ASE_Lab8 Report

JWT Authentication

March 23, 2019

Documented by:

Dharani Muli (Class ID : 18),

Chakra Pavan Kumar (Class ID: 13)

Introduction

Objectives

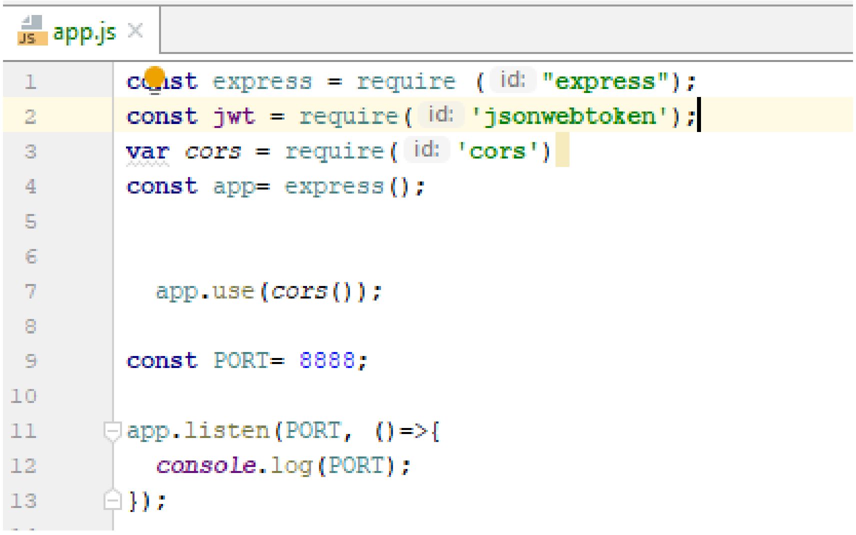
User Authentication using JWT token

Design/Implementation

We have followed below steps to successfully complete this lab assignment:

Step -1: We make sure all the pre-requisites are ready before start of the project and below are the technologies/languages used:

1. WebStorm IDE
2. Node
3. Npm
4. Expressjs :
   a. **Installation**: npm install express
   b. **Importing**: const app= express();
5. Angular
6. JWT:
   a. **Installation**: npm install jsonwebtoken
   b. **Importing**: const jwt= require('jsonwebtoken');
7. Cors: below are commands used
   a. **Installation**: npm install cors
   b. **Importing**: const cors = require('cors');
   c. **Enabling cors**: app.use(cors());
8. Also need to setup nodejs for porting



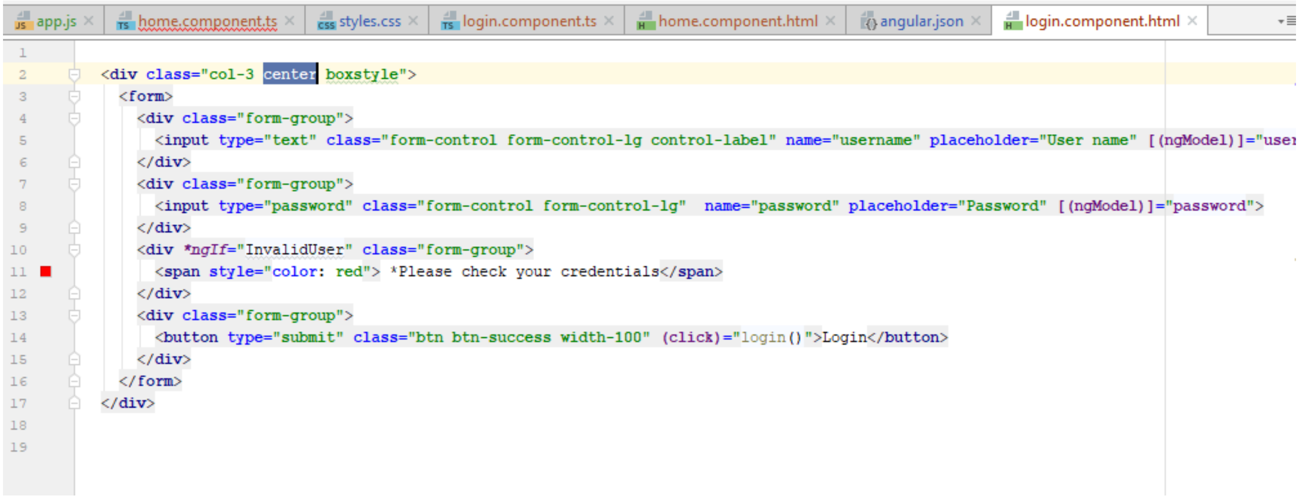
```
const express = require ( id: "express");
const jwt = require( id: 'jsonwebtoken');
var cors = require( id: 'cors')
const app= express();


  app.use(cors());

const PORT= 8888;

app.listen(PORT, ()=>{
  console.log(PORT);
});
```

