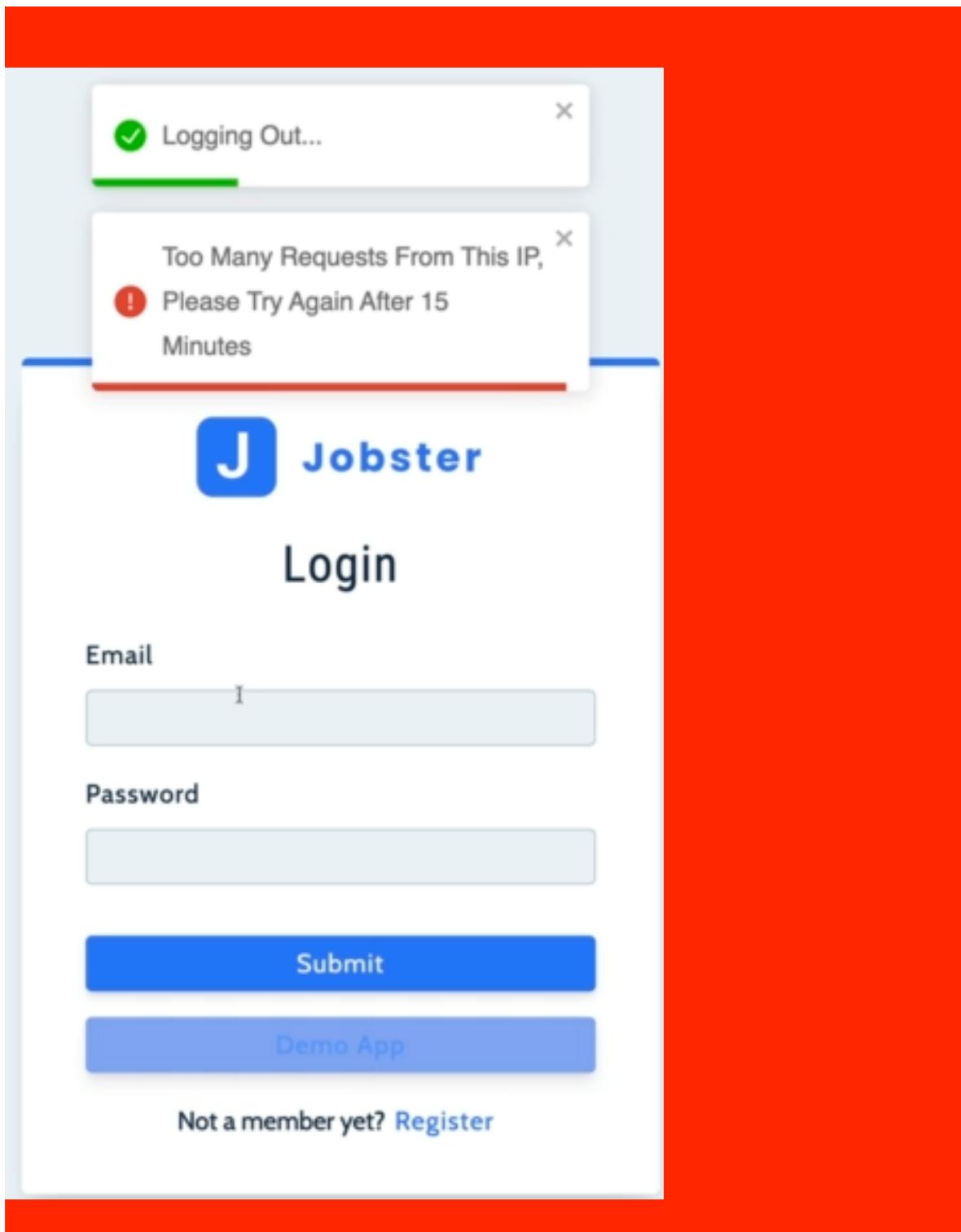
No Item Found With Id : Stats

Pending Applications 0

Interviews Scheduled 0

Jobs Declined 0

Logout



```
532 app.js
533
534 ````js
535 app.set('trust proxy', 1);
536
537 app.use(express.static(path.resolve(__dirname, './client/build')));
538 ````
```

## app.js

---

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

```
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.set('trust proxy', 1);
22
23 app.use(express.static(path.resolve(__dirname, './client/build')));
24 app.use(express.json());
25 app.use(helmet());
26
27 app.use(xss());
28
29 // routes
30 app.use('/api/v1/auth', authRouter);
31 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
```

## 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 aggregation pipeline



All News Videos Images Shopping More

About 397,000 results (0.49 seconds)

<https://www.mongodb.com/docs/manual/core/aggregation-pipeline/> ::

### Aggregation Pipeline — MongoDB Manual

An aggregation pipeline consists of one or more stages that process documents: ... Starting in MongoDB 4.2, you can update documents with an aggregation pipeline ...

[Aggregation Reference](#) · [Aggregation Pipeline Limits](#) · [Aggregation with User...](#)

You visited this page on 8/11/22.

<https://www.mongodb.com/reference/operator/aggregation/> ::

### Aggregation Pipeline Stages — MongoDB Manual

Writes the resulting documents of the aggregation pipeline to a collection. The stage can incorporate (insert new documents, merge documents, replace documents, ...

[mongodb.com/docs/manual/core/aggregation-pipeline/](https://www.mongodb.com/docs/manual/core/aggregation-pipeline/)

MongoDB Products Solutions Resources Company Pricing

**MongoDB Documentation**

Docs Home → Develop Applications → MongoDB Manual

← Back To Develop Applications

**MongoDB Manual**

6.0 (current) ▾

- ▶ Introduction
- ▶ Installation
- MongoDB Shell (mongosh)
- ▶ MongoDB CRUD Operations

An aggregation pipeline consists of one or more [stages](#) that process documents:

- Each stage performs an operation on the input documents. For example, a stage can filter 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 example, to calculate the maximum, and minimum values.

Starting in MongoDB 4.2, you can update documents with an aggregation pipeline [Updates with Aggregation Pipeline](#).



[redux-toolkit-jobster.netlify.app](https://redux-toolkit-jobster.netlify.app)

**Jobster** Dashboard Test User

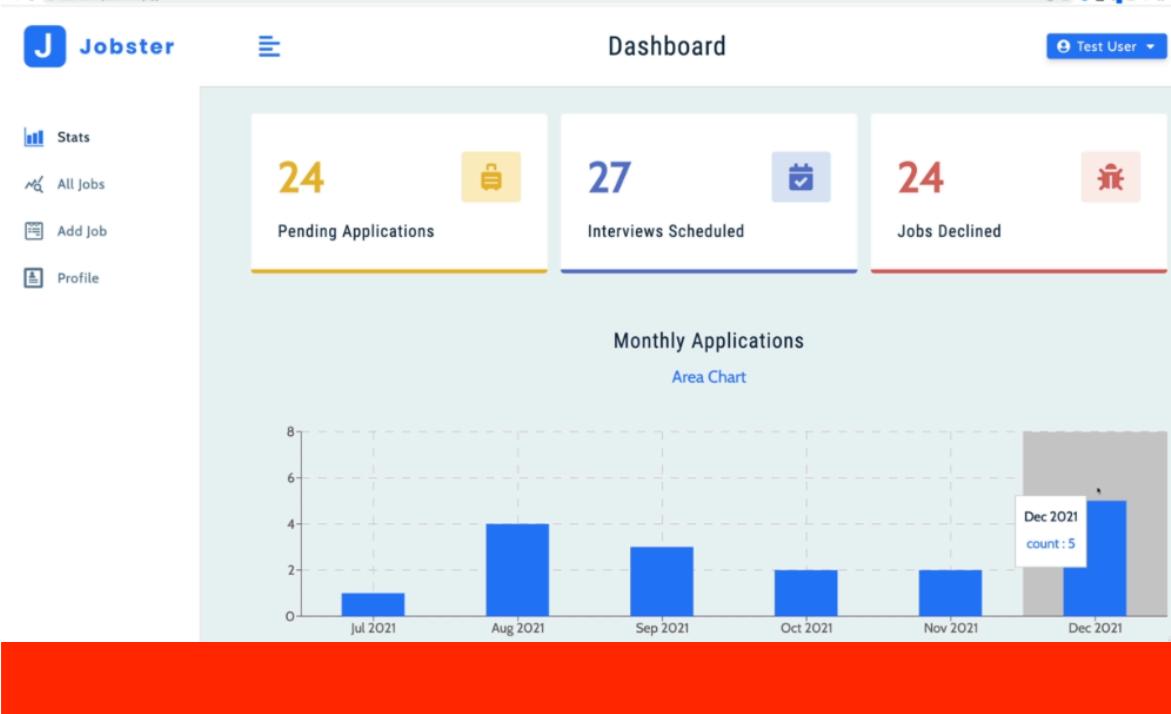
Stats All Jobs Add Job Profile

**Pending Applications** 24

**Interviews Scheduled** 27

**Jobs Declined** 24

**Monthly Applications** Area Chart



Month	Count
Jul 2021	1
Aug 2021	4
Sep 2021	3
Oct 2021	2
Nov 2021	2
Dec 2021	5

## MongoDB Documentation

[← Back To Develop Applications](#)

