
PASSWORD MANAGER

TEAM 1 C2

By:

Mahdi Bathallath – {2236809}

Taha Barzanji – {2135092}

Mohammed AlShehri – {2042124}



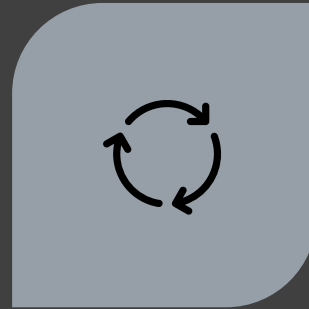
PROCESS



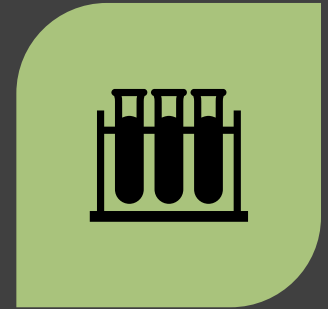
SPECIFICATIONS



DESIGN

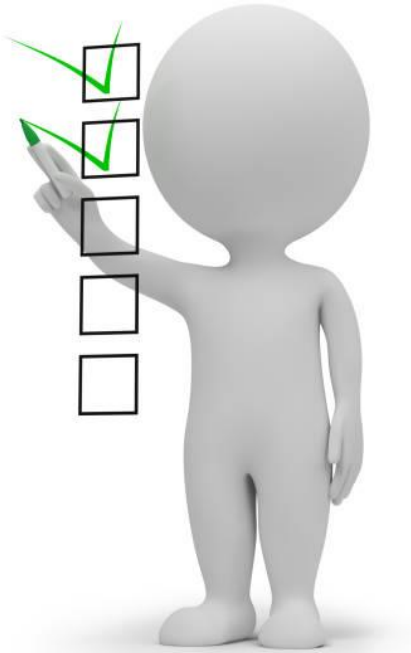


IMPLEMENTATION



TESTING

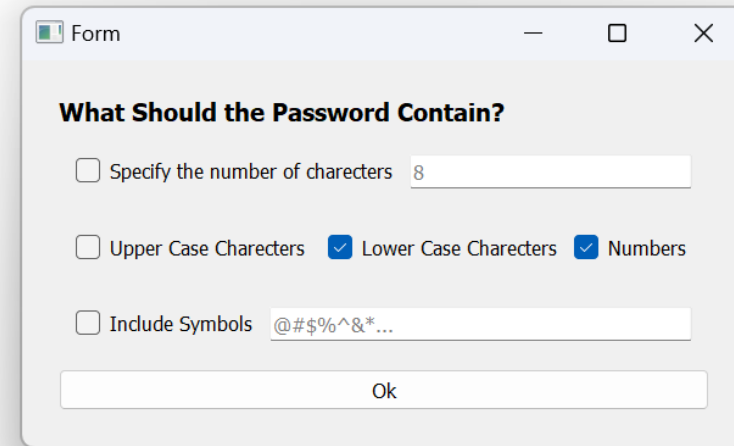
SPECIFICATIONS



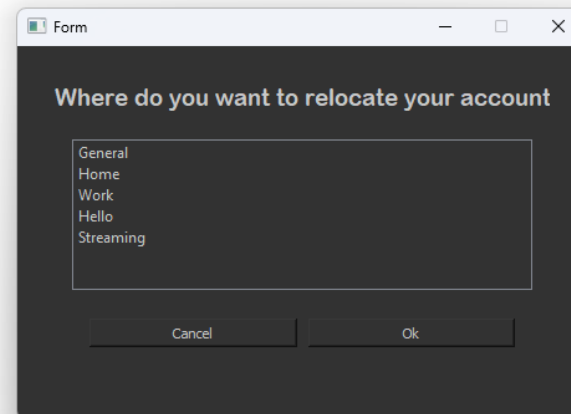
- GUI using PyQt5
- Create and associate passwords
- Master lists
- Encryption
- Accept Remote files
- Local files

