

Finance Tracker

Requirements and Test Document

Pablo Bandera Lopez

03/28/2025

CS 225, Spring 25

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

INTRODUCTION:

This document will detail the various requirements and test cases needed to have a fully functional application. This program will be a personal monthly finance tracker where the user will be able to create entries, edit entries throughout the month and view all entries that have been entered. Further details can be found in the Software Design Document.

The requirements and test cases that follow break down the application into testable units. Some requirements and test cases are designed to test program reliability and adequate function of the back-end code. Other requirements and test cases are designed to test the user interface and its ability to interact with the back-end program. Both of these types of test cases requirements are necessary for the application to function.

BACKGROUND INFORMATION:

As previously stated, further information about the specific categories, sub-categories, accounts, types, etc. Is available in the software design document. Refer to tables 1-6 to see the list of available selections and when they should become available.

REQUIREMENTS:

Table 1: Requirement Specifications

ID	Requirement Specification
1	As a tester I want to be able to create an instance of Entry and set all its required information: dollarAmount, type, category, account, comment.
	1.1: A new instance can only be created when a valid entryID, entryID>0, is passed in the constructor. 1.2: A new instance of Entry shall be initialized by setting all its values to -1 and amount to 0.0. 1.3: The dollar amount, type, category, subcategory1-4, account, attributes shall only be settable to a value greater than 0 after being initialized. An InvalidEntryException shall be thrown if attempted.
2	As a tester I want to be able to create and instance of Entries to then add and remove entries from.
	2.1: A new Entry may only be added if the entry is valid. 2.2: An Entry shall be able to be removed based on and entryNumber. 2.3: An Entry shall be able to be found by entryNumber.
3	As a tester want to be able to Create a new instance of EntryFrame with or without an Entry.

	<p>3.1: The program shall only display selection menus for dollarAmount, type, account, an area for a comment, and the submit or cancel buttons if instantiated with a null entry.</p> <p>3.2: The program shall load the Entry data if instantiated and display all the previously selected and entered values for the user to change.</p> <p>3.3: The program shall add the Entry when the submit button is pressed only if the Entry contains valid information.</p>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>

TEST CASES:

Table 1: Test Case Summary

User Story ID	Requirement ID	Test Case ID	Date	Status Pass/Fail/Pending
1	1.1	1.1	03.28.25	PASS
1	1.2	1.2	03.28.25	PASS
1	1.3	1.3	03.28.25	PASS
2	2.1	2.1	03.28.25	PASS
2	2.2	2.2	03.28.25	PASS
2	2.3	2.3	03.28.25	PASS
3	3.1	3.1	03.28.25	PASS
3	3.2	3.2	03.28.25	PASS
3	3.3	3.3	03.28.25	PASS

Table 2: Test Case Template and Results

Test Case ID: 1.1		Current Status: PASS	Date: 03.28.25
Req. ID: 1.1 A new instance can only be created when a valid entryID, entryID>0, is passed in the constructor.			
Step #	Operator Action	Expected Results	Comments
1	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 0.	Program should throw an InvalidEntryException with message: Invalid Entry Number.	<pre> /*TEST MAIN*/ Run main Debug main Run Debug public static void main(String[] args) { try { Entry testEntry = new Entry(entryNumber:0); System.out.println(testEntry.getEntryString()); } catch (InvalidEntryException e) { System.out.println(e); } } </pre>
2	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 1.	Program should print the new entry instance in the format: id,0.0,-1,-1,-1,-1,-1,-1,-1,-1, This format represents an empty entry.	<pre> /*TEST MAIN*/ Run main Debug main Run Debug public static void main(String[] args) { try { Entry testEntry = new Entry(entryNumber:1); System.out.println(testEntry.getEntryString()); } catch (InvalidEntryException e) { System.out.println(e); } } </pre>

Screenshots:

```

159  /*TEST MAIN*/
160  Run main | Debug main | Run | Debug
161  public static void main(String[] args) {
162      try {
163          Entry testEntry = new Entry(entryNumber:0);
164          System.out.println(testEntry.getEntryString());
165      } catch (InvalidEntryException e) {
166          System.out.println(e);
167      }
168  }
169  }
170
171

```

```

159  /*TEST MAIN*/
160  Run main | Debug main | Run | Debug
161  public static void main(String[] args) {
162      try {
163          Entry testEntry = new Entry(entryNumber:1);
164          System.out.println(testEntry.getEntryString());
165      } catch (InvalidEntryException e) {
166          System.out.println(e);
167      }
168  }
169  }
170
171