Step-2: As we are using Angular, we should create project using command “ ng new JWTAuthentication” and Using this command I have created components :
ng g c login and mg g c home

Step-3: We have done the required configurations for components in the app.routing.module.ts file

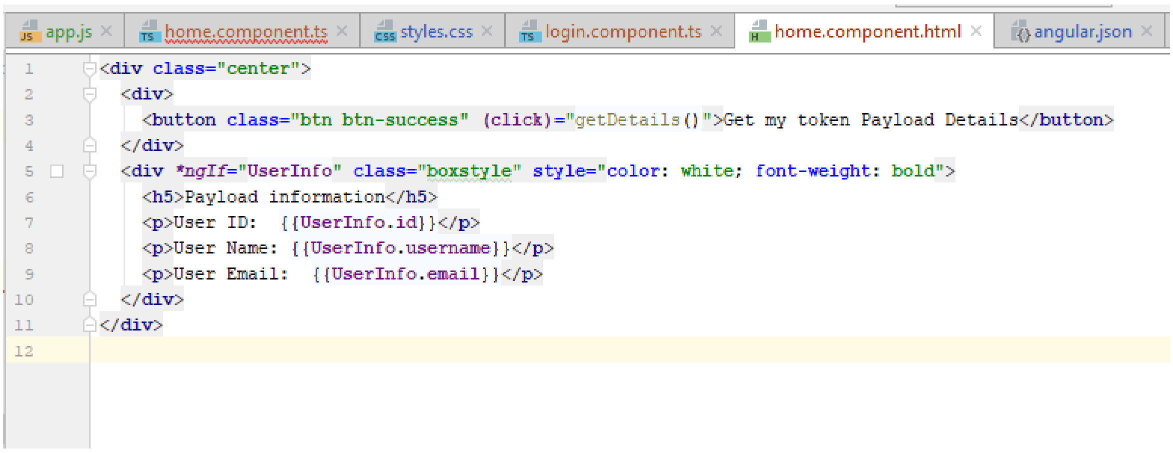
Step-4: We have written below html code for home and login page as below

Login.html



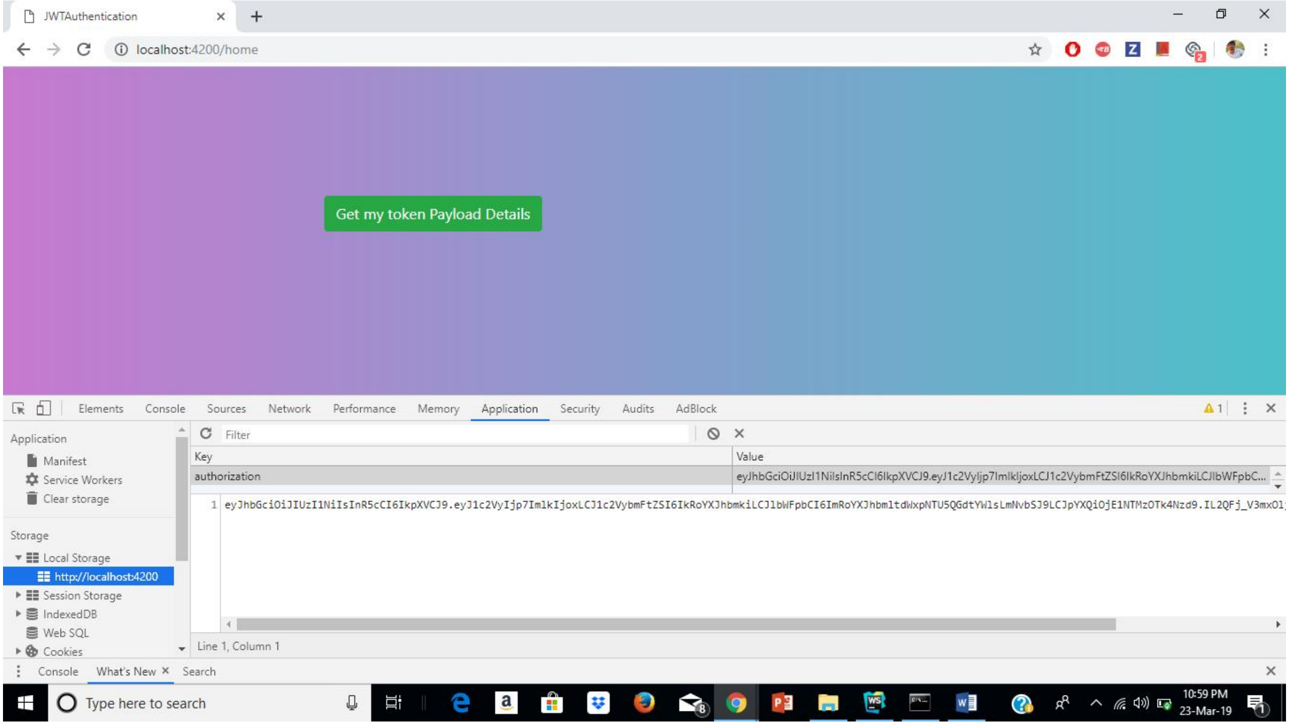
```
<div class="col-3 center boxstyle">
  <form>
    <div class="form-group">
      <input type="text" class="form-control form-control-lg control-label" name="username" placeholder="User name" [(ngModel)]="user
    </div>
    <div class="form-group">
      <input type="password" class="form-control form-control-lg"  name="password" placeholder="Password" [(ngModel)]="password">
    </div>
    <div *ngIf="InvalidUser" class="form-group">
      <span style="color: red"> *Please check your credentials</span>
    </div>
    <div class="form-group">
      <button type="submit" class="btn btn-success width-100" (click)="login()">Login</button>
    </div>
  </form>
</div>
```