EXTRA FEATURES

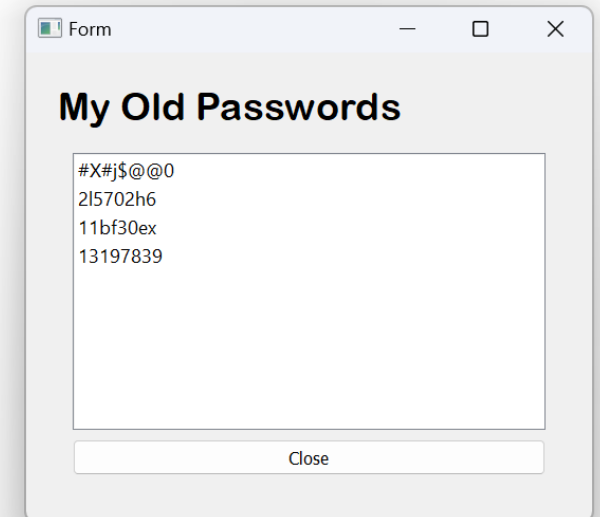
- Password Generator with spec.
- View Old Passwords
- Recently Deleted
- Restore Recently Deleted
- Shortcuts



A screenshot of a Windows-style dialog box titled "Form". The title bar has standard minimize, maximize, and close buttons. The dialog content is titled "What Should the Password Contain?". It features three rows of options, each with a checkbox and a text input field. The first row is "Specify the number of charecters" (note the typo) with a text box containing "8". The second row has "Upper Case Characters" (unchecked), "Lower Case Characters" (checked), and "Numbers" (checked). The third row is "Include Symbols" with a text box containing "@#\$\$%^&*...". At the bottom is an "Ok" button.



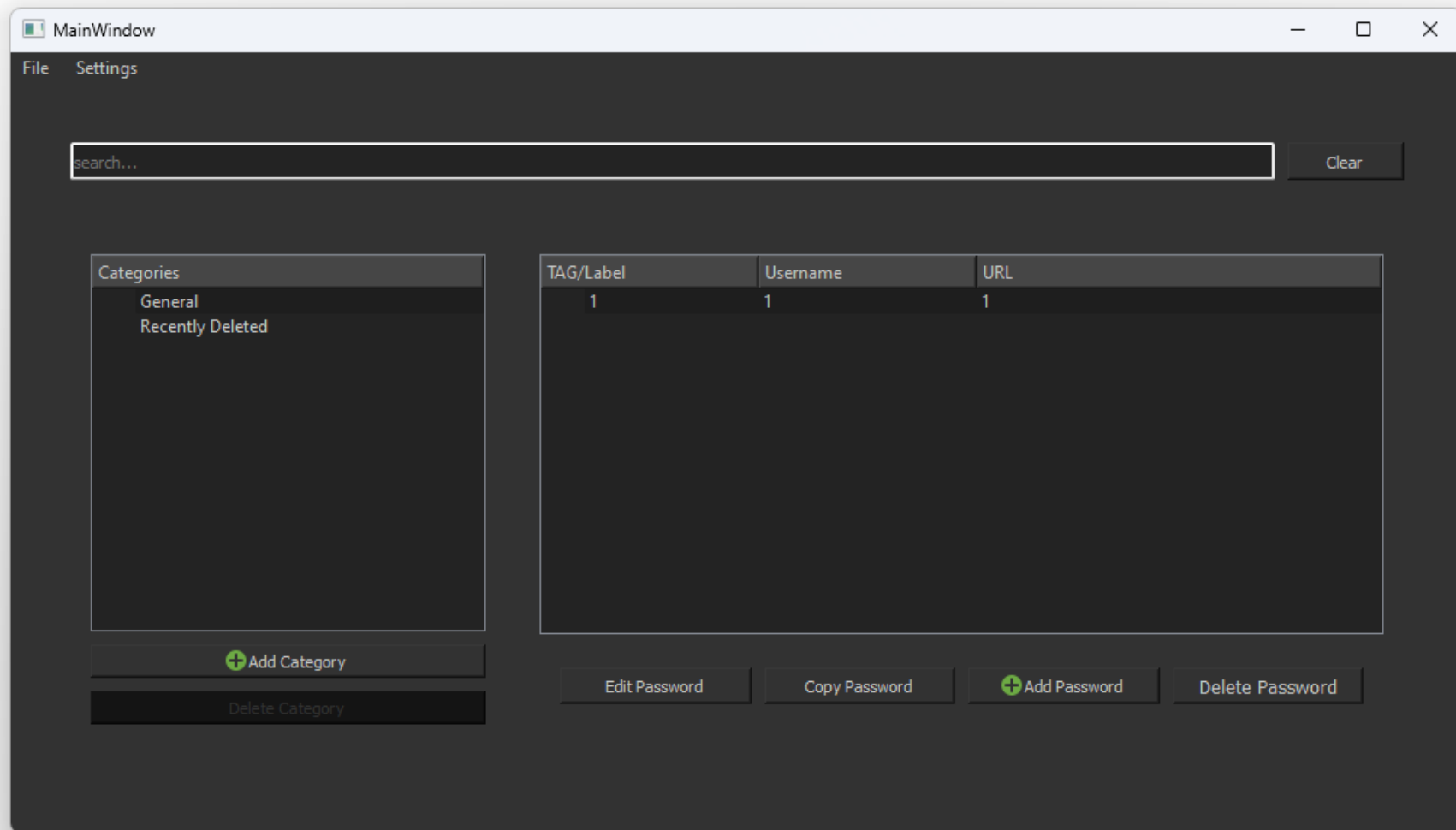
A screenshot of a Windows-style dialog box titled "Form". The title bar has standard minimize, maximize, and close buttons. The dialog content is titled "Where do you want to relocate your account". It features a list box containing the following items: "General", "Home", "Work", "Hello", and "Streaming". At the bottom are "Cancel" and "Ok" buttons.



