# Test Plan

I will be testing the REST API in the following 3 ways:

* Manual Testing using Postman
* Unit Testing using Visual Studio
* Build Testing using Github Workflows

# Test Methods

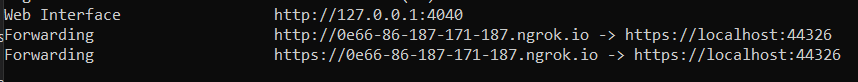
## Postman

Postman is a piece of software used to develop and test API’s. It allows you to build the requests and test each individual endpoint. To test, I run the application as usual and start an ngrok tunnel using Visual Studio’s ngrok extension.

Graphical user interface, application

Description automatically generated

Ngrok allows me to make a secure connection to localhost. I can then use the supplied URLs to use as the base url.



I can then plug a request into postman like as follows, and send a request for a result.

Graphical user interface, text, application, email

Description automatically generated

I can then make amendments, such as the headers to test each request.

Graphical user interface, text, application, email

Description automatically generated

## Unit Testing

## Github Workflows

# Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Description | Type | Expected Result | Actual Result | Pass/Fail |
| 1 | Acquire User information without Authorization Header | Postman | Server to respond with 401 | As expected | Pass |
| 2 | Encrypt a given string, then decrypt it. | Unit | Decrypted string to match the original string. | As expected | Pass |
| 3 | Decrypt a given string, then re-encrypt it. | Unit | Encrypted string to match the original string. | As expected | Pass |
| 4 | Convert string to base64 and back again using extension method | Unit | String converted back from base64 to match original string. | As expected | Pass |
| 5 | Acquire User information with Authorization header without logging in. | Postman | Server to respond with 401 | As expected | Pass |
| 6 | Login a user with empty credentials | Postman | Server to respond 400 and an error message | As expected | Pass |
| 7 | Login a user with invalid credentials | Postman | Server to respond with 401 | As expected | Pass |
| 8 | Login a user with correct credentials | Postman | Server to respond with 200 and a welcome message including the user’s name | As expected | Pass |
| 9 | Acquire user information after logging in | Postman | Server to respond with User information | Server responded 401 (Please Log in) | Fail |
| 10 | Action on test no. 9: modified code in BaseController – made the dictionary housing authenticated users static. | Debug | As with Test no. 9 | As expected | Pass |
| 11 | Send logout message | Postman | Server to respond with 200 and a goodbye message | As expected | Pass |
| 12 | Send logout message while already logged out | Postman | Server to respond with “You are already logged out” | As expected | Pass |
| 13 | Login with user that doesn’t exist | Postman | Server to respond with 401 and to request the user registers | As expected | Pass |
| 14 | Register a new user with missing fields | Postman | Server to respond with 400 and an error message | Responded with 401 and asked to login or register | Fail |
| 15 | Action on test no. 14: modified code in BaseContoller to no longer enforce BasicAuth for every endpoint. Now enforcing on UserController and the Login/Logout methods in the Authentication controller | Debug | As with Test no. 14 | As expected | Pass |
| 16 | Register a new user | Postman | Server to respond with 200 | As expected | Pass |
| 17 | Attempt to register a user that already exists | Postman | Server to respond with 400 | As expected | Pass |
| 18 | Login with newly created user | Postman | As with Test no. 8 | As expected | Pass |
| 19 | Build with Github workflows | Workflows | Build to pass | Build failed | Fail |
| 20 | Action on Test no. 19: specify solution file in the build script | Workflows | As with Test no. 19 | Build succeeded but unit tests failed | Fail |
| 21 | Action on Test no. 20: Specify the correct .NET version | Workflows | Build to still pass but unit tests to also pass | As expected | Pass |