|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Data Type | Example Input | Expected Output | | Actual Output |
| First Name | Valid Data | Jeff, Bob | | Will only accept ascii / letters. | Accepts input and passes it on. Submits to the receipts without error |
| First Name | Invalid Data | 110, \*(&#, “” (blank) | | First name must be letters, no symbols or numbers or blanks | Throws error (calls error box). Clears input  Non-text character |
| Last Name | Valid Data | Jeff, Bob | | Will only accept ascii / letters | Accepts input and passes it on. Submits to the receipts without error |
| Last Name | Invalid Data | 110, \*(&#, “” (blank) | | Last name must be letters, no symbols or numbers or blanks | Throws error (calls error box). Clears input |
| Checkbox/ Toggle | Boolean | True or False | | Accepts either true or false and updates values accordingly | Gracefully handles toggles |
| Item quantity | Valid Data | 1-500 | | Will only accept numbers from 1 to 500 | Accepts input and passes it on |
| Item quantity | invalid Data | -1,(\*&@\*&!#, 892oiuqw, | | Doesn’t accept values other than 1 to 500 | Throws error (calls error box). Clears input |
| Submit Button | Valid Data | User click | | User clicks on the buttons, and an appropriate command / function will run. | Gracefully handles button presses |
| Message box | Invalid Data | User enters incorrect / invalid input | | Clears the input if necessary and tells the user where the invalid entry and item is, and it’s index ( if applicable to the entry) | User can continue after pressing OK / accepting the error |
| JSON Read and Write | Valid data | Program reads and writes to JSON accordingly | | Will output values to a function, which then uses them. | Gracefully handles exceptions |

Testing

**Debugging (Below add your debugging/fixes/modifications made)**

|  |  |
| --- | --- |
| Error | Modifications / Fixes / Debug |
| Traceback (most recent call last):  File "c:\Users\23040\OneDrive - Lynfield College\2025\2PAD\Lee\_AS91897\_AS91896\test1.py", line 21, in <module>  for position, item in l:  ^^^^^^^^^^^^^^  ValueError: too many values to unpack (expected 2) | Can’t iterate through the list (for l because the there are too many values. I am using a counter and a variable. Therefore it needs a tuple as the input for the for loop . To fix this, I have to use the enumerate function, which turns each object in the list to a counter AND the list item.  Changed:  for position, item in l:  To:  for position, item in enumerate(l): |
| Traceback (most recent call last):  File "c:\Users\23040\OneDrive - Lynfield College\2025\2PAD\Lee\_AS91897\_AS91896\test1.py", line 38, in <module>  quantity\_var.trace\_add("write", check\_totals())  ^^^^^^^^^^^^  NameError: name 'check\_totals' is not defined | Can’t find function check\_totals(). This is because the function appears after the this element. To fix this I have to add lambda.  Changed:  quantity\_var.trace\_add("write", check\_totals())  To:  quantity\_var.trace\_add("write", lambda: check\_totals()) |
| Exception in Tkinter callback  Traceback (most recent call last):  File "C:\Program Files\Python312\Lib\tkinter\\_\_init\_\_.py", line 1967, in \_\_call\_\_  return self.func(\*args)  ^^^^^^^^^^^^^^^^  TypeError: <lambda>() takes 0 positional arguments but 3 were given | Too many arguments into the lambda function. I need to add \*args so I can input more arguments.  Changed:  quantity\_var.trace\_add("write", lambda: check\_totals())  To:  quantity\_var.trace\_add("write", lambda \*args: check\_totals()) |
| Traceback (most recent call last):  File "c:\Users\23040\OneDrive - Lynfield College\2025\2PAD\Lee\_AS91897\_AS91896\main.py", line 375, in <module>  app = App()  ^^^^^  File "c:\Users\23040\OneDrive - Lynfield College\2025\2PAD\Lee\_AS91897\_AS91896\main.py", line 75, in \_\_init\_\_  self.display\_items()  File "c:\Users\23040\OneDrive - Lynfield College\2025\2PAD\Lee\_AS91897\_AS91896\main.py", line 128, in display\_items  command=lambda pos=postion: self.on\_check(pos),  ^^^^^^^  NameError: name 'postion' is not defined. Did you mean: 'position'? | Spelt position wrong. I had to fix the spelling |
| def check\_totals(self):          print("tracevar ran")          for i, item in enumerate(self.items):              value\_str = self.quantity\_vars[i].get()              if value\_str == "":                  self.total\_labels[i].configure(text="$0.00")                  continue              try:                  quantity = int(value\_str)                  if quantity < 0:                      self.error\_message("negative", item, i)                      self.total\_labels[i].configure(text="$0.00")                  elif quantity > 500:                      self.error\_message("too big", item, i)                      self.total\_labels[i].configure(text="$0.00")                  else:                      total = self.price[i] \* quantity                      self.total\_labels[i].configure(text=f"${total:.2f}")              except ValueError:                  self.total\_labels[i].configure(text="$0.00")                  self.error\_message("type", item, i) | Code fully works. I had to add continue to the end the for loop because it stopped the code and doesn’t continue.  Also added the try and except to further validate data |
| def check\_name(self, var, first\_last):          value = var.get()          if not value.isalpha() and value != "":              if first\_last == "last":                  self.error\_message("name type", "last name")              if first\_last == "first":                  self.error\_message("name type", "first name") | To be able to see if the input is only text, I used .isalpha () to scan the entire string. |
| if self.first\_name\_var.get() == "" or self.last\_name\_var.get() == "":              self.error\_message("names empty")              return | Check if name is empty and throws a message |
| receipts\_file = os.path.join(self.BASE\_DIR, "receipts.json")          try:              with open(receipts\_file, "r") as f:                  all\_receipts = json.load(f)          except (FileNotFoundError, json.JSONDecodeError):              all\_receipts = [] | I had to add os.path.join to make the code compatible with all operating systems (eg windows, mac, linux)  I also added the try and except to validate if there was a problem parsing / opening the json file. |
| with open(receipts\_file, "w") as f:              json.dump(all\_receipts, f, indent=4) | Write to json file using the previous receipts\_file, with an indent of 4. |
| def delete\_selected\_receipts(self):          receipts\_file = os.path.join(self.BASE\_DIR, "receipts.json")          try:              with open(receipts\_file, "r") as f:                  all\_receipts = json.load(f)          except Exception:              all\_receipts = []          # Keep only receipts that are NOT checked          all\_receipts = [r for i, r in enumerate(all\_receipts) if not self.receipt\_checkbox\_vars[i].get()]          with open(receipts\_file, "w") as f:              json.dump(all\_receipts, f, indent=4) | Delete receipts FROM json, not just from the display. Get the receipts that are not checked and saves them to a variable, then overwrites the entire file.  I had to use for loop compression to simplify the code and make it easier to understand.  I had to use json.dump because of the way my code works. |