A screenshot of a Windows-style dialog box titled "Form". The title bar has standard minimize, maximize, and close buttons. The dialog content is titled "My Old Passwords". It features a text area containing a list of passwords: "#X#j\$@@0", "2l5702h6", "11bf30ex", and "13197839". At the bottom is a "Close" button.

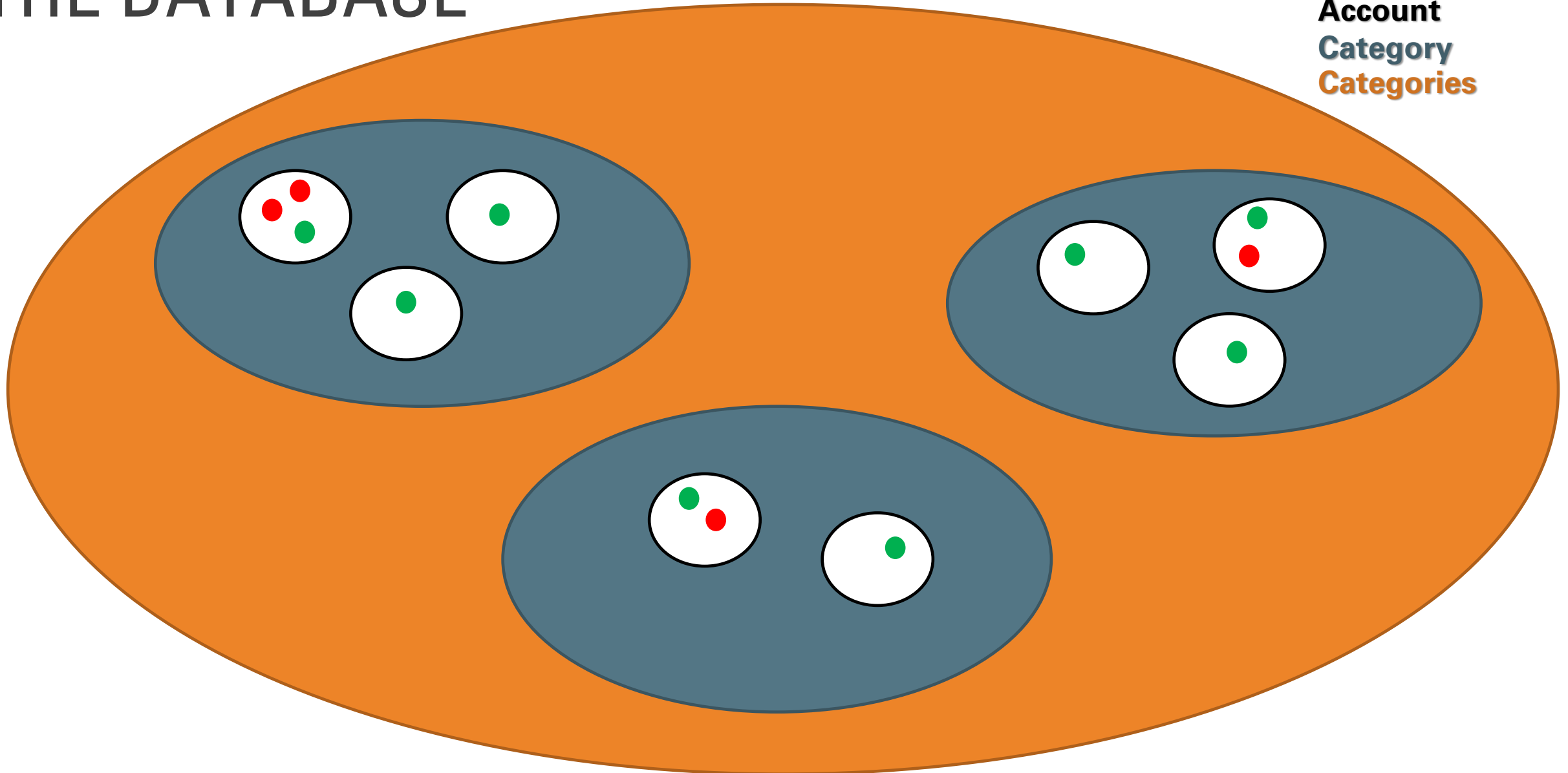
DESIGN AND IMPLEMENTATION





THE DATABASE

Passwords
Account
Category
Categories



ACCESSING OLD PASSWORDS

```
102     def append_to_passwords(self, new_password, category_index, acc_index):
103         for password in self.categories[category_index].Accounts[acc_index].Password:
104             if password == new_password:
105                 self.categories[category_index].Accounts[acc_index].Password.remove(password)
106         self.categories[category_index].Accounts[acc_index].Password.append(new_password)
```

```
def copy_password(self):
    try:
        acc_index = self.database.get_account_index(self.category_index, self.acc_tag_name)
        copied = self.database.categories[self.category_index].Accounts[acc_index].Password[-1]

        copy = QApplication.clipboard()
        copy.clear(mode = copy.Clipboard)
        copy.setText(copied, mode = copy.Clipboard)

        self.main.statusbar.showMessage("Copied")
        QTest.QTest.qWait(2000)
        self.main.statusbar.showMessage("")
```


TAG DUPLICATES

```
64     def check_duplicate_tags(self, tag, category_index=-1, acc_index=-1):
65         for i in range(len(self.categories)):
66             for j in range(len(self.categories[i].Accounts)):
67                 if category_index == i and acc_index == j:
68                     continue
69                 elif self.categories[i].Accounts[j].Tag == tag:
70                     raise Exception()
```

```
88     def edit_account(self, category_index, old_tag, new_tag, username, password, url):
89         account_index = self.get_account_index(category_index, old_tag)
90         try:
91             self.check_duplicate_tags(new_tag, category_index, account_index)
92             self.categories[category_index].Accounts[account_index].Tag = new_tag
93             self.categories[category_index].Accounts[account_index].Username = username
94             self.append_to_passwords(password, category_index, account_index)
95             self.categories[category_index].Accounts[account_index].Url = url
96             print(self.categories)
97         except:
98             raise Exception()
```