Home.html



```
<div class="center">
  <div>
    <button class="btn btn-success" (click)="getDetails()">Get my token Payload Details</button>
  </div>
  <div *ngIf="UserInfo" class="boxstyle" style="color: white; font-weight: bold">
    <h5>Payload information</h5>
    <p>User ID:  {{UserInfo.id}}</p>
    <p>User Name: {{UserInfo.username}}</p>
    <p>User Email:  {{UserInfo.email}}</p>
  </div>
</div>
```

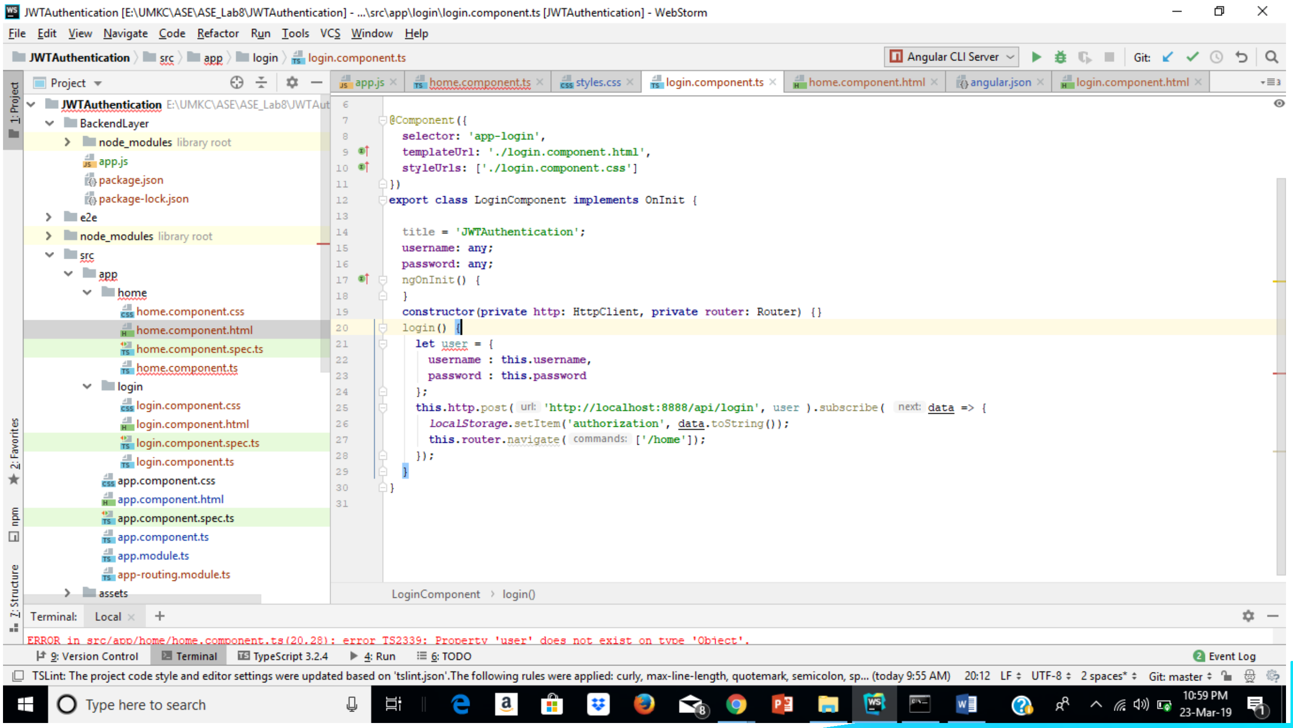
Step-5: Our login.components.ts is sending a request to nodejs server using url: `http://localhost:8888/api/login` for authentication and then it store received response into local storage and navigate to home page along with response data received.