```

PROBLEMS 4

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

COMMENTS

InvalidEntryException: Invalid Entry Number

PROBLEMS 4

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

COMMENTS

1,0.0,-1,-1,-1,-1,-1,-1,-1,

Test Case ID: 1.2		Current Status: pending	Date: 03.28.25
Req. ID: A new instance of Entry shall be initialized by setting all its values to −1 and amount to 0.0.			
Step #	Operator Action	Expected Results	Comments
1	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int > 0.	Program should print the new entry instance in the format: id,0.0,-1,-1,-1,-1,-1,-1, This format represents an empty entry.	<pre>/*TEST MAIN*/ Run main Debug main Run Debug public static void main(String[] args) { try { Entry testEntry = new Entry(entryNumber:1); System.out.println(testEntry.getEntryString()); } catch (InvalidEntryException e) { System.out.println(e); } }</pre>

Screenshots:

```

159      /*TEST MAIN*/
160      Run main | Debug main | Run | Debug
161      public static void main(String[] args) {
162          try {
163              Entry testEntry = new Entry(entryNumber:1);
164              System.out.println(testEntry.getEntryString());
165          } catch (InvalidEntryException e) {
166              System.out.println(e);
167          }
168      }
169  }
170

```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

1,0.0,-1,-1,-1,-1,-1,-1,-1,

Test Case ID: 1.3		Current Status: PASS	Date: 03.28.25
Req. ID: 1.3 The dollar amount, type, category, subcategory1-4, account, attributes shall only be settable to a value greater than 0 after being initialized. An InvalidEntryException shall be thrown if attempted.			
Step #	Operator Action	Expected Results	Comments
1	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 1. Use the setter methods to set all parameters to a value greater than 0 or non-empty string.	Program should print the new entry instance in the format: id,dollarAmount,type,category,subcategory,subcategory2,subcategory3,subcategory4,accout,comment	
2	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 1. Use the setter methods to set all parameters to a value greater than 0 or non-empty string. Using one of the setter	Program should throw and InvalidEntryException and print the corresponding error message. Example: InvalidEntryException: No Category Selected	

	methods set a value of 0 for any of the elements. Example: testEntry.setCategory(0);	
--	--	--

Screenshots:

159

/*TEST MAIN*/

Run main | Debug main | Run | Debug

160 public static void main(String[] args) {

161 try {

162 Entry testEntry = new Entry(entryNumber:1);

163 testEntry.setAmount(amount:12.22);

164 testEntry.setType(type:2);

165 testEntry.setCategory(category:2);

166 testEntry.setSubcategory(subcategory:1);

167 testEntry.setSubcategory2(subcategory2:2);

168 testEntry.setSubcategory3(subcategory3:1);

169 testEntry.setSubcategory4(subcategory4:3);

170 testEntry.setAccount(account:2);

171 testEntry.setComment(comment:"Test Comment");

172 System.out.println(testEntry.getEntryString());

173 } catch (InvalidEntryException e) {

174 System.out.println(e);

175 }

176 }

177 }

178 }

179 }

180 }

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

1,12.22,2,2,1,2,1,3,2,Test Comment

159

/*TEST MAIN*/

Run main | Debug main | Run | Debug

160 public static void main(String[] args) {

161 try {

162 Entry testEntry = new Entry(entryNumber:1);

163 testEntry.setAmount(amount:12.22);

164 testEntry.setType(type:2);

165 testEntry.setCategory(category:2);

166 testEntry.setSubcategory(subcategory:1);

167 testEntry.setSubcategory2(subcategory2:2);

168 testEntry.setSubcategory3(subcategory3:1);

169 testEntry.setSubcategory4(subcategory4:3);

170 testEntry.setAccount(account:2);

171 testEntry.setComment(comment:"Test Comment");

172 testEntry.setCategory(category:0);

