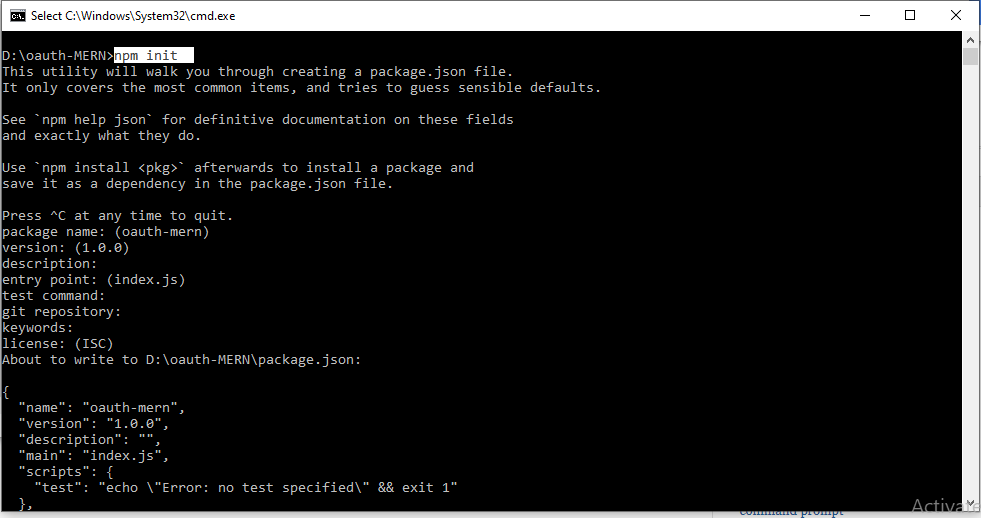
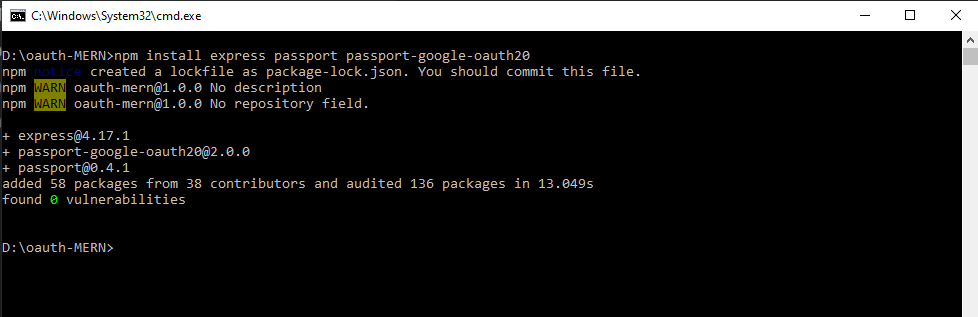
**Google OAUTH LOGIN NODE JS**

STEP1 – Create a directory 🡪 open a command line

npm init 🡪 enter 🡪 enter 🡪 develop a package.json file



STEP2 – npm install express passport passport-google-oauth20



Npm install nodemon

Google Console Developer

<https://www.loom.com/share/f12ccf6ce9b4485191c91e86d0ccf086>

<https://www.loom.com/share/aa2591345fd945948b1d4360605fcbc6>

**Basic Setup Node JS**

**Install 2 packages –**

Npm install passport

Npm install passport-google-oauth20

**Node END**

Index.js (newly created)

const express = require("express");

const passport = require("passport");

const GOOGLE\_STRATEGY = require("passport-google-oauth20").Strategy;

const app = express();

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

app.get("/", (req, res) => {

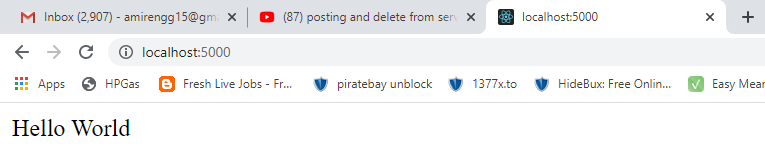
  res.send("Hello World");

});

app.listen(port, () => {

  console.log(`Node server started in port ${port}`);

});



Setting up Google Console Developers **(Google Auth End)**

<https://www.loom.com/share/f12ccf6ce9b4485191c91e86d0ccf086>

<https://www.loom.com/share/f6741234dcb64b079c9346bc3485de16> (See this one first)

NOTE: In Google login App :