Key	Value
authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7ImlkIjoxLCJ1c2VybmFtZSI6IkRoYXJhbmkiLCJlbWFpbC...

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7ImlkIjoxLCJ1c2VybmFtZSI6IkRoYXJhbmkiLCJlbWFpbCI6ImRoYXJhbmltdWxpNTU5QGdtYWlsLmNvbSJ9LCJpYXQiOjE1NTMzOTk4Nzd9.IL2QFj_V3mxO1
```

Step-6: We have our backend layer which handles authentication and creation of JWT token. By receiving request from “login” it creates token and send as the same with the response header and when it receives request from “getDetails” it verify the token received and authenticate and sends the payload as one of the response.

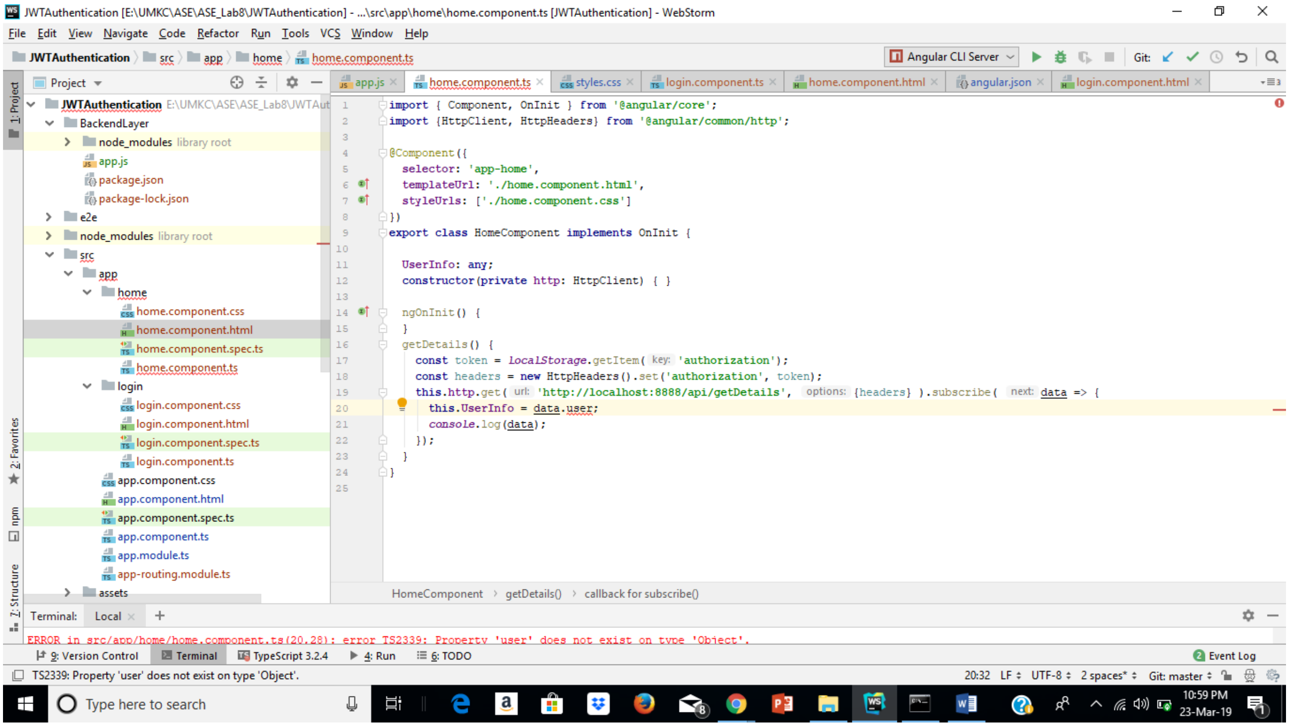


```
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  title = 'JWTAuthentication';
  username: any;
  password: any;
  ngOnInit() {
  }
  constructor(private http: HttpClient, private router: Router) {}
  login() {
    let user = {
      username : this.username,
      password : this.password
    };
    this.http.post( 'http://localhost:8888/api/login', user ).subscribe( data => {
      localStorage.setItem('authorization', data.toString());
      this.router.navigate( ['/home']);
    });
  }
}
```