173 System.out.println(testEntry.getEntryString());

174 } catch (InvalidEntryException e) {

175 System.out.println(e);

176 }

177 }

178 }

179 }

180 }

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

InvalidEntryException: No Category Selected

Test Case ID: 2.1		Current Status: PASS	Date: 03.28.25
Req. ID: 2.1 A new Entry may only be added if the entry is valid. Valid is not null.			
Step #	Operator Action	Expected Results	Comments
1	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set its data. Add e1 to entries.	The Entry object e1 should be added to the entries attribute.	<pre> 124 /*TEST MAIN */ 125 Run main Debug main Run Debug 126 public static void main(String[] args) { 127 Entries entries = new Entries(); 128 try { 129 Entry e = new Entry(entryNumber:1); 130 e.setAmount(amount:100.0); 131 e.setCategory(category:1); 132 e.setSubcategory(subcategory:1); 133 e.setSubcategory2(subcategory2:1); 134 e.setSubcategory3(subcategory3:1); 135 e.setSubcategory4(subcategory4:1); 136 e.setAccount(account:1); 137 e.setComment(comment:"This is a comment1"); 138 entries.addEntry(e); 139 140 } catch (InvalidEntryException e) {System.err.println(e);} 141 142 for(Entry e : entries.entries) 143 { 144 System.out.println(e.getEntryString()); 145 } 146 </pre>
2	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set it equal to null.	The Entry object e1 should be rejected from the entries.	<pre> 124 /*TEST MAIN */ 125 Run main Debug main Run Debug 126 public static void main(String[] args) { 127 Entries entries = new Entries(); 128 Entry e1 = null; 129 entries.addEntry(e1); 130 131 for(Entry e : entries.entries) 132 { 133 System.out.println(e.getEntryString()); 134 } </pre>

Screenshots:

```

124  /*TEST MAIN */
125  Run main | Debug main | Run | Debug
126  public static void main(String[] args) {
127      Entries entries = new Entries();
128      try {
129          Entry e = new Entry(entryNumber:1);
130          e.setAmount(amount:100.0);
131          e.setCategory(category:1);
132          e.setSubcategory(subcategory:1);
133          e.setSubcategory2(subcategory2:1);
134          e.setSubcategory3(subcategory3:1);
135          e.setSubcategory4(subcategory4:1);
136          e.setAccount(account:1);
137          e.setComment(comment:"This is a comment1");
138          entries.addEntry(e);
139
140          } catch (InvalidEntryException e) {System.err.println(e);}
141
142          for(Entry e : entries.entries)
143          {
144              System.out.println(e.getEntryString());
145          }
146

```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```

1,100.0,-1,1,1,1,1,1,1,This is a comment1
1,100.0,-1,1,1,1,1,1,1,This is a comment1

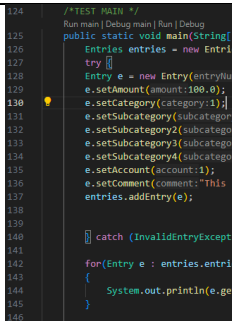
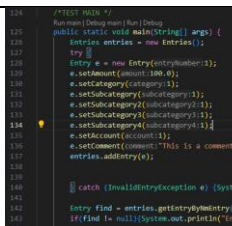
```


Test Case ID: 2.2		Current Status: pending	Date: 03.28.25
Req. ID: 2.2 An Entry shall be able to be removed based on and entryNumber.			
Step #	Operator Action	Expected Results	Comments
1	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set its data. Add e1 to entries.	Program should print the new entry instance in the format: id,dollarAmount,type,category,subcategory,subcategory2,subcategory3,subcategory4,accout,comment	<pre>124 /*TEST MAIN */ 125 Run main Debug main Run Debug 126 public static void main(String 127 entries entries = new Entr 128 try { 129 Entry e = new Entry(entry 130 e.setAmount(amount:100.0) 131 e.setCategory(category:1) 132 e.setSubcategory(subcateg 133 e.setSubcategory2(subcateg 134 e.setSubcategory3(subcateg 135 e.setSubcategory4(subcateg 136 e.setAccount(account:1); 137 e.setComment(comment:"Thi 138 entries.addEntry(e); 139 140 } catch (InvalidEntryExce 141 142 for(Entry e : entries.ent 143 { 144 System.out.println(e. 145 } 146 }</pre>
2	Use the removeEntry ByNumber() function to remove the entry by ID.	Program should print the rest of the entries in the array in this case nothing should be printed.	<pre>124 /*TEST MAIN */ 125 Run main Debug main Run Debug 126 public static void main(String 127 entries entries = new Entr 128 try { 129 Entry e = new Entry(entry 130 e.setAmount(amount:100.0) 131 e.setCategory(category:1) 132 e.setSubcategory(subcateg 133 e.setSubcategory2(subcateg 134 e.setSubcategory3(subcateg 135 e.setSubcategory4(subcateg 136 e.setAccount(account:1); 137 e.setComment(comment:"Thi 138 entries.addEntry(e); 139 140 } catch (InvalidEntryExce 141 142 for(Entry e : entries.ent 143 { 144 System.out.println(e. 145 } 146 147 entries.removeEntry(entry 148 System.out.println(s:"Aft 149 for(Entry e : entries.ent 150 { 151 System.out.println(e. 152 }</pre>
Screenshots:			