## MongoDB Manual

6.0 (current)

[Introduction](#)[Installation](#)[MongoDB Shell \(mongosh\)](#)[MongoDB CRUD Operations](#)[Aggregation Operations](#)

## Aggregation Pipeline

[Aggregation Pipeline Optimization](#)[Aggregation Pipeline Limits](#)[Aggregation Pipeline and Sharded Collections](#)[Example with ZIP Code](#)

```
        quantity: 10, date : ISODate( "2022-01-12T05:08:13Z" ) },
        { _id: 6, name: "Vegan", size: "small", price: 17,
          quantity: 10, date : ISODate( "2021-01-13T05:08:13Z" ) },
        { _id: 7, name: "Vegan", size: "medium", price: 18,
          quantity: 10, date : ISODate( "2021-01-13T05:10:13Z" ) }
    ] )
```

## Calculate Total Order Quantity

The following aggregation pipeline example contains two [stages](#) and returns the total order quantity of medium size pizzas grouped by pizza name:

```
db.orders.aggregate([
    // Stage 1: Filter pizza order documents by pizza size
    {
        $match: { $size: "medium" }
    },
    // Stage 2: Group remaining documents by pizza name and calculate total quantity
    {
        $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }
    }
])
```

The [\\$match](#) stage:

## 24 - Show Stats Controller

---



Jobster



Stats

All Jobs

Add Job

Profile

0



Pending Applications

Elements Console Recorder Performance insights Sources Network Performance

● ⏹ 🔍 □ Preserve log  Disable cache No throttling WiFi Upload Download

50 ms 100 ms 150 ms 200 ms

Name	Headers	Preview	Response	Initiator	Timing
stats	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				

1 / 10 requests | 800 B / 1.4 MB total

```
540 ##### Setup Stats Route
541
542 controllers/jobs
543
544 ````js
545 const showStats = (req, res) => {
546   res
547     .status(StatusCodes.OK)
548     .json({ defaultStats: {}, monthlyApplications: [] });
549 }
550
551 module.exports = {
552   createJob,
553   deleteJob,
554   getAllJobs,
555   updateJob,
556   getJob,
557   showStats,
558 };
559
560
561 routes/jobs.js
562
563 ````js
564 const express = require('express');
565
566 const router = express.Router();
567 const {
568   createJob,
569   deleteJob,
570   getAllJobs,
571   updateJob,
572   getJob,
573   showStats,
574 } = require('../controllers/jobs');
575
576 router.route('/').post(createJob).get(getAllJobs);
577 router.route('/stats').get(showStats);
578 router.route('/:id').get(getJob).delete(deleteJob).patch(updateJob);
579
580 module.exports = router;
581 ````
```

## controllers/jobs.js

---

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

```
};
```

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

```
06.5-jobster-api > starter > controllers > JS jobs.js > [?] <unknown>
106  };
107
108 const showStats = async (req, res) => {
109   res
110     .status(StatusCodes.OK)
111     .json({ defaultStats: {}, monthlyApplications: [] });
112 }
113
114 module.exports = [
115   createJob,
116   deleteJob,
117   getAllJobs,
118   updateJob,
119   getJob,
120   showStats,
121 ];
```

## routes/jobs.js

---

```
06.5-jobster-api > starter > routes > js jobs.js > ...
10  getJob,
11  showStats,
12 } = require('../controllers/jobs');
13
14 router.route('/').post(testUser, createJob);
15 router.route('/stats').get(showStats);
16
17 router
18   .route('/:id')
19   .get(getJob)
20   .delete(testUser, deleteJob)
21   .patch(testUser, updateJob);
22
23 module.exports = router;
```

```
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   showStats,
12 } = require('../controllers/jobs');
13
14 router.route('/').post(testUser, createJob).get(getAllJobs);
15 router.route('/stats').get(showStats);
16
17 router
18   .route('/:id')
19   .get(getJob)
20   .delete(testUser, deleteJob)
21   .patch(testUser, updateJob);
22
23 module.exports = router;
```

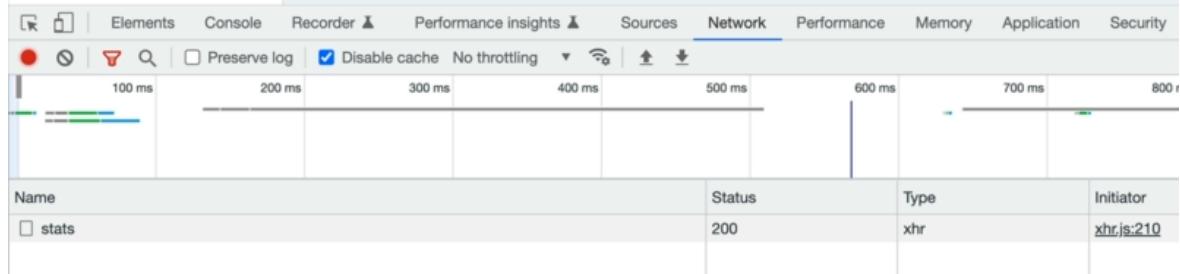
```
const express = require('express');
const testUser = require('../middleware/testUser');

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');
````
```

```
smilga@work-iMac starter % install moment
usage: install [-bCcpSsv] [-B suffix] [-f flags] [-g group] [-m mode]
                [-o owner] file1 file2
                install [-bCcpSsv] [-B suffix] [-f flags] [-g group] [-m mode]
                [-o owner] file1 ... fileN directory
                install -d [-v] [-g group] [-m mode] [-o owner] directory ...
smilga@work-iMac starter %
```

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL  
smilga@work-iMac starter % npm install moment  
((#####)) :: reify:moment: sill audit bulk request {
```

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL  
smilga@work-iMac starter % npm run dev
```

## controllers/jobs.js

---

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

```
06.5-jobster-api > starter > controllers > js jobs.js > ...  
1 const Job = require('../models/Job');  
2 const { StatusCodes } = require('http-status-codes');  
3 const { BadRequestError, NotFoundError } = require('../errors');  
4 const mongoose = require('mongoose');  
5 const moment = require('moment');  
6  
7  
8 const getAllJobs = async (req, res) => {  
9   const { search, status, jobType, sort } = req.query;  
10  
11  const queryObject = {  
12    createdBy: req.user.userId,  
13  };  
14  
15  if (search) {  
16    queryObject.position = { $regex: search, $options: 'i' };
```

**if user id is Demo**

The dashboard shows three metrics:

- Pending Applications: 0
- Interviews Scheduled: 0
- Jobs Declined: 0

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

The dashboard shows three metrics:

- Pending Applications: 24
- Interviews Scheduled: 27
- Jobs Declined: 24

```
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);
```

```
06.5-jobster-api > starter > controllers > JS jobs.js > (e) showStats > (e) stats > $group > $_id (, ,  
108 };  
109  
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   console.log(stats);  
116   res  
117     .status(StatusCodes.OK)  
118     .json({ defaultStats: {}, monthlyApplications: [] });  
119 };  
120  
121 module.exports = {  
122   createJob,  
123   deleteJob,  
124 };
```

PROBLEMS ⑥ OUTPUT DEBUG CONSOLE TERMINAL

