# Node js Live stream App

## Group members


- Muhammad Amir Hassan (19-CS-60)
- Wajahat Hanif (19-CS-58)
- Muhammad Usman (19-CS-91)
- Muhammad Yahya Sajjad (19-CS-87)


Submitted To: Sir Rashid Amin
Date: 27-January-2022


**Department of Computer Science, UET TAXILA**

# Overview

- Basic User Authentication using Passport JS
- RTMP streaming server and ffmpeg library in node js
- Displaying live streams
- Issue stream keys

## Web Server with Basic Authentication

Let's set up a basic node server with passport local strategy authentication. We will use MongoDB with Mongoose for persistent storage. Initialize a new project by running

In your project directory, create two folders client and server. We will place our react components inside the client directory and backend code in the server directory. For this part, we will be working inside the server directory. We are using passport.js for authentication.
We have already installed **passport** and **passport-local** modules. Before we define our **local strategy** for authenticating users. Let us create an **app.js** file and add the necessary code to run a basic web server. Make sure you have **MongoDB**(DATABASE) installed on your system and running as a service.

## Node js Main App file:

```
const express = require('express'),
   Session = require('express-session'),
   bodyParse = require('body-parser'),
   mongoose = require('mongoose'),
   middleware = require('connect-ensure-login'),
   FileStore = require('session-file-store')(Session),
   config = require('./config/default'), //server configuration for streaming
   flash = require('connect-flash'),
   port = 3333,
   app = express();


mongoose.connect('mongodb://127.0.0.1/nodeStream' , { useNewUrlParser: true });


app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, './views'));
app.use(express.static('public'));
app.use(flash());
app.use(require('cookie-parser')());
```

```
app.use(bodyParse.urlencoded({extended: true}));
app.use(bodyParse.json({extended: true}));

app.use(Session({
   store: new FileStore({
      path : './server/sessions'
   }),
   secret: config.server.secret,
   maxAge : Date().now + (60 * 1000 * 30)
}));

app.get('*', middleware.ensureLoggedIn(), (req, res) => {
   res.render('index');
});

app.listen(port, () => console.log(`App listening on ${port}!`));
```

All the necessary middle wares for our application, connected to MongoDB and configured **express session** to use the file storage for session persistence in case of the web server restart. Now we will define our **passport strategies** for registering and authenticating users.

# User Login and register system

## Passport JS code

```
const passport = require('passport'),
   LocalStrategy = require('passport-local').Strategy,
   User = require('../database/Schema').User,
   shortid = require('shortid');
```

```
passport.serializeUser( (user, cb) => {
   cb(null, user);
});

passport.deserializeUser( (obj, cb) => {
   cb(null, obj);
});

// Passport strategy for handling user registration
passport.use('localRegister', new LocalStrategy({
      usernameField: 'email',
      passwordField: 'password',
      passReqToCallback: true
   },
   (req, email, password, done) => {
      User.findOne({$or: [{email: email}, {username: req.body.username}]},  (err, user) => {
         if (err)
            return done(err);
         if (user) {
            if (user.email === email) {
               req.flash('email', 'Email is already taken');
            }
            if (user.username === req.body.username) {
               req.flash('username', 'Username is already taken');
            }

            return done(null, false);
         } else {
            let user = new User();
            user.email = email;
            user.password = user.generateHash(password);
```

```
            user.username = req.body.username;
            user.stream_key = shortid.generate();
            user.save( (err) => {
               if (err)
                  throw err;
               return done(null, user);
            });
         }
      });
   }));

// Passport strategy for authenticating users
passport.use('localLogin', new LocalStrategy({
      usernameField: 'email',
      passwordField: 'password',
      passReqToCallback: true
   },
   (req, email, password, done) => {

      User.findOne({'email': email}, (err, user) => {
         if (err)
            return done(err);

         if (!user)
            return done(null, false, req.flash('email', 'Email doesn\'t exist.'));

         if (!user.validPassword(password))
            return done(null, false, req.flash('password', 'Oops! Wrong password.'));

         return done(null, user);
      });
```

```
}));
```

```
module.exports = passport;
```

# DATABASE SCHEMA DESIGN

```
let mongoose = require('mongoose'),
    bcrypt   = require('bcrypt-nodejs'),
    shortid = require('shortid'),
    Schema = mongoose.Schema;

let UserSchema = new Schema({
    username: String,
    email : String,
    password: String,
    stream_key : String,
});

UserSchema.methods.generateHash = (password) => {
    return bcrypt.hashSync(password, bcrypt.genSaltSync(8), null);
};

UserSchema.methods.validPassword = function(password){
    return bcrypt.compareSync(password, this.password);
};

UserSchema.methods.generateStreamKey = () => {
    return shortid.generate();
};

module.exports = UserSchema;
```

**GenerateHash** method will convert plain text password to bcrypt hash. **ValidPassword** method will take in a plain text password and validate it by comparing it to bcrypt hash stored in our database. **GenerateStreamKey** method will generate a unique string that we will issue to users as their streaming key for RTMP clients.

# DATABASE model export code

**let mongoose = require('mongoose');**

**exports.User = mongoose.model('User', require('./UserSchema'));**

Now we will add our passport strategy, user schema of database model in Main app.js

Add following in **App.js**

```
const passport = require('./auth/passport');
app.use(passport.initialize());
app.use(passport.session());

// Register routes in main app

app.use('/login', require('./routes/login'));
app.use('/register', require('./routes/register'));
```

Create a **login.js** and **register.js** file under routes directory where we will define these routes and use passport middleware for registration and authentication.

## Creating Routes for login and register in Backend code

```
const express = require('express'),
    router = express.Router(),
    passport = require('passport');

router.get('/',
    require('connect-ensure-login').ensureLoggedOut(),
    (req, res) => {
        res.render('login', {
```

```
            user : null,
            errors : {
                email : req.flash('email'),
                password : req.flash('password')
            }
        });
    });

router.post('/', passport.authenticate('localLogin', {
    successRedirect : '/',
    failureRedirect : '/login',
    failureFlash : true
}));

module.exports = router;
```

## REGISTER ROUTE FILE code

```
const express = require('express'),
    router = express.Router(),
    passport = require('passport');

router.get('/',
    require('connect-ensure-login').ensureLoggedOut(),
    (req, res) => {
        res.render('register', {
            user : null,
            errors : {
                username : req.flash('username'),
                email : req.flash('email')
            }
        });
    });
```

```
router.post('/',
    require('connect-ensure-login').ensureLoggedOut(),
    passport.authenticate('localRegister', {
        successRedirect : '/',
        failureRedirect : '/register',
        failureFlash : true
    })
);


module.exports = router;
```

## FRONT END CODE for same both above files to show on Web

```
<!doctype html>
<html lang="en">
<% include header.ejs %>
<body>

<div class="container app mt-5">
    <h4>Login</h4>

    <hr class="my-4">
    <div class="row">
        <form action="/login" method="post" class="col-xs-12 col-sm-12 col-md-8 col-lg-6">
            <div class="form-group">
                <label>Email address</label>
                <input type="email" name="email" class="form-control" placeholder="Enter email" required>
                <% if (errors.email.length) { %>
                    <small class="form-text text-danger"><%= errors.email %></small>
                <% } %>
            </div>
            <div class="form-group">
```

```
            <label>Password</label>
            <input type="password" name="password" class="form-control" placeholder="Password"
required>
            <% if (errors.password.length) { %>
                <small class="form-text text-danger"><%= errors.password %></small>
            <% } %>
          </div>
          <div class="form-group">
            <div class="leader">
                Don't have an account? Register <a href="/register">here</a>.
            </div>
          </div>
          <button type="submit" class="btn btn-dark btn-block">Login</button>
        </form>
    </div>
</div>

<% include footer.ejs %>
</body>
</html>
```

**REGISTER FRONT END COED**

```
<!doctype html>
<html lang="en">
<% include header.ejs %>
<body>

<div class="container app mt-5">
    <h4>Register</h4>

    <hr class="my-4">

    <div class="row">
```

```
<form action="/register"
    method="post"
    class="col-xs-12 col-sm-12 col-md-8 col-lg-6">

    <div class="form-group">
      <label>Username</label>
      <input type="text" name="username" class="form-control" placeholder="Enter username" required>
      <% if (errors.username.length) { %>
        <small class="form-text text-danger"><%= errors.username %></small>
      <% } %>
    </div>

    <div class="form-group">
      <label>Email address</label>
      <input type="email" name="email" class="form-control" placeholder="Enter email" required>
      <% if (errors.email.length) { %>
        <small class="form-text text-danger"><%= errors.email %></small>
      <% } %>
    </div>

    <div class="form-group">
      <label>Password</label>
      <input type="password" name="password" class="form-control" placeholder="Password" required>
    </div>

    <div class="form-group">
      <div class="leader">
        Have an account? Login <a href="/login">here</a>.
      </div>
    </div>
```

```
        <button type="submit" class="btn btn-dark btn-block">Register</button>
      </form>
    </div>
</div>

<% include footer.ejs %>
</body>
</html>
```

# Setting Up RTMP server Configuration

Now we will move onto the next part of this tutorial and set up our RTMP server.

**Real-Time Messaging Protocol (RTMP)** was designed for high-performance transmission of video, audio, and data between broadcaster and server.

We are using node-media-server, a Node.js implementation of RTMP media server. It accepts RTMP streams and remux them to HLS/DASH using ffmpeg.

We must have installed **ffmpeg and RTMP** inour systems.

```
const config = {
  server: {
    secret: 'kjVkuti2xAyF3JGCzSZTk0YWM5JhI9mgQW4rytXc'
  },
  rtmp_server: {
    rtmp: {
      port: 1935,
      chunk_size: 60000,
      gop_cache: true,
      ping: 60,
      ping_timeout: 30
    },
    http: {
      port: 8888,
```

```
        mediaroot: './server/media',
        allow_origin: '*'
    },
    trans: {
        ffmpeg: '/usr/bin/ffmpeg',
        tasks: [
            {
                app: 'live',
                hls: true,
                hlsFlags: '[hls_time=2:hls_list_size=3:hls_flags=delete_segments]',
                dash: true,
                dashFlags: '[f=dash:window_size=3:extra_window_size=5]'
            }
        ]
    }
  }
};

module.exports = config;
```

Create **media_server.js** file and add this code.

```
const NodeMediaServer = require('node-media-server'),
    config = require('./config/default').rtmp_server;

nms = new NodeMediaServer(config);

nms.on('prePublish', async (id, StreamPath, args) => {
    let stream_key = getStreamKeyFromStreamPath(StreamPath);
    console.log('[NodeEvent on prePublish]', `id=${id} StreamPath=${StreamPath}
args=${JSON.stringify(args)}`);
});

const getStreamKeyFromStreamPath = (path) => {
```

```
    let parts = path.split('/');
    return parts[parts.length - 1];
};


module.exports = nms;
```

**NodeMediaServer** module usage is pretty straight forward. It runs **an RTMP server** and lets you listen to connection events. Invalid streams can be rejected. We will be listening to its **prePublish** event.

We will add more code inside **prePublish** event listener closure to reject incoming connections with invalid streaming keys. For now, we are accepting all incoming connection on default **1935** RTMP port. Now, all we have to do is import **nms** object in the app.js file and call its run method.

```
// Add this on the top of app.js file
// next to all imports
const node_media_server = require('./media_server');


// and call run() method at the end
// file where we start our web server
node_media_server.run();
```

Now We must have OBS studio downloaded in our system and from stream setting we will add

**Rtmp://127.0.0.1:1935/live** and in next field we add stream key generated.

# Front End Part of Project

So, why webpack? Webpack provides a great developer experience as it not only bundles JavaScript applications (supporting both EcmaScript Modules and CommonJS), but when paired with plugins and loaders it can be used for taking care of dependencies and assets, such as images, fonts, SASS, CSS, etx.

### INDEX JS File code

```
import React from "react";
import ReactDOM from 'react-dom';
```

```
import {BrowserRouter} from 'react-router-dom';
import 'bootstrap';
require('./index.scss');
import Root from './components/Root.js';

if(document.getElementById('root')){
ReactDOM.render(
<BrowserRouter>
<Root/>
</BrowserRouter>,
document.getElementById('root')
);
}
```

## BOOTSTRAP CSS File

```
@import '~bootstrap/dist/css/bootstrap.css';
@import '~video.js/dist/video-js.css';

@import url('https://fonts.googleapis.com/css?family=Dosis');

html,body{
  font-family: 'Dosis', sans-serif;
}
```

We are using **react-router** for routing and **bootstrap** on the frontend along with **video.js** for displaying live streams. Add components directory with **Root.js**

## ROOT JS FILE CODE (our main frontend HomePage)

```
import React from "react";
import {Router, Route} from 'react-router-dom';
import Navbar from './Navbar';
import LiveStreams from './LiveStreams';
import Settings from './Settings';

import VideoPlayer from './VideoPlayer';
```

```
const customHistory = require("history").createBrowserHistory();

export default class Root extends React.Component {

    constructor(props){
        super(props);
    }

    render(){
        return (
            <Router history={customHistory} >
                <div>
                    <Navbar/>
                    <Route exact path="/" render={props => (
                        <LiveStreams  {...props} />
                    )}/>

                    <Route exact path="/stream/:username" render={(props) => (
                        <VideoPlayer {...props}/>
                    )}/>

                    <Route exact path="/settings" render={props => (
                        <Settings {...props} />
                    )}/>
                </div>
            </Router>
        )
    }
}
```

**<Root/>** component renders a react **<Router/>** to hold three sub **<Route/>** components.  **<LiveStreams/>** component will render all the live streams.

**<VideoPlayer/>** will render video.js player components. **<Settings/>** component will provide an interface for generating a new streaming key

## LIVE STREAM JS File code

```
import React from 'react';

import axios from 'axios';

import {Link} from 'react-router-dom';

import './LiveStreams.scss';

import config from '../../server/config/default';



export default class Navbar extends React.Component {


constructor(props) {

super(props);

this.state = {

live_streams: []

}

}


componentDidMount() {

this.getLiveStreams();

}


getLiveStreams() {

axios.get('http://127.0.0.1:' + config.rtmp_server.http.port + '/api/streams')

.then(res => {

let streams = res.data;

if (typeof (streams['live'] !== 'undefined')) {

this.getStreamsInfo(streams['live']);

}

});

}
```

```
getStreamsInfo(live_streams) {
axios.get('/streams/info', {
params: {
streams: live_streams
}
}).then(res => {
this.setState({
live_streams: res.data
}, () => {
console.log(this.state);
});
});
}

render() {
let streams = this.state.live_streams.map((stream, index) => {
return (
<div className="stream col-xs-12 col-sm-12 col-md-3 col-lg-4" key={index}>
<span className="live-label">LIVE</span>
<Link to={'/stream/' + stream.username}>
<div className="stream-thumbnail">
<img src={'/thumbnails/' + stream.stream_key + '.png'}/>
</div>
</Link>

<span className="username">
<Link to={'/stream/' + stream.username}>
{stream.username}
</Link>
</span>
</div>
```

```
        );
    });

    return (
      <div className="container mt-5">
        <h4>Live Streams</h4>
        <hr className="my-4"/>

        <div className="streams row">
          {streams}
        </div>
      </div>
    )
  }
}
```

## BACKEND FILE CODE stream js

```
const express = require('express'),
router = express.Router(),
User = require('../database/Schema').User;

router.get('/info',
require('connect-ensure-login').ensureLoggedIn(),
(req, res) => {
if(req.query.streams){
let streams = JSON.parse(req.query.streams);
let query = {$or: []};
for (let stream in streams) {
if (!streams.hasOwnProperty(stream)) continue;
query.$or.push({stream_key : stream});
}
```

```
User.find(query,(err, users) => {
if (err)
return;
if (users) {
res.json(users);
}
});
}
});

module.exports = router;
```

Export this file to APP.JS

```
app.use('/streams', require('./routes/streams'));
```

We will render thumbnail of users live streams

## VideoPlayerJS code

```
import React from 'react';
import videojs from 'video.js'
import axios from 'axios';
import config from '../../server/config/default';

export default class VideoPlayer extends React.Component {

constructor(props) {
super(props);

this.state = {
stream: false,
videoJsOptions: null
```

```
}
}

componentDidMount() {

axios.get('/user', {
params: {
username: this.props.match.params.username
}
}).then(res => {
this.setState({
stream: true,
videoJsOptions: {
autoplay: false,
controls: true,
sources: [{
src: 'http://127.0.0.1:' + config.rtmp_server.http.port + '/live/' + res.data.stream_key + '/index.m3u8',
type: 'application/x-mpegURL'
}],
fluid: true,
}
}, () => {
this.player = videojs(this.videoNode, this.state.videoJsOptions, function onPlayerReady() {
console.log('onPlayerReady', this)
});
});
})
}

componentWillUnmount() {
if (this.player) {
```

```
this.player.dispose()
}
}

render() {
return (
<div className="row">
<div className="col-xs-12 col-sm-12 col-md-10 col-lg-8 mx-auto mt-5">
{this.state.stream ? (
<div data-vjs-player>
<video ref={node => this.videoNode = node} className="video-js vjs-big-play-centered"/>
</div>
) : ' Loading ... '}
</div>
</div>
)
}
}
```

## GENERATE STREAM KEYS

```
componentDidMount() {
this.getStreamKey();
}

generateStreamKey(e){
axios.post('/settings/stream_key')
.then(res => {
this.setState({
stream_key : res.data.stream_key
});
})
}
```

```
getStreamKey(){
axios.get('/settings/stream_key')
.then(res => {
this.setState({
stream_key : res.data.stream_key
});
})
}
```

## Server Side Setting for Stream keys generation

```
const express = require('express'),
router = express.Router(),
User = require('../database/Schema').User,
shortid = require('shortid');

router.get('/stream_key',
require('connect-ensure-login').ensureLoggedIn(),
(req, res) => {
User.findOne({email: req.user.email}, (err, user) => {
if (!err) {
res.json({
stream_key: user.stream_key
})
}
});
});

router.post('/stream_key',
require('connect-ensure-login').ensureLoggedIn(),
(req, res) => {
```

User.findOneAndUpdate({

email: req.user.email

}, {

stream_key: shortid.generate()

}, {

upsert: true,

new: true,

}, (err, user) => {

if (!err) {

res.json({

stream_key: user.stream_key

})

}

});

});

module.exports = router;

**HLS streaming thumbnail**

const generateStreamThumbnail = (stream_key) => {

const args = [

'-y',

'-i', 'http://127.0.0.1:8888/live/'+stream_key+'/index.m3u8',

'-ss', '00:00:01',

'-vframes', '1',

'-vf', 'scale=-2:300',

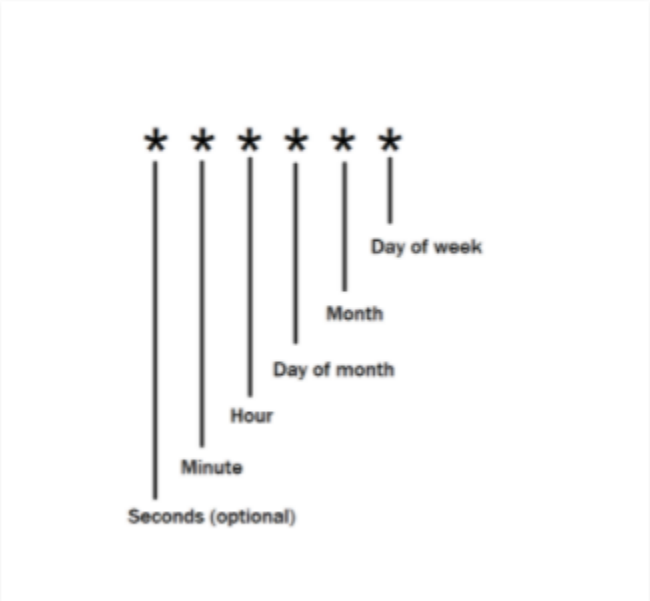
'server/thumbnails/'+stream_key+'.png',

];

spawn(cmd, args, {

```
detached: true,
stdio: 'ignore'
}).unref();
};
```

## Time formatting for Cron jobs:




```
const job = new CronJob('*/5 * * * * *', function () {
request
.get('http://127.0.0.1:' + port + '/api/streams', function (error, response, body) {
let streams = JSON.parse(body);
if (typeof (streams['live'] !== undefined)) {
let live_streams = streams['live'];
for (let stream in live_streams) {
if (!live_streams.hasOwnProperty(stream)) continue;
helpers.generateStreamThumbnail(stream);
}
}
});
}, null, true);
```

# Conclusions

Node js Live stream App is based on the Authentication before we dive into App where user will put credentials and register it self and then log In to this app.

Where user can start live stream using any free open source video streaming Application. Like OBS Studio which used in this project.