

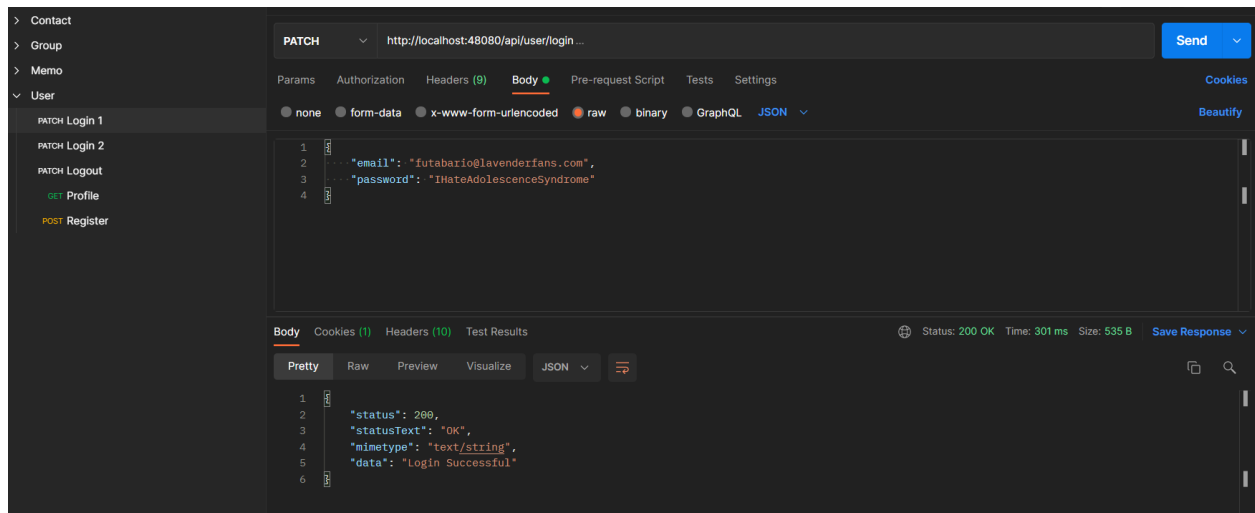
# Testing Report

## Overview

We have conducted countless tests in accordance with the testing plan that we've established. Due to the very high volume of tests, it would be infeasible to generate a test report for each test. Therefore, this report serves as a reflective and holistic summary of the outcomes of the tests along with the problems encountered (and how we resolved them) which we believe is more important.

## Unit testing using Postman

The unit testing with Postman went on seamlessly since it is very easy, straightforward and intuitive to use Postman. No scripts need to be written and the test is GUI based and very user-friendly. This usually means that if Postman tests fail, it is (almost) guaranteed that the issue lies in the implementation of the controller functions (assuming the server connection is fine). As so Postman serves as the main tool for debugging for our controller functions, which some of the problems we were able to catch include race conditions (with asynchronous functions).



*Testing login controller function*

## Integration testing using Jest

The integration testing with Jest is where we start to encounter problems that are not necessarily due to a bug in our implementation. In this stage we have to write our test files which in itself may have some bugs such that it is often not trivial to know whether the problem

lies with our implementation or the test itself. Non-implementation problems (i.e. not because our source code has bugs) that occur include:

- Jest timeout when connecting to the MongoDB database
  - Resolved by clearing the database after each test
- Permission errors when trying to update the MongoDB database
  - Resolved by creating a dedicated database for Jest and giving it full permissions to read and write
- Race conditions when running tests in parallel
  - Resolved by disabling parallel test runs so tests run sequentially, one by one

Overall compared to unit testing with Postman, it is much faster and simpler to run all the tests we need since we only have to do “npm run test”, and Jest will automatically run all the tests for us and generate the report. We ended up creating 22 test suites (groups), comprising 122 tests in total which are inspired by the acceptance criteria.

```
COMP30022-DocMcStuffins/backend$ npm run test

> comp30022-backend@0.0.0 test
> jest --config jestconfig.json --forceExit --runInBand

console.log
  Successfully connected to MongoDB test database.

    at NativeConnection.<anonymous> (src/config/databaseConfig.ts:22:13)

console.log
  HTTPError {
    status: undefined,
    statusText: "I'm a teapot",
    body: 'Would you like to have tea?'
  }

    at HTTPErrorHandler (src/middlewares/HTTPErrorHandler.ts:10:17)

console.log
  HTTPError {
    status: 418,
    statusText: undefined,
    body: 'Would you like to have tea?'
  }

    at HTTPErrorHandler (src/middlewares/HTTPErrorHandler.ts:10:17)

console.log
  HTTPError { status: 418, statusText: "I'm a teapot", body: undefined }

    at HTTPErrorHandler (src/middlewares/HTTPErrorHandler.ts:10:17)

PASS tests/classes/errorHandling.test.ts (11.819 s)
Error handling tests
  ✓ Trigger a well formed 418 error (479 ms)
  ✓ Trigger a 418 error with missing status field (52 ms)
```