```
Server is listening on port 5000...  
[  
  { _id: 'declined', count: 22 },  
  { _id: 'interview', count: 25 },  
  { _id: 'pending', count: 28 }  
]
```

## 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 }

```

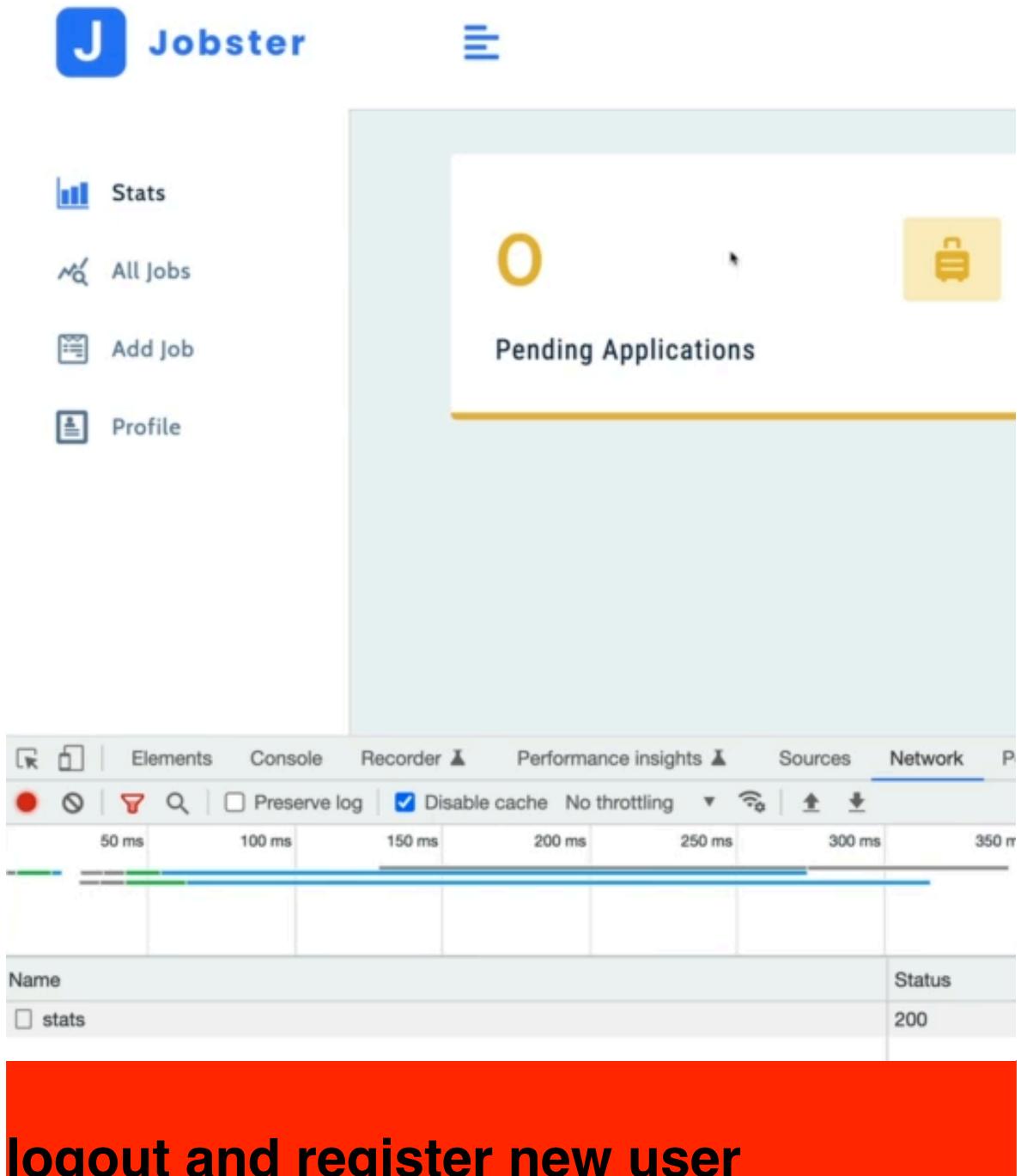
**stats = stats.reduce((acc, curr) => {  
 const { \_id: title, count } = curr;  
 acc[title] = count;  
 return acc;  
 }, {});**

```
108  };
109
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   console.log(stats);
123   res
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

```
{ _id: 'pending', count: 28 }
]
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Server is listening on port 5000...
{ interview: 25, declined: 22, pending: 28 }
```

**send request in the browser to see the above result**



 Jobster

Logging Out...

Login

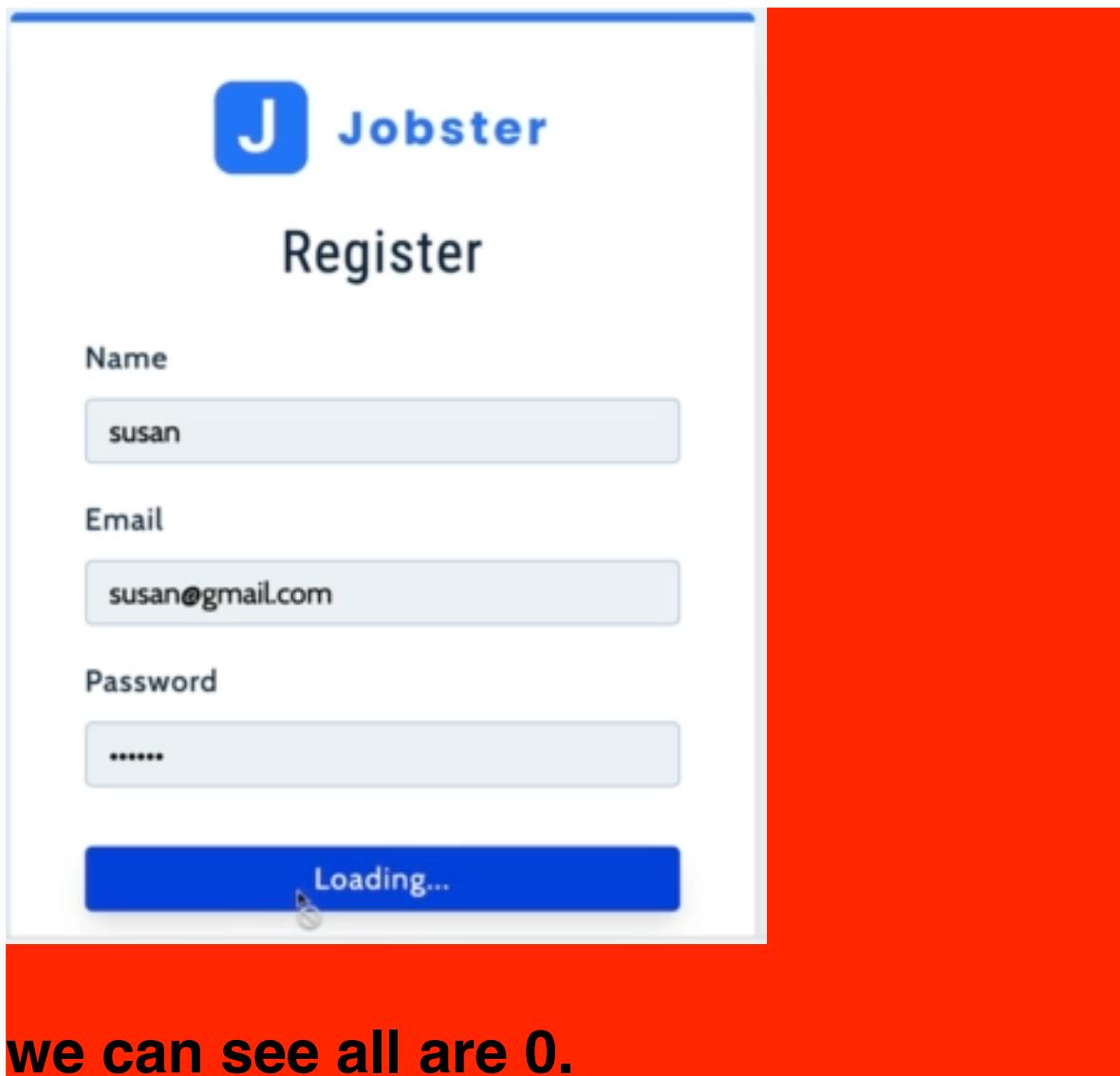
Email

Password

Submit

Demo App

Not a member yet? [Register](#)



we can see all are 0.

The image shows the Jobster dashboard for a user named Susan. At the top, there is a message "Hello There Susan". Below it, three cards provide summary statistics: "Pending Applications" (0), "Interviews Scheduled" (0), and "Jobs Declined" (0). The dashboard has a light blue header and a white body.

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

PROBLEMS 6

OUTPUT

DEBUG CONSOLE

TERMINAL