Create Credentials 🡪 Oauth client ID 🡪 Fill form

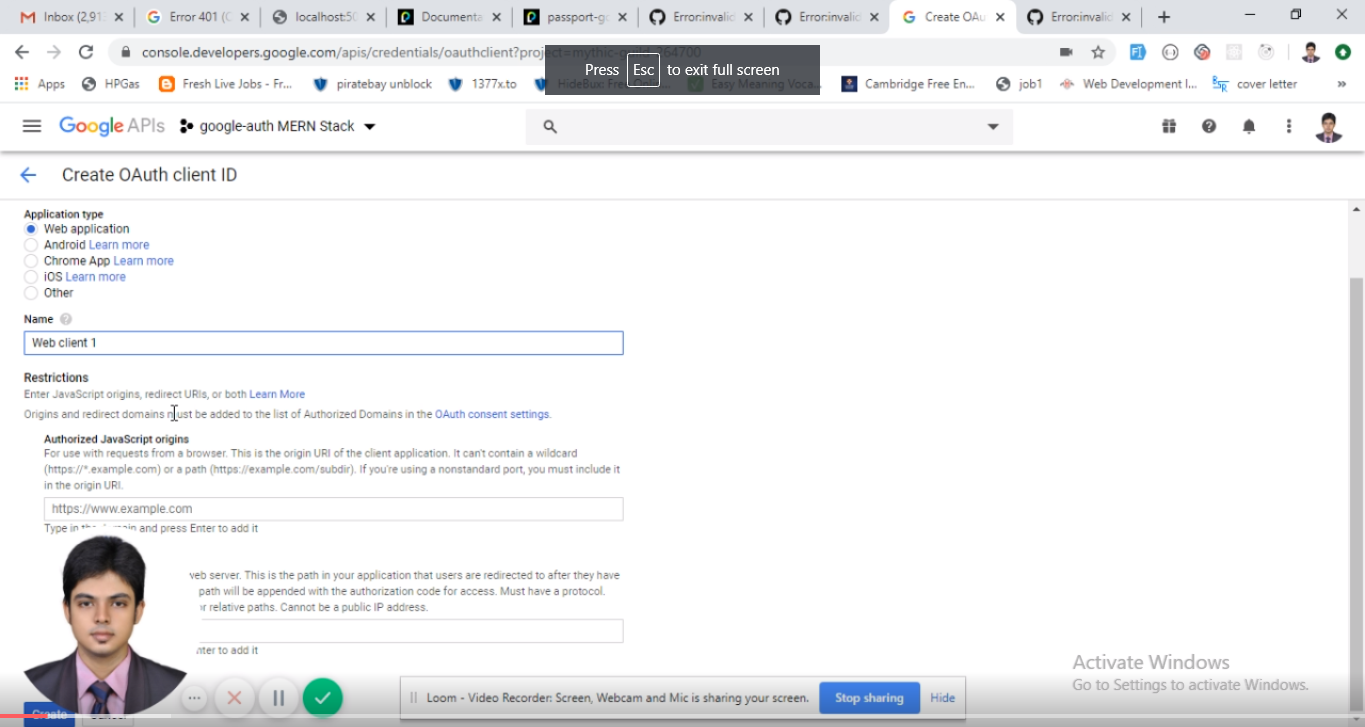
Select

App type – Web Application

Name – Your App name

**Authorized JS Origins – Your project domain eg.** [**http://locahost:5000**](http://locahost:5000)

**Authorized Redirect URI –** [**http://localhost:5000/auth/google**](http://localhost:5000/auth/google)

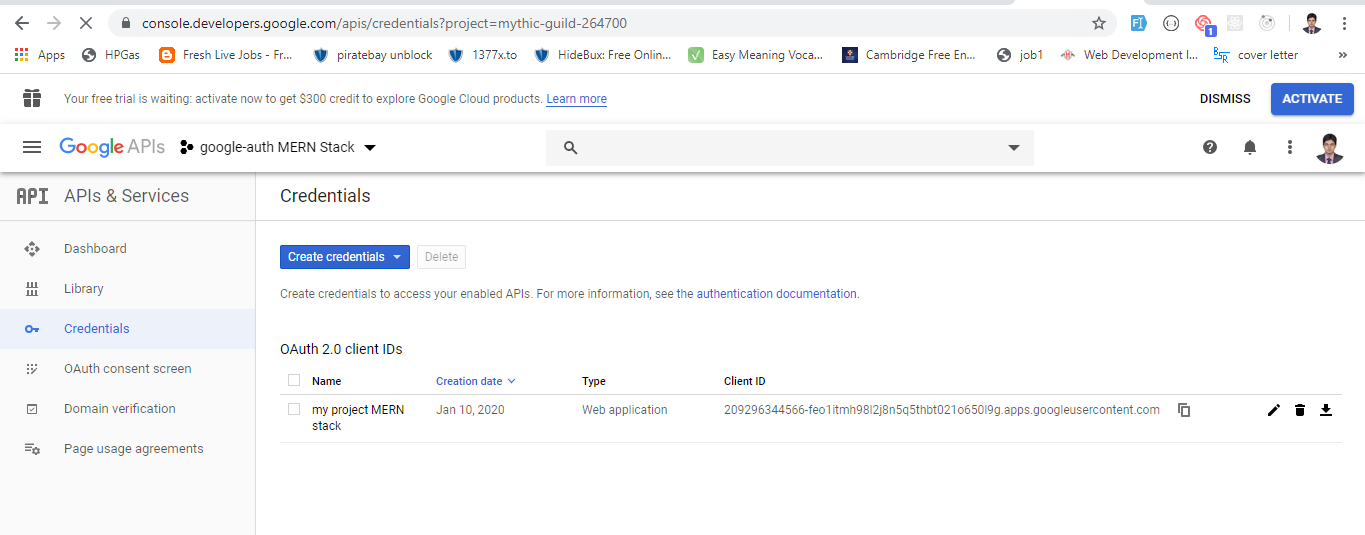


After signing in we will get client id, secret key and callback url we set there

Eg.

**client id = 209296344566-feo1itmh98l2j8n5q5thbt021o650l9g.apps.googleusercontent.com**

**client secret = c6-\_Drf55EjGRHKBmDsSOHX4**



Passport JS Doc- <http://www.passportjs.org/packages/passport-google-oauth20/>

Here we will use passport to authenticate

index.js

const express = require("express");

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const keys = require("./config/keys"); // client id, secret id

const app = express();

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

// client id = 209296344566-feo1itmh98l2j8n5q5thbt021o650l9g.apps.googleusercontent.com

// client secret = c6-\_Drf55EjGRHKBmDsSOHX4

passport.use(

  new GoogleStrategy(

    {

      clientID: keys.googleClientID,

      clientSecret: keys.googleClientSecret,

      callbackURL: "/auth/google/callback" // below create node routes

    },

    (accessToken, refreshToken, profile, done) => {

      console.log("Access Token", accessToken);

      console.log("Refresh Token", refreshToken);

      console.log("profile", profile);

      console.log("done", done);

    }

  )

);

app.get("/", (req, res) => {

  res.status(200).json({

    googleClientID: keys.googleClientID,

    googleCLientSecret: keys.googleClientSecret

  });

});

// node routes

app.get( // REQUEST

  "/auth/google",

  passport.authenticate("google", { scope: ["profile", "email"] })

);

// RESPONSE

app.get("/auth/google/callback", passport.authenticate("google"));

app.listen(port, () => {

  console.log(`Node server started in port ${port}`);

});

* We get data in terminal



* Now we will refractor code.
* We will use some dev keys in development and fetch key from heroku in production