```
ERROR in src/app/home/home.component.ts(20,28): error TS2339: Property 'user' does not exist on type 'Object'.
```

TSLint: The project code style and editor settings were updated based on 'tslint.json'.The following rules were applied: curly, max-line-length, quotemark, semicolon, sp... (today 9:55 AM)

Step-7: We are using local storage to get data from login component and sending a request along with "token" for verification using HttpHeaders and receiving payload as a response.

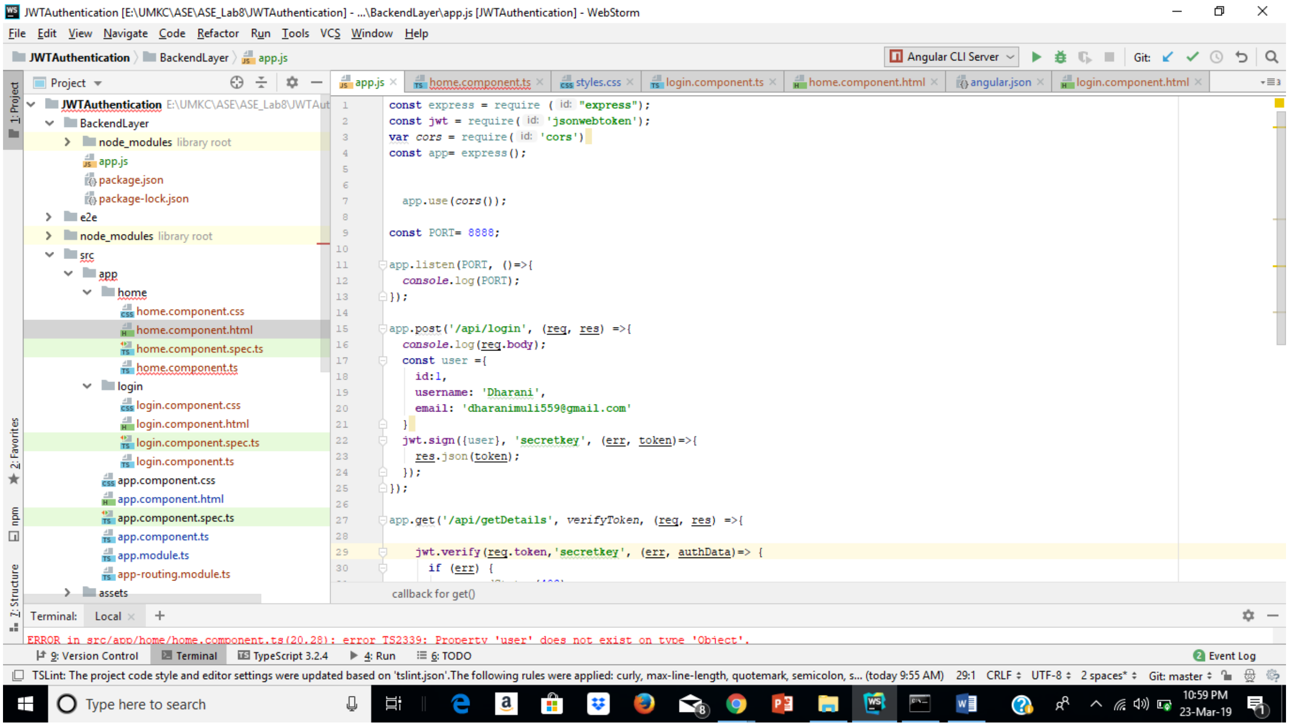


JWTAuthentication [E:\UMKC\ASE\ASE_Lab8\JWTAuthentication] - ...\src\app\home\home.component.ts [JWTAuthentication] - WebStorm

```
import { Component, OnInit } from '@angular/core';
import {HttpClient, HttpHeaders} from '@angular/common/http';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  UserInfo: any;
  constructor(private http: HttpClient) { }

  ngOnInit() {
  }
  getDetails() {
    const token = localStorage.getItem('authorization');
    const headers = new HttpHeaders().set('authorization', token);
    this.http.get('http://localhost:8888/api/getDetails', {headers} ).subscribe( data => {
      this.UserInfo = data.user;
      console.log(data);
    });
  }
}
```

