

Example M5: Testing and Code Review

1. Change History

Change Date	Modified Sections	Rationale
March 30, 2025	3.2	Updated required response time to 3s to match test implementation
March 30, 2025	2.3 & 2.4	Updated screenshots of coverage to reflect changes in implementation
March 30, 2025	5.1 - 5.3	Updated screenshots of codacy and commit hash where run to reflect changes in implementation
March 30, 2025	2.1	Updated links of backend tests and commit hash where run to reflect changes in implementation
March 30, 2025	3.1	Updated links of non-functional tests to reflect changes in implementation
April 2, 2025	3.2	Update description to mention 2 second response time requirement, update description with justification of 2 seconds and source, and update test logs to match new implementation
April 2, 2025	5.*	Update codacy screenshots and commit hash to account for code updates
April 2, 2025	2.1.3	Add database setup instructions based on feedback from M5
April 3, 2025	3.1	Update frontend non-functional test location to reflect new file
April 3, 2025	3.2	Update frontend usability test log for new test
April 3, 2025	4.2	Update frontend functional test logs with most recent run
April 3, 2025	5.*	Update codacy screenshots and commit hash to account for code updates

2. Back-end Test Specification: APIs

2.1. Locations of Back-end Tests and Instructions to Run Them

2.1.1. Tests

Interface	Describe Group Location, No Mocks	Describe Group Location, With Mocks
POST /discounts	backend/___tests___/no_mock/NoMockDiscount.test.ts#L13	backend/___tests___/with_mock/WithMock
GET /discounts/:id	backend/___tests___/no_mock/NoMockDiscount.test.ts#L172	backend/___tests___/with_mock/WithMock
GET /discounts	backend/___tests___/no_mock/NoMockDiscount.test.ts#L252	backend/___tests___/with_mock/WithMock
DELETE /discounts/:id	backend/___tests___/no_mock/NoMockDiscount.test.ts#L341	backend/___tests___/with_mock/WithMock
POST /notifications	backend/___tests___/no_mock/NoMockNotification.test.ts#L10	backend/___tests___/with_mock/WithMock
DELETE /notifications/:id	backend/___tests___/no_mock/NoMockNotification.test.ts#L107	backend/___tests___/with_mock/WithMock
POST /preferences/allergies	backend/___tests___/no_mock/NoMockPreference.test.ts#L10	backend/___tests___/with_mock/WithMock
GET /preferences/allergies/:id	backend/___tests___/no_mock/NoMockPreference.test.ts#L74	backend/___tests___/with_mock/WithMock
DELETE /preferences/allergies/:id/:allergy	backend/___tests___/no_mock/NoMockPreference.test.ts#L114	backend/___tests___/with_mock/WithMock
POST /recipes	backend/___tests___/no_mock/NoMockRecipe.test.ts#L171	backend/___tests___/with_mock/WithMock
GET /recipes/:id	backend/___tests___/no_mock/NoMockRecipe.test.ts#L423	backend/___tests___/with_mock/WithMock

Interface	Describe Group Location, No Mocks	Describe Group Location, With Mocks
DELETE /recipes/:id	backend/___tests___/no_mock/NoMockRecipe.test.ts#L495	backend/___tests___/with_mock/WithMock
POST /routes	backend/___tests___/no_mock/NoMockRoute.test.ts#L52	backend/___tests___/with_mock/WithMock
GET /routes/:id	backend/___tests___/no_mock/NoMockRoute.test.ts#L341	backend/___tests___/with_mock/WithMock
DELETE /routes/:id	backend/___tests___/no_mock/NoMockRoute.test.ts#L402	backend/___tests___/with_mock/WithMock
GET /users/:userID/routes	backend/___tests___/no_mock/NoMockUser.test.ts#L11	backend/___tests___/with_mock/WithMock

2.1.2. Commit Hash Where Tests Run

252aa8c8160a75e9a6a253ed465435ff039d71bc

2.1.3. Explanation on How to Run the Tests

1. Clone the Repository:
- Open your terminal and run:

```
git clone https://github.com/CPEN321-FoodTrip/FoodTrip.git
```
2. Navigate to the backend directory:
- In the terminal, run:

```
cd backend/
```
3. Install the required dependencies:
- In the terminal, run:

```
npm ci
```
4. Setup a .env in the backend directory:
- In the terminal, run:

```
touch .env
```