1. Refractoring Keys for dev and prod

project/config/dev.js

module.exports = {

  googleClientID:

    "209296344566-fglo7jedn2nqjmb8me6o0kg5jel4qjan.apps.googleusercontent.com",

  googleClientSecret: "V4GhGJcLXCvtgSqdiOCR93Zm"

};

project/config/prod.js (here the keys are set in heroku)

module.exports = {

  googleClientID: process.env.GOOGLE\_CLIENT\_ID,

  googleClientSecret: process.env.GOOGLE\_CLIENT\_SECRET

};

project/config/keys.js

if (process.env.NODE\_ENV === "production") {

  module.exports = require("./prod");

} else {

  module.exports = require("./dev");

}

1. Refractoring Passport JS Codes

We will keep all passport js codes in one directory

Project/services/passport.js

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const keys = require("../config/keys");

passport.use(

  new GoogleStrategy(

    {

      clientID: keys.googleClientID,

      clientSecret: keys.googleClientSecret,

      callbackURL: "/auth/google/callback"

    },

    (accessToken, refreshToken, profile, done) => {

      console.log("Access Token", accessToken);

      console.log("Refresh Token", refreshToken);

      console.log("profile", profile);

      console.log("done", done);

    }

  )

)

Index.js

const express = require("express");

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const keys = require("./config/keys");

const app = express();

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

require("./services/passport");

require("./routes/authRoute")(app);

app.listen(port, () => {

  console.log(`Node server started in port ${port}`);

});

routes/authRoute.js

const express = require("express");

const passport = require("passport");

const keys = require("../config/keys");

module.exports = app => {

  app.get("/", (req, res) => {

    res.status(200).json({

      googleClientID: keys.googleClientID,

      googleCLientSecret: keys.googleClientSecret

    });

  });

  app.get(

    "/auth/google",

    passport.authenticate("google", { scope: ["profile", "email"] })

  );

  app.get("/auth/google/callback", passport.authenticate("google"));

};

1. Mongo DB (DATABASE)

Install the driver package for the node

npm install mongoose

We will use mongodb Atlas Login/Signup

Click Create Project 🡪 Create Cluster 🡪 Database Access (create user) 🡪 Network Access (select IP from anywhere) 🡪 Cluster(click connect 🡪 Connect your app 🡪 Copy the MongoDB url)

Follow this video for Mongo DB SRV

<https://www.loom.com/share/57d80d7684894d84b590fc75c1a75e29>

Eg.

**mongodb+srv://amirengg15:<password>@cluster0-cwvsk.mongodb.net/test?retryWrites=true&w=majority**

* Create directory models/User.js (Schema)

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const userSchema = new Schema({

  googleId: String

});

mongoose.model("users", userSchema); // setting Schema – users(table name)

* Database connect

index.js (database connect)

const express = require("express");

const app = express();

const mongoose = require("mongoose");

const keys = require("./config/keys");

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

require("./models/User"); // Note model must be imported before passport

require("./services/passport");

mongoose.connect(

  keys.mongoURI,

  { useNewUrlParser: true, useUnifiedTopology: true },

  () => {

    console.log("connnected to mongo DB");

  }

);

require("./routes/authRoute")(app);

app.listen(port, () => {

  console.log(`Node server started in port ${port}`);

});

* Insert to db code. This we will write in passport code as we have access from google profile

Services/passport.js

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const keys = require("../config/keys");

const mongoose = require("mongoose");

const User = mongoose.model("users"); // getting Schema

// module.exports = {

passport.use(

  new GoogleStrategy(

    {

      clientID: keys.googleClientID,

      clientSecret: keys.googleClientSecret,

      callbackURL: "/auth/google/callback"

    },

    (accessToken, refreshToken, profile, done) => {

      // console.log(profile);

      // profile has all google login data

      /\* === DATABASE CHECK PRE EXIST AND INSERT QUERY: START ====  \*/

      // check if user id already inserted

      User.findOne({ googleId: profile.id }).then(existingUser => {

        if (existingUser) {

          done(null, existingUser);

        } else {

          // new user case

          // insert new user id

          new User({ googleId: profile.id }).save().then(user => {

            done(null, user);

          });

        }

      });

      /\* === DATABASE CHECK PRE EXIST AND INSERT QUERY: END ====  \*/

    }

  )

);

// };

Video - <https://www.loom.com/share/a0f05aba98384a4a96eef46f9b888d10>

Serialize/Deserialize User

-🡪 Now one more error comes when we sign in from Google API. It runs infinite times. Database entry is successful. For this we have to write passport serialize deserialize

🡪 Just think this as extra security when foreign requests google to signin it asks why should I sign in. Give me the google id you got from sign in – SERIALIZE

* Now Google gives all data with respect to that google id - DESERIALIZE

Project/services/passport.js

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const keys = require("../config/keys");

const mongoose = require("mongoose");

const User = mongoose.model("users"); // getting Schema

passport.serializeUser((user, done) => {

  done(null, user.id);

});

passport.deserializeUser((id, done) => {

  User.findById(id).then(user => {

    done(null, user);

  });

});

passport.use(

  new GoogleStrategy(

    {

      clientID: keys.googleClientID,

      clientSecret: keys.googleClientSecret,

      callbackURL: "/auth/google/callback"

    },

    (accessToken, refreshToken, profile, done) => {

      // console.log(profile);

      // profile has all google login data

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: START =====  \*/

      // check if user id already inserted

      User.findOne({ googleId: profile.id }).then(existingUser => {

        if (existingUser) {

          done(null, existingUser);

        } else {

          // new user case

          // insert new user id

          new User({ googleId: profile.id }).save().then(user => {

            done(null, user);

          });

        }

      });

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: END =========  \*/

    }

  )

);

Install a module cookie-session: npm i cookie-session

This cookie session is sent in header as request in google session where ever passport uses and destroys in logout case

Index.js

const express = require("express");

const app = express();

const mongoose = require("mongoose");

const keys = require("./config/keys");

let cookieSession = require("cookie-session");

const passport = require("passport");