*Running the test files automatically using “npm run test”*

```

Login Tests
  ✓ Login successfully (171 ms)
  ✓ Missing a field (36 ms)
  ✓ Wrong email (48 ms)
  ✓ Wrong password (100 ms)
Logout Tests
  ✓ Logout when authenticated (273 ms)
  ✓ Logout when unauthenticated (20 ms)
Get User Profile Tests
  ✓ Get profile when authenticated (229 ms)
  ✓ Get profile when unauthenticated (35 ms)

console.log
  Successfully connected to MongoDB test database.

    at NativeConnection.<anonymous> (src/config/databaseConfig.ts:22:13)
PASS tests/controllers/passport/passport.test.ts
Registration tests
  ✓ 1. Register with empty data (354 ms)
  ✓ 2. Register only with required fields (440 ms)
  ✓ 3. Register by sending all fields (165 ms)
  ✓ 4. Register with less than required password length (38 ms)
  ✓ 5. Register with pre-existing email address (237 ms)
Login tests
  ✓ 1. Login with empty data (58 ms)
  ✓ 2. Login with valid email and password (116 ms)
  ✓ 3. Login with invalid email (68 ms)
  ✓ 4. Login with incorrect password (130 ms)

Test Suites: 22 passed, 22 total
Tests:       122 passed, 122 total
Snapshots:   0 total
Time:        63.57 s
Ran all test suites.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all tests finished?

```

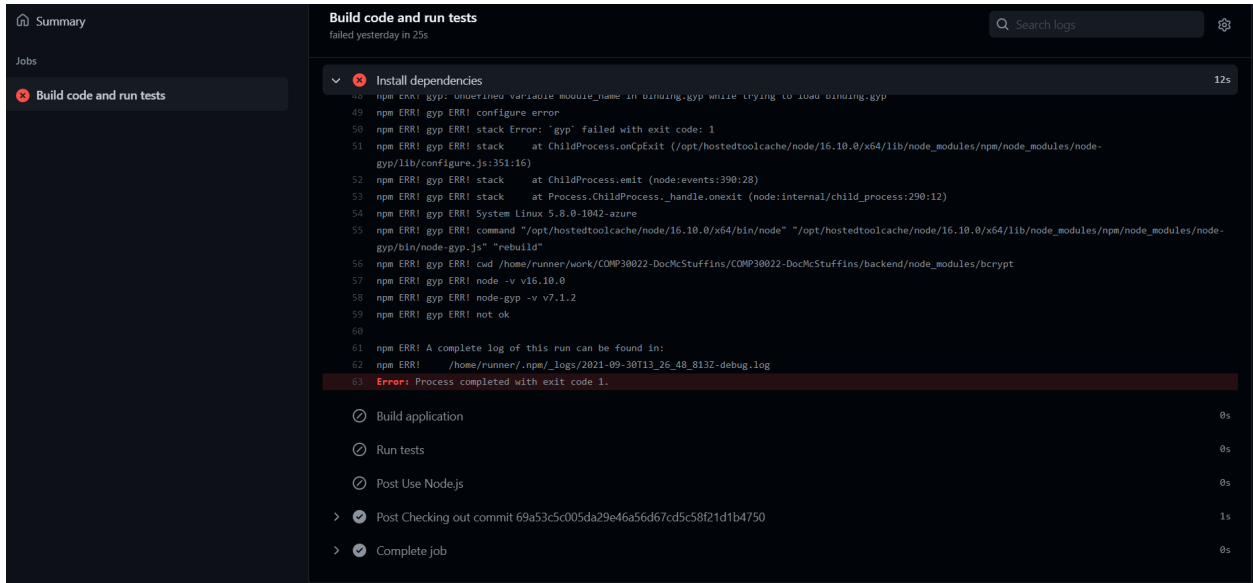
*Jest reporting the results of the test (in this case all tests pass)*

## CI

We configured the CI to build the server application and run all the Jest test suites. Effectively, CI would be the same as integration testing with Jest with the addition of checking whether our application can be built correctly (which is crucial for development), and that it lives in Github. Theoretically, CI should pass if the integration testing with Jest is successful in our local machine, but it turns out we encounter several problems including:

- Jest timeout (again)
  - Resolved by increasing the default Jest timeout value from 5s to 10s since we figured that CI runs much slower than our local machine
- Error binding the server to the specified port
  - We found out the problem lies in each test file attempting to bind the server to the same port repeatedly (which doesn't cause a problem in our local machines)
  - Resolved by restructuring the source code such that when Jest imports our server app, it binds and listens to the specified port only once
- Npm version errors
  - Resolved by making sure that the package-lock.json is unchanged

On the bright side, these annoying problems would mean that if we resolve them, then we can be sure the problems we might run to during deployment would be minimal.



