PASSWORD MANAGER

TEAM 1 C2

By:

Mahdi Bathallath – {2236809}

Taha Barzanji – {2135092}

Mohammed AlShehri – {2042124}

PROCESS

SPECIFICATIONS

DESIGN

IMPLEMENTATION

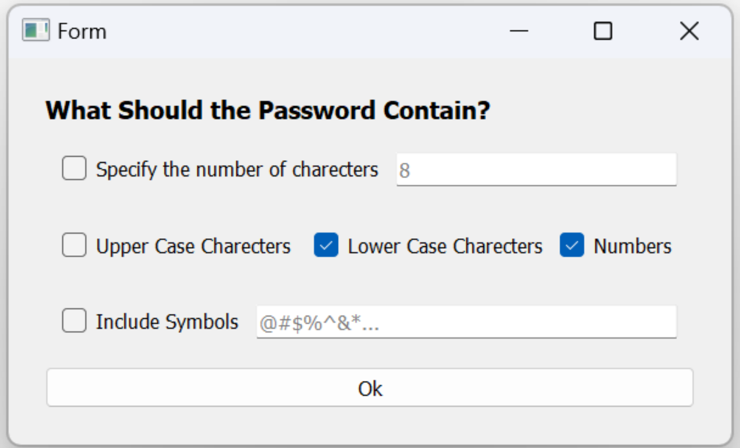
TESTING

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

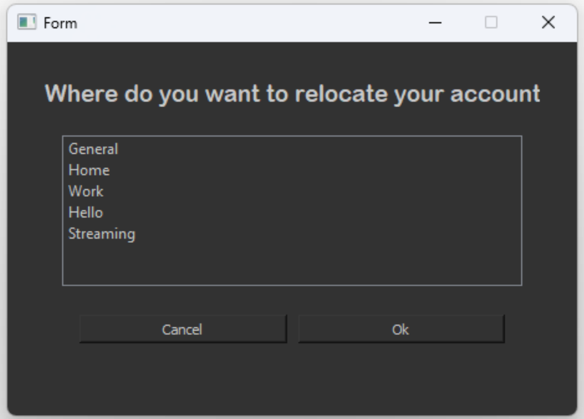

Form

What Should the Password Contain?

☐ Specify the number of charecters 8

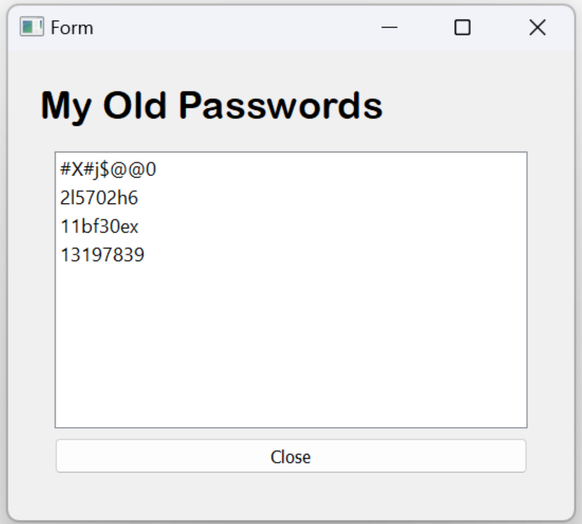
☐ Upper Case Charecters ✓ Lower Case Charecters ✓ Numbers

☐ Include Symbols @#$%^&*...