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

require("./models/User"); // Note model must be imported before passport

require("./services/passport");

// connect with mongo db

mongoose.connect(

  keys.mongoURI,

  { useNewUrlParser: true, useUnifiedTopology: true },

  () => {

    console.log("connnected to mongo DB");

  }

);

/\* ====== Creating Cookie Key and link with Passport JS: Start ========  \*/

We have connected cookie session with the keys which will auto expire

app.use(

  cookieSession({

    maxAge: 30 \* 86400 \* 1000, // expire in 30 days(milli seconds)

    keys: [keys.cookieKey]

  })

);

app.use(passport.initialize());

app.use(passport.session());

/\* ======= Creating Cookie Key and link with Passport JS: End =========  \*/

require("./routes/authRoute")(app);

app.listen(port, () => {

  console.log(`Node server started in port ${port}`);

});

routes/authRoutes.js

const express = require("express");

const passport = require("passport");

const keys = require("../config/keys");

module.exports = app => {

  app.get("/", (req, res) => {

    res.status(200).json({

      googleClientID: keys.googleClientID,

      googleCLientSecret: keys.googleClientSecret

    });

  });

  app.get(

    "/auth/google",

    passport.authenticate("google", { scope: ["profile", "email"] })

  );

  app.get("/auth/google/callback", passport.authenticate("google"));

  app.get("/current\_user", (req, res) => {

    res.send(req.user); // req.user has google data after session created

  });

  app.get("/api/logout", (req, res) => {

    req.logout();

    res.send(req.user);

  });

};

React End

* Install react inside node directory – npx create-react-app **client**

Create the basic layout navigation links using react-router-dom

Setup proxy from React JS to Node JS

In React JS (project/client)

* npm install http-proxy-middleware
* Inside src folder create setupProxy.js(auto called)

const proxy = require("http-proxy-middleware");

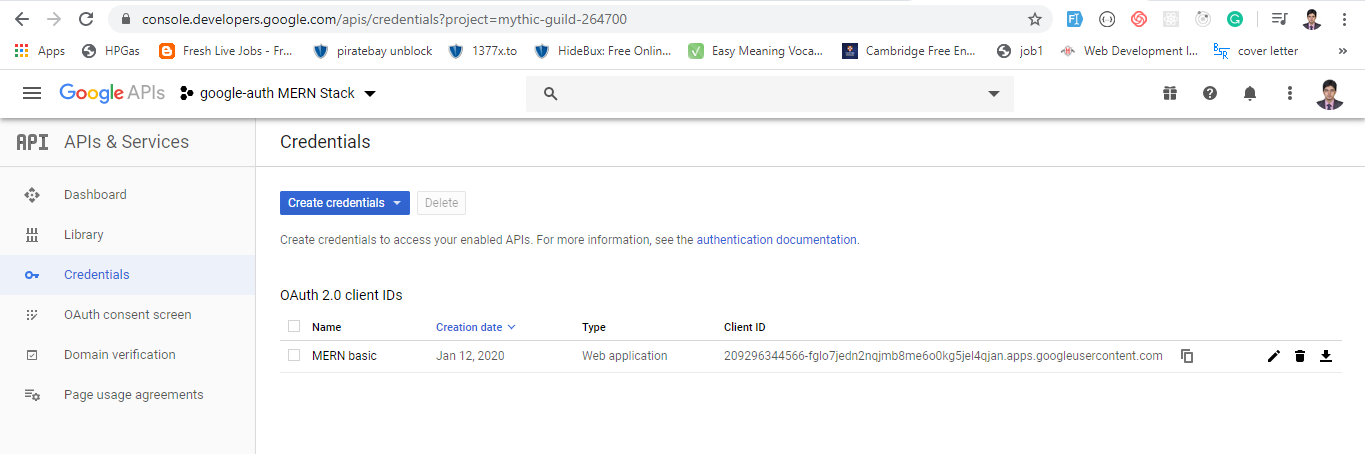
// write all your routes for sending to node

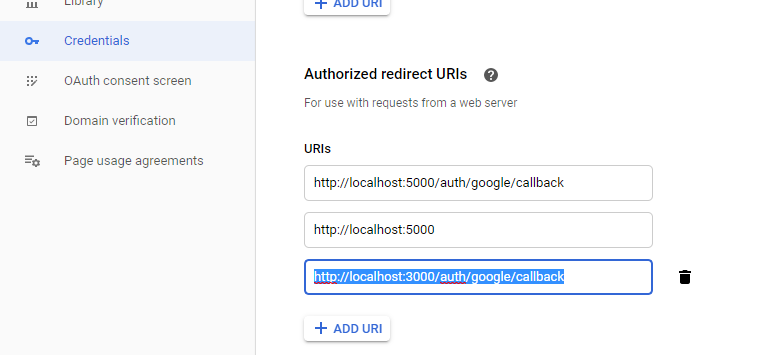
module.exports = function(app) {

  app.use(proxy("/auth/google", { target: "http://localhost:5000/" }));

};

* Last if you get google error callback localhost:3000/callback – add this in google console developers





Setting up UI in React JS

Project/client – npm install react-router-dom

* Materialize CSS – keep in client/public/index.js/

<!-- Compiled and minified CSS -->

     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">

     <!-- Compiled and minified JavaScript -->

     <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>

Adding Redux to the Script (Redux)

* In redux store basically we will check whether user is logged in or not.

i.e. we will check node current route

i.e. authRoute.js –

app.get("/api/current\_user", (req, res) => {

    res.send(req.user);

  });

* Connecting Redux with React

client/index.js

import React from "react";

import ReactDOM from "react-dom";

import "./index.css";

import App from "./App";

import \* as serviceWorker from "./serviceWorker";

// Redux Connectivity

import { Provider } from "react-redux";

import thunk from "redux-thunk";

import { createStore, applyMiddleware, combineReducers } from "redux";

import { composeWithDevTools } from "redux-devtools-extension";

import { authReducer } from "./reducers/authReducer";

// Combine Reducer

const rootReducer = combineReducers({

  auth: authReducer

});

