**Database**

Steps to create database and tables in MySQL,

1. In the root folder of <project-title>-app, create a folder dbscripts
2. In dbscripts, create a sql file to store the schema creation and table data
3. Create table user, role, and user\_role
4. Insert few records to the tables

**Microservices**

Steps to code AuthenticationService end point in Spring Boot (package => com.cognizant.authentication)

1. In the root folder of <project-title>-app, generate a spring boot project authentication-service by using <https://start.spring.io/>
2. Create a sub package com.cognizant.authentication.security and place the following classes
   1. SecurityConfig.java
   2. JwtAuthorizationFilter.java
   3. AppUser.java
   4. AppUserDetailsService.java
3. Create a sub package com.cognizant.authentication.model and place the following classes
   1. User.java
   2. Role.java
4. Create a sub package com.cognizant.authentication.repository and place the following class
   1. UserRepository
5. Create a sub package com.cognizant.authentication.service and place the following class
   1. UserService
6. Create a sub package com.cognizant.authentication.controller and place the following class
   1. AuthenticationController
7. Modify application.properties to include configuration
8. Implement Mockito test for the Authentication

**Angular – Reference: truYum application**

1. In the root folder of <project-title>-app, create an angular app
2. Create a new component login-component
3. Create a new service authentication-service
4. Create a new guard with name auth inside the auth folder
5. Implement Karma testing to test the login-component