```
]
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Server is listening on port 5000...
{ interview: 25, declined: 22, pending: 28 }
{}
```

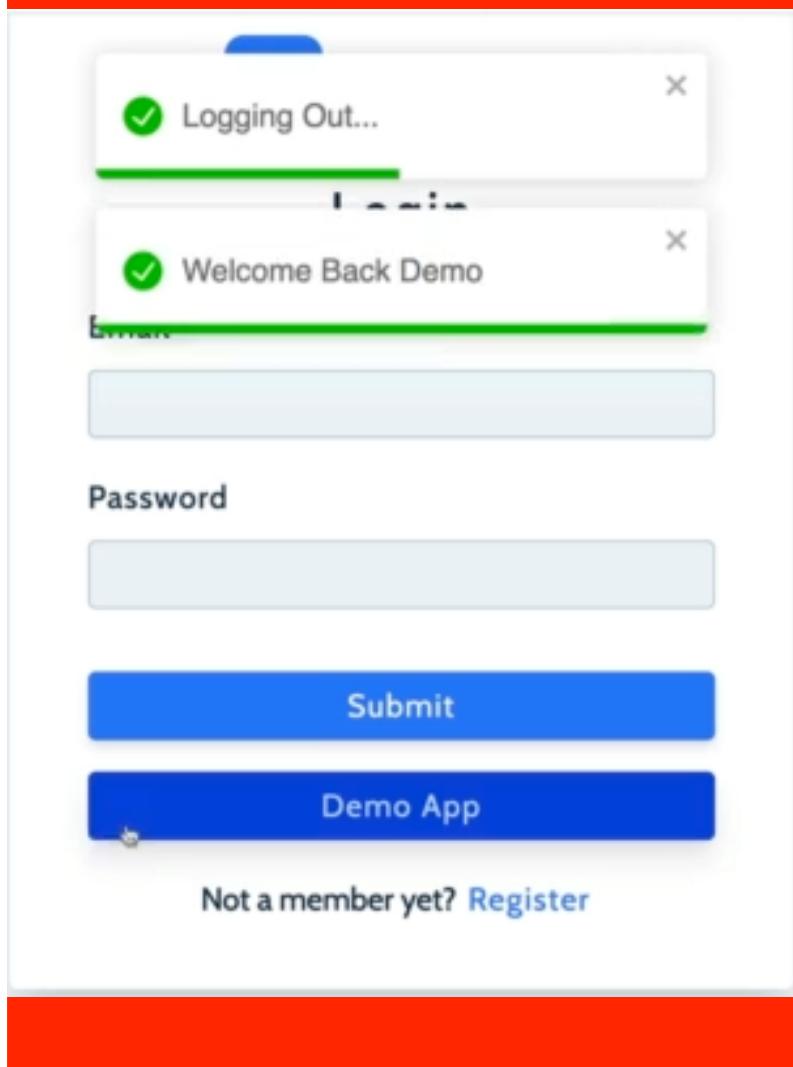
```
const defaultStats = {
  pending: stats.pending || 0,
  interview: stats.interview || 0,
  declined: stats.declined || 0,
};

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

```
06.5-jobster-api > starter > controllers > JS jobs.js > [e] showStats
  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   res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications: [] });
  128 };
  129
  130 module.exports = {
  131   createJob,
  132   deleteJob,
```

log out from susan user and login as

# demo



Dashboard

Demo ▾

28



Pending Applications

25



Interviews Scheduled

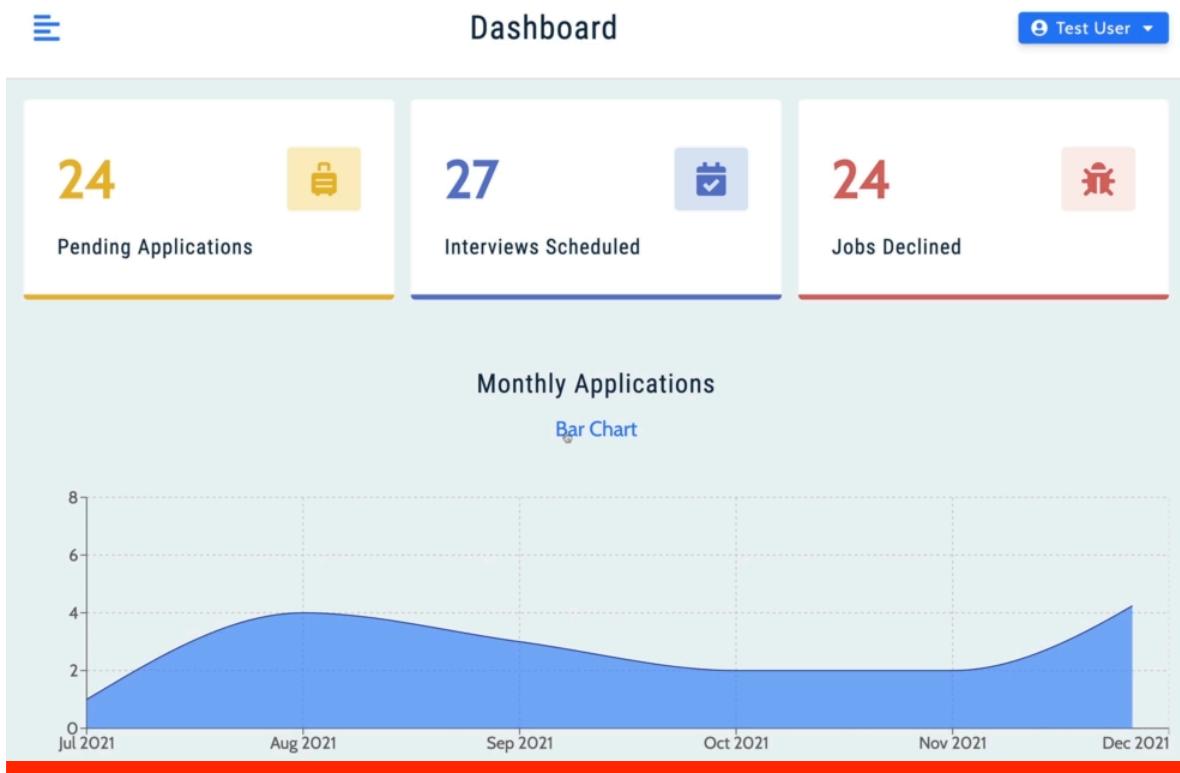
22



Jobs Declined

## 27 - Setup Monthly Applications Aggregation Pipeline

## Bar chart



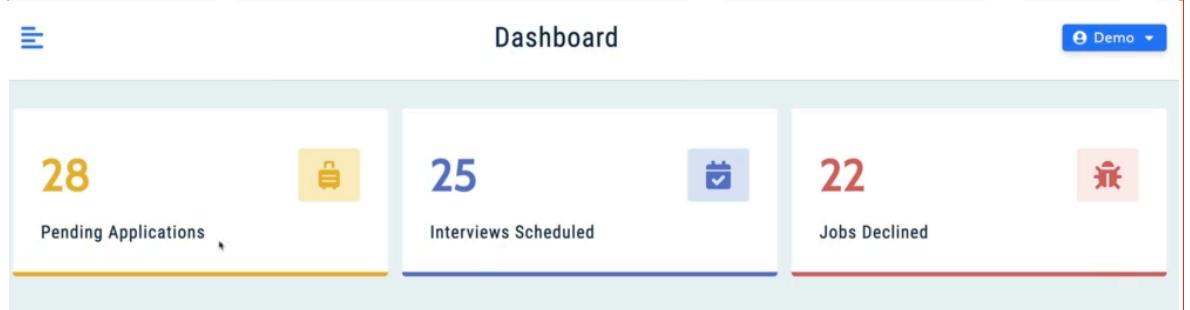
## Area Chart



```
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);
```

```
06.5-jobster-api > starter > controllers > jobs.js > (e) showStats  
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 ]);  
137 console.log(monthlyApplications);  
138 res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications: [] });
```



```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

[nodemon] starting `node app.js`
Server is listening on port 5000...
[
  { _id: { year: 2022, month: 3 }, count: 8 },
  { _id: { year: 2022, month: 5 }, count: 4 },
  { _id: { year: 2022, month: 6 }, count: 5 },
  { _id: { year: 2021, month: 8 }, count: 5 },
```