- First check
- Then change

```
723         try:
724             self.database.add_account(self.category_index, tag, username, password, URL)
725             new_item = QtWidgets.QTreeWidgetItem()
726             new_item.setText(0, tag)
727             new_item.setText(1, username)
728             new_item.setText(2, URL)
729             new_item.setText(3, self.category_title)
730             self.main.accountsTreeWidget.addTopLevelItem(new_item)
731             self.addEditWindow.close()
732         except:
733             error_msg = QMessageBox()
734             error_msg.setWindowTitle("Error")
735             error_msg.setText("The tag must have a unique title")
736             error_msg.setIcon(QMessageBox.Warning)
737             error_msg.setStandardButtons(QMessageBox.Ok)
738             error_msg.show()
739             error_msg.exec_()
740             error_msg.buttonClicked.connect(lambda: error_msg.close())
```

ACCOUNT AND BUTTON ACCESSIBILITY

```
585     def accounts_buttons_disabler(self):
586         if self.database.categories[self.category_index].Accounts == []:
587             self.main.editPasswordButton.setEnabled(False)
588             self.main.copyPasswordButton.setEnabled(False)
589             self.main.deletePasswordButton.setEnabled(False)
590         else:
591             self.main.editPasswordButton.setEnabled(True)
592             self.main.copyPasswordButton.setEnabled(True)
593             self.main.deletePasswordButton.setEnabled(True)
```

```
607     if self.category_index == 0:
608         self.selectedCat.setFlags(self.selectedCat.flags() & ~QtCore.Qt.ItemIsEditable)
609         self.main.editPasswordButton.setText("Edit Password")
610         self.main.addPasswordButton.setEnabled(True)
611         self.main.deleteCategoryButton.setEnabled(False)
612     elif self.category_index == 1:
613         self.main.editPasswordButton.setText("Restore Account")
614         self.selectedCat.setFlags(self.selectedCat.flags() & ~QtCore.Qt.ItemIsEditable)
615         self.main.addPasswordButton.setEnabled(False)
616         self.main.deleteCategoryButton.setEnabled(False)
617     else:
618         self.main.editPasswordButton.setText("Edit Password")
619         self.main.deleteCategoryButton.setEnabled(True)
620         self.selectedCat.setFlags(self.selectedCat.flags() | QtCore.Qt.ItemIsEditable)
621         self.main.addPasswordButton.setEnabled(True)
622
623     self.check_category_name()
624     self.refresh_accounts_treewidget()
```

ACCESSING INDEX OF ITEM IN CATEGORY

- Edit Account
- Del Account
- Copy Account

```
72     def get_account_index(self, category_index, tag):
73         for i in range(len(self.categories[category_index].Accounts)):
74             if self.categories[category_index].Accounts[i].Tag == tag:
75                 return i
```

```
47     def relocate(self, category_index, tag):
48         account_index = self.get_account_index(1, tag)
49         temp = self.categories[1].Accounts[account_index]
50         self.categories[1].Accounts.pop(account_index)
51         self.categories[category_index].Accounts.append(temp)
```

```
77     def del_account(self, category_index, tag):
78         account_index = self.get_account_index(category_index, tag)
79         account = self.categories[category_index].Accounts[account_index]
80
81         if category_index == 1:
82             pass
83         else:
84             self.categories[1].Accounts.append(account)
85             self.categories[category_index].Accounts.remove(account)
```



TESTING

SET WINDOW MODALITY

- Change category index
- Change account index
- => end up in unwanted place

```
646     def add_account(self):
647         self.edit = False
648         self.addEditWindow.setWindowModality(QtCore.Qt.ApplicationModal)
649         self.addEditWindow.show()
650         self.add_edit.tagLineEdit.clear()
651         self.add_edit.usernameLineEdit.clear()
652         self.add_edit.passwordLineEdit.clear()
653         self.add_edit.urlLineEdit.clear()
654         self.add_edit.viewOldButton.setEnabled(False)
```