Form

Where do you want to relocate your account

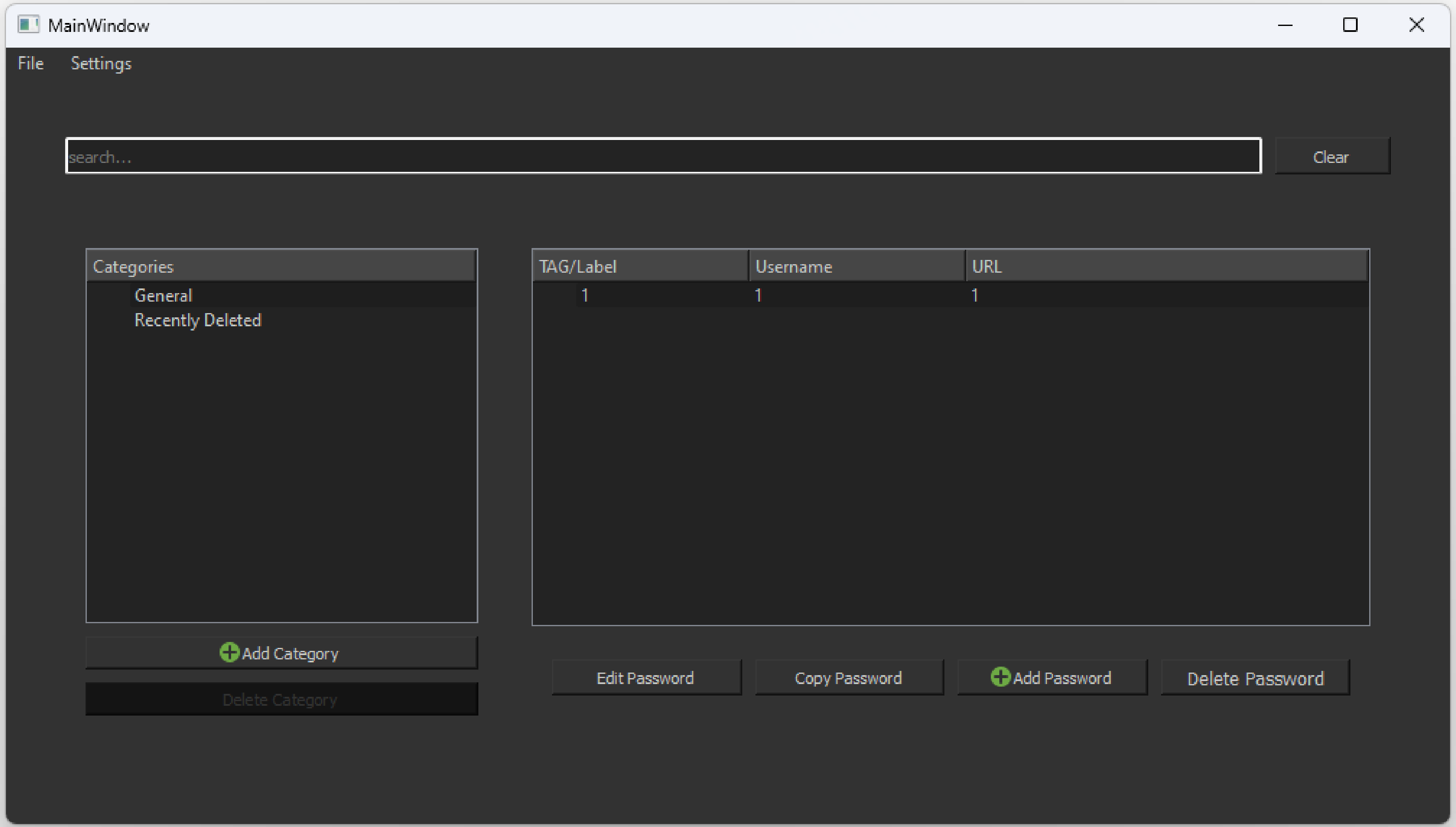
General
Home
Work
Hello
Streaming


Form

My Old Passwords