Summary

Jobs

**Build code and run tests**

**Build code and run tests**  
failed yesterday in 25s

Search logs

Install dependencies 12s

```
48 npm ERR! gyp: undefined variable module_name in binding.gyp while trying to load binding.gyp
49 npm ERR! gyp ERR! configure error
50 npm ERR! gyp ERR! stack Error: "gyp" failed with exit code: 1
51 npm ERR! gyp ERR! stack     at ChildProcess.onCpExit (/opt/hostedtoolcache/node/16.10.0/x64/lib/node_modules/npm/node_modules/node-gyp/lib/configure.js:351:16)
52 npm ERR! gyp ERR! stack     at ChildProcess.emit (node:events:390:28)
53 npm ERR! gyp ERR! stack     at Process.ChildProcess._handle.onexit (node:internal/child_process:290:12)
54 npm ERR! gyp ERR! System Linux 5.8.0-1042-azure
55 npm ERR! gyp ERR! command "/opt/hostedtoolcache/node/16.10.0/x64/bin/node" "/opt/hostedtoolcache/node/16.10.0/x64/lib/node_modules/npm/node_modules/node-gyp/bin/node-gyp.js" "rebuild"
56 npm ERR! gyp ERR! cwd /home/runner/work/COMP30022-DocMcStuffs/COMP30022-DocMcStuffs/backend/node_modules/bcrypt
57 npm ERR! gyp ERR! node -v v16.10.0
58 npm ERR! gyp ERR! node-gyp -v v7.1.2
59 npm ERR! gyp ERR! not ok
60
61 npm ERR! A complete log of this run can be found in:
62 npm ERR!     /home/runner/.npm/_logs/2021-09-30T13_26_48_813Z-debug.log
63 Error: Process completed with exit code 1.
```

Build application 0s

Run tests 0s

Post Use Node.js 0s

Post Checking out commit 69a53c5c005da29e46a56d67cd5c58f21d1b4750 1s

Complete job 0s

*CI failing*



Implemented controller functions related to amendment and their respective tests Backend REST API CI #66

Summary

Jobs

**Build code and run tests**

**Build code and run tests**  
succeeded 7 days ago in 3m 43s

Search logs

Set up job 4s

Checking out commit e822eae3be97412a0097cd6dc47010ed30cb768d 4s

Use Node.js 5s

Install dependencies 5s

Build application 4s

Run tests 3m 20s

Post Use Node.js 0s

Post Checking out commit e822eae3be97412a0097cd6dc47010ed30cb768d 0s

Complete job 1s

*CI succeeding*

## CD

As projected in the CI section, we didn't encounter many problems during the phase of CD. As long as CI is able to build and serve the web app, so should CD. We did encounter a minor issue, but it was diagnosed and resolved swiftly. The issue we encountered was an error in the npm version (similar to the problem in CI, although for some reason CI didn't catch this), and again was resolved by using the correct version of package-lock.json file.

The screenshot shows the Heroku dashboard for the 'doc-mcstuffins' application. The 'Build Log' tab is selected, displaying a failed build. The log output indicates that the build failed because the Node.js version was not specified in the package.json file. The error message is: 'npm ERR! A complete log of this run can be found in: /tmp/npmcache.TYBTV/\_logs/2021-09-30T20\_17\_51\_887Z-debug.log'. The log also provides a link to the Heroku Devcenter for troubleshooting Node.js deployment issues. The build finished with the error 'Push rejected, failed to compile Node.js app. Push failed'.

heroku.com | Blogs | Careers | Documentation | **Support** | Terms of Service | Privacy | Cookies | © 2021 Salesforce.com

## CD failing

The screenshot shows the Heroku dashboard for the 'doc-mcstuffins' application. The 'Build Log' tab is selected, displaying a successful build and deployment. The log output shows the build process, including pruning devDependencies, auditing packages, and discovering process types. The build succeeded, and the application was deployed to Heroku. The log output is: '-----> Pruning devDependencies up to date, audited 207 packages in 2s 32 packages are looking for funding run `npm fund` for details found 0 vulnerabilities -----> Build succeeded! -----> Discovering process types Profile declares types -> (none) Default types for buildpack -> web -----> Compressing... Done: 129.7M -----> Launching... Released v19 https://doc-mcstuffins.herokuapp.com/ deployed to Heroku Build finished'.

heroku.com | Blogs | Careers | Documentation | **Support** | Terms of Service | Privacy | Cookies | © 2021 Salesforce.com

## CD succeeding

# end-to-end testing

During this end-to-end testing we have encountered several issues including:

- Backend API was blocked by CORS policy due to a mistake of the specified URL
  - Resolved by fixing the URL of the API
- A minor mistake in HTTP methods for RESTful services
  - Resolved by double checking with the controllers in the backend file and matching them one by one
- When an already assigned group, we are not able to unassign it completely and needs to still be assigned to some group

- Unresolved problem that would be dealt in Sprint 2