HYKE Database - Backend Unit Testing

Submitted By: Abigayle Hickey

HYKE

100 Signal Hill Road
St. John's, NL, Canada
A1A 1B3

T 1-709-769-1986
hello@hykeup.com

April 30th, 2021
Claire Avery

Co-operative Education Office
Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, NL
A1B 3X5

Dear Ms. Avery,

During this work term (Engineering 004W) I was employed with HYKE as an undergraduate co-operative student under the software development team. This is my first work term with HYKE and I was supervised by Tim Mather.

HYKE is a local start-up designed to personalize consumer offers based on user patterns and desired price ranges. Consumers save more by using HYKE and are more likely to engage with HYKEs listed companies, creating a win-win for consumers and vendors. This company has allowed me to contribute my own designs and code to their front-end software, as well as aid in the completion of their front and backend tests.

The enclosed presentation package titled *HYKE Database – Backend Unit Testing* outlines the set up and testing process of a softwares database. This outline draws from my own experience and research from working on HYKEs database tests.

If there are any questions concerning this report, I would be pleased to discuss them further with you.

Sincerely,

Abigayle Hickey

HYKE Database - Backend Unit Testing

Work Term Four Presentation Package

HYKE

Submitted To:

Ms. Claire Avery

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

April 30th, 2021

Created By;

Abigayle Hickey

201732641

Table of Contents

Presentation Overview:

1.0 – Introduction

1.1– Who is HYKE

HYKE is a local start up, based through the Genesis Enterprise program, aiming to launch their product this spring. HYKE has developed an app tailored to finding what consumers are looking for at their desired price point, without the need for coupons or store points. HYKE offers a unique discount for the item the consumer currently wants, making their purchase more likely and desirable than random sales not tailored to the consumers wishes. HYKE is teamed with various companies, ranging from grocery, restaurant, and goods stores to allow consumers more options in their purchases [1].

Upon release, the company is also aiming to gain insight in the purchasing patterns of consumers. As consumers use the app more frequently, HYKE will offer suggested discounted items, based on what they think the consumer wants at a time they are likely to buy it. This app is a great way for consumers to seek offers that can truly aid them, by discounting items they habitually buy and would otherwise seek elsewhere at a higher price. As well, suppliers gain more customers through HYKEs offers and exposure, creating a great win-win for both consumers and suppliers [1].

1.2 – What is unit testing

Unit testing is an essential part of any software project as a way to improve code, fix bugs and ensure stability in the coding. Through unit testing, each component is individually tested to isolate errors to single components before the software is combined and run together.

Once projects launch and are in full use, bugs are an inevitable step. Unit testing allows these bugs to be maneuvered more easily, as the process for checking each component has already been implemented. Without unit testing, a simple error in one component can lead to every intertwined component to fail, causing server downtime. This would lead to both unhappy consumers and profit losses, which can be easily prevented by taking the correct precautions.

2.0– Backend Unit Testing

Backend unit testing is designed to ensure that the database, which holds all of the consumer and suppliers information, is working as expected. A testing database is used, which is simply an empty database which can then be manipulated accordingly for each test. In order to mock the real database, the testing database requires its own set up to fully replicate the software environment.

2.1– Setting up the database

To begin backend unit testing the first step is to create a utils file, which will hold set up functions for the testing database. Before each testing file, a unique testing database needs to be created. As tests are run in parallel, to ensure the database is not teared down in one file while simultaneously being worked on in another, this set up allows each file to generate their own unique database.

After each individual test of a file, the database is wiped of all its information. This is to ensure more accuracy in testing, as to prevent the database from being too convoluted with manipulated information by other tests. This ensures the initial state of every test thus the expected response is much easier to guarantee. Once all the tests are complete for a file, that respective database is then deleted and disconnected.

2.2– Manipulating the database

Once the database is set up, it should be filled with data to test against. This data can be filled using seeder files, which hold manual information for components, such as the information for a given user. These seeders can then be run inside a testing file, which will automatically run before each individual test, allowing every test to begin with the same seeded database.

The utils file, in addition to database setup, also holds functions that are used in manipulating database data. The databases information is only accessible to those registered within the database, and most functions require a login token to allow interaction with the database. This login token can be generated by calling the method to login and sending a correct email and password. This email and password must be in the database, using a user seeder, to receive the correct token. As this process is needed for every test, it can be simplified by a login function in the utils file, and then called for every test to receive a token.

2.3– Knowing what to test

Backend functions work by sending a request to the database and then awaiting a response. A request requires a method name and route, simply called a routing method, which combined create a function that can interact with the database. Depending on the routing method something can be retrieved from the database, in the form of a response body, or can be sent, as a request body, which would then update the database.

Each component has their own routes file, which holds all the possible routing methods they can perform. These are further implemented as functions in their respective controllers file, which are the basis of knowing what to test. Generally, if the function is

successful in interacting with the database, the response will have a status code of 200, whereas a failure is represented by a status code of 500. To fully test each routing method, both statuses should be checked, by testing both expected responses as well as planned failures.

3.0– Conclusion

Unit testing is an essential part of any software project as a way to eliminate errors early, gain stability in software code and reduce the risk of server downtime and money loss. Testing allows full exposure to each of the individual components, before combining all parts together, and uncovers errors or disadvantages that would have been hidden originally under the layers of software.

Backend unit testing requires a testing database, which must be set up before and after tests. Testing databases should be wiped clean after every test to ensure accuracy when testing and avoid a convoluted database. Testing databases must be manually filled in using a seeder, which is then used as a cross check with tests. Each routing method of a component is required to be tested, which is done using the request method and route and then cross checking the expected response with the given response body from the database.

4.0 - References

[1] "HYKE," Hyke Technologies Inc, 2021. [Online]. Available: https://hykeup.com/. [Accessed March 2021].