```
124  /*TEST MAIN */
125  Run main | Debug main | Run | Debug
126  public static void main(String[] args) {
127      Entries entries = new Entries();
128      try {
129          Entry e = new Entry(entryNumber:1);
130          e.setAmount(amount:100.0);
131          e.setCategory(category:1);
132          e.setSubcategory(subcategory:1);
133          e.setSubcategory2(subcategory2:1);
134          e.setSubcategory3(subcategory3:1);
135          e.setSubcategory4(subcategory4:1);
136          e.setAccount(account:1);
137          e.setComment(comment:"This is a comment1");
138          entries.addEntry(e);
139
140      } catch (InvalidEntryException e) {System.err.println(e);}
141
142      for(Entry e : entries.entries)
143      {
144          System.out.println(e.getEntryString());
145      }
146
147      entries.removeEntry(entryNumber:1);
148      System.out.println(x:"After removing entry 1:");
149      for(Entry e : entries.entries)
150      {
151          System.out.println(e.getEntryString());
152      }
153
154      //Entry e2 = new Entry(2);
155      // e2.setAmount(200.0);
156      ...
157  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
1,100.0,-1,1,1,1,1,1,1,This is a comment1
1,100.0,-1,1,1,1,1,1,1,This is a comment1
After removing entry 1:
```

Test Case ID: 2.3		Current Status: PASS	Date: 03.28.25
Req. ID: An Entry shall be able to be found by entryNumber.			
Step #	Operator Action	Expected Results	Comments
1	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set its data. Add e1 to entries.	Program should print the new entry instance in the format: id,dollarAmount,type,category,subcategory,subcategory2,subcategory3,subcategory4,account,comment	 <pre>124 /*TEST MAIN */ 125 Run main Debug main Run Debug 126 public static void main(String[] args) { 127 Entries entries = new Entries(); 128 try { 129 Entry e = new Entry(entryNumber:1); 130 e.setAmount(amount:100.0); 131 e.setCategory(category:1); 132 e.setSubcategory(subcategory:1); 133 e.setSubcategory2(subcategory:1); 134 e.setSubcategory3(subcategory:1); 135 e.setSubcategory4(subcategory:1); 136 e.setAccount(account:1); 137 e.setComment(comment:"This is a comment"); 138 entries.addEntry(e); 139 } catch (InvalidEntryException e) { 140 System.out.println(e.getMessage()); 141 } 142 for (Entry e : entries.getEntries()) { 143 System.out.println(e.getId() + " " + e.getAmount() + " " + e.getCategory() + " " + e.getSubcategory() + " " + e.getSubcategory2() + " " + e.getSubcategory3() + " " + e.getSubcategory4() + " " + e.getAccount() + " " + e.getComment()); 144 } 145 } 146 }</pre>
2	Use the getEntryByNmEntry function with parameter 1.	Program should return the entry and print it.	 <pre>124 /*TEST MAIN */ 125 Run main Debug main Run Debug 126 public static void main(String[] args) { 127 Entries entries = new Entries(); 128 try { 129 Entry e = new Entry(entryNumber:1); 130 e.setAmount(amount:100.0); 131 e.setCategory(category:1); 132 e.setSubcategory(subcategory:1); 133 e.setSubcategory2(subcategory:1); 134 e.setSubcategory3(subcategory:1); 135 e.setSubcategory4(subcategory:1); 136 e.setAccount(account:1); 137 e.setComment(comment:"This is a comment"); 138 entries.addEntry(e); 139 } catch (InvalidEntryException e) { 140 System.out.println(e.getMessage()); 141 } 142 Entry find = entries.getEntryByNmEntry(1); 143 if (find != null) { 144 System.out.println("Found entry: " + find.getId() + " " + find.getAmount() + " " + find.getCategory() + " " + find.getSubcategory() + " " + find.getSubcategory2() + " " + find.getSubcategory3() + " " + find.getSubcategory4() + " " + find.getAccount() + " " + find.getComment()); 145 } 146 }</pre>
Screenshots:			