// Passing reducer to store and using chrome dev tools + async thunk

const store = createStore(

  rootReducer,

  composeWithDevTools(applyMiddleware(thunk))

);

// Wrapping component with Provider

ReactDOM.render(

  <Provider store={store}>

    <App />

  </Provider>,

  document.getElementById("root")

);

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

* Store page

reducers/authReducer.js

// auth Reducer will basically check that whether user is logged in or not - use node's /api/current user route

export const authReducer = (state = null, action) => {

  console.log(action.payload); // this lines is returned from action creator

  switch (action.type) {

    case "GET\_USER":

      return action.payload;

    default:

      return state;

  }

};

* App Component subscribing to Store

Client/Components/App.js

import React, { useEffect } from "react";

import "./App.css";

import Header from "./components/Header";

import { BrowserRouter, Switch, Route } from "react-router-dom"; // for react router

import Home from "./components/Home";

import Profile from "./components/Profile";

// App subscribing to redux store

import { connect } from "react-redux";

import { fetchUserAction } from "./actions/myaction";

  // App component

function App(props) {

  // this is similar to componentDidMount of class component - here we call mapDispatchToProps

  useEffect(() => {

    props.fetchUser();

  }, []);

  return (

// React router-setting react routes for component navigation

    <BrowserRouter className="App">

      <Header />

      <Switch>

        <Route path="/" component={Home} exact></Route>

        <Route path="/profile" component={Profile}></Route>

      </Switch>

    </BrowserRouter>

  );

}

const mapDispatchToProps = dispatch => {

  return {

    fetchUser: () => {

      dispatch(fetchUserAction());

    }

  };

};

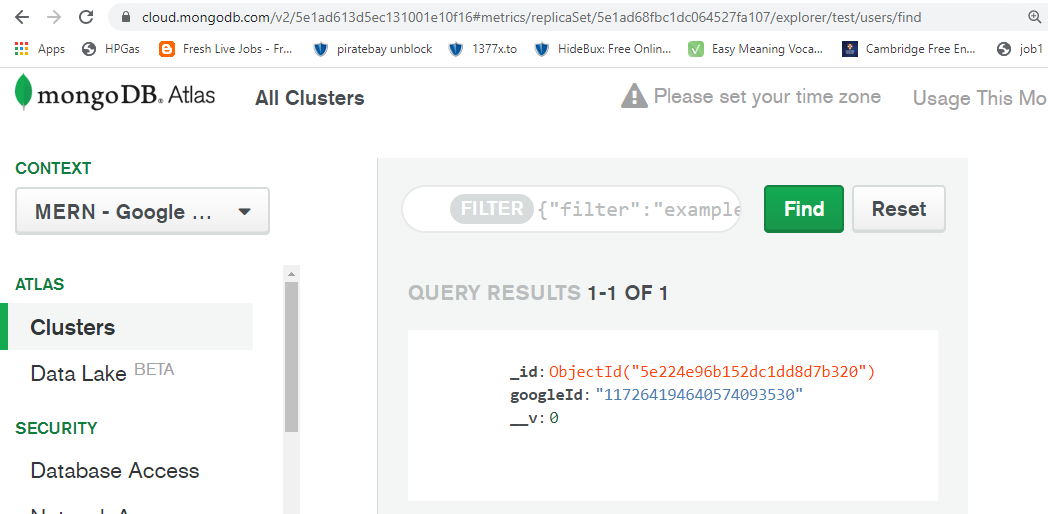
// para 1 is null as we don't need mapStateToProps

export default connect(null, mapDispatchToProps)(App);

* Action Creator

This will hold type and payload. This will call /api/current\_user proxy

As we know we have set proxy for client/src/setupProxy.js to go to node’s 5000 port instead of react’s 3000.



Npm install axios

Here we have also used Axios for connecting to node instead of fetch (basically we skipped .json line of fetch)

Client/actions/myaction.js

We get data of logged in user – initially empty, after login with google will get google id, \_id etc

import axios from "axios";

export const fetchUserAction = () => {

  return dispatch => {

    axios.get("/api/current\_user").then(res => {

      dispatch({ type: "GET\_USER", payload: res.data });

    });

    // fetch("/api/current\_user")

    // .then(res => res.json())