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

{
  { _id: { year: 2022, month: 2 }, count: 3 },
  { _id: { year: 2022, month: 7 }, count: 6 },
  { _id: { year: 2022, month: 8 }, count: 5 },
  { _id: { year: 2022, month: 1 }, count: 10 },
  { _id: { year: 2021, month: 9 }, count: 4 }
```

**to sort for last six months, from higher to low level**

```
{ $sort: { '_id.year': -1, '_id.month': -1 } },
```

```
16.5-jobster-api > starter > controllers > JS jobs.js > (e) showStats > (e) monthlyApplications > ⚡ $sort > ⚡ '_id.month'  
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 ]);  
138 console.log(monthlyApplications);  
139 res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications: [] });  
140 };  
141  
142 module.exports = {  
143   createJob,  
144   deleteJob,  
145   getAllJobs,  
146   updateJob,  
147 };  
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL node - starter +
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

```
[  
  { _id: { year: 2022, month: 8 }, count: 5 },  
  { _id: { year: 2022, month: 7 }, count: 6 },  
  { _id: { year: 2022, month: 6 }, count: 5 },  
  { _id: { year: 2022, month: 5 }, count: 4 },  
  { _id: { year: 2022, month: 4 }, count: 6 },  
  { _id: { year: 2022, month: 3 }, count: 8 },
```

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

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 console.log(monthlyApplications);  
140 res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications: [] });  
141 };  
142  
143 module.exports = {  
144 createJob  
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL node - starter +  
  
Server is listening on port 5000...  
[  
{ \_id: { year: 2022, month: 8 }, count: 5 },  
{ \_id: { year: 2022, month: 7 }, count: 6 },  
{ \_id: { year: 2022, month: 6 }, count: 5 },  
{ \_id: { year: 2022, month: 5 }, count: 4 },  
{ \_id: { year: 2022, month: 4 }, count: 6 },  
{ \_id: { year: 2022, month: 3 }, count: 5 },  
{ \_id: { year: 2022, month: 2 }, count: 4 },  
{ \_id: { year: 2022, month: 1 }, count: 5 },  
{ \_id: { year: 2021, month: 12 }, count: 5 },  
{ \_id: { year: 2021, month: 11 }, count: 4 },  
{ \_id: { year: 2021, month: 10 }, count: 5 },  
{ \_id: { year: 2021, month: 9 }, count: 4 },  
{ \_id: { year: 2021, month: 8 }, count: 5 },  
{ \_id: { year: 2021, month: 7 }, count: 4 },  
{ \_id: { year: 2021, month: 6 }, count: 5 },  
{ \_id: { year: 2021, month: 5 }, count: 4 },  
{ \_id: { year: 2021, month: 4 }, count: 5 },  
{ \_id: { year: 2021, month: 3 }, count: 4 },  
{ \_id: { year: 2021, month: 2 }, count: 3 },  
{ \_id: { year: 2021, month: 1 }, count: 4 },  
{ \_id: { year: 2020, month: 12 }, count: 4 },  
{ \_id: { year: 2020, month: 11 }, count: 3 },  
{ \_id: { year: 2020, month: 10 }, count: 4 },  
{ \_id: { year: 2020, month: 9 }, count: 3 },  
{ \_id: { year: 2020, month: 8 }, count: 4 },  
{ \_id: { year: 2020, month: 7 }, count: 3 },  
{ \_id: { year: 2020, month: 6 }, count: 4 },  
{ \_id: { year: 2020, month: 5 }, count: 3 },  
{ \_id: { year: 2020, month: 4 }, count: 4 },  
{ \_id: { year: 2020, month: 3 }, count: 3 },  
{ \_id: { year: 2020, month: 2 }, count: 2 },  
{ \_id: { year: 2020, month: 1 }, count: 3 },  
{ \_id: { year: 2019, month: 12 }, count: 3 },  
{ \_id: { year: 2019, month: 11 }, count: 2 },  
{ \_id: { year: 2019, month: 10 }, count: 3 },  
{ \_id: { year: 2019, month: 9 }, count: 2 },  
{ \_id: { year: 2019, month: 8 }, count: 3 },  
{ \_id: { year: 2019, month: 7 }, count: 2 },  
{ \_id: { year: 2019, month: 6 }, count: 3 },  
{ \_id: { year: 2019, month: 5 }, count: 2 },  
{ \_id: { year: 2019, month: 4 }, count: 3 },  
{ \_id: { year: 2019, month: 3 }, count: 2 },  
{ \_id: { year: 2019, month: 2 }, count: 1 },  
{ \_id: { year: 2019, month: 1 }, count: 2 },  
{ \_id: { year: 2018, month: 12 }, count: 2 },  
{ \_id: { year: 2018, month: 11 }, count: 1 },  
{ \_id: { year: 2018, month: 10 }, count: 2 },  
{ \_id: { year: 2018, month: 9 }, count: 1 },  
{ \_id: { year: 2018, month: 8 }, count: 2 },  
{ \_id: { year: 2018, month: 7 }, count: 1 },  
{ \_id: { year: 2018, month: 6 }, count: 2 },  
{ \_id: { year: 2018, month: 5 }, count: 1 },  
{ \_id: { year: 2018, month: 4 }, count: 2 },  
{ \_id: { year: 2018, month: 3 }, count: 1 },  
{ \_id: { year: 2018, month: 2 }, count: 1 },  
{ \_id: { year: 2018, month: 1 }, count: 1 }]

# 28 - Refactor Monthly Applications Data

**we have to show last months as first  
one**

## 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 (deprecated)
```

## Format Dates

```
moment().format('MMM Do YYYY, h:mm:ss a'); // August 14th 2022, 11:37:00 am
moment().format('dddd');
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
```

## Monthly Applications

Area Chart



**monthlyApplications =**

# monthlyApplications

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

```
06.5-jobster-api > starter > controllers > JS_jobs.js > showStats > monthlyApplications.map() callback
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.map((item) => {
141   const {
142     _id: { year, month },
143     count,
144   } = item;
145   const date = moment()
146     .month(month - 1)
147     .year(year)

{ 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 }
]
```

PROBLEMS ⑥ OUTPUT DEBUG CONSOLE TERMINAL

node - starter +

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 }
```

]

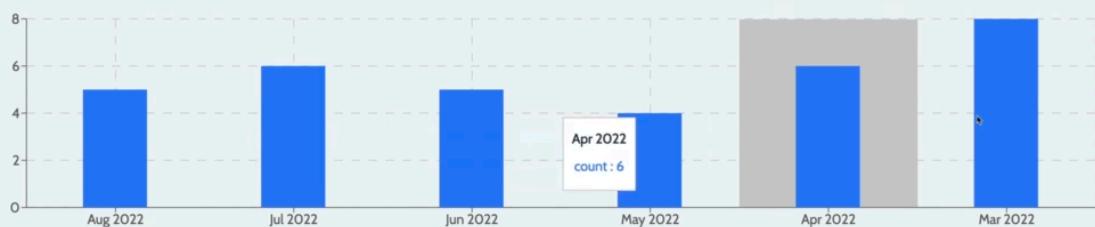


Dashboard

[Demo ▾](#)

Monthly Applications

Area Chart



.reverse();

```
controllers > JS jobs.js > [o] showStats > [o] monthlyApplications.map() callback
144   count,
145 } = item;
146 const date = moment()
147   .month(month - 1)
148   .year(year)
149   .format('MMM (property) count: any')
150 return { date, count };
151 }
152 .reverse();
153
154 res.status(StatusCodes.OK).json({ defaultStats, monthlyApplications });
155 }
```