◦ With the editor of your choice, add the following environment variables:

```
EDAMAM_APP_ID=
EDAMAM_API_KEY=
FIREBASE_SERVICE_ACCOUNT_KEY=
(Optional, only for performance test) GATEWAY_BASE_URL=
```
5. Setup database for testing:
- The backend tests use an in-memory MongoDB server for testing. No actual database setup is required. However, ensure that your environment is set up to support it:

▪ The tests use the `mongodb-memory-server` package to spin up a temporary in-memory MongoDB server for each test run.

▪ The in-memory MongoDB server is automatically initialized and torn down before and after the tests. Ensure that MongoDB is not running on your system as the in-memory server will use available resources dynamically.
- Note: If you want to change the default MongoDB database names for testing, you can modify the `testDBs` array in the jest setup file.
6. Run the tests with or without coverage:
- To run the tests **with** coverage, use the following command in the terminal:

```
npm run test:coverage
```

- To run the tests **without** coverage, use the following command in the terminal:

```
npm run test
```

7. [Optional] Run only the mocked or unmocked tests:

- To run the tests **with** mocks, use the following command in the terminal:

```
npm run test __tests__/with_mock/
```

- To run the tests **without** mocks, use the following command in the terminal:

```
npm run test __tests__/no_mock/
```

8. View the full coverage repo:

- Within the backend directory navigate to the `coverage/lcov-report` directory.
- Open `index.html` in the browser.

2.2. GitHub Actions Configuration Location

`~/.github/workflows/test-backend.yml`

2.3. Jest Coverage Report Screenshots With Mocks

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	96.86	92.85	94.04	96.64	
backend	71.42	58.82	54.54	71.42	
index.ts	70.45	83.33	44.44	70.45	43-66, 71
jest.config.ts	0	100	100	0	3-22
services.ts	90	45.45	100	90	10
backend/controllers	100	100	100	100	
DiscountController.ts	100	100	100	100	
NotificationController.ts	100	100	100	100	
PreferenceController.ts	100	100	100	100	
RecipeController.ts	100	100	100	100	
RouteController.ts	100	100	100	100	
UserController.ts	100	100	100	100	
backend/helpers	100	100	100	100	
DiscountHelper.ts	100	100	100	100	
NotificationHelper.ts	100	100	100	100	
PreferenceHelper.ts	100	100	100	100	
RecipeHelper.ts	100	100	100	100	
RouteHelpers.ts	100	100	100	100	
UserHelper.ts	100	100	100	100	
backend/routes	100	100	100	100	
DiscountRoutes.ts	100	100	100	100	
NotificationRoutes.ts	100	100	100	100	
PreferenceRoutes.ts	100	100	100	100	
RecipesRoutes.ts	100	100	100	100	
RouteRoutes.ts	100	100	100	100	
UserRoutes.ts	100	100	100	100	
Test Suites: 6 passed, 6 total					
Tests: 86 passed, 86 total					
Snapshots: 0 total					
Time: 29.292 s					
Ran all test suites matching /__tests__\/with_mock/i.					

The `index.ts` file does not have 100% coverage because the server does not start or open a port during tests, and a different MongoDB client is used. As a result, the `startServer()` function is not executed.

```

40 133x      try {
41 133x        await route.action(req, res, next);
42      } catch (err) {
43        console.error(err);
44        res.status(500);
45      }
46    }
47  );
48  });
49
50  function startServer() {
51    initializeClient();
52    client
53      .connect()
54      .then(async () => {
55        console.debug("Connected to MongoDB");
56
57        initializeFirebaseAdmin();
58        await initializeGeoNamesDatabase();
59
60        app.listen(process.env.PORT, () => {
61          console.debug(`Server is running on port ${process.env.PORT}`);
62        });
63      })
64      .catch((err: Error) => {
65        console.error(err);
66        client.close();
67      });
68  }
69
70 12x  I if (process.env.NODE_ENV !== "test") {
71    startServer();
72  }

```

The reason the `jest.config.ts` file does not have any coverage is simply because it is for configuring our test environment and does not contribute to the functionality of the app.

