**Test Plan**

1. **Unit Testing**

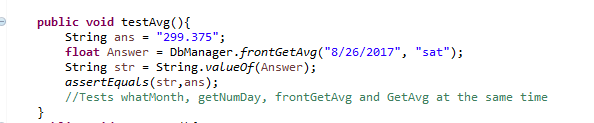
* **Backend Junit Test Report**

**testAvg():**

testAvg will input a specific date and day of the week, then AssertEquals the returned value with the expected value. This test will test multiple methods within the frontGetAvg method.

Methods being tested:

* frontGetAvg
* GetAvg
* whatMonth
* getNumDay

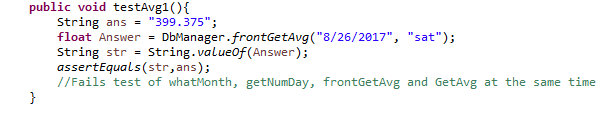




**testAvg1():**

testAvg1 is designed to cause the test to fail. It works the exact same way as testAvg, but AssertEquals the returned value with the wrong expected value to ensure that the test does not return true regardless of the input.

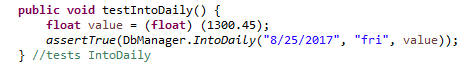
testAvg1 tests the exact same methods as testAvg.





**testIntoDaily():**

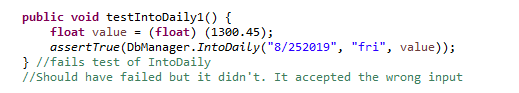
testIntoDaily will test the ability of the program to input new data into the database. It is inputting a date, day of the week, and a float gross sales value.





**testIntoDaily1():**

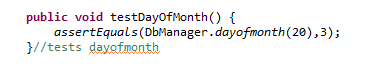
testIntoDaily1 is taking an invalid input and entering it into the database. This test is designed to fail, however it does not and IntoDaily is accepting the invalid date and entering it into the database. This highlights an issue that needs to be fixed.





**testDayOfMonth():**

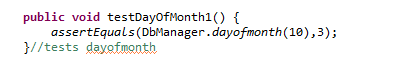
testDayOfMonth takes an input of the day of the month and tests the output of DayOfMonth against the expected value. DayOfMonth tests the 20th of any given month to return which iteration of the day of the week of a given month it is. The 20th should be the third time that specific day of the week has been seen that month.





**testDayOfMonth1():**

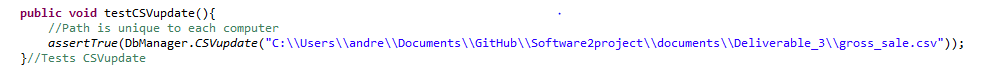
testDayOfMonth is designed to fail by testing against the wrong expected value. It fails as expected, validating the DayOfMonth works as expected.





**testCSVupdate():**

testCSVupdate tests that the CSV file is properly updated given a user specific file path. The test works as expected.





**testCSVupdate1():**

testCSVupdate1 attempts to run CSVupdate with an improper input. This test fails as expected.

**testMonthUpdate():**

testMonthUpdate tests the MonthUpdate method to update the table for that entire month. It inserts a set date into MonthUpdate and AssertTrue the returned value.





**testMonthUpdate1():**

testMonthUpdate1 works the same as testMonthUpdate but inserts an invalid input. This test predictably fails.





* **Encryption Junit Test Report**

**testlogin():**

testlogin will have a test user login, then assert Equals will test to ensure that the CheckPass login passes. This means the user in testing can login returning a true and the position of the user in question.

**testloginfail():**

testloginfail will have a test user login, then assert Equals will test to ensure that the CheckPass login fails with improper credentials. This means the user in testing cannot login, this returns a null for both position and login. In this case a Junit pass is a user login failure.

**testnewuser():**

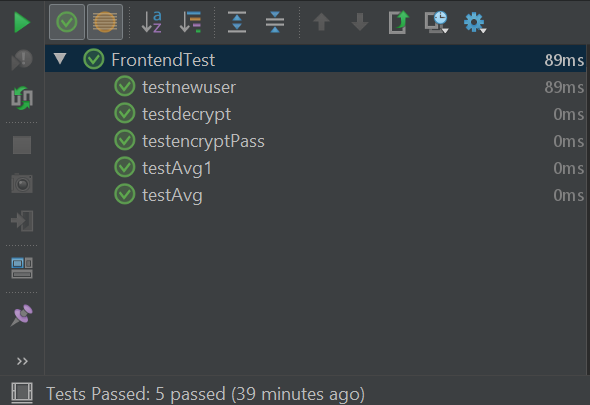
testnewuser is trying to create a new user that already exist in the database that is meant to fail, this returns a false because the system is unable to create the new user with the new permissions.