```
ERROR in src/app/home/home.component.ts(20,28): error TS2339: Property 'user' does not exist on type 'Object'.
```

TS2339: Property 'user' does not exist on type 'Object'.

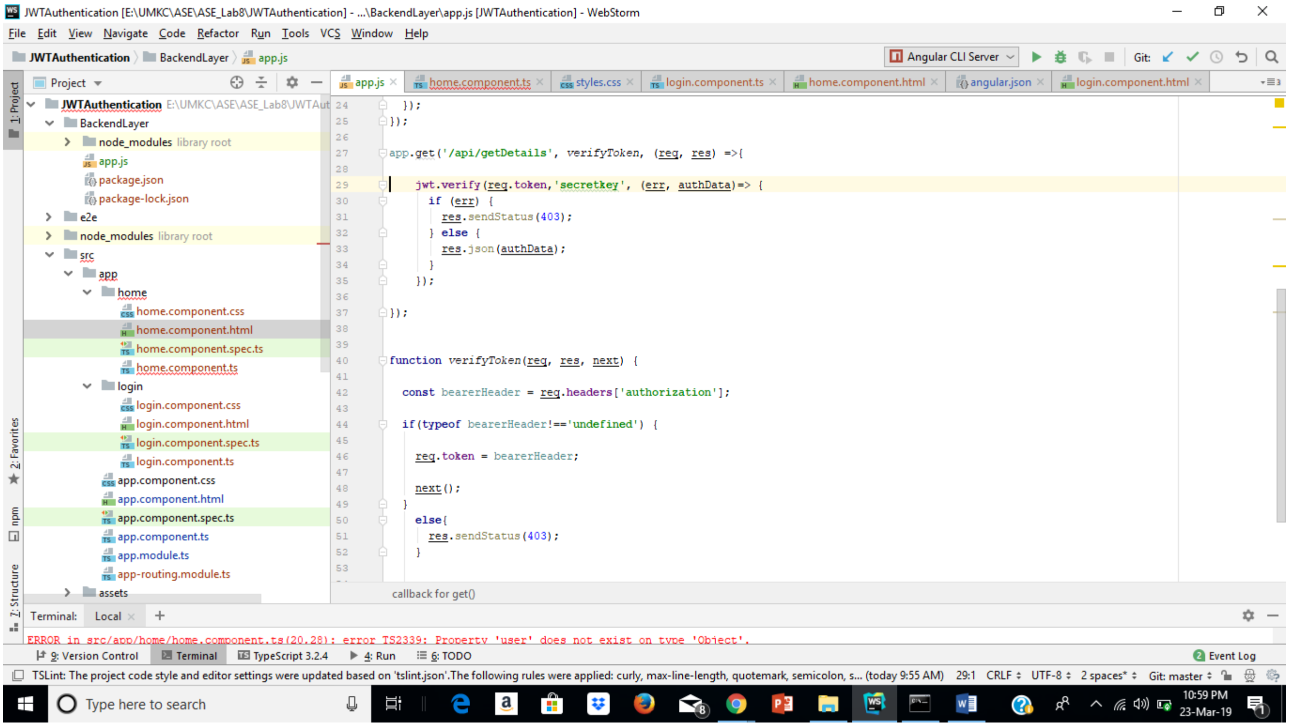


JWTAuthentication [E:\UMKC\ASE\ASE_Lab8\JWTAuthentication] - ...\BackendLayer\app.js [JWTAuthentication] - WebStorm

```
const express = require ("express");
const jwt = require('jsonwebtoken');
var cors = require('cors')
const app= express();


  app.use(cors());

const PORT= 8888;

app.listen(PORT, ()=>{
  console.log(PORT);
});

app.post('/api/login', (req, res) =>{
  console.log(req.body);
  const user ={
    id:1,
    username: 'Dharani',
    email: 'dharanimuli559@gmail.com'
  }
  jwt.sign({user}, 'secretkey', (err, token)=>{
    res.json(token);
  });
});