#X#j$@@0
2l5702h6
11bf30ex
13197839

DESIGN AND IMPLEMENTATION


MainWindow

File Settings

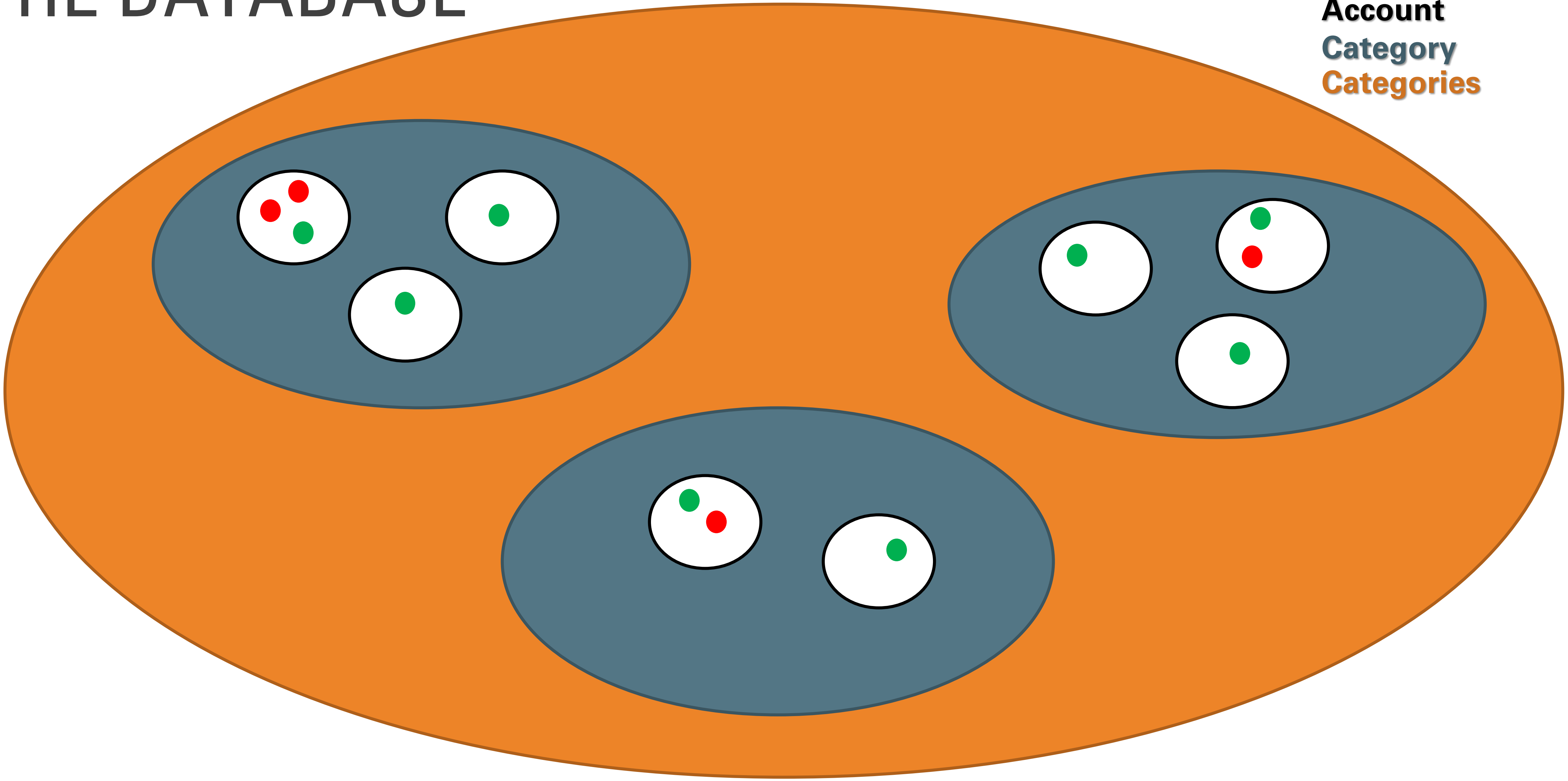
search...

