**MERN JWTs, Axios, Splitting Code, and Mongoose**

These instructions are for Linux, specifically Ubuntu was used.

This assumes that you have already completed sections A and B

**Split Code**

In the cards directory create a file named api.js.

In server.js take the 3 API endpoints. Copy them, past them into api.js, and then comment them out.

Add the exports into api.js as follows:

require('express');

require('mongodb');

exports.setApp = function ( app, client )

{

app.post('/api/addcard', async (req, res, next) =>

{

// incoming: userId, color

// outgoing: error

const { userId, card } = req.body;

const newCard = {Card:card,UserId:userId};

var error = '';

try

{

const db = client.db('COP4331Cards');

const result = db.collection('Cards').insertOne(newCard);

}

catch(e)

{

error = e.toString();

}

cardList.push( card );

var ret = { error: error };

res.status(200).json(ret);

});

app.post('/api/login', async (req, res, next) =>

{

// incoming: login, password

// outgoing: id, firstName, lastName, error

var error = '';

const { login, password } = req.body;

const db = client.db('COP4331Cards');

const results = await db.collection('Users').find({Login:login,Password:password}).toArray();

var id = -1;

var fn = '';

var ln = '';

if( results.length > 0 )

{

id = results[0].UserID;

fn = results[0].FirstName;

ln = results[0].LastName;

}

var ret = { id:id, firstName:fn, lastName:ln, error:''};

res.status(200).json(ret);

});

app.post('/api/searchcards', async (req, res, next) =>

{

// incoming: userId, search

// outgoing: results[], error

var error = '';

const { userId, search } = req.body;

var \_search = search.trim();

const db = client.db('COP4331Cards');

const results = await db.collection('Cards').find({"Card":{$regex:\_search+'.\*', $options:'i'}}).toArray();

var \_ret = [];

for( var i=0; i<results.length; i++ )

{

\_ret.push( results[i].Card );

}

var ret = {results:\_ret, error:error};

res.status(200).json(ret);

});

}

In server.js add the following (I put this right below the database connection code.):

var api = require('./api.js');

api.setApp( app, client );

Now remove the API code from server.js

Create Path.js in src/components

const app\_name = 'cop4331-5.com'

exports.buildPath =

function buildPath(route:string) : string

{

if (process.env.NODE\_ENV != 'development')

{

return 'http://' + app\_name + ':5000/' + route;

}

else

{

return 'http://localhost:5000/' + route;

}

}

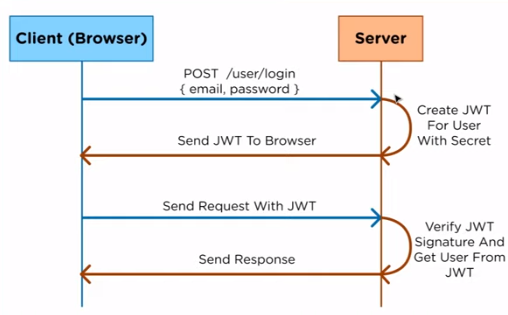
Replace (in Login.tsx and CardUI.tsx) the calls to build path.

import { buildPath } from './Path';

const response = await fetch(buildPath('api/login'),

{method:'POST',body:js,headers:{'Content-Type': 'application/json'}});

**JWTs**



From cmd line:

sudo npm install jsonwebtoken (in root and frontend)

sudo npm install dotenv 🡨 you may have done this in section B

🡨 also do this in the server directory and the

Frontend directory

Add an environmental variable named ACCESS\_TOKEN\_SECRET (make sure you do this for local in .env)

Create new file in cards directory named createJWT.js

const jwt = require("jsonwebtoken");

require("dotenv").config();

exports.createToken = function ( fn, ln, id )

{

    return \_createToken( fn, ln, id );

}

\_createToken = function ( fn, ln, id )

{

    try

    {

      const expiration = new Date();

      const user = {userId:id,firstName:fn,lastName:ln};

      const accessToken =  jwt.sign( user, process.env.ACCESS\_TOKEN\_SECRET);

      // In order to exoire with a value other than the default, use the

       // following

      /\*

      const accessToken= jwt.sign(user,process.env.ACCESS\_TOKEN\_SECRET,

         { expiresIn: '30m'} );

                       '24h'

                      '365d'

      \*/

      var ret = {accessToken:accessToken};

    }

    catch(e)

    {

      var ret = {error:e.message};

    }

    return ret;

}

exports.isExpired = function( token )

{

   var isError = jwt.verify( token, process.env.ACCESS\_TOKEN\_SECRET,

     (err, verifiedJwt) =>

   {

     if( err )