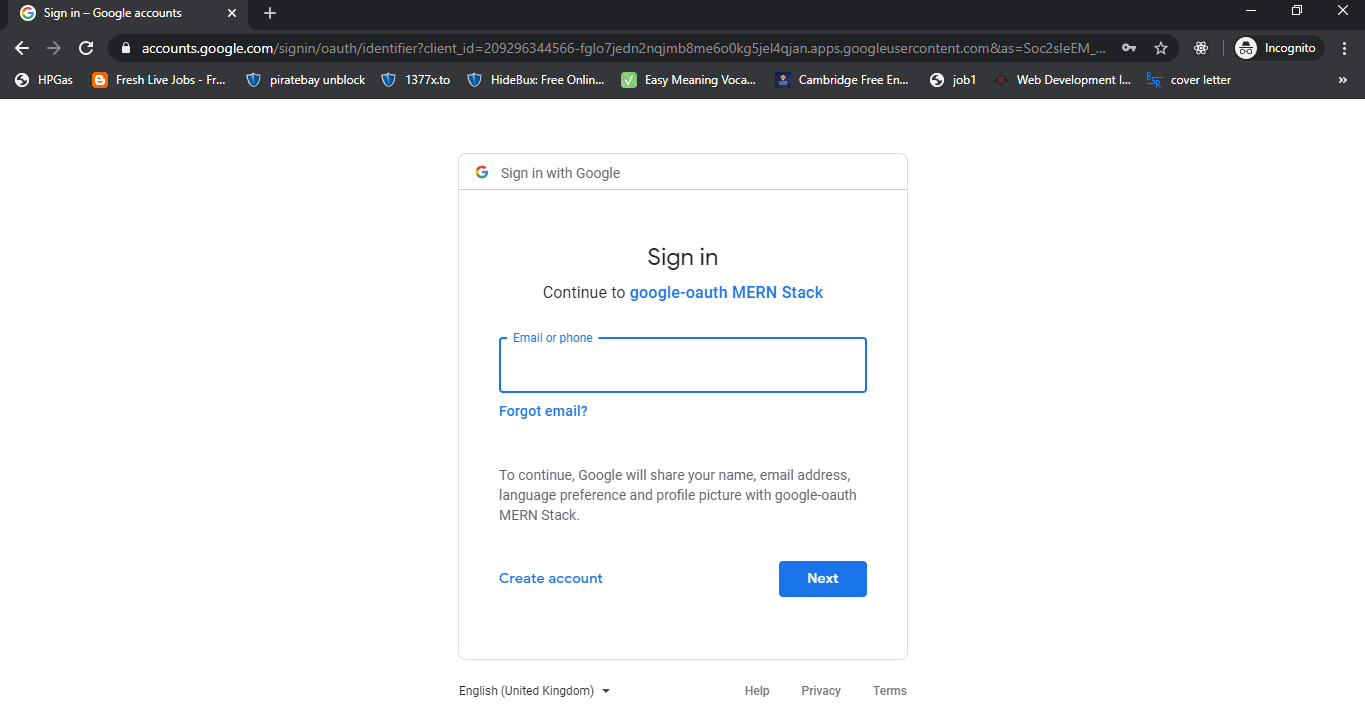
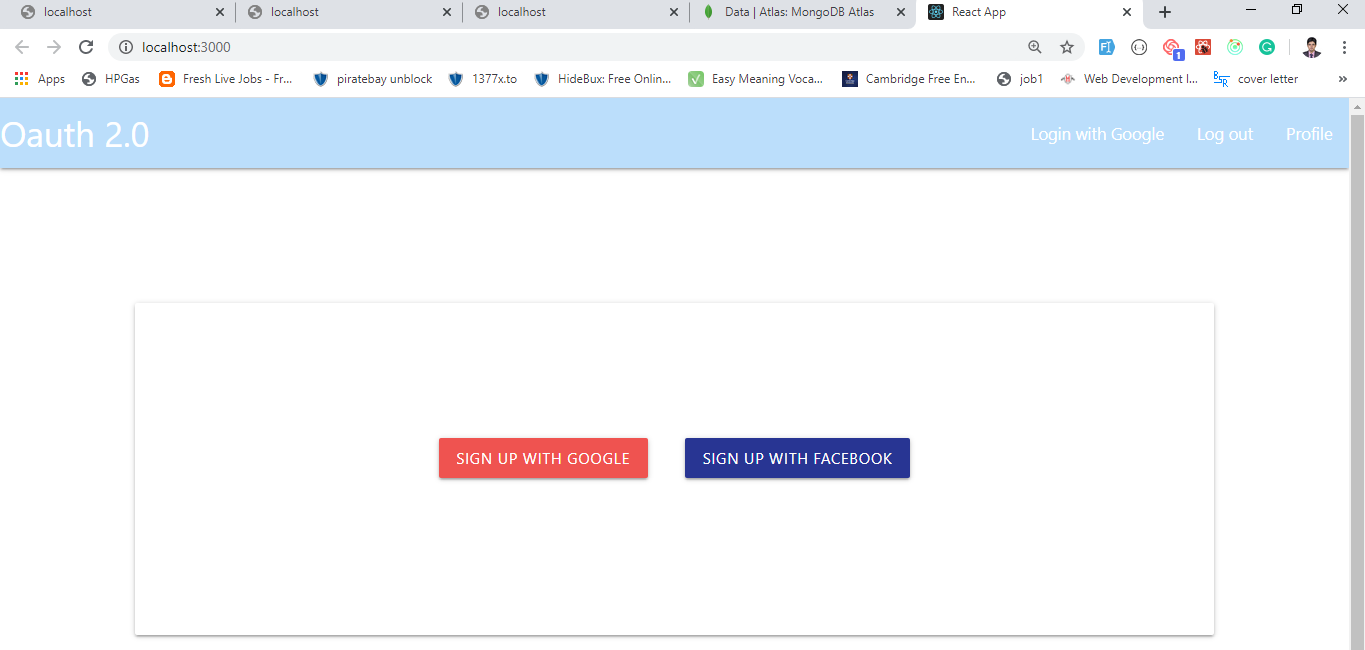
    // .then(data => {

    // });

  };

};

* Action creator returns to reducer data as action.payload



* Now we will work on header links for initial case show google login link and after signin show logout and profile button and for
* For this we have written code in Header.js component
* We have used redux here. Basically from redux store we get data using mapStateToProps (i.e. auth), if auth is false – signup case else signin case

client/src/components/Header.js (React header)

import React from "react";

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

// connecting to redux store

import { connect } from "react-redux";

const Header = props => {

  // logic for logout case show signup and login case show logout and profile tab

  const renderContent = () => {

    switch (props.user) {

      case null:         // when net is slow - node url in process

 return (

          <li>

            <a href="/">loading..</a>

          </li>

        );

      case false: // when not log in case

        return (

          <li>

            <a href="/auth/google">Login with Google</a>

          </li>

        );

      default: // log in case

        return (

          <React.Fragment>

            <li>

              <a href="/api/logout">Log out</a>

            </li>

            <li>

              <Link to="/profile">Profile</Link>

            </li>

          </React.Fragment>

        );

    }

  };

  return (

    <nav>

      <div className="nav-wrapper  blue lighten-4">

        <Link to="/" className="brand-logo">

          Oauth 2.0

        </Link>

        <ul id="nav-mobile" className="right">

          {renderContent()}

          {/\* <li>

            <a href="/auth/google">Login with Google</a>

          </li>

          <li>

            <Link to="/api/logout">Log out</Link>

          </li>

          <li>

            <Link to="/profile">Profile</Link>

          </li> \*/}

        </ul>

      </div>

    </nav>

  );

};

// getting data of auth from redux store

const mapStateToProps = state => {

  return {

    user: state.auth // this auth is in store (see client/index.js or chrome in chrome dev tools)

