Basic Skills Test

Overview

The purpose of this test is to determine if the applicant is capable of building a decent looking website with a REST API. Requirements, outcomes, and API documentation is provided below.

Requirements:

1. A website with at least a login page and a user management page
2. The code must be HTML/CSS/Javascript. The applicant may use frameworks such as Angular, React, etc. No server-side rendering is allowed eg. no PHP

Expected Outcomes

1. There must be a minimum of two pages: 1 login page, and at least one page for user management
2. The user starts at the login page. On successful login, the user must proceed to the management page
3. On the management page, the user must be able to perform basic CRUD operations on users
4. The design of this site is left to the applicant. We are primarily interested in ease of use of the website. Ideally, the design should be responsive eg. if a user is added, the list of users should be updated without having to refresh the page. The layout and style should feel intuitive.
5. The code must be deployed as you would for a live site eg. uglify the code, etc.
6. A copy of the code must also be submitted for analysis

API Documentation

Overview

The user API calls documented below allow admin users to manage other users of the system. While you will be able to log in as any valid user, attempts to manage users as a non-admin user will result in an error.

I have created the following account on the system for you to get started:

username: test

password: test12345

It is only possible to change passwords for users. The username and role cannot be modified.

Server Address

All requests must target the following server:

http://31.222.178.244:80

Success Messages

On success, HTTP code 200 is received with an optional payload in JSON format, depending on the endpoint

Error Messages

By default, and unless otherwise stated, failed REST calls return HTTP response code 406 with a JSON message indicating the reason for the error. The error message structure is:

```
{
	“msg”: “string”
}
```

Additional error codes:

1. 5xx – These are errors from the API gateway. This means the API could not be contacted. These are not accompanied by a JSON response.
2. 404 – The endpoint in question does not exist. This is typically not a API gateway error. These are not accompanied by a JSON response.
3. 401 – Authentication error. You are not logged in, or your JWT token has expired.
4. 403 – You are not allowed to access the specified resource.

Login

Endpoint: /auth/rest/login
Method: POST
Request:

```
{
	"username":"string",
	"password":"string"
}
```

Response:

```
{
	"token": "JWT Token as a string",
```

```
        "username": "string",
        "role": "string"
    }
```

This endpoint is used to log a user into the system. The username and password in the request are string values.

On success, a JWT token is returned, along with the username and the user's role.

"token" is a JWT token. This must be used in subsequent HTTP requests by adding the following header to each request:

Authorization: Bearer <token>

where <token> is the token received in the login response. For more details on JWT, please see jwt.org

"username" in the response matches the username in the request.

"role" is the user role. This may be "admin" for administrators, "tech" for technicians, or "adv_tech" for advanced technicians.

Add User

Endpoint: /admin/rest/users/add

Method: POST
Request:

```
    {
        "username":"string",
        "password":"string",
        "role":"string"
    }
```

Response: None

Adds a user to the system.

"username" - A string at least 5 characters long, and no more than 16 characters. Alphanumeric characters are allowed only.

"password" - A string at least 8 characters long. Printable characters are allowed

"role" - must be one of "admin" for administrators, "tech" for technicians, or "adv_tech" for advanced technicians

List Users

Endpoint: /admin/rest/users/list

Method: GET

Request: None

Response:

```
[
      {
            "username":"string",
            "password":null,
            "role":"string"
       },
      ...,
      ...,
      ...
]
```

This endpoint lists all users on the system and returns an array of users. The data model is the same as for adding a user. The password field will always be null. The password is never returned.

Change Password

Endpoint: /admin/rest/users/changepw/user

Method: PUT

Request:

```
{
      "username":"string",
      "password":"string",
      "role":null
}
```

Response: None

Change a user's password. The data model is the same as for adding a user.

"username" - must contain the name of the user whose password is being changed
"password" - the new password for the user
"role" - this must be null.

Delete User

Endpoint: /admin/rest/users/delete/{username}
Method: DELETE
Request: None
Response: None

Deletes a user from the system. {username} is the name of the user to delete.

End Of Document