## 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)
|| 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 });
};

const getJob = async (req, res) => {
```

```
const {  
  user: { userId },  
  params: { id: jobId },  
} = req;  
  
const job = await Job.findOne({  
  _id: jobId,  
  createdBy: userId,  
});  
if (!job) {  
  throw newNotFoundError(`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).js
```

```
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.userId) } },
    { $group: { _id: '$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.userId) } },  
    {  
        $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();

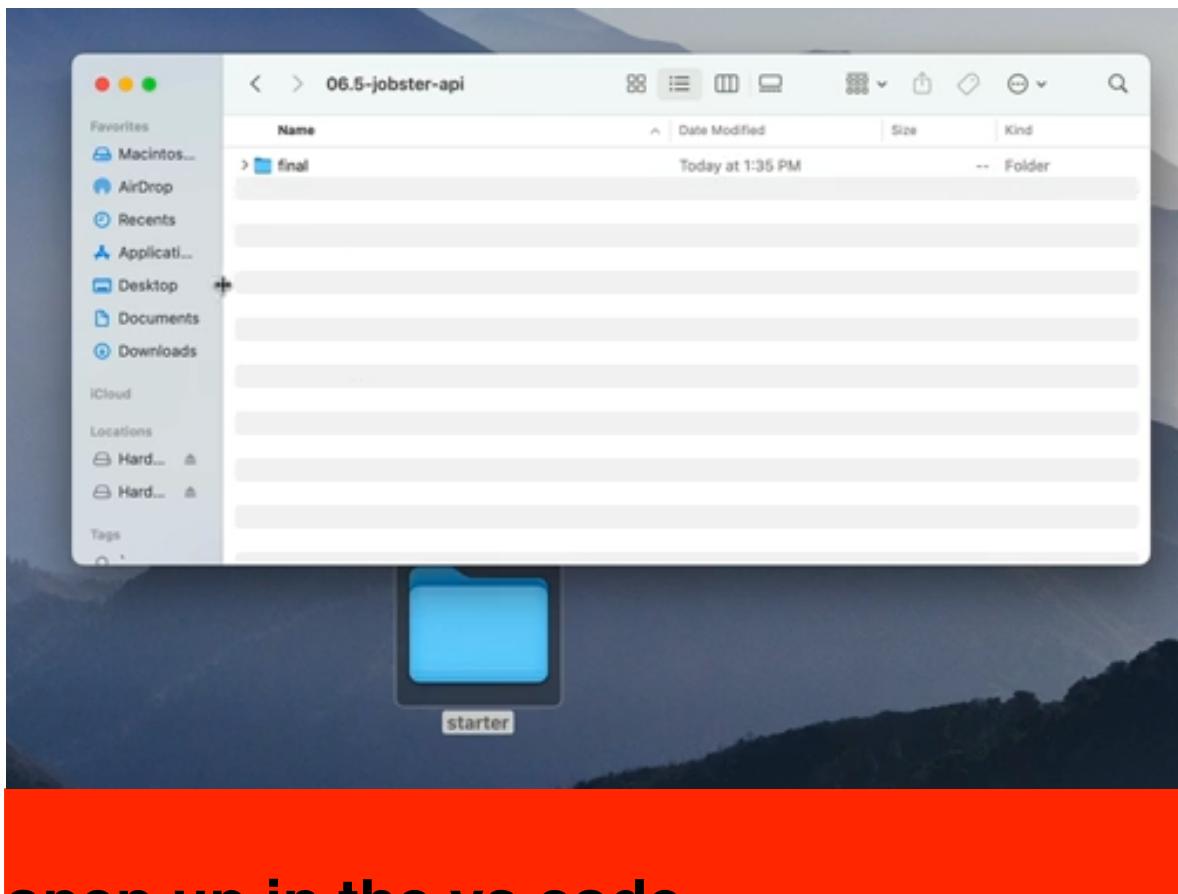
res.status(StatusCodes.OK).json({
    defaultStats, monthlyApplications
});
}

