To send individual headers:

```
//user.service.ts
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from "@angular/common/http";
import { User } from './user';

@Injectable()
export class UserService {

    constructor(private http:HttpClient) { }

    registerUser(user:User){
        return this.http.post("/api/registerUser",user);
    }
    listUsers(){
        return this.http.get("/api/listusers",{
          headers:new HttpHeaders().append('Autherization','Bearer chandhu')
        });
    }
    editUser(id:number){
        return this.http.get("/api/editUser/"+id);
    }
    updateUser(user:User){
        return this.http.put("/api/updateUser",user);
    }
    deleteUser(id:number){
        return this.http.delete("/api/deleteUser/"+id);
    }
    loginUser(user:User){
        return this.http.post("/api/loginUser",user);
    }
    logOut(){
      return this.http.get("/api/logout");
    }
    authStatus(){
      return this.http.get("/api/authStatus");
    }

}
```

Interceptors:

A request **interceptor** is a piece of code that gets activated for every single HTTPrequest received by your application. **Interceptors** are very useful when you need to perform some common processing for every HTTP request.

```
//auth.interceptor.ts
import { Injectable } from '@angular/core';
import { HttpEvent, HttpInterceptor, HttpHandler, HttpRequest } from
'@angular/common/http';
import { Observable } from "rxjs";

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
{
    console.log("interceptor request...");
    const idToken = localStorage.getItem("id_token");
    if (idToken) {
      console.log(idToken);
      //dont hit direct req, clone it and add header
      const cloned = req.clone({
        headers: req.headers.set("Autherization", "Bearer " + idToken)
      })
      return next.handle(cloned);//if token found then send this headers request
    }
    else {
      return next.handle(req);//if no token send as it is
    }
  }

}
```

Now in app.module.ts

```
import { LoginComponent } from './login.component';
import { FooterComponent } from './footer.component';
import { routes } from './app.routes';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { RouterModule } from "@angular/router";
import { FormsModule } from "@angular/forms";
import { HttpClientModule, HTTP_INTERCEPTORS } from "@angular/common/http";
import { AppComponent } from './app.component';
import { HomeComponent } from './home.component';
```

```
import { AboutComponent } from './about.component';
import { RegisterComponent } from './register.component';
import { MenuComponent } from './menu';
import { UserService } from './user.service';
import { UserListComponent } from './userList.component';
import { EditUserComponent } from './editUser.component';
import { ProtectDataGaurd } from './protectData.gaurd';
import { HashLocationStrategy, LocationStrategy } from '@angular/common';
import { AuthGaurd } from './auth.gaurd';
import { CapitalPipe } from './capital.pipe';
import { ColorDirective } from './colorDirective';
import { DisableInputDirective } from './disableInputDirective';
import { AuthService } from './auth.service';
import { AdminModule } from './admin/admin.module';
import { AuthInterceptor } from './auth.interceptor';

@NgModule({
  declarations: [
    AppComponent, HomeComponent, AboutComponent, RegisterComponent,
MenuComponent, FooterComponent, LoginComponent, UserListComponent,
EditUserComponent, CapitalPipe, ColorDirective, DisableInputDirective
  ],
  imports: [
    BrowserModule, RouterModule.forRoot(routes), FormsModule, HttpClientModule,
AdminModule
  ],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: AuthInterceptor,
      multi: true
    },
    {
      provide: LocationStrategy,
      useClass: HashLocationStrategy
    },
    UserService, ProtectDataGaurd, AuthGaurd, AuthService,
  ],
  bootstrap: [AppComponent],
  exports: [RouterModule]
})
export class AppModule { }
```

```
//login.component.ts
import { Router } from '@angular/router';
import { User } from './user';
import { Component, OnInit } from '@angular/core';
import { UserService } from './user.service';

@Component({
    selector: 'login',
    templateUrl: 'login.component.html'
})

export class LoginComponent implements OnInit {
    user:User;
    data1:User;
    token:string;
    constructor(private cs:UserService,private router:Router) {
        this.user = new User();
     }

    ngOnInit() { }
    login(){
        this.cs.loginUser(this.user).subscribe((data)=>{
            if(data){

                this.data1 = <User>(data as any).user;
                this.token = (data as any).token;
                console.log("this.data1",this.data1)
                localStorage.setItem("id_token",this.token);
                this.router.navigate(['/userList']);
            }
        })
    }
}
```