Clear

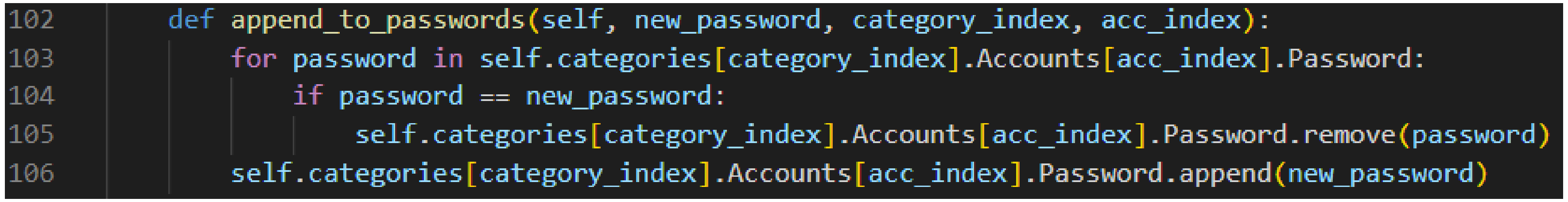
TAG/Label	Username	URL
1	1	1

THE DATABASE

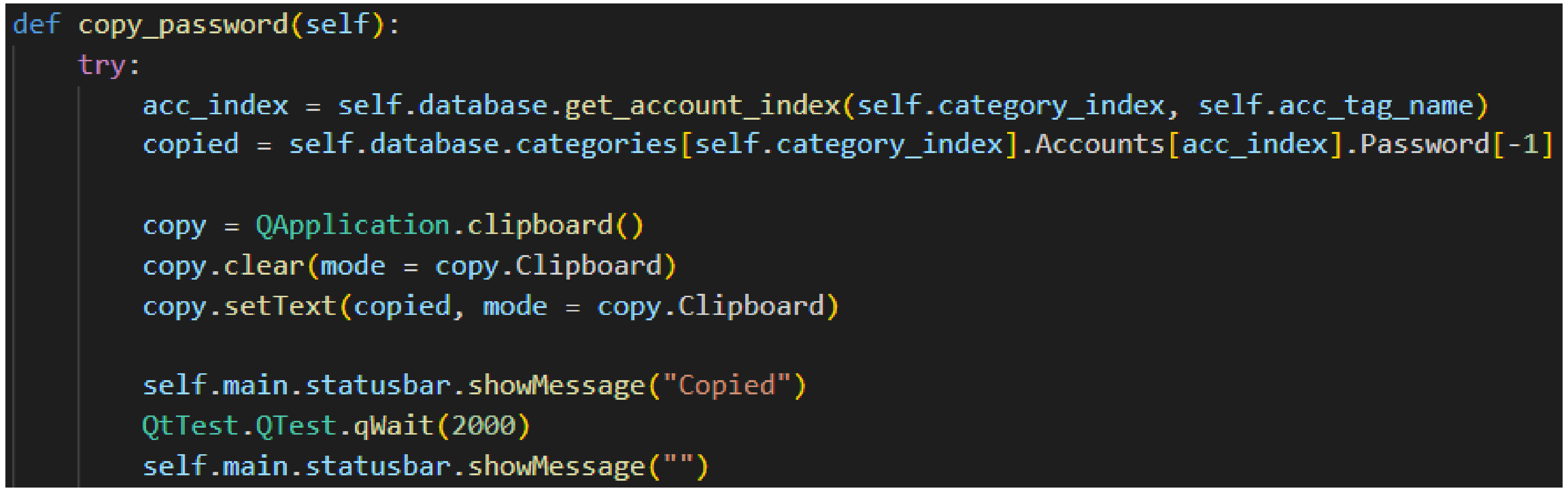
Passwords
Account
Category
Categories



ACCESSING OLD PASSWORDS

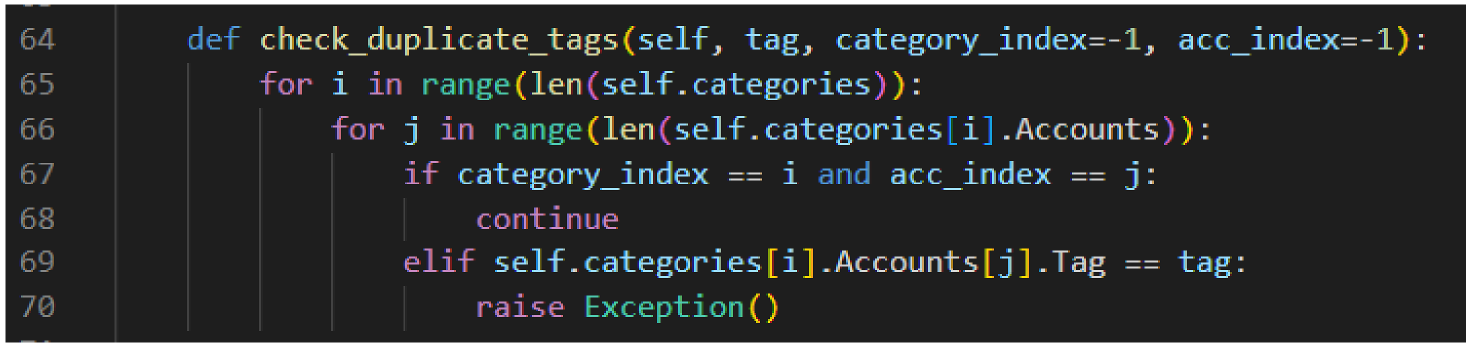


```
def append_to_passwords(self, new_password, category_index, acc_index):
    for password in self.categories[category_index].Accounts[acc_index].Password:
        if password == new_password:
            self.categories[category_index].Accounts[acc_index].Password.remove(password)
    self.categories[category_index].Accounts[acc_index].Password.append(new_pas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

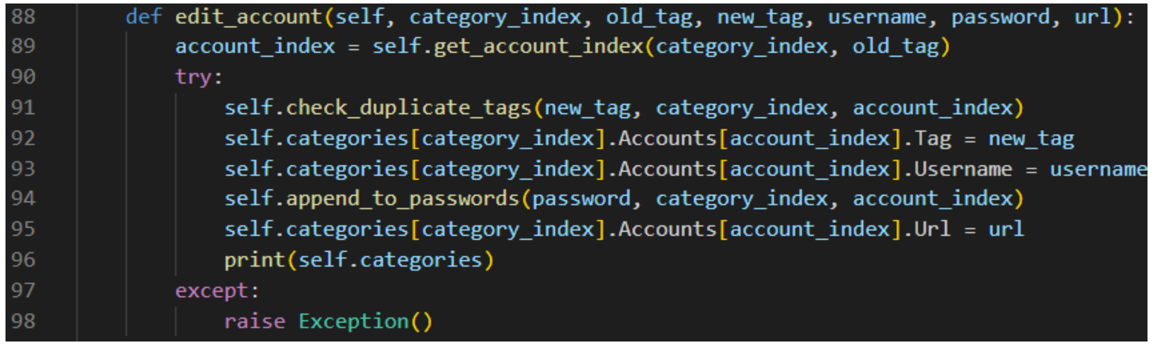