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

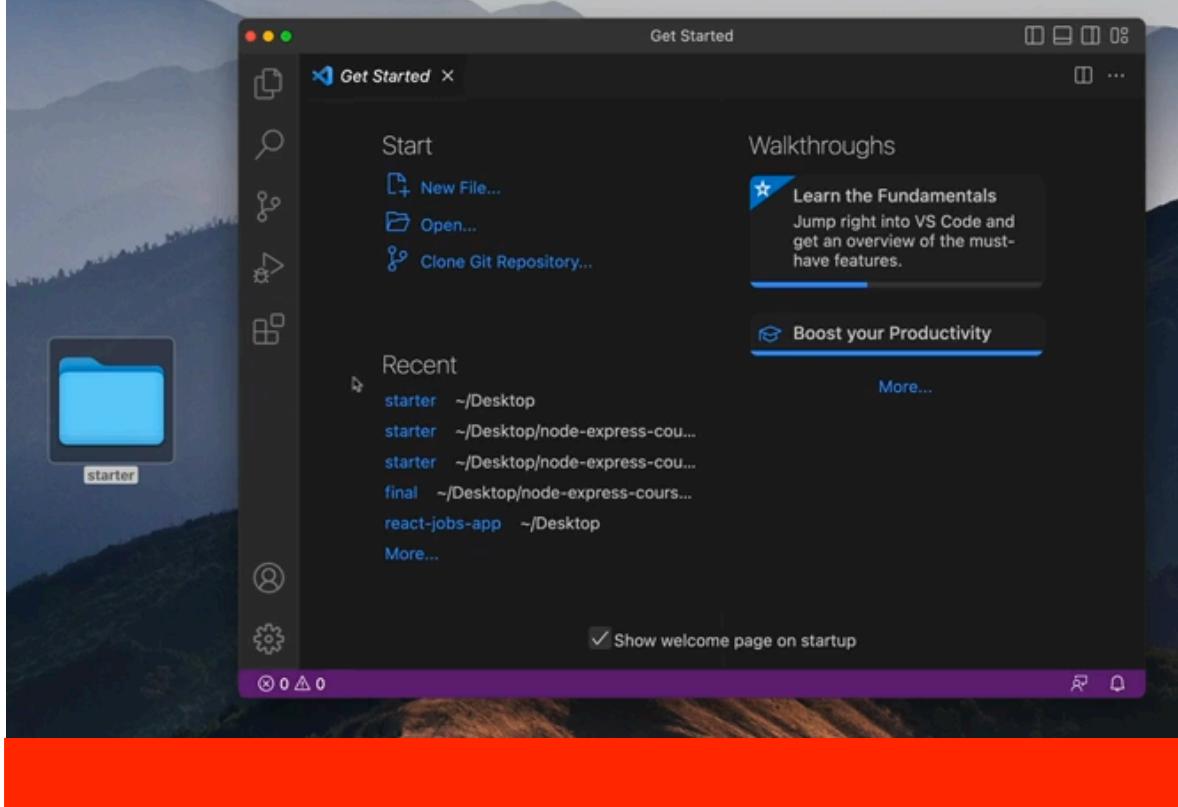
## 20 - Deployment

---

move this application to Desktop



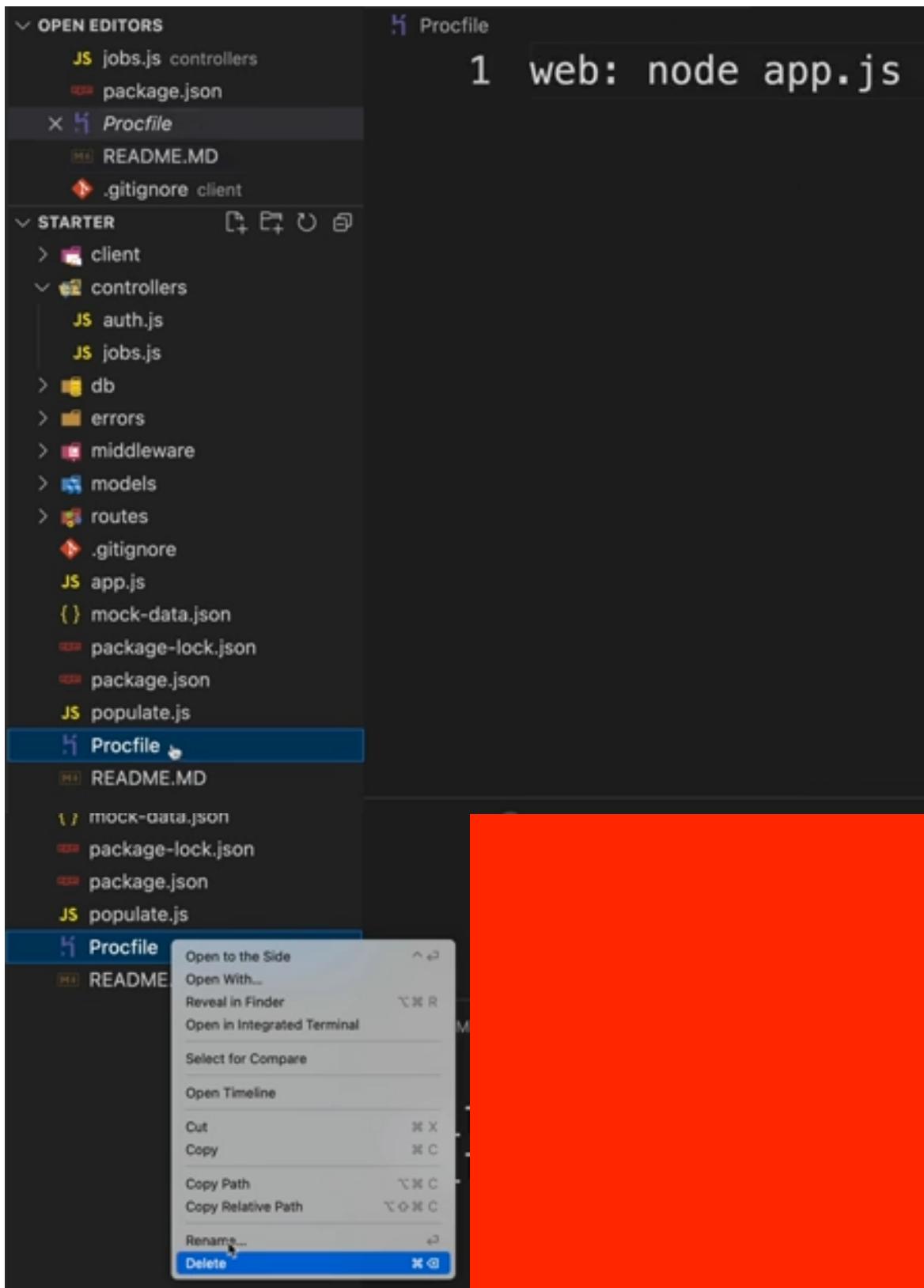
open up in the vs code



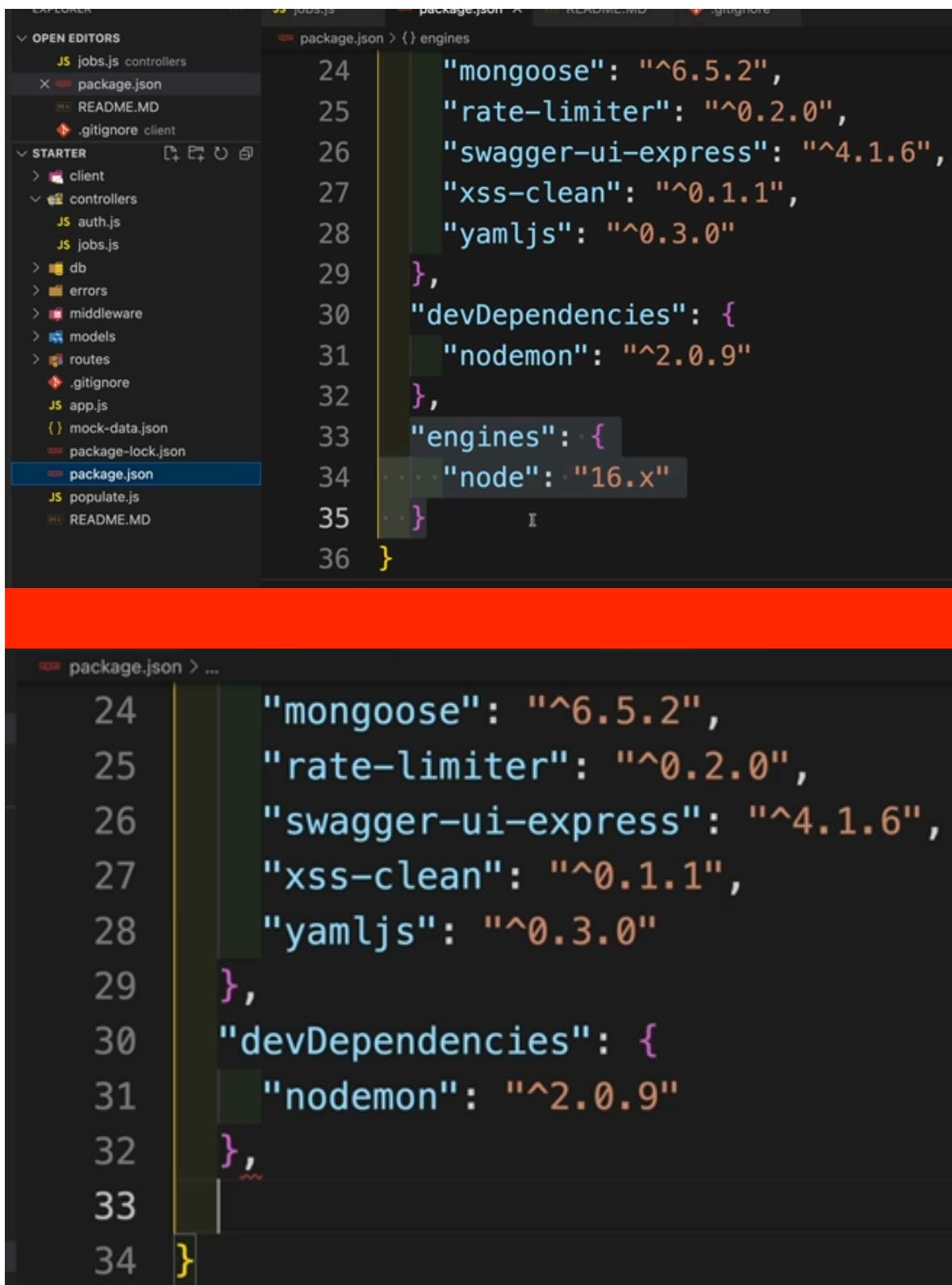
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL

- smilga@work-iMac starter % rm -rf .git
- smilga@work-iMac starter % █

```
670  - remove Procfile
671  - remove engines from package.json|
672
673  ````json
674  "engines": {
675    "node": "16.x"
676  }
677  `````
678
679  - fix build folder (remove /build from client/.gitignore)
680  - setup new github repo
681  - deploy to render
```



**remove the engines**



```
24 "mongoose": "^6.5.2",
25 "rate-limiter": "^0.2.0",
26 "swagger-ui-express": "^4.1.6",
27 "xss-clean": "^0.1.1",
28 "yamljs": "^0.3.0"
29 },
30 "devDependencies": {
31   "nodemon": "^2.0.9"
32 },
33 "engines": {
34   "node": "16.x"
35 }
36 }
```

```
24 "mongoose": "^6.5.2",
25 "rate-limiter": "^0.2.0",
26 "swagger-ui-express": "^4.1.6",
27 "xss-clean": "^0.1.1",
28 "yamljs": "^0.3.0"
29 },
30 "devDependencies": {
31   "nodemon": "^2.0.9"
32 },
33 "engines": {
34   "node": "16.x"
35 }
36 }
```

**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**

The screenshot shows a code editor with a dark theme. On the left is a file tree under 'OPEN EDITORS' and 'STARTER'. The 'client' folder is expanded, showing 'build', 'public', and 'src' subfolders. Inside 'src', there are files: '.gitignore', 'package-lock.json', 'package.json', and 'README.md'. The '.gitignore' file is selected and open in the main editor area. The code in the editor is as follows:

```
2
3 # dependencies
4 /node_modules
5 /.pnp
6 .pnp.js
7
8 # testing
9 /coverage
10
11 # production
12 /build
13
14 # misc
```

**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**

```
JS app.js > ⚡ app.get('*') callback
29 // routes
30 app.use('/api/v1/auth', authRouter);
31 app.use('/api/v1/jobs', authenticateUser, jobsRouter);
32
33 app.get('*', (req, res) => {
34   res.sendFile(path.resolve(__dirname, './client/build', 'index.html'));
35 });
36
37 app.use(notFoundMiddleware);
38 app.use(errorHandlerMiddleware);
39
40 const port = process.env.PORT || 5000;
```

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

The screenshot shows the Visual Studio Code interface with two main panes. The left pane displays a file tree under the heading 'OPEN EDITORS' with 1 unsaved file. The right pane shows a code editor for a '.gitignore' file.