  };

};

export default connect(mapStateToProps)(Header);

project/authRoute.js (node route)

app.get(

    "/auth/google/callback",

    passport.authenticate("google"),

    (req, res) => {

      res.redirect("/profile"); // login case - redirect to profile page

    }

  );

app.get("/api/logout", (req, res) => {

    req.logout();

    res.redirect("/"); // log out case

    // res.send(req.user);

  });

Making Profile Page

* Firstly in mongoose schema we will add username and picture for saving in mongo DB

project/models/User.js

const mongoose = require("mongoose"); // this is schema page, we will use this at the time of db insert/fetch/delete etc

const Schema = mongoose.Schema;

const userSchema = new Schema({

  googleId: String,

  username: String,

  picture: String

});

mongoose.model("users", userSchema); // setting Schema

* Now in out db insert code we will update username and picture

Project/services/passport.js

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const keys = require("../config/keys");

const mongoose = require("mongoose");

const User = mongoose.model("users"); // getting Schema

/\* =================== Handeling Infinite run: Start ===================  \*/

passport.serializeUser((user, done) => {

  done(null, user.id);

});

passport.deserializeUser((id, done) => {

  User.findById(id).then(user => {

    done(null, user);

  });

});

passport.use(

  new GoogleStrategy(

    {

      clientID: keys.googleClientID,

      clientSecret: keys.googleClientSecret,

      callbackURL: "/auth/google/callback"

    },

    (accessToken, refreshToken, profile, done) => {

      console.log(profile);

      // profile has all google login data

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: START =========  \*/

      // check if user id already inserted

      User.findOne({ googleId: profile.id }).then(existingUser => {

        if (existingUser) {

          done(null, existingUser);

        } else {

          // new user case

          // insert new user id

          new User({

            googleId: profile.id,

            username: profile.displayName,

            picture: profile.\_json.picture // do console.log(profile) for google receive data

          })

            .save()

            .then(user => {

              done(null, user);

            });

        }

      });

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: END =========  \*/

    }

  )

);

Profile.js

import React, { useState, useEffect } from "react";

import { connect } from "react-redux";

const Profile = props => {

  const [userdata, setUserData] = useState({

    name: "loading",

    picture: ""

  });

  useEffect(() => {

    if (props.user) {

      setUserData({

        name: props.user.username,

        picture: props.user.picture

      });

    }

  }, []);

  // if (!props.user) {

  //   props.history.push("/");

  // }

  return (

    <div>

      <div

        className="card"

        style={{ margin: "10%", padding: "10px", textAlign: "center" }}

      >

        <h1>My Profile</h1>

        <h2>{userdata.name}</h2>

        <img

          className="circle"

          src={userdata.picture}

          style={{ width: "50%" }}

        />

      </div>

    </div>

  );

};

const mapStateToProps = state => {

  console.log("state", state.auth);

  return {

    user: state.auth || {

      \_id: "5e2f8e93dccdb213d8d7ce55",

      googleId: "117264194640574093530",

      username: "Amir Mustafa",

      picture:

        "https://lh3.googleusercontent.com/a-/AAuE7mCn-ml9gLue1SgHTHCi2ECdu\_jx1lMjKn7tNOqZ",

      \_\_v: 0

    }

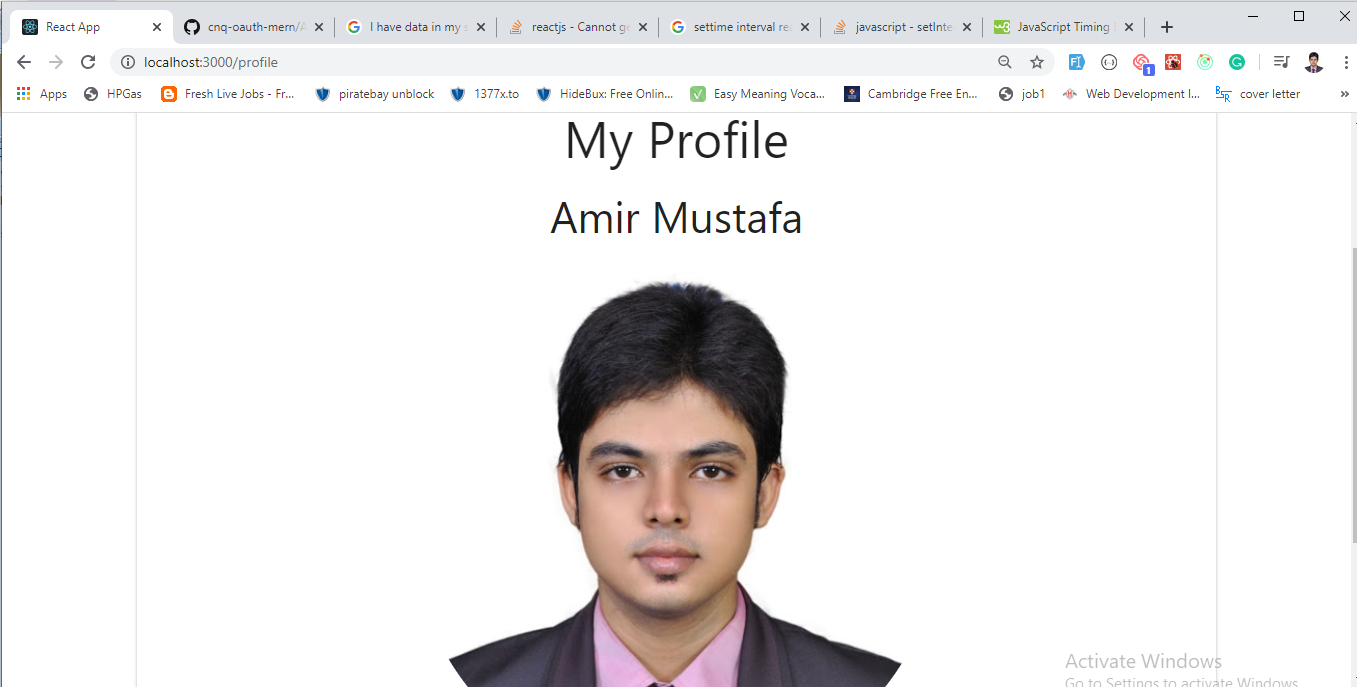
  };