app.get('/api/getDetails', verifyToken, (req, res) =>{

    jwt.verify(req.token,'secretkey', (err, authData)=> {
      if (err) {
```

```
ERROR in src/app/home/home.component.ts(20,28): error TS2339: Property 'user' does not exist on type 'Object'.
```

TSLint: The project code style and editor settings were updated based on 'tslint.json'.The following rules were applied: curly, max-line-length, quotemark, semicolon, s... (today 9:55 AM)



JWTAuthentication [E:\UMKC\ASE\ASE_Lab8\JWTAuthentication] - ...\BackendLayer\app.js [JWTAuthentication] - WebStorm

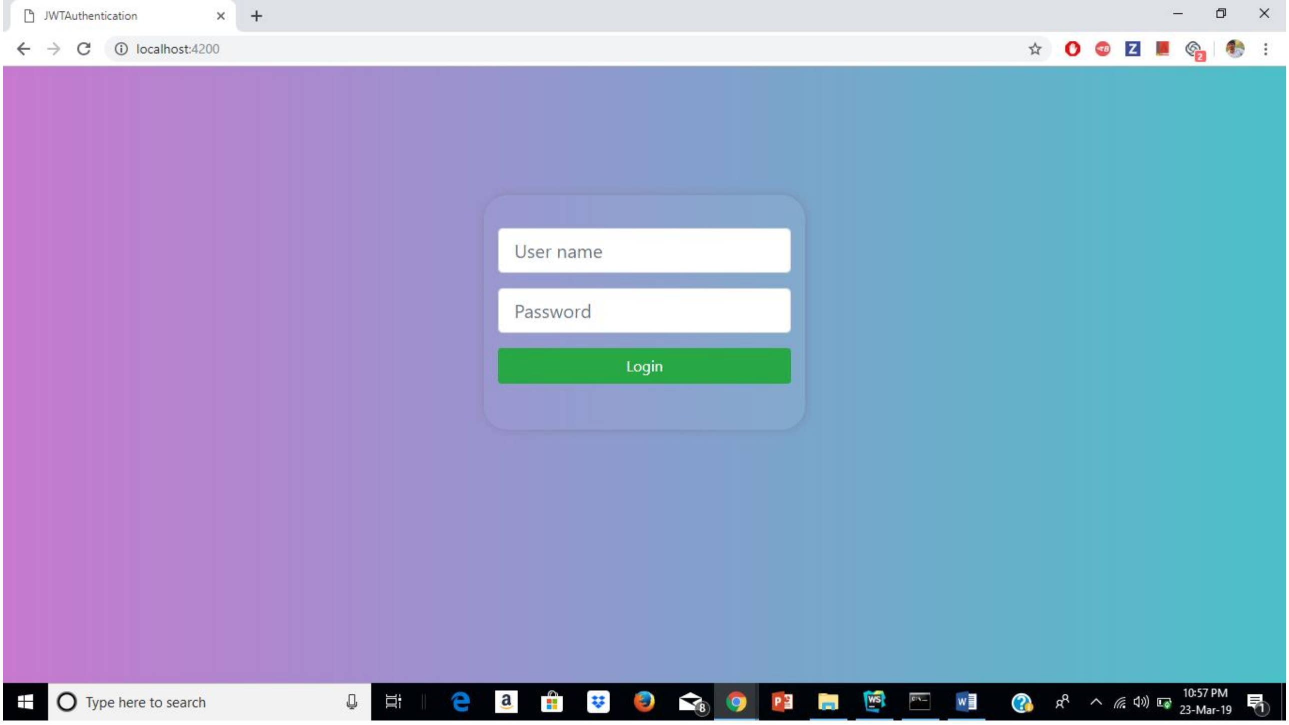
```
  });
});

app.get('/api/getDetails', verifyToken, (req, res) =>{

    jwt.verify(req.token,'secretkey', (err, authData)=> {
      if (err) {
        res.sendStatus(403);
      } else {
        res.json(authData);
      }
    });

});


function verifyToken(req, res, next) {

  const bearerHeader = req.headers['authorization'];

  if(typeof bearerHeader!=='undefined') {

    req.token = bearerHeader;