```

124  /*TEST MAIN */
125  Run main | Debug main | Run | Debug
126  public static void main(String[] args) {
127      Entries entries = new Entries();
128      try {
129          Entry e = new Entry(entryNumber:1);
130          e.setAmount(amount:100.0);
131          e.setCategory(category:1);
132          e.setSubcategory(subcategory:1);
133          e.setSubcategory2(subcategory2:1);
134          e.setSubcategory3(subcategory3:1);
135          e.setSubcategory4(subcategory4:1);
136          e.setAccount(account:1);
137          e.setComment(comment:"This is a comment1");
138          entries.addEntry(e);
139
140      } catch (InvalidEntryException e) {System.err.println(e);}
141
142      Entry find = entries.getEntryByNmEntry(entryNumber:1);
143      if(find != null){System.out.println("Entry Found: "+find.getEntryString());}
144
145      // for(Entry e : entries.entries)

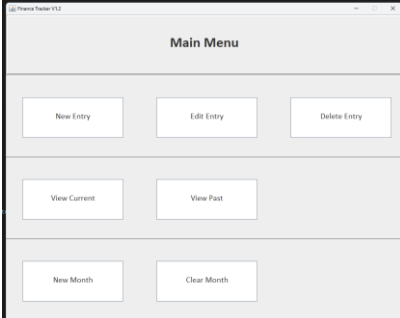
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

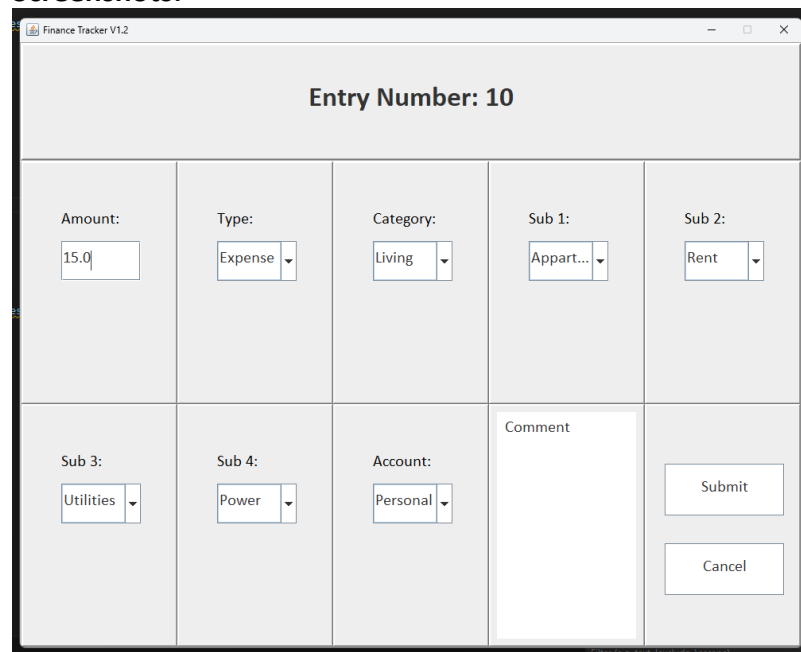
```

1,100.0,-1,1,1,1,1,1,1,This is a comment1
Entry Found: 1,100.0,-1,1,1,1,1,1,1,This is a comment1

```


Test Case ID: 3.2		Current Status: Pass	Date: 03.28.25
Req. ID: 3.2 The program shall load the Entry data if instantiated and display all the previously selected and entered values for the user to change.			
Step #	Operator Action	Expected Results	Comments
1	Run the main method of FinanceTracker.	The Main Menu Should Display.	
2	Select Edit Entry.	A new instance of Entry Frame should be created with a fabricated entry passed in the constructor.	<pre> public void addEditEntry() { System.out.println("Edit Entry"); Entry en = null; try{en = new Entry(entryNumber=10);en.setAmount(amount=15.0); en.setType(type=2);en.setCategory(category=1); en.setSubcategory(subcategory=1); en.setSubcategory2(subcategory2=3); en.setSubcategory3(subcategory3=3); en.setSubcategory4(subcategory4=3); en.setAccount(account=3); en.setComment(comment="Comment"); }catch(InvalidEntryException e){System.out.println(e);} new EntryFrame(title:"Entry Number: "+ new GridLayout(1,2,cols=5), en, this.entries); } </pre>

Screenshots:

