Documentation for Test Results

Overview

The test results indicate the outcome of various functions within the Python code. The tests are categorized into "Passed" and "Failed" with details about each individual test case. This documentation provides an overview of the test results, highlighting the successful and failed tests along with relevant details.

Summary

- Total Tests: 1669
- Passed Tests: 1289
- Failed Tests: 380

Test:

1 Database Tests

1.1 Test: Check Database Existence

1.1.1 Goal: Ensure that the SQLite database file 'routeplanning.db' exists in the 'db' folder.

1.1.2 Outcome: Passed if the database file exists; failed otherwise.

1.2 Test: Database Connection

1.2.1 Goal: Verify successful connection to the SQLite database.

1.2.2 Outcome: Passed if the connection is successful; failed otherwise.

1.3 Test: Get Amenities from Database

1.3.1 Goal: Confirm the ability to retrieve amenities from the database for various types.

1.3.2 Outcome: Passed if amenities are successfully retrieved; failed otherwise.

1.4 Test: Get Closest Nodes from Database

1.4.1 Goal: Validate the functionality to retrieve the closest nodes to a given location.

1.4.2 Outcome: Passed if closest nodes are successfully retrieved; failed otherwise.

1.5 Test: Get Node Information

1.5.1 Goal: Test the capability to retrieve information about a specific node.

1.5.2 Outcome: Passed if node information is successfully retrieved; failed otherwise.

1.6 Test: Get Way Information

1.6.1 Goal: Verify the ability to retrieve information about a way.

1.6.2 Outcome: Passed if way information is successfully retrieved; failed otherwise.

1.7 Test: Get Node Neighbors

1.7.1 Goal: Ensure the ability to retrieve neighbors of a specific node.

1.7.2 Outcome: Passed if node neighbors are successfully retrieved; failed otherwise.

1.8 Test: Get Node Coordinates
1.8.1 Goal: Confirm the capability to retrieve coordinates for a specific node.
1.8.2 Outcome: Passed if node coordinates are successfully retrieved; failed otherwise.

1.9 Test: Get Walking Neighbors
1.9.1 Goal: Validate the ability to retrieve walking neighbors of a node with different risk factors.
1.9.2 Outcome: Passed if walking neighbors are successfully retrieved; failed otherwise.

1.10 Test: Get Biking Neighbors
1.10.1 Goal: Validate the ability to retrieve biking neighbors of a node with different risk factors.
1.10.2 Outcome: Passed if biking neighbors are successfully retrieved; failed otherwise.

1.11 Test: Node Walkability
1.11.1 Goal: Verify the determination of whether a node is walkable.
1.11.2 Outcome: Passed if walkability is correctly determined; failed otherwise.

1.12 Test: Node Bikeability
1.12.1 Goal: Verify the determination of whether a node is bikeable.
1.12.2 Outcome: Passed if bikeability is correctly determined; failed otherwise.

1.13 1.13 Test: Database Closure
1.13.1 Goal: Confirm the ability to close the database connection.
1.13.2 Outcome: Passed if the database connection is successfully closed; failed otherwise.

2 GPX Export Tests
2.1 Test: Parse String to List
2.1.1 Goal: Verify the correct parsing of a string representing a path to a list.
2.1.2 Outcome: Passed if the string is successfully parsed into a list; failed otherwise.

2.2 Test: GPX Export
2.2.1 Goal: Confirm the successful export of a path to a GPX file.
2.2.2 Outcome: Passed if the GPX file is successfully exported; failed otherwise.

3 Pathfinder Tests
3.1 Test: Pathfinder Execution
3.1.1 Goal: Execute the pathfinder algorithm for random start and end nodes with different transportation types and risk factors.
3.1.2 Outcome: Passed if the pathfinder algorithm successfully finds a path; failed otherwise.

3.2 Test: Risk Tolerance with Fixed Locations
3.2.1 Goal: Test the behavior of the pathfinder with different risk tolerance levels for fixed start and end locations.

3.2.2 Outcome: Passed if the paths are different for different risk tolerances; failed otherwise.

4 Folder Structure Generation

4.1.1 Goal: Generate and verify the correctness of the project folder structure.

4.1.2 Outcome: Passed if the folder structure is generated correctly; failed otherwise.

5 Overall Test Results

5.1.1 Goal: Provide a summary of passed and failed tests.

5.1.2 Outcome: Display the number of tests passed and failed along with details.

Failed Tests

The Reason why we have some failed tests, specifically next node or neighboring node (bikeable / walkable) is simply because OSM did not provide connected nodes with path or ways. So in some cases there is a Path in the real world, but there is no connection in the Database from OSM. Since we pulled the data from OSM and build our own database based on the data from OSM, we therefore end up having none connected nodes. While running the Program and Pathfinder we never had a Problem where we couldn't find a path / way.

We handle those problems by increasing the radius on where we look for the next node. These Problems would only occur when we set the Start and End locations. When we can't find a connection we just increase the distance around the start / end node to find a different node which will be a connection one. And then start from there on. While looking for an path we only go down connected nodes. Which we saw in the passing of ALL random nodes we did. Therefore the failed tests only failed because they're specific dead node entries in the database.

Conclusion

Most of the tests passed, indicating the overall health of the code. However, specific database-related tests failed, and further investigation is required to address the issues related to column existence, missing results, and unsuccessful bike ability checks. The code execution ended with exit code 0, suggesting that the process completed without errors. Some of the Nodes are not Connected on the OSM-Database. Since we pulled most of our data from OSM we therefore have the same problem. We did not have the time to check all ~53.000 Nodes one by one and connect them manually if needed. This is something Modeshift can spend time and resources into, if they want to do so. We found that most of the results found a good route. Sometimes with some weird connections but in reality, People could just take a shortcut around a weird looking part of the found path.