EXPECTATION HANDLING

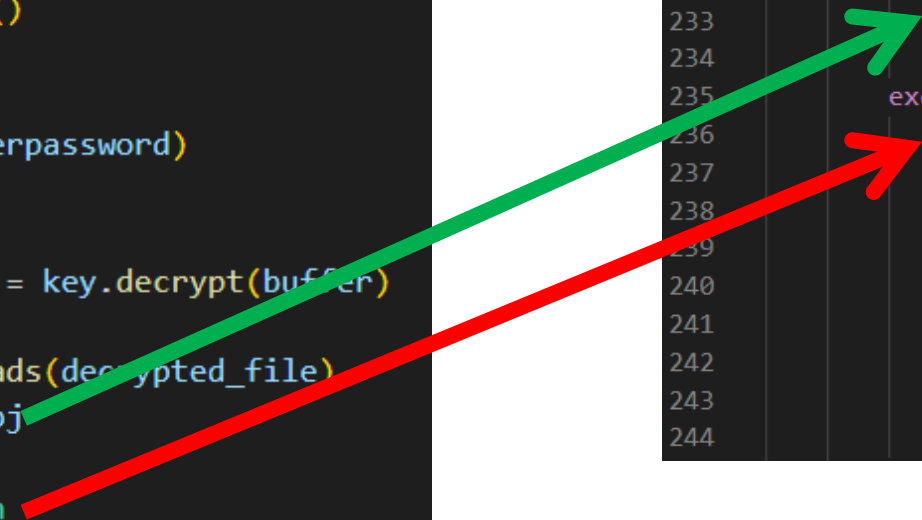
```
def Open(self, path, masterpassword):
    file = open(path, 'rb')
    buffer = file.read()
    file.close()

    key = Encrypt(masterpassword)

    try:
        decrypted_file = key.decrypt(buffer)

        openedobj = loads(decrypted_file)
        return openedobj
    except:
        raise Exception
```

```
228 def return_opened_local_file(self):
229     path = self.path
230     masterpassword = self.MP.passwordLineEdit.text()
231     try:
232         self.database = self.database.Open(path, masterpassword)
233         self.MPWindow.close()
234         self.refrsh_ui()
235     except:
236         error_msg = QMessageBox()
237         error_msg.setWindowTitle("Error")
238         error_msg.setText("Wrong Password! Try again")
239         error_msg.setIcon(QMessageBox.Warning)
240         error_msg.setStandardButtons(QMessageBox.Ok)
241         error_msg.show()
242         error_msg.exec_()
243         error_msg.buttonClicked.connect(lambda: error_msg.close())
244         self.MPWindow.show()
```

A diagram illustrating the flow of control between two code blocks. A green arrow originates from the 'return openedobj' line in the first code block and points to the 'try' block in the second code block. A red arrow originates from the 'raise Exception' line in the first code block and points to the 'except' block in the second code block.

THEME PREFERENCES

```
184 def theme_preferences(self, mode):
185     f = open("Theme", "w")
186     f.write(mode)
187     f.close()
188
189 def check_theme_preferences(self):
190     theme = open("Theme", "r")
191     Theme_mode = theme.read()
192     theme.close()
193     if Theme_mode == "DarkMode":
194         return 1
195     elif Theme_mode == "DefaultMode":
196         return 0
```

__INIT__

```
155 if self.database.check_theme_preferences() == 1:
156     self.DarkTheme()
157 else:
158     self.DefaultTheme()
```

CONCLUSION



LEARNED



USED



FLAWS



NEXT

COMPLETED WORK ALONE

Mahdi Bathallath	Taha Barzanji	Mohammed AlShehri
		

TASK WEIGHTING

TASKS	Mahdi Bathallath	Taha Barzanji	Mohammed AlShehri
UI Sketch {5%}	✓	✓	✗
UI Creation {5%}	✓	✓	✗
Data Structure {20%}	✓	✓	✗
UI with triggers {10%}	✓	✓	✗
Open/Save files {10%}	✓	✓	✗
Open URL {5%}	✓	✓	✗
Encryption {5%}	✓	✓	✗
Packaging {20%}	✓	✓	✗
Extra Features {20%}	✓	✓	✗
TOTAL Contribution:	51%	49%	0%

ANY QUESTIONS