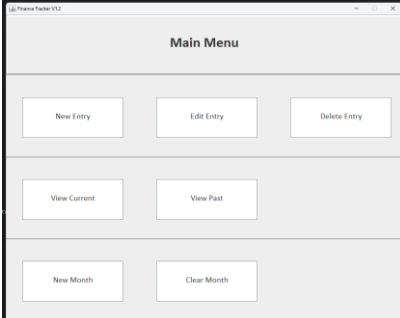
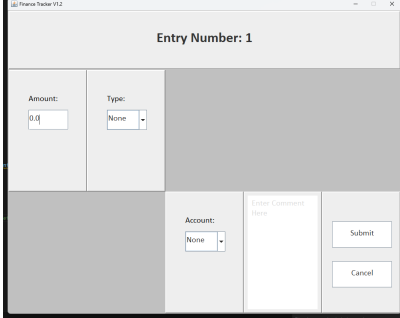
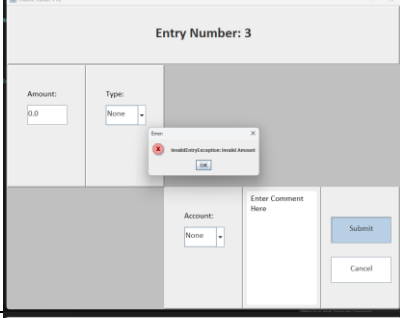
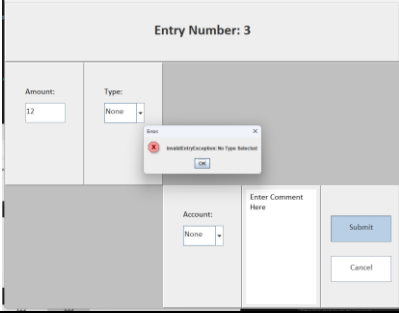


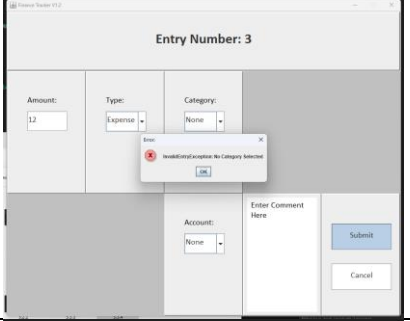
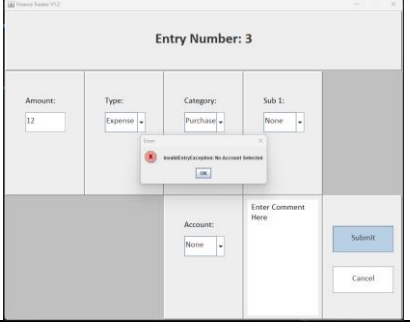
Finance Tracker V1.2

Entry Number: 10

Amount: 15.0	Type: Expense	Category: Living	Sub 1: Appart...	Sub 2: Rent
Sub 3: Utilities	Sub 4: Power	Account: Personal	Comment Submit Cancel	

Enter id, type, newSub, newAccount

Test Case ID: 3.3		Current Status: pending	Date: 03.28.25
Req. ID: 3.3 The program shall add the Entry when the submit button is pressed only if the Entry contains valid information.			
Step #	Operator Action	Expected Results	Comments
1	Run the main method of FinanceTracker.	The Main Menu Should Display.	
2	Select New Entry.	A new instance of Entry Frame should be created with a null entry passed in the constructor.	<pre>// EntryFrame.java public void addNewEntry() { System.out.println("New Entry"); Entry en = null; new EntryFrame(11111, "Entry Number: ", new BorderLayout(10, 10), en, this.entries); }</pre> 
3	Click Submit	An Error message should appear, invalid amount.	
4	Enter Amount > 0 and click Submit.	An Error message should appear, invalid type.	

5	Select type expense and click submit.	An Error message should appear, invalid category.	
6	Select category personal and click submit.	An Error message should appear, no account selected.	
7	Select Account personal and click submit.	The frame should close and return to the main menu.	
Screenshots:			

Test Case ID:		Current Status: << pass / fail / pending >>		Date:
Req. ID: Written requirement...				
Step #	Operator Action	Expected Results	Comments	
N	Describe the actions the tester needs to take in this step. Be sure to include specifics: filename, specific value/input, file to be used (put in the appendices), prompt and response...	Describe the expected response of the system based on the operator's action for this step.		
N				
N				
Screenshots:				

*[Shall be **completed** for user stories actively worked, and completed, in the current sprint.]*

REFERENCES:

Oracle. (n.d.). *Trail: Creating a GUI with swing*. Trail: Creating a GUI With Swing (The Java™ Tutorials). <https://docs.oracle.com/javase/tutorial/uiswing/index.html>

APPENDICES: