

## My Courses > M220JS

**Course Ends:**

59d:03hr:27m

15 juin à 17:00 UTC

**Chapter Labs Due:**

59d:03hr:27m

15 juin à 17:00 UTC

Chapter 0: Introduction and... ▾

LESSONS

HANDOUTS

### Lessons



Lecture: Welcome To M220JS



Lecture: MongoDB URI



Quiz



Lecture: Setting Up Atlas

Lab (Ungraded): Create or Reuse  
Atlas Cluster

Lecture: OVERVIEW

Lecture: README: Setting Up  
mflix

Ticket: Connection

Next Chapter

Report an issue



Course Overview



View Discussion

Chapter 0: Introduction and Setup  
README: Setting Up mflix

---

## Download Course Materials

---

### Handouts (1)

 [m220/mflix-js.zip](#)

## Table of Contents

### Setting Up mflix:

1. Project Structure
2. Node Library Dependencies
3. Running the Application
4. Running the Unit Tests

In order to run properly, the MFlix software project has some installation requirements and environmental dependencies.

These requirements and dependencies are defined in this lesson, and they can also be found in the **README.rst** file from the **mflix-js** project, which you will download shortly. This lesson serves as a guide for setting up these necessary tools. After following this README, you should be able to successfully run the MFlix application. First, you will need to download the **mflix-js** project, as described below.

## Download the mflix-js.zip file

You can download the **mflix-js.zip** file by clicking the link in the "Handouts" section of this page. Downloading this handout may take a few minutes. When the download is complete, unzip the file and **cd** into the project's root directory, **mflix-js**.

```
cd ~/Downloads
unzip mflix-js.zip
cd mflix-js
```

 COPY

Note: The handout contains some hidden files like **.babelrc**. Please ensure you copy the entire directory to a new location rather than selecting all files within your system's file browser to copy and paste into a new directory.

# Project Structure

Most of your work will be implementing methods in the **dao** directory, which contains all database interfacing methods. The API will make calls to Data Access Objects (DAOs) that interact directly with MongoDB.

The unit tests in **test** will test these database access methods directly, without going through the API. The UI will run these methods in integration tests, and therefore requires the full application to be running.

The lesson handouts can be found in the **test/lessons** directory. These files will look like **<lesson-name>.spec.js**, and can be run with **npm test -t <lesson-name>**.

The API layer is fully implemented, as is the UI. The application is programmed to run on port **5000** by default - if you need to run on a port other than 5000, you can edit the **dotenv\_win** (if on Windows) or the **dotenv\_unix** file (if on Linux or Mac) in the root directory to modify the value of **PORT**.


Please do not modify the API layer in any way, under the **mflix-js/src/api** directory. This may result in the front-end application failing to validate some of the labs.

## Node Library Dependencies

The dependencies for the MFlix application should be downloaded using the **npm** command-line tool. You can get this tool by [downloading Node.js](#). Make sure to choose the correct option for your operating system.

Once the installation is complete, you may need to restart your computer before using the command line tools. You can test that it's installed by running the following command:

```
node -v
```

 COPY

This should print out the version of **node** you currently have - we recommend using version 10 or later, so this command should print something like **v10.x**.

Once **npm** is installed, you can install the MFlix dependencies by running the following command from the **mflix-js** directory:

```
npm install
```

 COPY

You must run this from the top level of the project, so **npm** has access to the

**package.json** file where the dependencies are.

While running `npm install`, you might encounter the below error regarding **node-gyp rebuild**. Although, it is completely harmless and you can start the application by running `npm start`.

```
mflix-js$ npm install
> weak-napi@1.0.3 install /Users/kanika/university/m220js-august/mflix-js/node_modules/weak-napi
> node-gyp rebuild

CC(target) Release/obj.target/nothing/../node-addon-api/src/nothing.o
LIBTOOL-STATIC Release/nothing.a
Traceback (most recent call last):
  File "/gyp-mac-tool", line 611, in <module>
    sys.exit(main(sys.argv[1:]))
  File "/gyp-mac-tool", line 28, in main
    exit_code = executor.Dispatch(args)
  File "/gyp-mac-tool", line 43, in Dispatch
    return getattr(self, method)(args[1:])
  File "/gyp-mac-tool", line 246, in ExecFilterLibtool
    if not libtool_re.match(line) and not libtool_re.match(line):
TypeError: cannot use a string pattern on a bytes-like object
make: *** [Release/nothing.a] Error 1
gyp ERR! build error
gyp ERR! stack Error: 'make' failed with exit code: 2
gyp ERR! stack   at ChildProcess.onExit (/usr/local/lib/node_modules/npm/node_modules/node-gyp/lib/build.js:262:23)
gyp ERR! stack   at ChildProcess.emit (events.js:200:13)
gyp ERR! stack   at Process.ChildProcess._handle.onexit (internal/child_process.js:272:12)
gyp ERR! System Darwin 18.7.0
gyp ERR! command "/usr/local/Cellar/node/12.4.0/bin/node" "/usr/local/lib/node_modules/npm/node_modules/node-gyp/bin/node-gyp.js" "rebuild"
gyp ERR! cwd /Users/kanika/university/m220js-august/mflix-js/node_modules/weak-napi
gyp ERR! node -v v12.4.0
gyp ERR! node-gyp -v v3.8.0
gyp ERR! not ok
npm WARN eslint-config-react-app@2.1.0 requires a peer of babel-eslint@^7.2.3 but none is installed. You must install peer dependencies yourself.
npm WARN eslint-config-react-app@2.1.0 requires a peer of eslint-plugin-jsx-ally@^5.1.1 but none is installed. You must install peer dependencies yourself.
npm WARN server@1.0.0 No repository field.

npm ERR! Maximum call stack size exceeded


npm ERR! A complete log of this run can be found in:
npm ERR!   /Users/kanika/.npm/_logs/2019-08-27T11_55_34_838Z-debug.log
```

## Running the Application

In order for the application to use Atlas, you will need a file called **.env** to contain the connection information. In the **mflix-js** directory you can find two files, **dotenv\_unix** (for Unix users) and **dotenv\_win** (for Windows users).

Open the file for your chosen operating system and enter your Atlas SRV connection string as directed in the comment. This is the information the driver will use to connect. Make sure **not** to wrap your Atlas SRV connection between quotes:


```
MFLIX_DB_URI = mongodb+srv://...
```

 COPY

It's highly suggested you also change the **SECRET\_KEY** to some very long, very random string. While this application is only meant for local use during this course, software has a strange habit of living a long time.


When you've edited the file, rename it to **.env** with the following command:

```
mv dotenv_unix .env    # on Unix
ren dotenv_win .env    # on Windows
```

 COPY

*Note:* Once you rename this file to **.env**, it will no longer be visible in Finder or File Explorer. However, it will be visible from Command Prompt or Terminal, so if you need to edit it again, you can open it from there:


```
vi .env          # on Unix
notepad .env     # on Windows
```

 COPY

In the **mflix-js** directory, run the following commands:

```
# install MFlix dependencies
npm install

# start the MFlix application
npm start
```

 COPY


This will start the application. You can then access the MFlix application at <http://localhost:5000/>.

## Running the Unit Tests

To run the unit tests for this course, you will use **Jest**. Jest has been included in this project's dependencies, so **npm install** should install everything you need.


Each course lab contains a module of unit tests that you can call individually with **npm test** from **mflix-js** directory like the following:

```
npm test -t TICKET_TEST_NAME
```

 COPY

Each ticket will contain the exact command to run that ticket's specific unit tests. You can run these commands from anywhere in the **mflix-js** project. Bear in mind that a tests will fail until the corresponding ticket is completed. For example to run the Connection Ticket test your shell command will be:

```
npm test -t db-connection
```

 COPY

Proceed to next section