    next();
  }
    else{
      res.sendStatus(403);
    }
```

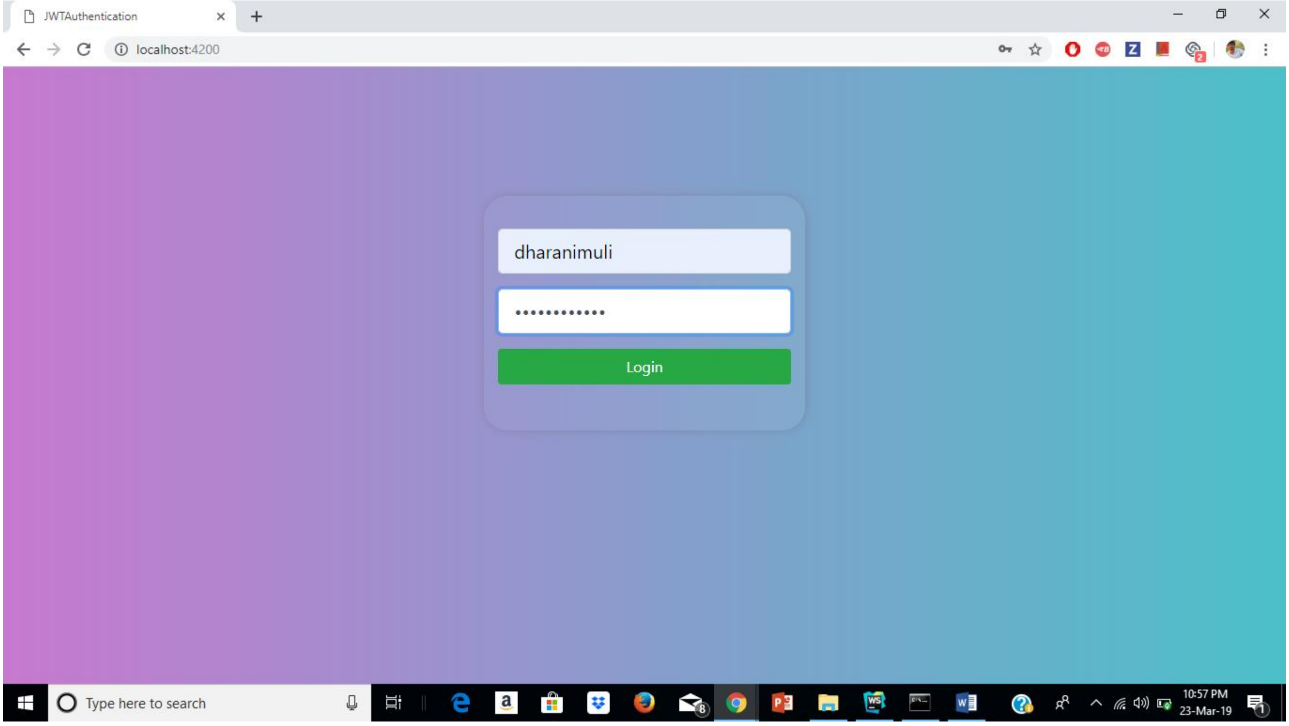
```
ERROR in src/app/home/home.component.ts(20,28): error TS2339: Property 'user' does not exist on type 'Object'.
```

TSLint: The project code style and editor settings were updated based on 'tslint.json'.The following rules were applied: curly, max-line-length, quotemark, semicolon, s... (today 9:55 AM)

Sample output

1. Login page



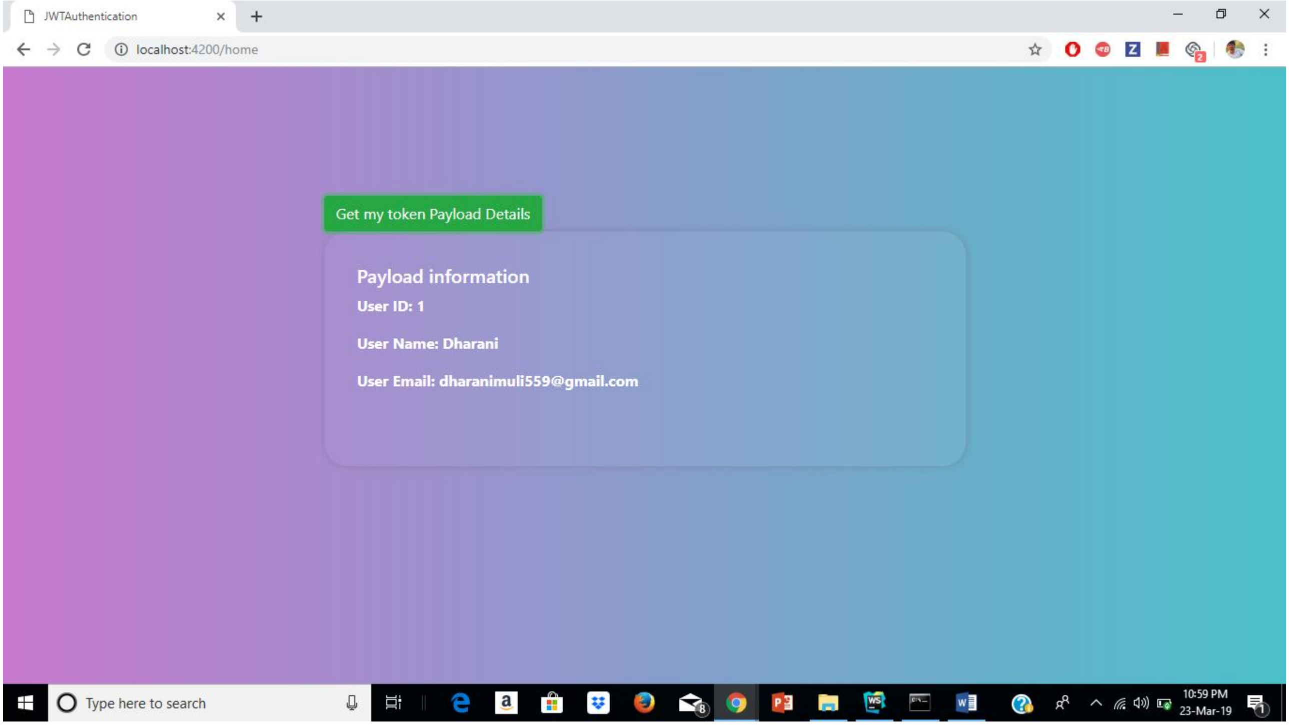


dharanimuli

••••••••••••

Login

2. Home Page



Get my token Payload Details

Payload information

User ID: 1

User Name: Dharani

User Email: dharanimuli559@gmail.com

Issues/Limitations:

None

Team Contribution

Dharani:

1. Implemented Login and JWToken generation.
2. Contributed in the creation of Wiki page, report and lab submission.

Chakra Pavan Kumar Kota:

1. Implemented Home page and JWT verification part.
2. Contributed creation of Wiki page, report and lab submission.

Conclusion:

As part of this exercise we gained new technique that is token based Authentication using JWT which we can also use this in our upcoming projects.