};

// window.setTimeout(mapStateToProps, 5000);

export default connect(mapStateToProps)(Profile);

O/P – receiving data from passport



Login with Facebook – Passport JS

* For this we log in passport facebook package

<http://www.passportjs.org/packages/passport-facebook/>

* Install passport facebook package – npm install passport-facebook
* Get client id, client secret key from facebook developers

Video(get key from fb developers) - <https://www.loom.com/share/80a8ca2b0d6b4290a51b308412c7022f>

Project/services/passport.js

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

const FacebookStrategy = require("passport-facebook").Strategy;

const keys = require("../config/keys");

const mongoose = require("mongoose");

const User = mongoose.model("users"); // getting Schema

/\* =================== Handeling Infinite run: Start ===================  \*/

passport.serializeUser((user, done) => {

  done(null, user.id);

});

passport.deserializeUser((id, done) => {

  User.findById(id).then(user => {

    done(null, user);

  });

});

// For Google

passport.use(

  new GoogleStrategy(

    {

      clientID: keys.googleClientID,

      clientSecret: keys.googleClientSecret,

      callbackURL: "/auth/google/callback"

    },

    (accessToken, refreshToken, profile, done) => {

      console.log(profile);

      // profile has all google login data

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: START =========  \*/

      // check if user id already inserted

      User.findOne({ userId: profile.id }).then(existingUser => {

        if (existingUser) {

          done(null, existingUser);

        } else {

          // new user case

          // insert new user id

          new User({

            userId: profile.id,

            username: profile.displayName,

            picture: profile.\_json.picture

          })

            .save()

            .then(user => {

              done(null, user);

            });

        }

      });

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: END =========  \*/

    }

  )

);

// For facebook

passport.use(

  new FacebookStrategy(

    {

      clientID: keys.FACEBOOK\_APP\_ID,

      clientSecret: keys.FACEBOOK\_APP\_SECRET,

      callbackURL: "/auth/facebook/callback"

    },

    (accessToken, refreshToken, profile, done) => {

      console.log(profile);

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: START =========  \*/

      // check if user id already inserted

      User.findOne({ userId: profile.id }).then(existingUser => {

        if (existingUser) {

          done(null, existingUser);

        } else {

          // new user case

          // insert new user id

          new User({

            userId: profile.id,

            username: profile.displayName,

            picture: profile.\_json.picture

          })

            .save()

            .then(user => {

              done(null, user);

            });

        }

      });

      /\* ========= DATABASE CHECK PRE EXIST AND INSERT QUERY: END =========  \*/

    }

  )

);

Project/config/dev.js

module.exports = {

  googleClientID:

    "209296344566-fglo7jedn2nqjmb8me6o0kg5jel4qjan.apps.googleusercontent.com",

  googleClientSecret: "V4GhGJcLXCvtgSqdiOCR93Zm",

  mongoURI:

    "mongodb+srv://amirengg15:Amir123456@cluster0-cwvsk.mongodb.net/test?retryWrites=true&w=majority",

  cookieKey: "hsdghsdghsd",

  FACEBOOK\_APP\_ID: "2183655625275997",

  FACEBOOK\_APP\_SECRET: "ba716e187750acf9eddaf74ba9fc37dc"

};

Project/client/src/component/profile.js(page after login)

import React, { useState, useEffect } from "react";

import { connect } from "react-redux";

const Profile = props => {

  const [userdata, setUserData] = useState({

    name: "loading",

    picture: ""

  });

  useEffect(() => {

    if (props.user) {

      setUserData({

        name: props.user.username,

        picture: props.user.picture

      });

    }

  }, []);

  // if (!props.user) {

  //   props.history.push("/");

  // }

  return (

    <div>

      <div

        className="card"

        style={{ margin: "10%", padding: "10px", textAlign: "center" }}

      >

        <h1>My Profile</h1>

        <h2>{userdata.name}</h2>

        <img

          className="circle"

          src={userdata.picture}

          style={{ width: "50%" }}

        />

      </div>

    </div>

  );

};

const mapStateToProps = state => {

  console.log("state", state.auth);

  return {

    user: state.auth || {

      \_id: "5e2f8e93dccdb213d8d7ce55",

      googleId: "117264194640574093530",

      username: "Amir Mustafa",

      picture:

        "https://lh3.googleusercontent.com/a-/AAuE7mCn-ml9gLue1SgHTHCi2ECdu\_jx1lMjKn7tNOqZ",

      \_\_v: 0

    }

  };

};

// window.setTimeout(mapStateToProps, 5000);

export default connect(mapStateToProps)(Profile);

We get a Facebook image by using the Facebook Graph API.

Video - [https://www.loom.com/share/fc3744ce61444435bfd0f6ecba64054e](https://www.youtube.com/redirect?redir_token=3rFO9Nt2LR9f1tM5Gre9koh6wER8MTU4MDYxMzgwOEAxNTgwNTI3NDA4&event=comments&q=https%3A%2F%2Fwww.loom.com%2Fshare%2Ffc3744ce61444435bfd0f6ecba64054e&stzid=Ugxp67grxAWHxXsfkxF4AaABAg)

Two steps:

a. get access token from Facebook for developers

b. Copy-paste token here: [https://graph.facebook.com/me/picture?width=200&height=200&access\_token=<your\_token>](https://www.youtube.com/redirect?redir_token=3rFO9Nt2LR9f1tM5Gre9koh6wER8MTU4MDYxMzgwOEAxNTgwNTI3NDA4&event=comments&q=https%3A%2F%2Fgraph.facebook.com%2Fme%2Fpicture%3Fwidth%3D200%26height%3D200%26access_token%3D%253Cyour_token%253E&stzid=Ugxp67grxAWHxXsfkxF4AaABAg)