**Personal Budget Manager Application**

**Test Plan**

**Version 2.0**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 31/January/2019 | 1.0 | Test Plan for deliverable 1 | Xinjie Zeng |
| 18/March/2019 | 2.0 | Test Plan for deliverable 2 (add test cases for composite expenses and expense hide/show) | Xinjie Zeng |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

Table of Contents

[1. Objectives 3](#_Toc536779765)

[1.1 Purpose 3](#_Toc536779766)

[1.2 Scope 3](#_Toc536779767)

[2. Requirements for Test 3](#_Toc536779768)

[2.1 Function Testing 3](#_Toc536779769)

[2.2 User Interface Testing 3](#_Toc536779770)

[2.3 Security and Access Control Testing 4](#_Toc536779771)

[3. Test Strategy 4](#_Toc536779772)

[3.1 Testing Types 4](#_Toc536779773)

[3.1.1 Function Testing 4](#_Toc536779774)

[3.1.2 User Interface Testing 4](#_Toc536779775)

[3.1.3 Configuration Testing 5](#_Toc536779776)

[4. Test cases 5](#_Toc536779777)

[5. Resources & Milestone 9](#_Toc536779778)

[5.1 Roles 9](#_Toc536779779)

[5.2 Project Milestones 9](#_Toc536779780)

[6. Deliverables 10](#_Toc536779781)

**Test Plan**

# **1.** **Objectives**

## **1.1 Purpose**

This document describes the plan for testing Personal Budget Manager Application. This Test Plan document supports the following objectives:

* + Identify existing project information and the software that should be tested.
  + List the recommended test requirements (high level).
  + Recommend and describe the testing strategies to be employed.
  + Identify the required resources and provide an estimate of the test efforts.
  + List the deliverable elements of the test activities.

## **1.2** **Scope**

This Test Plan describes the tests that will be conducted on Personal Budget Manager Application. The purpose is to test its interfaces, functionalities.

The interfaces between the following subsystems will be tested:

* + Main panel for the Application
  + Add expense panel of the Application

The most critical functionalities to test are:

* + Add a new expense
  + Mark an expense as paid(unpaid)
  + Remove an expense
  + View all expenses of the spending list (in arbitrary order)

# **2.** **Requirements for Test**

## **2.1 Function Testing**

The functions to test are:

* Verify correct addition of expense to database
* Verity correct update of expense to database
* Verify correct removal of expense to database
* Verify correct retrieval of update of database data

## **2.2 User Interface Testing**

The interfaces to test are:

* Verify ease of navigation through interface
* Verify ease of use for users
* Verify sample screens conform to GUI standards.

## **2.3 Security and Access Control Testing**

None at this time

# **3.****Test Strategy**

## **3.1** **Testing Types**

### **3.1.1 Function Testing**

|  |  |
| --- | --- |
| Test Objective: | * Ensure successful data entry, data update, data removal and retrieval |
| Technique: | * Execute each use case, use case flow, or function, using valid and invalid data, to verify the following: * The expected results occur when valid data is used * The appropriate error / warning messages are displayed when invalid data is used. |
| Completion Criteria: | * The updated data are successfully displayed on the screen * All identified defects have been addressed |
| Special Considerations: | * None |

### **3.1.2** **User Interface Testing**

|  |  |
| --- | --- |
| Test Objective: | * Proper navigation to application, ease of use for users, the interface conforms to the JAVA GUI standard: |
| Technique: | * Navigate through window to window, verify key and mouse movement * Verify menus, size, position, fonts, colors that conform to standards |
| Completion Criteria: | * Each window successfully verified to remain consistent with benchmark version or within acceptable standard |
| Special Considerations: | * None |

### **3.1.3 Configuration Testing**

|  |  |
| --- | --- |
| Test Objective: | * Validate and verify that the applications function properly on the Windows and Mac. |
| Technique: | * Execute application on Windows and Mac to test all functionalities |
| Completion Criteria: | * All functionalities are successfully completed without failure. * Users can see all the updated expense on the screen |