The `services.ts` file has no coverage because, during testing, an in-memory MongoDB client is injected. As a result, one branch of an if statement, responsible for assigning the production client, is never executed.

```

6 5x  export function initializeClient(customClient?: MongoClient) {
7 5x    if (customClient) {
8 5x      client = customClient;
9      } else {
10     client = new MongoClient(process.env.DB_URI ?? "mongodb://localhost:27017");
11   }
12 }

```

2.4. Jest Coverage Report Screenshots Without Mocks

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	92.74	84.69	94.04	92.24	
backend	71.42	58.82	54.54	71.42	
index.ts	70.45	83.33	44.44	70.45	43-66,71
jest.config.ts	0	100	100	0	3-22
services.ts	90	45.45	100	90	10
backend/controllers	92.45	100	100	91.75	
DiscountController.ts	92.45	100	100	91.83	59,84,106,132
NotificationController.ts	90.9	100	100	90	28,48
PreferenceController.ts	90	100	100	88.46	22,43,63
RecipeController.ts	93.61	100	100	93.18	72-73,96
RouteController.ts	93.87	100	100	93.47	82,108,134
UserController.ts	90.9	100	100	88.88	18
backend/helpers	97.77	80.95	100	97.61	
DiscountHelper.ts	100	100	100	100	
NotificationHelper.ts	100	100	100	100	
PreferenceHelper.ts	100	100	100	100	
RecipeHelper.ts	94.23	75	100	94.11	27,39-40
RouteHelpers.ts	98.16	83.33	100	98.05	113,124
UserHelper.ts	100	100	100	100	
backend/routes	100	100	100	100	
DiscountRoutes.ts	100	100	100	100	
NotificationRoutes.ts	100	100	100	100	
PreferenceRoutes.ts	100	100	100	100	
RecipesRoutes.ts	100	100	100	100	
RouteRoutes.ts	100	100	100	100	
UserRoutes.ts	100	100	100	100	
Test Suites: 6 passed, 6 total					
Tests: 61 passed, 61 total					
Snapshots: 0 total					
Time: 36.362 s					
Ran all test suites matching /__tests__\/no_mock/i.					

The reason for not having 100% coverage on files in the `backend/controllers` and `backend/helpers` directories is because of internal error handling related to database and external api failures which cannot be tested without mocks. Typically, they are handled within try/catch blocks that call `next()` to pass the error to a central error handler or the helper function which is then caught in the controller.

The `index.ts`, `jest.config.ts` and `services.ts` files do not have 100% coverage due to the exact same reasons as the mocked tests.

3. Back-end Test Specification: Tests of Non-Functional Requirements

3.1. Test Locations in Git

Non-Functional Requirement	Location in Git
Performance (Response Time)	FoodTrip/backend/__tests__/non_functional_requirements/performance.test.ts
Usability (Clicks to navigate)	FoodTrip/frontend/app/src/androidTest/java/com/example/FoodTripFrontend/Non-FunctionalTests.kt

3.2. Test Verification and Logs

- **Performance (Response Time)**
 - **Verification:** This test suite evaluates the performance of critical API endpoints in an unmocked environment, simulating real-world user interactions. It measures execution times for creating and deleting routes, recipes, discounts, notifications, and allergy preferences, ensuring each operation completes within 2 seconds. This is important for maintaining a smooth user experience, preventing delays, and ensuring the system can handle expected traffic. By logging execution times and validating responses, the tests help identify performance issues and ensure the system is quick enough. 2 seconds was selected as the ideal time based on this report (<https://odown.com/blog/what-is-a-good-api-response-time/>) which states that for web applications you strive for response times under 2 seconds and for mobile applications, aim for 1-3 seconds.
 - **Log Output**

```
> backend@1.0.0 test
> NODE_ENV=test jest --testPathPattern=(/non_functional_requirements/)

console.debug
  Route Execution time: 1865ms
```

```
    at __tests__/non_functional_requirements/performance.test.ts:29:15
console.debug
  Recipe Execution time: 864ms
    at __tests__/non_functional_requirements/performance.test.ts:52:15
console.debug
  Route Teardown Execution time: 62ms
    at __tests__/non_functional_requirements/performance.test.ts:76:15
console.debug
  Recipe Teardown Execution time: 85ms
    at __tests__/non_functional_requirements/performance.test.ts:96:15
console.debug
  Add Discount Execution time: 301ms
    at __tests__/non_functional_requirements/performance.test.ts:125:13
console.debug
  Get Discount Execution time: 88ms
    at __tests__/non_functional_requirements/performance.test.ts:144:13
console.debug
  Delete Discount Execution time: 85ms
    at __tests__/non_functional_requirements/performance.test.ts:165:13
console.debug
  Add 10 Discounts Average Execution time: 191.5ms
    at __tests__/non_functional_requirements/performance.test.ts:205:13
console.debug
  Get All Discounts Execution time: 76ms
    at __tests__/non_functional_requirements/performance.test.ts:228:13
console.debug
  Delete 10 Discounts Average Execution time: 52.4ms
    at __tests__/non_functional_requirements/performance.test.ts:260:13
console.debug
  Add Notification Execution time: 34ms
    at __tests__/non_functional_requirements/performance.test.ts:287:13
console.debug
  Delete Notification Execution time: 31ms
    at __tests__/non_functional_requirements/performance.test.ts:313:13
console.debug
  Add 10 Notifications Average Execution time: 45.6ms
    at __tests__/non_functional_requirements/performance.test.ts:348:13
console.debug
  Delete 10 Notifications Average Execution time: 41.5ms
    at __tests__/non_functional_requirements/performance.test.ts:383:13
console.debug
  Add Allergy Execution time: 85ms
    at __tests__/non_functional_requirements/performance.test.ts:412:13
console.debug
  Get Allergy Execution time: 33ms
    at __tests__/non_functional_requirements/performance.test.ts:431:13
console.debug
  Delete Allergy Execution time: 68ms
```

```
at __tests__/non_functional_requirements/performance.test.ts:456:13

PASS __tests__/non_functional_requirements/performance.test.ts (11.024 s)
Performance test
  ✓ Single route, 3 stops (2911 ms)
  ✓ Single discount (482 ms)
  ✓ 10 discount (2527 ms)
  ✓ Single notification (70 ms)
  ✓ 10 notification (878 ms)
  ✓ Single allergy (192 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        11.064 s
Ran all test suites matching /(\non_functional_requirements\)/i.
```

• Usability (Clicks to Navigate)

- **Verification:** This test suite simulates using the frontend app along with Espresso to mimic a user's behavior. The test is done by counting the number of valid click in a use case test. At the end of the test, it checks the number of clicks performed in the test. If the number of clicks is within 3 clicks, it passes, and vice versa.

◦ Log Output

Tests	Duration	Medium_Phone_API_31
✓ Test Results	32 s	5/5
✓ MainActivityTest	32 s	5/5
✓ checkAccount	14 s	✓
✓ checkViewRecipe	6 s	✓
✓ checkManageTrip	4 s	✓
✓ checkPastTrips	4 s	✓
✓ signOut	2 s	✓

> Task :app:connectedDebugAndroidTest

Starting 5 tests on Medium_Phone_API_31(AVD) - 12

Medium_Phone_API_31(AVD) - 12 Tests 0/5 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 1/5 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 3/5 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 5/5 completed. (0 skipped) (0 failed)

Finished 5 tests on Medium_Phone_API_31(AVD) - 12

BUILD SUCCESSFUL in 1m 4s

69 actionable tasks: 2 executed, 67 up-to-date

Build Analyzer results available

Tests	Duration	Medium_Phone_API_31
✓ Test Results	24 s	1/1
✓ TripActivityTest	24 s	1/1
✓ planRegularTripShort	24 s	✓

> Task :app:connectedDebugAndroidTest

Starting 1 tests on Medium_Phone_API_31(AVD) - 12

Medium_Phone_API_31(AVD) - 12 Tests 0/1 completed. (0 skipped) (0 failed)

Finished 1 tests on Medium_Phone_API_31(AVD) - 12

BUILD SUCCESSFUL in 48s

69 actionable tasks: 2 executed, 67 up-to-date

Build Analyzer results available

✓ Test Results	36 s	2/2
✓ PastTripActivityTestPersonTest	36 s	2/2
✓ GeneralRecipeViewPastTrip	22 s	✓
✓ backButton	13 s	✓

> Task :app:connectedDebugAndroidTest

Starting 2 tests on Medium_Phone_API_31(AVD) - 12

Medium_Phone_API_31(AVD) - 12 Tests 0/2 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 1/2 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 2/2 completed. (0 skipped) (0 failed)

Finished 2 tests on Medium_Phone_API_31(AVD) - 12

BUILD SUCCESSFUL in 1m 13s

69 actionable tasks: 2 executed, 67 up-to-date

Build Analyzer results available

Tests	Duration	Medium_Phone_API_31
✓ Test Results	34 s	2/2
✓ GroceryStoreActivityTest	34 s	2/2
✓ postAndDeleteDiscountTest	16 s	✓
✓ changeSelectedTest	17 s	✓

> Task :app:connectedDebugAndroidTest

Starting 2 tests on Medium_Phone_API_31(AVD) - 12

Medium_Phone_API_31(AVD) - 12 Tests 0/2 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 1/2 completed. (0 skipped) (0 failed)

Finished 2 tests on Medium_Phone_API_31(AVD) - 12

BUILD SUCCESSFUL in 59s

69 actionable tasks: 2 executed, 67 up-to-date

Build Analyzer results available

4. Front-end Test Specification

4.1. Location in Git of Front-end Test Suite:

`frontend/app/src/androidTest/java/com/example/FoodTripFrontend/ExampleInstrumentedTest.kt`

4.2. Tests

• Use Case: View Past Trips Main Success Scenario:

1. The user opens "Past Trip" screen.

- 2. The app shows a list of "past trip" text view, and a "Back" button.
- 3. The user clicks on any past trip.
- 4. A window pops up
- 5. The window shows the starting location, intermediate stops, destination, recipes for each stop, and a "Show Route" button.
- 6. The user presses the "Show Route" button
- 7. The user clicks on the first recipe

Extensions:

- 6a. The user is directed to the main page and a map of the route is displayed
- 7a. The window shows the details of the recipe and a recipe url
 - 7a1. The user clicks the recipe url.
 - 7a2. A webView shows up.

Failure Scenarios:

- 2a. The user has no past trip record
 - 2a1. The app shows no items in the list
- 2b. No internet connection
 - 2b1. The app displays an error message: "No internet connection"

Expected Behaviors:

Scenario Steps	Test Case Steps
1. User opens the view past trip.	Open Past Trip screen.
2. The app shows a list of past trip and a "Back" button.	Check the list is present on screen. Check the button is present on screen.
2a. The user has no past trip record	/
2a1. The app shows no items in the list	Check no items are present in the list on screen
3. The user click on any past trip.	Click any past trip text view
4. A window pops up.	Check the window activity is present on screen.
5. The window shows the starting location, intermediate stops, destination, recipes for each stop, and a "Show Route" button	Check first recipe is present on screen.
6. The user presses the show route button	Click on the show route button
6a. The user is directed to the main page and a map of the route is displayed	Check that the activity has switched to the Main Activity screen and check that the map is now displayed and visible on the screen
7. The user clicks on the first recipe	Click on the first recipe
7a. The window shows the details of the recipe and a recipe url	Check that the recipe text is displayed
7a1. The user clicks the recipe url	Click on the url text view
7a2. A webView of the recipe shows up	Check that the webView is now displayed

Test Logs:

Tests	Duration	Medium_Phone_API31	
✓ Test Results	37 s	3/3	> Task :app:connectedDebugAndroidTest Starting 3 tests on Medium_Phone_API_31(AVD) - 12
✓ PastTripActivityTestPersonTestFunctional	37 s	3/3	Medium_Phone_API_31(AVD) - 12 Tests 1/3 completed. (0 skipped) (0 failed)
✓ checkElements	5 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 2/3 completed. (0 skipped) (0 failed)
✓ GeneralRecipeViewPastTrip	18 s	✓	Finished 3 tests on Medium_Phone_API_31(AVD) - 12
✓ backButton	13 s	✓	BUILD SUCCESSFUL in 1m 4s 69 actionable tasks: 2 executed, 67 up-to-date Build Analyzer results available
✓ Test Results	24 s	3/3	> Task :app:connectedDebugAndroidTest Starting 3 tests on Medium_Phone_API_31(AVD) - 12
✓ PastTripActivityEmptyTestFunctional	24 s	3/3	Medium_Phone_API_31(AVD) - 12 Tests 1/3 completed. (0 skipped) (0 failed)
✓ checkElements	7 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 2/3 completed. (0 skipped) (0 failed)
✓ emptyPastTrip	3 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 3/3 completed. (0 skipped) (0 failed)
✓ backButton	14 s	✓	Finished 3 tests on Medium_Phone_API_31(AVD) - 12
			BUILD SUCCESSFUL in 53s 69 actionable tasks: 2 executed, 67 up-to-date Build Analyzer results available

• Use Case: Manage Trip

Main Success Scenario:

- 1. User Opens the Manage Trip screen
- 2. The app shows three text input fields and a Create Trip button

- 3. The user enters a starting city, ending city, and the number of desired stops, then presses the Create Trip button
- 4. The app opens the Main Screen
- 5. A google map is displayed showing the created route from the start to end city

Failure Scenarios:

- 3a. The user enters an invalid start/end city or an invalid number of stops and attempts to create a trip
 - 3a1. The app displays a pop-up saying that the associated field is invalid
- 3b. The user doesn't enter a input into any one of the text inputs and attempts to create a trip
 - 3b1. The app displays a pop-up saying that the associated field is missing
- 3c. The user enters the same city in the start and end fields and attempts to create a trip
 - 3c1. The app displays a pop-up saying that there can't be the same start and end city

Expected Behaviors:

Scenario Steps	Test Case Steps
1. User Opens the Manage Trip screen	Open Manage Trip screen
2. The app shows three text input fields and a Create Trip button	Check that there are three text input fields labeled Start City, End City, and Number of Stops and check that there is button with text Create Trip
3. The user enters a starting city, ending city, and the number of desired stops, then presses the Create Trip button	Input "Calgary" into the start city text field, "New York" into the end city text field, and "3" into the number of stops, then press the Create Trip button
3a. The user enters an invalid start/end city or an invalid number of stops and attempts to create a trip	Input adjfiolej as the start/end city or enter 0 as the number of stops, then click Create Trip button
3a1. The app displays a pop-up saying that the associated field is invalid	Check that a textfield is displayed with the text "Invalid Start City", "Invalid End City", or "Invalid Number of Stops"
3b. The user doesn't enter a input into any one of the text inputs and attempts to create a trip	Input a correct value (A real city or a number greater than or equal to 1) into two of the three text input fields and then click Create Trip Button
3b1. The app displays a pop-up saying that the associated field is missing	Check that a textfield is displayed with the text "Missing Start City", "Missing End City", or "Missing Number of Stops"
3c. The user enters the same city in the start and end fields and attempts to create a trip	Input "New York" into both the start and end city text fields and input 3 into the number of stop field. Press the Create Trip button
3c1. The app displays a pop-up saying that there can't be the same start and end city	Check that a textfield is displayed with the text "Same Start and End City"
4. The app opens the Main Screen	Check that the activity has switched to the Main Activity screen
5. A google map is displayed showing the created route from the start to end city	Check that the map is displayed and visible on the screen

Test Logs:

✓ Test Results	1m 15 s	8/8	> Task :app:connectedDebugAndroidTest Starting 8 tests on Medium_Phone_API_31(AVD) - 12
✓ TripActivityTestFunctional	1m 15 s	8/8	
✓ wrongEnd	13 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 0/8 completed. (0 skipped) (0 failed)
✓ planRegularTripShort	17 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 1/8 completed. (0 skipped) (0 failed)
✓ sameStartEnd	8 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 2/8 completed. (0 skipped) (0 failed)
✓ missingInputsEnd	7 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 4/8 completed. (0 skipped) (0 failed)
✓ wrongStart	8 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 5/8 completed. (0 skipped) (0 failed)
✓ missingInputsStart	6 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 6/8 completed. (0 skipped) (0 failed)
✓ missingInputsStops	6 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 7/8 completed. (0 skipped) (0 failed)
✓ wrongStopsAmount	7 s	✓	Medium_Phone_API_31(AVD) - 12 Tests 7/8 completed. (0 skipped) (0 failed)
			Finished 8 tests on Medium_Phone_API_31(AVD) - 12
			BUILD SUCCESSFUL in 1m 49s
			69 actionable tasks: 2 executed, 67 up-to-date
			Build Analyzer results available

Use Case: Manage Discounts Main Success Scenario:

- 1. The admin opens "Grocery Store" screen.
- 2. The app shows a list of discounts, an ingredient input text field, a price input text field, a "Delete" button, and a "Post" button.
- 3. The admin inputs a new ingredient and the price.
- 4. The admin presses the "Post" button.
- 5. The screen refreshes and the new discount is added into the list.

Extensions:

- 3a. The admin presses on a discount in the list.
 - 3a1. The admin presses the "Delete" button.
 - 3a2. The screen refreshes and the selected discount is removed.

Failure Scenarios

- 3ai. The admin doesn't select a discount to delete.

- 3ai(1). The app display an error message prompting the admin to select the discount to be deleted
- 3b. The admin doesn't enter an input into any one of the text input fields and attempts to post a discount.
 - 3b1. The app displays an error message prompting the admin for the valid input.
- 3c. The admin enter 0 as the price and attempts to post a discount.
 - 3c1. The app displays an error message prompting the admin for the valid input.

Expected Behaviors:

Scenario Steps	Test Case Steps
1. Admin opens "Grocery Store" screen.	Open Grocery Store screen.
2. The app shows a list of discounts, an ingredient input text field, a price input text field, a "Delete" button, and a "Post" button.	Check the list is present on screen. Check the ingredient input text view is present on screen. Check the price input text view is present on screen. Check the delete button is present on screen. Check the post button is present on screen.
3. Admin inputs a new ingredient and price	Input "snack" for the ingredient. Input "10" for the price
3a. Admin presses on a discount in the list.	Press a discount on the list.
3ai. Admin doesn't select a discount to delete	Press button labelled "Delete"
3ai(1). The app display an error message prompting the admin to select the discount to be deleted.	Check dialog is opened with text: "Please select discount to be deleted"
3b. Admin doesn't enter an input into any one of the text input fields and attempts to post a discount.	Input "snack" for ingredient or input "10" for price. Press the "Post" button"
3b1.The app displays an error message prompting the admin for a valid input.	Check dialog is opened with text: "Please enter valid ingredient and price"
3c. Admin input enter 0 as the price and attempts to post a discount.	Input "snack" for ingredient Input "0" for price Press the "Post" button
3c1. The app displays an error message prompting the admin for a valid input.	Check dialog is opened with text: "Please enter valid ingredient and price"
4. Admin presses the "Post" button	Press the button labelled "Post"
5. The screen refreshes and the new discount is added into the list	Check textView with text "snack: \$10" is present on screen.

Test Logs:

Test Results

1m 20s

7/7

GroceryStoreActivityTestFunctional

1m 20s

7/7

checkElements

5s

✓

emptyIngredientTest

6s

✓

postAndDeleteDiscountTest

13s

✓

zeroPriceTest

7s

✓

emptyPriceTest

5s

✓

deleteTest

21s

✓

changeSelectedTest

19s

✓

> Task :app:connectedDebugAndroidTest

Starting 7 tests on Medium_Phone_API_31(AVD) - 12

Medium_Phone_API_31(AVD) - 12 Tests 1/7 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 2/7 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 3/7 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 4/7 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 5/7 completed. (0 skipped) (0 failed)

Medium_Phone_API_31(AVD) - 12 Tests 6/7 completed. (0 skipped) (0 failed)

Finished 7 tests on Medium_Phone_API_31(AVD) - 12

BUILD SUCCESSFUL in 1m 47s

69 actionable tasks: 2 executed, 67 up-to-date

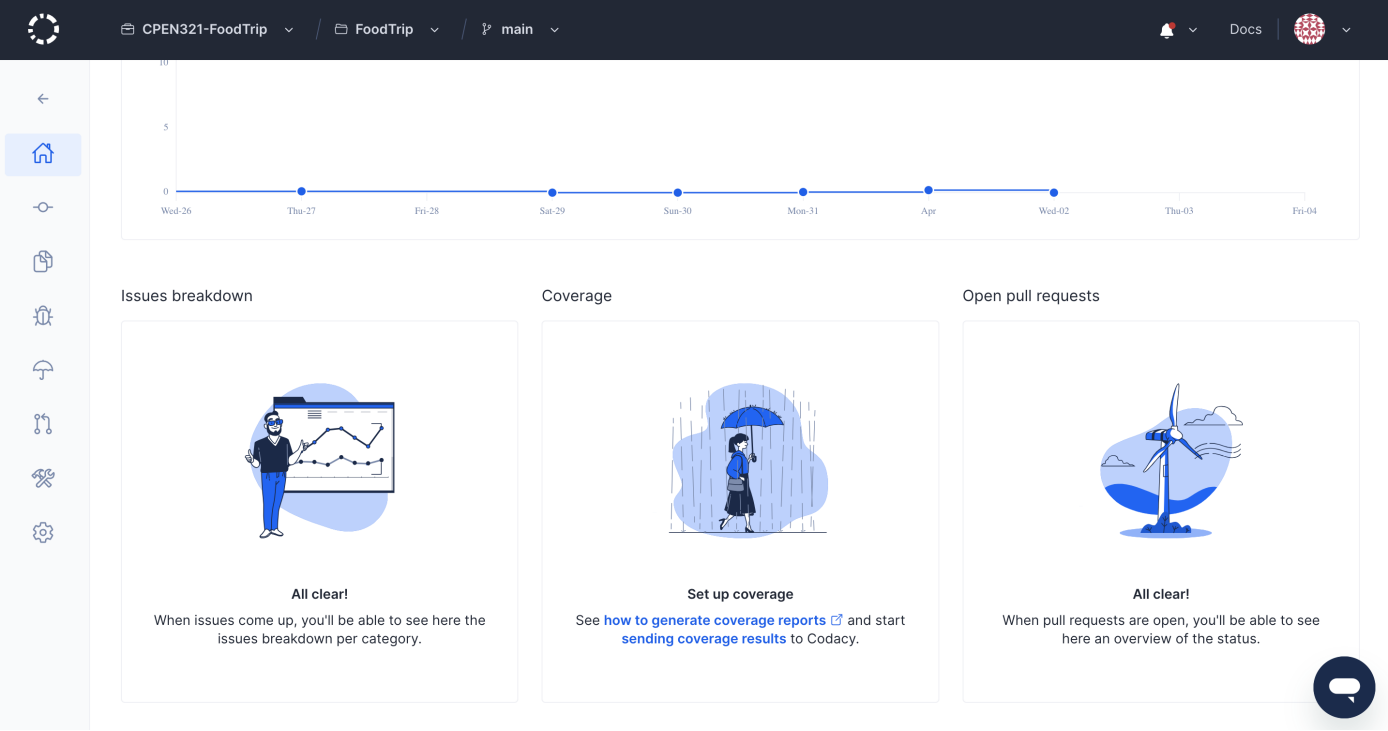
Build Analyzer results available

5. Automated Code Review Results

5.1. Commit Hash Where Codacy Ran

abfa2144599635f44dcd59fd106209fd80abb743

5.2. Unfixed Issues per Codacy Category



5.3. Unfixed Issues per Codacy Code Pattern