//server.js(JWT)

```
//server.js
var express = require("express");
var session = require('express-session');
var path = require("path");
var bodyParser = require("body-parser");
var mongoose = require("mongoose");
```

```
var app = express();
var router = express.Router();
var passport = require('passport');
var LocalStrategy = require('passport-local').Strategy;
const jwt = require('jsonwebtoken');

app.use(session({
  secret: 'keyboard cat',
  resave: false,
  saveUninitialized: false
}));

var config = require("./config/config");
mongoose.connect(config.local.db, { useNewUrlParser: true });

require("./models/user");
var User = mongoose.model("User");

app.use(express.static(path.join(__dirname, "../dist/myapp/")));
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use(passport.initialize());
app.use(passport.session());

router.post("/api/registerUser", function (req, res) {
  var theUser = new User(req.body);
  theUser.save(function (err) {

    if (err) {
      console.log(err);
    } else {
      res.send(theUser);
    }
  });
});
const VerifyToken = require("./VerifyToken");

router.get("/api/listusers", VerifyToken, function (req, res) {

  //console.log(req.isAuthenticated());
  //console.log(req.session);

  User.find({}).exec(function (err, users) {
    if (err) {
      res.failure(err)
```

```
    } else {
      res.json(users);
    }
  })
});

function isUserAllowed() {
  return function (req, res, next) {
    if (!req.isAuthenticated()) {
      res.redirect('/api/logout');
    }
    else {
      next();
    }
  }
}

router.get("/api/authStatus", function (req, res) {
  if (req.isAuthenticated()) {
    return res.status(200).json({
      status: true
    });
  } else {
    return res.status(500).json({
      status: false
    });
  }

});

router.get("/api/logout", function (req, res) {
  req.logout();
  res.json({ "message": 'success, you are now logged out!' });
});

router.get("/api/editUser/:id", VerifyToken, function (req, res) {

  User.findOne({ _id: req.params.id }).exec(function (err, user) {
    if (err) {
      res.failure(err)
    } else {
      res.json(user);
    }
  })
});
```

```
router.put("/api/updateUser", VerifyToken, function (req, res) {
  var sid = req.body._id;
  User.findById(sid, function (err, doc) {
    doc.fname = req.body.fname;
    doc.lname = req.body.lname;
    doc.save(function () {
      res.json(doc);
    })
  })
});

router.delete("/api/deleteUser/:id", VerifyToken, function (req, res) {
  var sid = req.params.id;
  User.findById(sid, function (err, doc) {
    doc.remove(function () {
      res.json({ message: "deleted" });
    })
  })
});

router.param('userId', function (req, res, next, id) {
  User
    .findOne({
      _id: id
    })
    .exec(function (err, theUser) {
      if (err) return next(err);
      if (!theUser) return res.failure('Failed to load User ' + id);
      req.theUser = theUser;
      console.log("theUser in userId", theUser);
      next();
    });
});

router.post("/api/loginUser", function (req, res, next) {
  passport.authenticate('local', function (err, user, info) {
    if (err) {
      return next(err);
    }
    if (!user) {
      console.log(info.message);
    }
    req.logIn(user, function (err) {
      if (err) {
```

```
        return next(err);
      }
      req.theUser = user;


      var token = jwt.sign({user }, config.local.secret);


      res.status(200).send({ auth: true, token: token,user:user });

      // res.send(user);

    });
  })(req, res, next);
});

//Use local strategy
passport.use(new LocalStrategy({
  usernameField: 'username',
  passwordField: 'password'
},
  function (username, password, done) {
    User.findOne({
      username: username
    }, function (err, user) {
      if (err) {
        return done(err);
      }
      if (!user) {
        return done(null, false, {
          message: 'Unknown user'
        });
      }

      if (!user.authenticate(password)) {
        return done(null, false, {
          message: 'Invalid passwordd'
        });
      }
      return done(null, user);
    });
  }
));

//Serialize sessions
```

```
passport.serializeUser(function (user, done) {
  //console.log('serialize user???');
  done(null, user.id);
});

passport.deserializeUser(function (id, done) {
  User.findOne({
    _id: id
  }, function (err, user) {
    if (user._id) user.userId = user._id;
    done(err, user);
  });
});

app.use(router);
app.listen(4200);
console.log("server is listening on port 4200");
```

```
//verifyToken.js
var jwt = require('jsonwebtoken');
var config = require('./config/config');
function verifyToken(req, res, next) {

  try{
    const token = req.headers.autherization.split(" ")[1];
    console.log("req.headers",req.headers);
  const decoded = jwt.verify(token,config.local.secret);
  console.log("decoded value",decoded);
  req.userData = decoded;
  next();
  } catch(error){
  return res.status(401).json({
    message:"Auth failed"
  });

  }
}
module.exports = verifyToken;
```

```
//config.js
module.exports={
```

```
  local:{
      db:"mongodb://localhost/trainDb",
      secret: 'keyboard cat',
  },
  prod:{
      db:"mongodb://10.112.21.23/trainDb"
  }
}
```

```
//user model
var mongoose = require('mongoose'),
    Schema = mongoose.Schema,
    crypto = require('crypto');

var UserSchema = new Schema({
    fname: String,
    lname:String,
    username: String,
    hashed_password: String,
    salt: String
});

UserSchema.virtual('password').set(function (password) {
    this._password = password;
    this.salt = this.makeSalt();
    this.hashed_password = this.encryptPassword(password);
}).get(function () {
    return this._password;
});

UserSchema.methods = {

    authenticate: function (plainText) {
        return this.encryptPassword(plainText) === this.hashed_password;
    },

    makeSalt: function () {
        return Math.round((new Date().valueOf() * Math.random())) + '';
    },

    encryptPassword: function (password) {
        if (!password) return '';
```

```
        return crypto.createHmac('sha1',
this.salt).update(password).digest('hex');
    }
};

mongoose.model('User', UserSchema);
```