     {

       return true;

     }

     else

     {

       return false;

     }

   });

   return isError;

}

exports.refresh = function( token )

{

  var ud = jwt.decode(token,{complete:true});

  var userId = ud.payload.id;

  var firstName = ud.payload.firstName;

  var lastName = ud.payload.lastName;

  return \_createToken( firstName, lastName, userId );

}

exports.refresh = function( token )

{

var ud = jwt.decode(token,{complete:true});

var userId = ud.payload.id;

var firstName = ud.payload.firstName;

var lastName = ud.payload.lastName;

return \_createToken( firstName, lastName, userId );

}

From the cards directory: **sudo npm install jsonwebtoken**

Add to login api endpoint

    app.post('/api/login', async (req, res, next) =>

    {

      // incoming: login, password

      // outgoing: id, firstName, lastName, error

     var error = '';

      const { login, password } = req.body;

      const db = client.db('COP4331Cards');

      const results = await db.collection('Users').find({Login:login,Password:password}).toArray();

      var id = -1;

      var fn = '';

      var ln = '';

      var ret;

      if( results.length > 0 )

      {

        id = results[0].UserId;

        fn = results[0].FirstName;

        ln = results[0].LastName;

        try

        {

          const token = require("./createJWT.js");

          ret = token.createToken( fn, ln, id );

        }

        catch(e)

        {

          ret = {error:e.message};

        }

      }

      else

      {

          ret = {error:"Login/Password incorrect"};

      }

      res.status(200).json(ret);

    });

Add to addcard and searchcard api endpoints

(above)

var token = require('./createJWT.js');

const { userId, card, jwtToken } = req.body;

try

{

if( token.isExpired(jwtToken))

{

var r = {error:'The JWT is no longer valid', jwtToken: ''};

res.status(200).json(r);

return;

}

}

catch(e)

{

console.log(e.message);

}

Change the end of the methods

var refreshedToken = null;

try

{

refreshedToken = token.refresh(jwtToken);

}

catch(e)

{

console.log(e.message);

}

var ret = { error: error, jwtToken: refreshedToken };

res.status(200).json(ret);

…and

var refreshedToken = null;

try

{

refreshedToken = token.refresh(jwtToken);

}

catch(e)

{

console.log(e.message);

}

var ret = { results:\_ret, error: error, jwtToken: refreshedToken };

app.post('/api/addcard', async (req, res, next) =>

{

// incoming: userId, color

// outgoing: error

const { userId, card, jwtToken } = req.body;

try

{

if( token.isExpired(jwtToken))

{

var r = {error:'The JWT is no longer valid', jwtToken: ''};

res.status(200).json(r);

return;

}

}

catch(e)

{

console.log(e.message);

}

const newCard = {Card:card,UserId:userId};

var error = '';

try

{

const db = client.db();

const result = db.collection('Cards').insertOne(newCard);

}

catch(e)

{

error = e.toString();

}

var refreshedToken = null;

try

{

refreshedToken = token.refresh(jwtToken);

}

catch(e)

{

console.log(e.message);

}

var ret = { error: error, jwtToken: refreshedToken };

res.status(200).json(ret);

});

app.post('/api/searchcards', async (req, res, next) =>

{

// incoming: userId, search

// outgoing: results[], error

var error = '';

const { userId, search, jwtToken } = req.body;

try

{

if( token.isExpired(jwtToken))

{

var r = {error:'The JWT is no longer valid', jwtToken: ''};

res.status(200).json(r);

return;

}

}

catch(e)

{

console.log(e.message);

}

var \_search = search.trim();

const db = client.db();

const results = await db.collection('Cards').find({"Card":{$regex:\_search+'.\*', $options:'i'}}).toArray();

var \_ret = [];

for( var i=0; i<results.length; i++ )

{

\_ret.push( results[i].Card );

}

var refreshedToken = null;

try

{

refreshedToken = token.refresh(jwtToken);

}

catch(e)

{

console.log(e.message);

}

var ret = { results:\_ret, error: error, jwtToken: refreshedToken };

res.status(200).json(ret);

});

In the src directory add a file names tokenStorage.js

export function storeToken( tok:any ) : any

{

    try

    {

      localStorage.setItem('token\_data', tok.accessToken);

    }

    catch(e)

    {

      console.log(e);

    }

}

export function retrieveToken() : any

{

    var ud;

    try

    {

      ud = localStorage.getItem('token\_data');

    }

    catch(e)

    {

      console.log(e);

    }

    return ud;

}

Install react-jwt – sudo npm install react-jwt

CardUI.tsx

import React, { useState } from 'react';

import { buildPath } from './Path';

import { retrieveToken, storeToken } from '../tokenStorage';

function CardUI()

