Team [0] Test Report

For our tests we used flasks recommendation which is using pytest. With pytest we created tests that would test our api functionality. We made sure our api calls succeeded when given proper data to succeed and failed when given improper data. The tests we created had a total coverage of 83% when running a `coverage pytest`.

In conclusion, the tests showed us a decent amount of insecurities in our code. We did multiple fixes and feel even more confident in our product. Below is an overall summary of each of our tests. They are separated into 4 different sections based on the 4 different files they tested.

Authentication (87% Coverage)

Test: test_login_valid_user
- Objective: Validate successful login of an existing user.
- Expected Outcome: Expects a successful response (HTTP status code 200) with user details.

Test: test_login_incorrect_username
- Objective: Test login with a non-existing username.
- Expected Outcome: Expects an error response indicating an incorrect username.

Test: test_login_incorrect_password
- Objective: Test login with an incorrect password for an existing user.
- Expected Outcome: Expects an error response indicating an incorrect password.

Test: test_login_missing_credentials
- Objective: Test login without providing any credentials.
- Expected Outcome: Expects an error response indicating missing credentials.

Test: test_register_valid_user
- Objective: Validate successful registration of a new user.
- Expected Outcome: Expects a successful response (HTTP status code 200) with a success message and the user's ID.

Test: test_register_existing_user
- Objective: Test registration with an existing username.
- Expected Outcome: Expects an error response indicating that the username is already taken.

Test: test_register_missing_data
- Objective: Test registration without providing necessary data.
- Expected Outcome: Expects an error response indicating missing data required for registration.

Customer (87% Coverage)

Test: test_menu

- Objective: Verify that the API returns the menu items available for customers.
- Expected Outcome:
  - Successful response (HTTP status code 200).
  - Expects the menu to contain three sections: cones, toppings, and ice cream, each with three items.

Test: test_checkout_post_get

- Objective: Test the checkout functionality by creating an order and retrieving order information.
- Expected Outcome:
  - Successful response (HTTP status code 200) for both the POST and GET requests related to checkout.
  - Expects the creation of a new order with a specific structure and presence of a full_order_id.
  - Expects retrieval of ordered cone information in the GET request.

Test: test_history

- Objective: Check if the API returns the order history for a specific customer.
- Expected Outcome:
  - Successful response (HTTP status code 200).
  - Expects retrieval of order history in the response data.

Test: test_account_get_put

- Objective: Test the functionality to update and retrieve customer account information.
- Expected Outcome:
  - Successful response (HTTP status code 200) for both PUT and GET requests related to account management.
  - Expects successful update of the customer's username and activation status.
  - Expects retrieval of updated customer information in the GET request.

Test: test_account_errors_get_put

- Objective: Test the errors when trying to get customer account information.
- Expected Outcome:
  - Successful response (HTTP status code 200) for GET requests related to account management.
  - Expects an error of the user does not exist.

Employee (62% Coverage)

Test: test_drones:

- Objective: Verify that the API returns the list of drones associated with an employee.

- Expected Outcome: Expects a successful response (HTTP status code 200) and a list of drones with a length of 3.

Test: test_no_drones_registered
- Objective: Validate the behavior when requesting drones for an employee with no registered drones.
- Expected Outcome: Expects a successful response (HTTP status code 200) with an error message indicating no registered drones for the employee.

Test: test_no_orders_earnings
- Objective: Validate the behavior when requesting earnings for an employee with no orders.
- Expected Outcome: Expects a successful response (HTTP status code 200) with an error message indicating no orders for the employee.

Test: test_earnings
- Objective: Validate the response for an employee's earnings.
- Expected Outcome: Expects a successful response (HTTP status code 200) with an error message in the earnings data.

Test: test_invalid_drone_data
- Objective: Test handling incomplete data for creating a drone.
- Expected Outcome: Expects a successful response (HTTP status code 200) and an error message indicating incomplete drone data.

Test: test_post_put_delete_drone
- Objective: Test adding, updating, and deleting drones.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200) when adding a drone.
    - Expects a successful response (HTTP status code 200) when getting all drones and the length of drones to increase by 1.
    - Expects a successful response (HTTP status code 200) when updating a drone.
    - Expects a successful response (HTTP status code 200) when deleting a drone.

Test: test_orders_history
- Objective: Request order history for an employee.
- Expected Outcome: Expects a successful response (HTTP status code 200) with a structure and content matching the historical orders for the employee.

Manager (74% Coverage)

Test: test_orders
- Objective: Test the /manager/orders endpoint.

- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects the presence of "orders" in the response data.

Test: test_users
- Objective: Test the /manager/users endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects the response data to be a list.

Test: test_product_get
- Objective: Test the /manager/product GET endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects the presence of "product" in the response data.

Test: test_product_errors_get
- Objective: Test error scenarios for the /manager/product GET endpoint.
- Expected Outcome:
    - Expects error handling when no ID is provided.
    - Expects an error response when an ID for a non-existing product is provided.

Test: test_product_put
- Objective: Test the /manager/product PUT endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects confirmation of the updated product in the response data.

Test: test_product_errors_put
- Objective: Test error scenarios for the /manager/product PUT endpoint.
- Expected Outcome:
    - Expects error handling when no ID is provided.
    - Expects error handling when incorrect data types are passed.

Test: test_product_post
- Objective: Test the /manager/product POST endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects a "success" message in the response data.

Test: test_product_errors_post
- Objective: Test error scenarios for the /manager/product POST endpoint.
- Expected Outcome:
    - Expects error handling when incomplete data is provided.

    - Expects error handling when incorrect data types are passed.

Test: test_product_delete
- Objective: Test the /manager/product DELETE endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects confirmation of the deleted product in the response data.

Test: test_product_errors_delete
- Objective: Test error scenarios for the /manager/product DELETE endpoint.
- Expected Outcome:
    - Expects error handling when no data is provided.

Test: test_user_get
- Objective: Test the /manager/user GET endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects the presence of "product" in the response data.

Test: test_user_errors_get
- Objective: Test error scenarios for the /manager/user GET endpoint.
- Expected Outcome:
    - Expects error handling when no ID or incorrect ID format is provided.
    - Expects error handling when an ID for a non-existing user is provided.

Test: test_user_put
- Objective: Test the /manager/user PUT endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects confirmation of the updated user in the response data.

Test: test_user_errors_put
- Objective: Test error scenarios for the /manager/user PUT endpoint.
- Expected Outcome:
    - Expects error handling when no ID or incorrect ID format is provided.
    - Expects error handling when an ID for a non-existing user is provided.

Test: test_history
- Objective: Test the /manager/history endpoint.
- Expected Outcome:
    - Expects a successful response (HTTP status code 200).
    - Expects the presence of "orders_history" in the response data.