Microservice

This is a app to authenticate user and generate a Token and use that token to patch json data and resize a given public image to 50x50 size. This service has one public api and two private apis that use token generated during authentication.

Setup and Run:

1. Please insure if node, mongodb is installed and mongodb is running.
2. Clone code from https://github.com/sunnyjain123/microservice
3. Then Run **npm install** inside the folder from terminal to install all the dependencies.
4. The run **npm start** to run the server. It will run on port 3000
5. To run test cases run **npm test**.

Structure :

1. Config :

This folder contains configurations to use during the run time of application.

- Config.js
    - This file contains name of application, Environment of application, port on which application will run, mongodb url to connect and secret key used during JWT Token generation

2. Server :

This folder contains all functions and database schema files

- models/user.js
    - This file is used to define schema for user collections. Schema contains username, password, token and expired_at fields. This file also contains function to verify if token is valid or not.
- services/imageSize.js
    - This file is used to resize image to 50x50 size to complete this task it uses request and fs to download file, lwip to resize image and cloudinary module to upload file to cloudinary. This function requires valid token in headers and image in body to complete task.
- services/patch.js
    - This file uses json-patch module to perform patch operations on a given object. This function requires obj and patch in body and valid token in header to complete task.

- services/user.js
    - This file uses jsonwebtoken module and secret from config file to generate JWT Token for authentication. This function requires username and password in body to complete task.
- Routes.js
    - This file contains all the routes to all the api and swagger specifications to generate swagger documentation for this operation it requires swagger-jsdoc module.

3. Test:

   This folder contains mocha and chai test case
   - Test.js
       - This file contains all test cases for 3 apis.
4. Dockerfile

   This file contains specification for running a docker service

5. Logger.js

   This file contains three log functions. These log functions are created using winston module.

6. Package.json

   This file contains all the node modules to install, server run and test case run instructions

7. Server.js

   This file contains specifications on how to run server.

API :

1. /api/1.0/generateToken :

   This api is used to generate access token to access other private apis. This api accepts **username** and **password** in request body as a object and uses **jsonwebtoken module** to generate **JWT Token** and returns it. Sample data..

   ```
   Example : {
           username : 'Sunny',
           password : 'sunnyPassword'
   }
   ```

   If any field is missing it will return a error. Generated token is valid for next 3 hours. After Generating Token it stores data in users collection.

2. /api/1.0/jsonPatch :

   This api is used to perform patch operations on a given data. This api accepts **obj** and **patch** data in request body and token in request header. This uses **json-patch module** to perform patch operations. Sample data..

   ```
   Example : {
           obj : {
                   "baz": "qux",
                   "foo": "bar"
           },
           patch : [{
                   "op": "replace", "path": "/baz", "value": "boo"
           },{
                   "op": "add", "path": "/hello", "value": "world"
           },{
                   "op": "remove", "path": "/foo"
           }]
   }
   ```

   If obj or patch data is not correct or provided token is not correct it will return error.

3. /api/1.0/imageSizeChange :

   This api is used to resize a given image to 50x50 size. This api accepts **image url** in body and **token** in request header. This api uses **request module** to download and save images, **lwip module** to resize image and **cloudinary module** to upload image to cloudinary. Sample data..

   ```
   Example : {
           image :
   'https://s3.amazonaws.com/silverpushcdn/320x50_Datawind-Calling-Tablet.jpg'
   }
   ```

   If image is private, url is not correct, token is not provided or invalid token is provided it will return error.

Testing :

This app uses **chai and mocha** to run test cases. Test cases are divided in three parts

1. POST /api/1.0/generateToken
   To test all the cases of user authentication and token generation
2. POST /api/1.0/jsonPatch
   To test all cases of performing patch operation on given object
3. POST /api/1.0/imageSizeChange
   To test all cases of resizing image to 50x50 size

Extra Documentation :

For extra documentation **swagger-jsdoc** module is used all the specifications are return in **routes.js file** inside services folder.

Logging/Monitoring :

Winston module is used to save logs. All the specification for logging are return in logger.js file. IT contains three function each for logging different apis.

Dockerize :

Dockerfile contains specifications for running in a docker environment

Code Coverage :

To run code coverage Istanbul can be used. Run **Istanbul cover test/test.js** inside application directory