**TASK 1**

**Exploratory testing charters to document your testing. You can use simple format of: Explore target, with resources, to discover information**

1. Verified user can add expenses by selecting the available options(Car, Food, etc)
2. Verified user can add expenses by clicking on the Expense button
3. Verified user can add income by clicking on the Income button
4. Verified the Pie chart displays the percentage of expense correctly
5. Verified the calculator displayed while adding Expense/Income is working correctly
6. Verified the user can add expense/Income by specifying the dates using the Calendar i.e., either by Day, Week, Month, Year
7. Verified the user can view the expense/Income using the Calendar i.e., either by Day, Week, Month, Year, Interval or by selecting a specific date
8. Verified the calendar displays the past as well as future date
9. Verified the user can transfer amount from one account to another
10. Verified user cannot transfer amount if the From and To Account are same
11. Verified the balance(Debit and Credit) is correctly displayed in the respective account after the amount is transferred
12. Verified user can add new account (Card, Cash, etc)
13. Verified user can export the transaction details to a file
14. Verified user can change the language and per his/her preferable language
15. Verified user can search for an expense/income category using the search bar
16. Verified user can store the data using Data Backup
17. Verified user can extract the transaction details to a file using export option

**Findings from your charters. Did everything work as expected? What bugs were discovered?**

Issue 1 : The expenses added in below account is less than the income and still the balance is displayed in red and negative (-$415.00)

Graphical user interface, application

Description automatically generated

**Issue 2:** Total expense added is $ 30.00 and Income is $20.00 **Expected - Balance should be displayed as ($-10.00) in Red**

**Actual - Balance should be displayed as ($ 10.00) in Green**Graphical user interface, application

Description automatically generated

**Issue 3: Expense and Income button displayed as + and -**

Chart

Description automatically generated

**Prioritisation of those charters - which area of the app or testing would you explore first and why?**

Verifying the Expense and Income section –

Since this is an important and main feature of the app. Any bug around this will make the users switch to other available apps in the market. So, it is very important to test this feature thoroughly and this should be part of smoke test. The team needs to ensure that expenses are correctly calculated as a deduction (In Red) and Income is calculated as an addition (In Green) when displaying the balance amount.

**How much time you have planned for each charter?**

The below modules were tested, and each took around 20 mins for exploring and understanding the features:

* Expense
* Income
* Transfer
* Calendar
* Settings (Budget Mode, carry over, Language, currency, File export, Data Backup)

**What kind of risks you need to mitigate for this type of application?**

The Money **Transfer** module is a high-risk area to take care of. If the amount deducted from one account is not transferred to another account or any mismatch in the amount (Debit Amount should match with the credit amount) transferred could lead to serious compliance issue against the organisation. Hence, an early and end to end test of this module will help to mitigate the risk.

**TASK 2**

**A list of ideas/bullet points you would test for. Focus on good coverage, rather than complete test cases.**

* Verifying the Expense Section ( Test by adding each available component in order to verify the components)
* Verifying the Income section ( Test for each available component- Salary, Deposits)
* Verifying the Transfer of amount from one account to another
* Verifying the Calculator which is displayed while adding income/expenses
* Verifying the segregation (Pie Chart ) of expenses is calculated and displayed correctly
* Verifying the Balance section (Shows the list of expenses/income added during the period)
* Verifying the Calendar ( Day, Week, Month, Year, Interval, etc)
* Verifying the search
* Verifying the Settings Option
* Verifying Accounts section (User is able to add/Delete/Update accounts)
* Verifying Categories section (User is able to add/Delete/Update Income/Expense Categories)

**A prioritization of such test cases according to their possible business impact in case of failure.**

Priority : 0 = High, 1 = Medium, 2 = Low

|  |  |
| --- | --- |
| Test | Priority |
| Verifying the Expense Section ( Test by adding each available component in order to verify the components) | 0 |
| Verifying the Income section ( Test for each available component- Salary, Deposits) | 0 |
| Verifying the Transfer of amount from one account to another | 0 |
| Verifying the Calculator which is displayed while adding income/expenses | 0 |
| Verifying the segregation (Pie Chart ) of expenses is calculated and displayed correctly | 1 |
| Verifying the Balance section (Shows the list of expenses/income added during the period) | 1 |
| Verifying the Calendar ( Day, Week, Month, Year, Interval, etc) | 1 |
| Verifying the search | 1 |
| Verifying the Settings Option | 2 |
| Verifying Accounts section (User is able to add/Delete/Update accounts) | 1 |
| Verifying Categories section (User is able to add/Delete/Update Income/Expense Categories) | 1 |

**Outline the possibilities of automating proposed test cases on different levels, together with a short summary of pros and cons of each of them.**

|  |  |  |
| --- | --- | --- |
| Test | Priority | Should be Automated |
| Verifying the Expense Section ( Test by adding each available component in order to verify the components) | 0 | Yes |
| Verifying the Income section ( Test for each available component- Salary, Deposits) | 0 | Yes |
| Verifying the Transfer of amount from one account to another | 0 | Yes |
| Verifying the Calculator which is displayed while adding income/expenses | 0 | Yes |

**Pros:**

Above test should always be a part of smoke test to ensure that the basic functionality of the application is not regressed when another feature is deployed

The tests should be Data driven so that every time a new or multiple set of data is used for verifying the tests

**Cons:**

If the test is automated using Data driven framework, It will be difficult to assert the expected value.

**TASK 3**

**A list of test cases proposed for automation.**

|  |  |
| --- | --- |
| **Test** | **RequestType** |
| Verify user is able to find a pet by status Pending | GET |
| Verify user is able to find a pet by status Available | GET |
| Verify user is able to find a pet by status Sold | GET |
| Verify user is able to find a pet by ID | GET |
| Verify user is able to find a pet by Tags | GET |
| Verify user is able to add a new Pet to the store | POST |
| Verify user is able to update an existing Pet | PUT |
| Verify user is able to upload an image for an existing Pet | POST |
| Verify user is able to update a pet in the store with form data using PetId | POST |
| Verify user is able to delete a pet using PetId | DELETE |
| Verify 4XX error is displayed when a user is searching for a pet with invalid status | GET |
| Verify 4XX error is displayed when a user is searching for a pet with invalid ID | GET |
| Verify 4XX error is displayed when a user is searching for a pet with invalid Tag | GET |
| Verify 4XX error is displayed when a user is deleting a pet with invalid ID | DELETE |
| Verify 4XX error is displayed when a user is updating pet's details with invalid ID | PUT |

**Please automate proposed test cases.**

Please check <https://github.com/moushmikhan/N26.git>