BurgerHub Assignment 3

PA1489

Group 19

Alem Zvirkic(alzv23), Amir Hindawi(amhi22),
Theodore Reangpusri(thre21), Tuva Rutberg(turu24)

Kom Igång

Vilka moduler? Vilka metoder? Vilka API-ändpunkter?

Modules:

- burgerorderer.js: This module manages burger-related database functions such as placing an order and fetching burger details.

- kitchenview.js: This module is responsible for displaying and retrieving all orders for kitchen management.

Methods:

- getBurger: This method fetches a specific burger by name from the database.
- placeOrder: Place a new order by inserting burger details, sides, and drink into the

database.
- getOrderId: Retrieves the latest order ID.
- showOrders: Fetches all orders placed.

API Endpoints:

- POST /place-order: This endpoint handles the submission of a burger order.
- GET /orders: This endpoint fetches all placed orders for display in the kitchen.

Hur skall de testas? Hur anropas de? Vad får ni för svar?

We decided to implement simple automated tests for both frontend and backend. Frontend tests consist of testing if buttons and navigation through the application works as intended and that the page renders correct information. Backend tests were created to check if access to the page works as intended and if the database saves orders after an order is placed.

Unit Testing (Jest):

- **Modules**: Tests functions such as getBurger, placeOrder, and showOrders by mocking database interactions with MySQL.
- **Methods**: Functions are called directly in the test files using Jest. For example, placeOrder is tested with mock data to simulate an order.
- **Expected Responses**: Mocked responses from the database such as inserted order IDs or fetched data are checked for accuracy of the application.

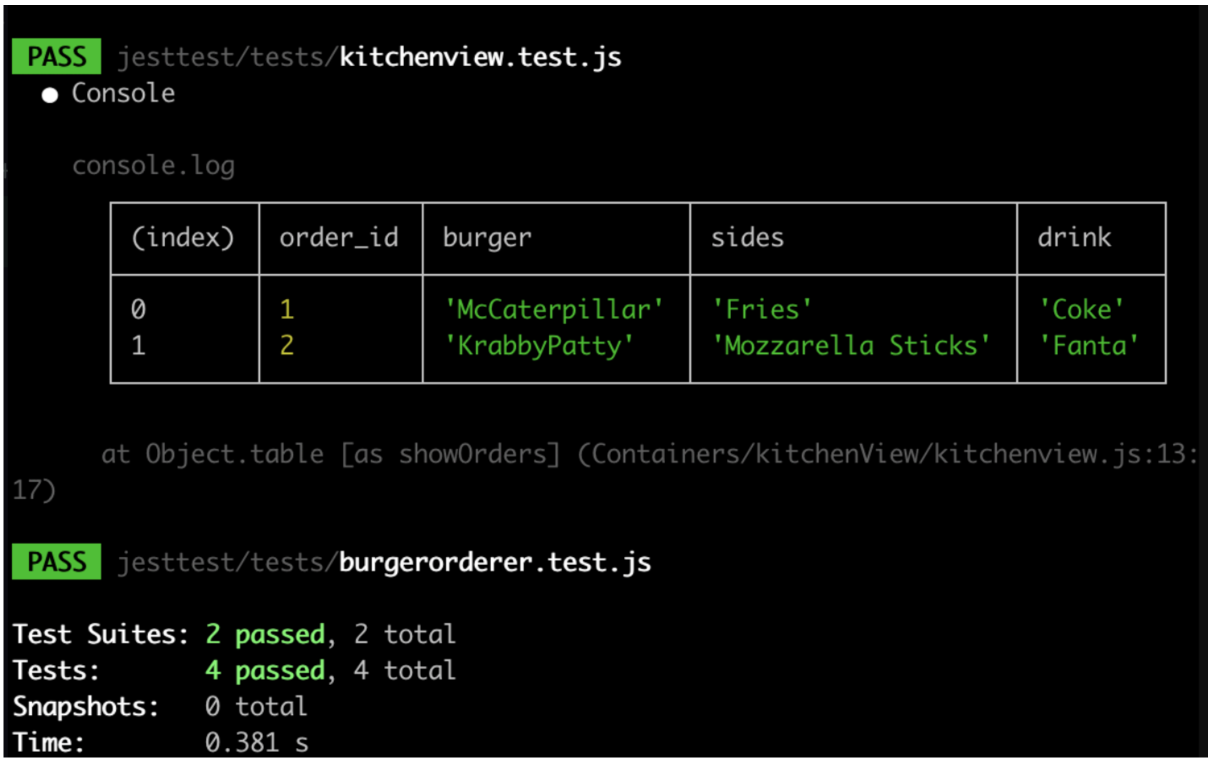
Cypress Testing:

- End-to-End Testing: Cypress tests ensure the UI flows, such as placing an order, work as expected. Each button click and form submission is tested to ensure proper redirection and successful order placement.

Under Arbetets Gång

Ideally we would have preferred to develop the application for this project following a test driven development plan (TDD) which in short means that no code should be written or changed without a test case. Using TDD is a way to ensure that the application delivered keeps a good standard. TDD can save time towards the end of a development process as it can help reduce the time spent debugging, reworking/remaking, and maintaining code by discovering errors early on. Unfortunately we were unable to work using TDD as we were short in time and needed to start implementing the application as soon as possible in order for us to deliver it by deadline. For future and similar projects we plan to focus more on tests and will suggest at least 1 member of the group to focus on tests early on in the project.

Genomför och Dokumentera en Debug-session
denna har jag ej gjort, vill ngn aspulla?



```
PASS  jesttest/tests/kitchenview.test.js
  ● Console

    console.log
      ┌─────────┬──────────┬─────────────────┬─────────────────────┬─────────┐
      │ (index) │ order_id │ burger          │ sides               │ drink   │
      ├─────────┼──────────┼─────────────────┼─────────────────────┼─────────┤
      │ 0       │ 1        │ 'McCaterpillar' │ 'Fries'             │ 'Coke'  │
      │ 1       │ 2        │ 'KrabbyPatty'   │ 'Mozzarella Sticks' │ 'Fanta' │
      └─────────┴──────────┴─────────────────┴─────────────────────┴─────────┘

      at Object.table [as showOrders] (Containers/kitchenView/kitchenview.js:13:
17)

PASS  jesttest/tests/burgerorderer.test.js

Test Suites: 2 passed, 2 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.381 s
```