# 4. Test cases

* **Add an expense**

|  |  |
| --- | --- |
| Test case ID | 1 |
| Test scenario | Add a purchase |
| test steps | 1. open the app  2. click “Add Expense” button on main panel  3. type in info  4. press “Add Expense button” on Add Expense panel |
| Test data | Type: purchase  Date: 2018-07-19  Name: candy  Amount: 2.62  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | store the info in the data base and show it on the main panel |
| Actual result | Successfully stores the info in the data base and update it on the main panel |

|  |  |
| --- | --- |
| Test case ID | 2 |
| Test scenario | Add a bill |
| test steps | 1. open the app  2. click “Add Expense” button on main panel  3. type in info  4. press “Add Expense button” on Add Expense panel |
| Test data | Type: bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly |
| Expected result | store the info in the data base and update it on the main panel |
| Actual result | Successfully stores the info in the data base and update it on the main panel |

|  |  |
| --- | --- |
| Test case ID | 3 |
| Test scenario | Add a purchase (invalid date) |
| test steps | 1. open the app  2. click “Add Expense” button on main panel  3. type in info  4. press “Add Expense button” on Add Expense panel |
| Test data | Type: purchase  Date: dkejide  Name: candy  Amount: 2.62  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | When user click the “Add Expense” button, have a prompt up to ask user to type in date in correct format like 2019-09-09. If the info is correct, the info is stored in the data base and is updated on the main panel. |
| Actual result | When user click the “Add Expense” button, the program freezes and it has no action nor feedback to users. |

|  |  |
| --- | --- |
| Test case ID | 4 |
| Test scenario | Add a purchase (invalid amount) |
| test steps | 1. open the app  2. click “Add Expense” button on main panel  3. type in info  4. press “Add Expense button” on Add Expense panel |
| Test data | Type: purchase  Date: 2019-03-07  Name: candy  Amount: dfsdf  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | When user click the “Add Expense” button, have a prompt up to ask user to type in amount in correct format like “74.55”. If the info is correct, the info is stored in the data base and is updated on the main panel. |
| Actual result | When user click the “Add Expense” button, the program freezes and it has no action nor feedback to users. |

|  |  |
| --- | --- |
| Test case ID | 5 |
| Test scenario | Add a bill (invalid date) |
| test steps | 1. open the app  2. click “Add Expense” button on main panel  3. type in info  4. press “Add Expense button” on Add Expense panel |
| Test data | Type: bill  Date: sdfsdfew  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly |
| Expected result | When user click the “Add Expense” button, have a prompt up to ask user to type in date in correct format like 2019-09-09. If the info is correct, the info is stored in the data base and is updated on the main panel. |
| Actual result | When user click the “Add Expense” button, the program freezes and it has no action nor feedback to users. |

|  |  |
| --- | --- |
| Test case ID | 6 |
| Test scenario | Add a bill (invalid amount) |
| test steps | 1. open the app  2. click “Add Expense” button on main panel  3. type in info  4. press “Add Expense button” on Add Expense panel |
| Test data | Type: purchase  Date: 2019-03-07  Name: candy  Amount: dfsdf  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | When user click the “Add Expense” button, have a prompt up to ask user to type in amount in correct format like “74.55”. If the info is correct, the info is stored in the data base and is updated on the main panel. |
| Actual result | When user click the “Add Expense” button, the program freezes and it has no action nor feedback to users. |

* **Create Composite Expense**

|  |  |
| --- | --- |
| Test case ID | 7 |
| Test scenario | Create a composite bill |
| test steps | 1. open the app  2. select multiple bills on the main panel  3. click the “Create Composite Expense” button |
| Test data | Type: bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly  Type: bill  Date: 2019-01-01  Name: Gym  Amount: 750  Status: paid  Vendor Name: ABC Fitness  Location:  Method:  Category: Default  Due date: 2019-04-01  Interval: Monthly |
| Expected result | These two bills are shown under the composite bill on the panel. |
| Actual result | Waiting for the implementation |

|  |  |
| --- | --- |
| Test case ID | 8 |
| Test scenario | Create a composite purchase |
| test steps | 1. open the app  2. select multiple purchases on the main panel  3. click the “Create Composite Expense” button |
| Test data | Type: purchase  Date: 2019-03-07  Name: candy  Amount: 30.5  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food  Type: purchase  Date: 2019-01-27  Name: cakes  Amount: cocobun  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | These two purchases are shown under the composite purchase on the panel. |
| Actual result | Waiting for the implementation |