{

    const [message,setMessage] = useState('');

    const [searchResults,setResults] = useState('');

    const [cardList,setCardList] = useState('');

    const [search,setSearchValue] = React.useState('');

    const [card,setCardNameValue] = React.useState('');

    var \_ud = localStorage.getItem('user\_data');

    var ud = JSON.parse(String(\_ud));

    var userId = ud.id;

//    var firstName = ud.firstName;

//    var lastName = ud.lastName;

    async function addCard(e:any) : Promise<void>

    {

        e.preventDefault();

        var obj = {userId:userId,card:card,jwtToken:retrieveToken()};

        var js = JSON.stringify(obj);

        try

        {

            const response = await fetch(buildPath('api/addcard'),

            {method:'POST',body:js,headers:{'Content-Type': 'application/json'}});

            let txt = await response.text();

            let res = JSON.parse(txt);

            if( res.error.length > 0 )

            {

                setMessage( "API Error:" + res.error );

            }

            else

            {

                setMessage('Card has been added');

                storeToken( res.jwtToken );

            }

        }

        catch(error:any)

        {

            setMessage(error.toString());

        }

    };

    async function searchCard(e:any) : Promise<void>

    {

        e.preventDefault();

        var obj = {userId:userId,search:search,jwtToken:retrieveToken()};

        var js = JSON.stringify(obj);

        try

        {

            const response = await fetch(buildPath('api/searchcards'),

            {method:'POST',body:js,headers:{'Content-Type': 'application/json'}});

            let txt = await response.text();

            let res = JSON.parse(txt);

            let \_results = res.results;

            let resultText = '';

            for( let i=0; i<\_results.length; i++ )

            {

                resultText += \_results[i];

                if( i < \_results.length - 1 )

                {

                    resultText += ', ';

                }

            }

            setResults('Card(s) have been retrieved');

            storeToken( res.jwtToken );

            setCardList(resultText);

        }

        catch(error:any)

        {

            alert(error.toString());

            setResults(error.toString());

        }

    };

    function handleSearchTextChange( e: any ) : void

    {

        setSearchValue( e.target.value );

    }

    function handleCardTextChange( e: any ) : void

    {

        setCardNameValue( e.target.value );

    }

    return(

<div id="cardUIDiv">

  <br />

  Search: <input type="text" id="searchText" placeholder="Card To Search For"

    onChange={handleSearchTextChange} />

  <button type="button" id="searchCardButton" className="buttons"

    onClick={searchCard}> Search Card</button><br />

  <span id="cardSearchResult">{searchResults}</span>

  <p id="cardList">{cardList}</p><br /><br />

  Add: <input type="text" id="cardText" placeholder="Card To Add"

    onChange={handleCardTextChange} />

  <button type="button" id="addCardButton" className="buttons"

    onClick={addCard}> Add Card </button><br />

  <span id="cardAddResult">{message}</span>

</div>

   );

}

export default CardUI;

Login.tsx

import React, { useState } from 'react';

import { buildPath } from './Path';

import { storeToken } from '../tokenStorage';

import { jwtDecode } from 'jwt-decode';

function Login()

{

  const [message,setMessage] = useState('');

  const [loginName,setLoginName] = React.useState('');

  const [loginPassword,setPassword] = React.useState('');

    async function doLogin(event:any) : Promise<void>

    {

        event.preventDefault();

        var obj = {login:loginName,password:loginPassword};

        var js = JSON.stringify(obj);

        try

        {

            const response = await fetch(buildPath('api/login'),

                {method:'POST',body:js,headers:{'Content-Type': 'application/json'}});

            var res = JSON.parse(await response.text());

        const { accessToken } = res;

        storeToken( res );

        const decoded = jwtDecode(accessToken);

        try

        {

          var ud = decoded;

          var userId = ud.iat;

          var firstName = ud.firstName;

          var lastName = ud.lastName;

          if( userId <= 0 )

          {

            setMessage('User/Password combination incorrect');

          }

          else

          {

            var user = {firstName:firstName,lastName:lastName,id:userId}

            localStorage.setItem('user\_data', JSON.stringify(user));

            setMessage('');

            window.location.href = '/cards';

          }

          }

          catch(e)

          {

            console.log( e );

            return;

          }

        }

        catch(error:any)

        {

            alert(error.toString());

            return;

        }

      };

    function handleSetLoginName( e: any ) : void

    {

      setLoginName( e.target.value );

    }

    function handleSetPassword( e: any ) : void

    {

      setPassword( e.target.value );