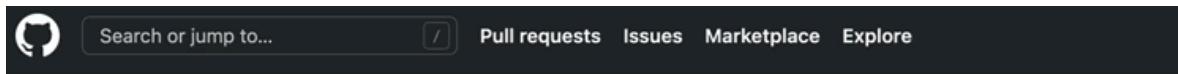
**File Tree (Left):**

- client > .gitignore
- 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

```
2
3 # dependencies
4 /node_modules
5 /.pnp
6 .pnp.js
7
8 # testing
9 /coverage
10
11 # production
12 |
13
14 # 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"
```



The image shows the top navigation bar of GitHub. It includes a search bar with placeholder text "Search or jump to...", a repository selection dropdown, and links for "Pull requests", "Issues", "Marketplace", and "Explore".

## Create a new repository

A repository contains all project files, including the revision history. Already have one elsewhere? [Import a repository](#).

Owner \*      Repository name \*

 john-smilga / temp-jobster-api ✓

Great repository names are `temp-jobster-api` is available. [I](#) [Inspiration?](#) [How abc](#)

Description (optional)

 Public  
Anyone on the internet can see this repository. You choose who can commit.

 Private  
You choose who can see and commit to this repository.

Initialize this repository with:  
 Add a README file  
This is where you can write a long description for your project. [Learn more](#).

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more](#).  
.gitignore template: None ▾

## ...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
```



The image shows a terminal window with the following text:

```
smilga@work-iMac starter % 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

**Code** Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

john-smilga first commit 823caa4 27 seconds ago 1 commit

| File        | Commit Message | Time           |
|-------------|----------------|----------------|
| client      | first commit   | 27 seconds ago |
| controllers | first commit   | 27 seconds ago |
| db          | first commit   | 27 seconds ago |
| errors      | first commit   | 27 seconds ago |
| middleware  | first commit   | 27 seconds ago |
| models      | first commit   | 27 seconds ago |
| routes      | first commit   | 27 seconds ago |
| .gitignore  | first commit   | 27 seconds ago |

## including /client/build files

main temp-jobster-api / client /

john-smilga first commit

..

| File   |
|--------|
| build  |
| public |
| src    |



go to the render

A screenshot of the Render dashboard at [dashboard.render.com](https://dashboard.render.com). The navigation bar includes "render", "Dashboard", "Blueprints", "Env Groups", "Docs", "Community", and a "New +" button. The "Dashboard" tab is selected. The "Overview" section shows a search bar with "Search services" and a table with one row: "NAME" (node-course-jobs-api) and "STATUS" (Deploy successful). To the right, a "Web Service" card is displayed with the following text:  
Web services are kept up and running at all times, with native SSL and HTTP/2 support. Add a persistent disk or custom domain. Scale up and down with ease.  
[Learn more.](#) Below the card is a sidebar with options: Static Site, Web Service (which is highlighted in blue), Private Service, Background Worker, Cron Job, PostgreSQL, Redis, and Blueprint.

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 change inside the root directory. Render runs commands  
e.g. src

**Environment**

The runtime environment for your web service.

**Node****Region**

The **region** where your web service runs. Services must be in the same region to communicate privately and you currently have services running in **Ohio**.

**Ohio (US East)****Branch**

The repository branch used for your web service.

**main****Build Command**

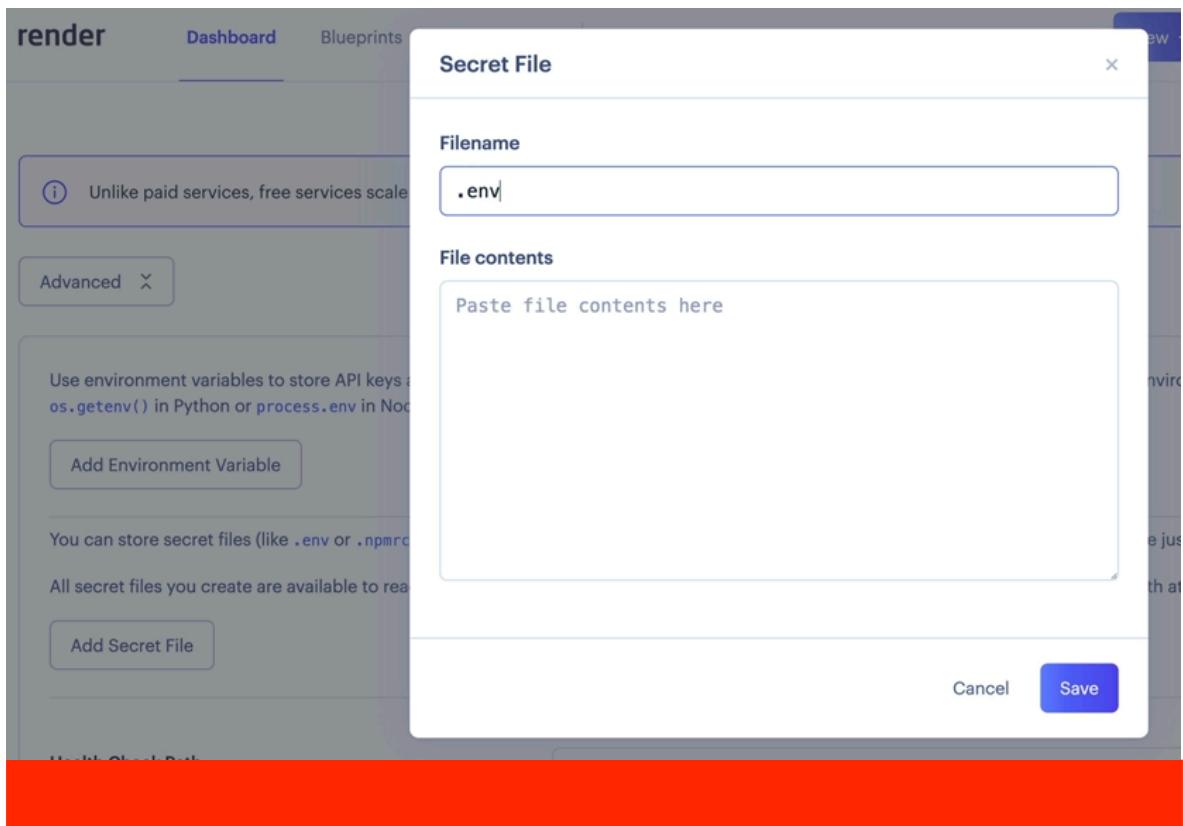
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

**\$ npm install****Start Command**

This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.

**\$ node app.js**

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



The screenshot shows the render.com dashboard for a "WEB SERVICE" named "node-course-jobster-api". The service is running on "Node" and has a "Free Plan". It is connected to a GitHub repository "john-smilga/temp-jobster-api" with a branch "main".

The left sidebar lists various service metrics: Events, Logs, Disks, Environment, Shell, PRs, Jobs, Sharing, Metrics, and Scaling. The "Logs" section is currently selected and expanded, showing log entries from October 31, 2022 at 2:47 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 found 0 vulnerabilities
- 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**

A screenshot of a web browser displaying a landing page for a job tracking application. The page has a light blue background. At the top left is a logo consisting of a blue square containing a white letter 'J'. To the right of the logo, the word 'Jobster' is written in a bold, blue, sans-serif font. Below the logo, there is a large, bold, dark blue heading that reads 'Job Tracking App'. Underneath this heading is a block of dark grey text that is mostly illegible due to being a placeholder or a dummy text. At the bottom of the page, there is a blue rectangular button with the text 'Login/Register' in white.

node-course-jobster-api.onrender.com/landing

J Jobster

# 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