* Mark Expense Paid/Unpaid

|  |  |
| --- | --- |
| Test case ID | 9 |
| Test scenario | Mark a purchase unpaid to paid |
| test steps | 1. open the app  2. select an expense  3. click “Mark Expense Paid/Unpaid” button |
| Test data | Type: purchase  Date: 2019-03-18  Name: gas  Amount: 62.94  Status: unpaid  Method: credit  Vendor: Petrol Canada  Location:  Category: default  Due Date: 2019-03-28 |
| Expected result | The purchase status changed from unpaid to paid. |
| Actual result | Successfully updated the purchase status to paid. |

|  |  |
| --- | --- |
| Test case ID | 10 |
| Test scenario | Mark a purchase paid to unpaid |
| test steps | 1. open the app  2. select an expense  3. click “Mark Expense Paid/Unpaid” button |
| Test data | Type: purchase  Date: 2019-03-18  Name: gas  Amount: 62.94  Status: paid  Method: credit  Vendor: Petrol Canada  Location:  Category: default  Due Date: 2019-03-28 |
| Expected result | The purchase status changed from paid to unpaid. |
| Actual result | Successfully updated the purchase status to unpaid. |

|  |  |
| --- | --- |
| Test case ID | 11 |
| Test scenario | Mark a bill unpaid to paid |
| test steps | 1. open the app  2. select an expense  3. click “Mark Expense Paid/Unpaid” button |
| Test data | Type: bill  Date: 2019-01-18  Name: Electricity  Amount: 576.93  Status: unpaid  Method:  Vendor: Hydro Quebec  Location:  Category: default  Due Date: 2019-03-18  Interval: Quarterly |
| Expected result | The bill status changed from unpaid to paid. |
| Actual result | Successfully updated the bill status to paid. |

|  |  |
| --- | --- |
| Test case ID | 12 |
| Test scenario | Mark a bill paid to unpaid |
| test steps | 1. open the app  2. select an expense  3. click “Mark Expense Paid/Unpaid” button |
| Test data | Type: bill  Date: 2019-03-18  Name: gas  Amount: 62.94  Status: paid  Method: credit  Vendor: Petrol Canada  Location:  Category: default  Due Date: 2019-03-28  Interval: Monthly |
| Expected result | The purchase status changed from paid to unpaid. |
| Actual result | Successfully updated the purchase status to unpaid. |

|  |  |
| --- | --- |
| Test case ID | 13 |
| Test scenario | mark a composite purchase unpaid to paid |
| test steps | 1. open the app  2. select composite purchase on the main panel  3. click the “Mark Expense Paid/Unpaid” button |
| Test data | Type: composite purchase  Date: 2019-03-07  Name: candy  Amount: 30.5  Status: unpaid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food  Date: 2019-01-27  Name: cakes  Amount: cocobun  Status: unpaid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | The status of two purchases shown under the composite purchase will be updated to paid. |
| Actual result | Waiting for the implementation |

|  |  |
| --- | --- |
| Test case ID | 14 |
| Test scenario | mark a composite purchase paid to unpaid |
| test steps | 1. open the app  2. select composite purchase on the main panel  3. click the “Mark Expense Paid/Unpaid” button |
| Test data | Type: composite purchase  Date: 2019-03-07  Name: candy  Amount: 30.5  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food  Date: 2019-01-27  Name: cakes  Amount: cocobun  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | The status of two purchases shown under the composite purchase will be updated to unpaid. |
| Actual result | Waiting for the implementation |