    }

    return(

      <div id="loginDiv">

        <span id="inner-title">PLEASE LOG IN</span><br />

        Login: <input type="text" id="loginName" placeholder="Username"

          onChange={handleSetLoginName} /><br />

        Password: <input type="password" id="loginPassword" placeholder="Password"

          onChange={handleSetPassword} />

        <input type="submit" id="loginButton" className="buttons" value = "Do It"

          onClick={doLogin} />

        <span id="loginResult">{message}</span>

     </div>

    );

};

export default Login;

**Axios – Optional**

From command line in the frontend directory use:

**npm install axios**

In Login.js and CardUI.js add the following:

import axios from 'axios'

Replace all network calls as follows:

bp.buildPath(‘api/login’)

bp.buildPath(‘api/addcard’)

bp.buildPath(‘api/searchcards’)

Remember: var storage = require('../tokenStorage.js');

import React, { useState } from 'react';

import axios from 'axios';

function Login()

{

var bp = require('./Path.js');

var storage = require('../tokenStorage.js');

var loginName;

var loginPassword;

const [message,setMessage] = useState('');

const doLogin = async event =>

{

event.preventDefault();

var obj = {login:loginName.value,password:loginPassword.value};

var js = JSON.stringify(obj);

var config =

{

method: 'post',

url: bp.buildPath('api/login'),

headers:

{

'Content-Type': 'application/json'

},

data: js

};

axios(config)

.then(function (response)

{

var res = response.data;

if (res.error)

{

setMessage('User/Password combination incorrect');

}

else

{

storage.storeToken(res);

var jwt = require('jsonwebtoken');

var ud = jwt.decode(storage.retrieveToken(),{complete:true});

var userId = ud.payload.userId;

var firstName = ud.payload.firstName;

var lastName = ud.payload.lastName;

var user = {firstName:firstName,lastName:lastName,id:userId}

localStorage.setItem('user\_data', JSON.stringify(user));

window.location.href = '/cards';

}

})

.catch(function (error)

{

console.log(error);

});

}

return(

<div id="loginDiv">

<span id="inner-title">PLEASE LOG IN</span><br />

<input type="text" id="loginName" placeholder="Username" ref={(c) => loginName = c} /><br />

<input type="password" id="loginPassword" placeholder="Password" ref={(c) => loginPassword = c} /><br />

<input type="submit" id="loginButton" class="buttons" value = "Do It"

onClick={doLogin} />

<span id="loginResult">{message}</span>

</div>

);

};

export default Login;

**Mongoose – Optional**

From command line use:

**npm install mongoose, npm install mongoose-int32**

Change the connection code

From

const MongoClient = require('mongodb').MongoClient;

const url = process.env.MONGODB\_URI;

const client = new MongoClient(url);

client.connect();

To:

const url = process.env.MONGODB\_URI;

const mongoose = require("mongoose");

mongoose.connect(url)

.then(() => console.log("Mongo DB connected"))

.catch(err => console.log(err));

Replace

api.setApp( app, client );

with

api.setApp( app, mongoose );

Create a models folder in the cards directory. Inside create user.js and card.js

user.js

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

//Create Schema

const UserSchema = new Schema({

UserId: {

type: Number

},

FirstName: {

type: String,

required: true

},

LastName: {

type: String,

required: true

},

Login: {

type: String,

required: true

},

Password: {

type: String,

required: true

}

});

module.exports = user = mongoose.model("Users", UserSchema);

card.js

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

// Create Schema

const CardSchema = new Schema({

UserId: {

type: Number

},

Card: {

type: String,

required: true

}

});

module.exports = Card = mongoose.model('Cards', CardSchema);

In api.js

//load user model

const User = require("./models/user.js");

//load card model

const Card = require("./models/card.js");

Change the database code to the following:

**addcard**

//const newCard = { Card: card, UserId: userId };

const newCard = new Card({ Card: card, UserId: userId });

var error = '';

try

{

// const db = client.db();

// const result = db.collection('Cards').insertOne(newCard);

newCard.save();

}

catch (e)

{

error = e.toString();

}

**login**

const { login, password } = req.body;

// const db = client.db();

// const results = await db.collection('Users').find({Login:login,Password:password}).toArray();

const results = await User.find({ Login: login, Password: password });

**searchcard**

var \_search = search.trim();

// const db = client.db();

// const results = await db.collection('Cards').find({ "Card": { $regex: \_search + '.\*', $options: 'r' } }).toArray();

const results = await Card.find({ "Card": { $regex: \_search + '.\*', $options: 'r' } });

In CardUI.js change line:

var userId = ud. id;

to the following:

var userId = ud. userId;