**Testencryptpass():**

Testencryptpass takes an input of an example password and makes that into a byte array that gets turned into a string and compared against a byte array string provided that we know to be correct. This test uses both encryptPass and byteToString this is intended to compare strings using assertequals.

**Testdecrypt():**

Testdecrypt takes an input of an example encrypted password that gets turned into a byte array and then decrypted into a string and compared against a string provided that we know to be the correct output. This test uses both decrypt and stringToByte this is intended to compare strings using assertequals making sure that it returns the correct pass from the encrypted data.



**Note:** Front end junit testing still to be done.

1. **Use Case Testing**
2. **Log in use case**

|  |  |  |
| --- | --- | --- |
| Main Success Scenario | Step | Description |
| A: Manager/Employee S: System | 1 | A: Enter username and password. |
| 2 | S: Validate username and password. |
| 3 | S: Allow account access to the next page. |
| Alternatives | 2a | **Username and/or password not valid.** Display error message. |

**Test Coverage:** Number of main and alternatives scenarios = 2.

100% coverage of use case.

1. **Create account use case**

|  |  |  |
| --- | --- | --- |
| Main Success Scenario | Step | Description |
| A: Manager S: System | 1 | A: Enter username and password. |
| 2 | S: Validate username and password. |
| 3 | S: Allow account access to the next page. |
| 4 | A: Selects to create account. |
| 5 | S: Loads create account screen. |
| 6 | A: Enter username, password, confirmation password, and selects the role. |
| 7 | S: Validates if all fields are filled. |
| 8 | S: Validate if a role was selected. |
| 9 | S: Validates if both passwords are the same. |
| 10 | S: Display a successful account creation message. |
| Alternatives | 2a | **Username and/or password not valid.** Display error message. |
| 7a | **One or all fields are empty.** Display error message. |
| 8a | **A role was not selected.** Display message and ask to select a role. |
| 9a | **Passwords are not the same.** Display message and ask to re-type passwords. |

**Test Coverage:** Number of main and alternatives scenarios = 5.

100% coverage of use case.

1. **Update/delete account use case**

|  |  |  |
| --- | --- | --- |
| Main Success Scenario | Step | Description |
| A: Manager S: System | 1 | A: Enter username and password. |
| 2 | S: Validate username and password. |
| 3 | S: Allow account access to the next page. |
| 4 | A: Selects account settings. |
| 5 | S: Load account settings screen. |
| 6 | A: Selects to load all accounts. |
| 7 | S: Load all account from the Database. |
| 8 | A: Selects one account and delete. |
| 9 | S: Deletes selected account of the database. |
| 10 | A: Selects one account and update information. |
| 11 | S: Validate if anything is changed or a field is empty. |
| 12 | S: Updates selected account information in the database. |
| Alternatives | 2a | **Username and/or password not valid.** Display error message. |
| 11a | **One or all fields are empty.** Display message and ask to re-try again. |

**Test Coverage:** Number of main and alternatives scenarios = 3.

100% coverage of use case.

1. **Generate sale prediction use case**

|  |  |  |
| --- | --- | --- |
| Main Success Scenario | Step | Description |
| A: Manager/Employee S: System | 1 | A: Enter username and password. |
| 2 | S: Validate username and password. |
| 3 | S: Allow account access to the next page. |
| 4 | A: Selects generate sale prediction. |
| 5 | S: Load generate sale prediction screen. |
| 6 | A: Choose a date or dates to generate prediction and selects to load. |
| 7 | S: Validate input dates. |
| 8 | S: Generate an accurate prediction, display a graphical view of sales, and fill a table with average sales for each date selected. |
| Alternatives | 2a | **Username and/or password not valid.** Display error message. |
| 7a | **Invalid Dates.** Display message and ask to re-try again. |

**Test Coverage:** Number of main and alternatives scenarios = 3.

100% coverage of use case.

1. **Upload sales data use case**

|  |  |  |
| --- | --- | --- |
| Main Success Scenario | Step | Description |
| A: Manager/Employee S: System | 1 | A: Enter username and password. |
| 2 | S: Validate username and password. |
| 3 | S: Allow account access to the next page. |
| 4 | A: Selects upload sales data. |
| 5 | S: Load upload sales data screen. |
| 6 | A: Choose a date and enter amount. |
| 7 | S: Validate input date and amount. |
| 8 | S: Uploads data in the database, close current screen, and go back to the home page screen. |
| Alternatives | 2a | **Username and/or password not valid.** Display error message. |
| 7a | **Invalid date or amount.** Display message and ask to re-try again. |

**Test Coverage:** Number of alternatives scenarios = 3.

100% coverage of use case.