|  |  |
| --- | --- |
| Test case ID | 15 |
| Test scenario | mark a composite bill unpaid to paid |
| test steps | 1. open the app  2. select composite bill on the main panel  3. click the “Mark Expense Paid/Unpaid” button |
| Test data | Type: composite bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly  Date: 2019-01-01  Name: Gym  Amount: 750  Status: unpaid  Vendor Name: ABC Fitness  Location:  Method:  Category: Default  Due date: 2019-04-01  Interval: Monthly |
| Expected result | The status of two bills shown under the composite bills will be updated to paid. |
| Actual result | Waiting for the implementation |

|  |  |
| --- | --- |
| Test case ID | 16 |
| Test scenario | mark a composite bill paid to unpaid |
| test steps | 1. open the app  2. select composite bill on the main panel  3. click the “Mark Expense Paid/Unpaid” button |
| Test data | Type: composite bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: paid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly  Date: 2019-01-01  Name: Gym  Amount: 750  Status: paid  Vendor Name: ABC Fitness  Location:  Method:  Category: Default  Due date: 2019-04-01  Interval: Monthly |
| Expected result | The status of two bills shown under the composite bills will be updated to unpaid. |
| Actual result | Waiting for the implementation |

* Remove Expense

|  |  |
| --- | --- |
| Test case ID | 17 |
| Test scenario | Remove a bill |
| test steps | 1. open the app  2. select a bill on the main panel  2. click “Remove Expense” button on the main panel |
| Test data | Type: bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly |
| Expected result | data is successfully removed from the main panel |
| Actual result | Data is successfully removed from the main panel |

|  |  |
| --- | --- |
| Test case ID | 18 |
| Test scenario | Remove a purchase |
| test steps | 1. open the app  2. select a purchase on the main panel  2. click “Remove Expense” button on the main panel |
| Test data | Type: purchase  Date: 2018-07-19  Name: candy  Amount: 2.62  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | data is successfully removed from the main panel |
| Actual result | Data is successfully removed from the main panel |

|  |  |
| --- | --- |
| Test case ID | 19 |
| Test scenario | Remove a composite bill |
| test steps | 1. open the app  2. select a composite bill  3. click the “Remove Expense” button |
| Test data | Type: composite bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: paid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly  Date: 2019-01-01  Name: Gym  Amount: 750  Status: paid  Vendor Name: ABC Fitness  Location:  Method:  Category: Default  Due date: 2019-04-01  Interval: Monthly |
| Expected result | These two bills should be removed from the main panel |
| Actual result | Waiting for the implementation |

|  |  |
| --- | --- |
| Test case ID | 20 |
| Test scenario | Remove a composite purchase |
| test steps | 1. open the app  2. select composite purchase on the main panel  3. click the “Remove Expense” button |
| Test data | Type: composite purchase  Date: 2019-03-07  Name: candy  Amount: 30.5  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food  Date: 2019-01-27  Name: cakes  Amount: cocobun  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food |
| Expected result | The two purchases will be both removed from the main panel |
| Actual result | Waiting for the implementation |

* Hide/Show Paid Expenses

|  |  |
| --- | --- |
| Test case ID | 21 |
| Test scenario | Hide Paid Purchase |
| test steps | 1. open the app  2. click “Hide/Show Paid Expenses” button on the main panel |
| Test data | Type: purchase  Date: 2018-07-19  Name: candy  Amount: 2.62  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food  Type: purchase  Date: 2018-03-09  Name: cakes  Amount: 5.5  Status: unpaid  Method: credit  Vendor Name: cocobun  Location: Downtown  Category: Food  Type: purchase  Date: 2018-02-19  Name: groceries  Amount: 100  Status: unpaid  Method: credit  Vendor Name: super c  Location: Downtown  Category: Food |
| Expected result | Paid purchases should be hidden from the main panel |
| Actual result | Paid purchases are successfully hidden from the main panel |