**ng new angular-app**

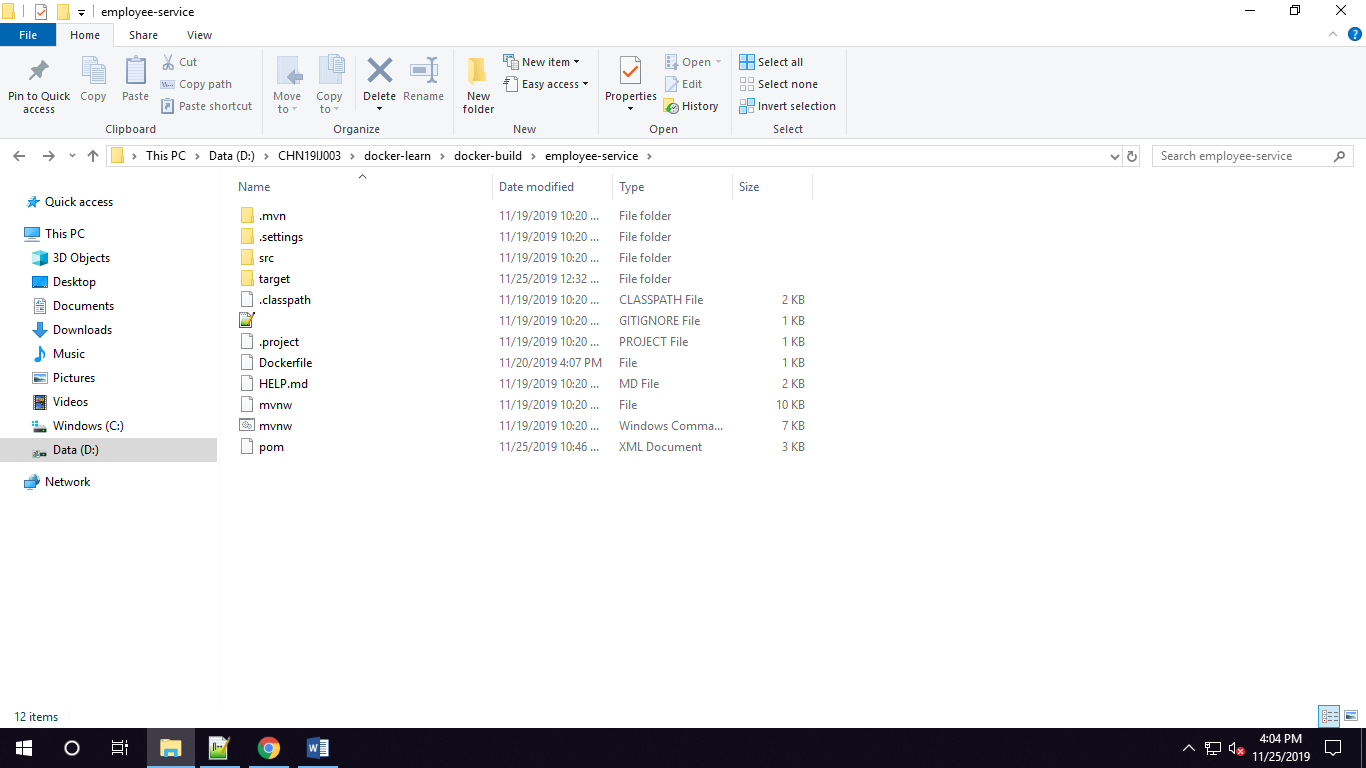
**ng g c login-component**

**ng g s authentication-service**

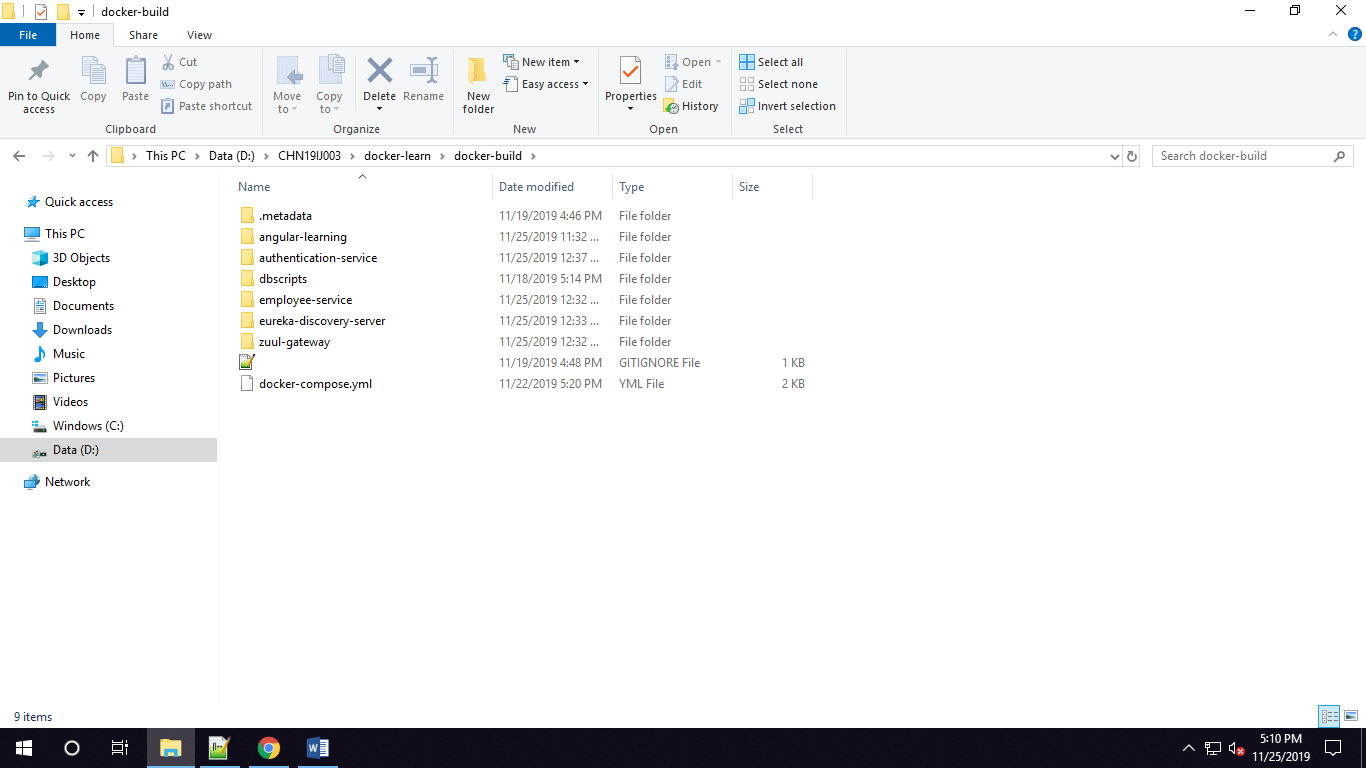
**ng g guard auth/auth**

**Docker**

1. In the root folder of <project-title>-app, create a **docker-compose.yml** to dockerize the dbscripts created earlier in db.
2. Inside authentication-service created earlier in microservices, create a new file named **Dockerfile** in the root folder of each microservices,



1. In the root folder of **<project-title>-app**,
   1. Upon successful completion of angular application in angular-app folder
   2. Execute **ng build** command which creates dist folder in the root folder of angular-app
   3. Create a Dockerfile in the same root folder of angular-app
2. Execute **docker-compose up - -build** command. Sample folder structure as mentioned below,



Few Docker Commands,

|  |  |
| --- | --- |
| **Command** | **Use** |
| docker images | List all images |
| docker ps | List all running containers |
| docker ps –a | List all running & stopped containers |
| docker start <container-name/container-id> | Run containers |
| docker stop <container-name/container-id> | Stop containers |
| docker container prune | Remove all stopped containers |
| docker system prune –a | Remove all images, stopped containers, & networks |

**Jenkins Setup**

1. Using Jenkins war file start the Jenkins (java –jar Jenkins.war) server
2. Using initialAdminPassword login, then create login credentials to manage Jenkins and Login to the Dashboard of Jenkins.
3. After stopping the Jenkins server, Copy all the extracted Jenkins-plugins in the Jenkins plugins folder.
4. Configure Git & Maven Path in Mange Jenkins-Global Tool Configuration.
5. Create Service-build item for all maven based micro-service projects, by configuring the GitLab url
6. Execute each build Item and Verify the Jar Creation after passing all the test cases.