|  |  |
| --- | --- |
| Test case ID | 22 |
| Test scenario | show Paid Purchase |
| test steps | 1. open the app  2. click “Hide/Show Paid Expenses” button on the main panel to hide the paid purchase  3. click “Hide/Show Paid Expenses” button on the main panel to show the paid purchase again |
| Test data | Type: purchase  Date: 2018-07-19  Name: candy  Amount: 2.62  Status: paid  Method: debit  Vendor Name: Tim Hortons  Location: Downtown  Category: Food  Type: purchase  Date: 2018-03-09  Name: cakes  Amount: 5.5  Status: unpaid  Method: credit  Vendor Name: cocobun  Location: Downtown  Category: Food  Type: purchase  Date: 2018-02-19  Name: groceries  Amount: 100  Status: unpaid  Method: credit  Vendor Name: super c  Location: Downtown  Category: Food |
| Expected result | Paid data should be reappeared on the main panel |
| Actual result | Paid data is successfully reappeared on the main panel |

|  |  |
| --- | --- |
| Test case ID | 23 |
| Test scenario | Hide Paid bill |
| test steps | 1. open the app  2. click “Hide/Show Paid Expenses” button on the main panel |
| Test data | Type: bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly  Type: bill  Date: 2018-08-19  Name: electricity  Amount: 150  Status: unpaid  Vendor Name: Hydro Quebec  Location:  Method: credit  Category: default  Due date: 2018-11-19  Interval: Quarterly  Type: bill  Date: 2019-01-19  Name: parking  Amount: 150  Status: paid  Vendor Name: Indigo  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-02-19  Interval: Monthly |
| Expected result | Paid bills should be hidden from the main panel |
| Actual result | Paid bills are successfully hidden from the main panel |

|  |  |
| --- | --- |
| Test case ID | 24 |
| Test scenario | show Paid Purchase |
| test steps | 1. open the app  2. click “Hide/Show Paid Expenses” button on the main panel to hide the paid bills  3. click “Hide/Show Paid Expenses” button on the main panel to show the paid bills again |
| Test data | Type: bill  Date: 2018-08-19  Name: jenny  Amount: 50  Status: unpaid  Vendor Name: Fido  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-09-19  Interval: Monthly  Type: bill  Date: 2018-08-19  Name: electricity  Amount: 150  Status: unpaid  Vendor Name: Hydro Quebec  Location:  Method: credit  Category: default  Due date: 2018-11-19  Interval: Quarterly  Type: bill  Date: 2019-01-19  Name: parking  Amount: 150  Status: paid  Vendor Name: Indigo  Location: Downtown  Method: credit  Category: Utilities  Due date: 2019-02-19  Interval: Monthly |
| Expected result | Paid bills should be reappeared on the main panel |
| Actual result | Paid bills are successfully reappeared on the main panel |

# **5.****Resources & Milestone**

This section presents the recommended resources for testing the application, their main responsibilities, and their knowledge or skill set.

## **5.1** **Roles**

This table shows the staffing assumptions for the testing.

|  |  |  |
| --- | --- | --- |
| **Human Resources** | | |
| **Role** | **Author** | **Specific Responsibilities/Comments** |
| Tester | Xinjie Zeng | Responsibilities:   * Generate test plan * Write test cases * Generate test suite * Execute tests |
|  |  |  |
|  |  |  |

## **5.2** **Project Milestones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone Task** | **Author** | **Start Date** | **End Date** |
| Initial Test Planning documentation | Xinjie Zeng | January 31, 2019 | January 31, 2019 |
| Prototype Test Execution | Xinjie Zeng | January 31, 2019 | January 31, 2019 |
| Update Test Plans corresponding deliverable 2 | Xinjie Zeng | March 18, 2019 | March 18, 2019 |

# **6.** **Deliverables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable** | **Owner** | **Review / Distribution** | **Due Date** |
| Test Plan for deliverable 1 | Xinjie Zeng | Internal Peer Review | February 2019 |
| Test Plan for deliverable 2 | Xinjie Zeng | Interval Peer Review | March 2019 |