```
def copy_password(self):
    try:
        acc_index = self.database.get_account_index(self.category_index, self.acc_tag_name)
        copied = self.database.categories[self.category_index].Accounts[acc_index].Password[-1]

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

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

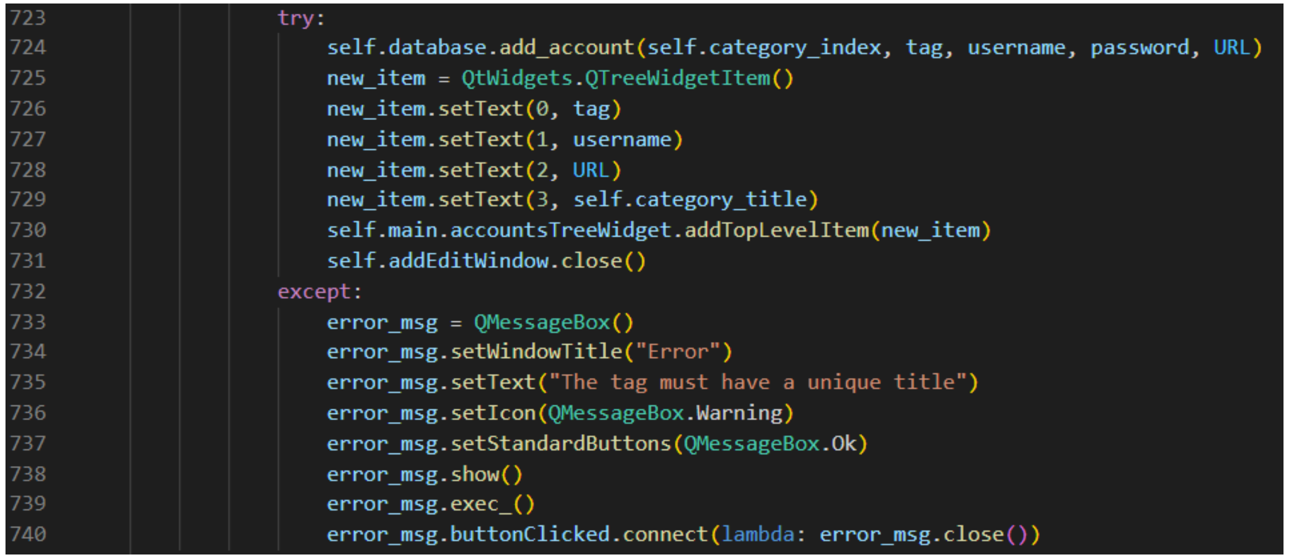
TAG DUPLICATES



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



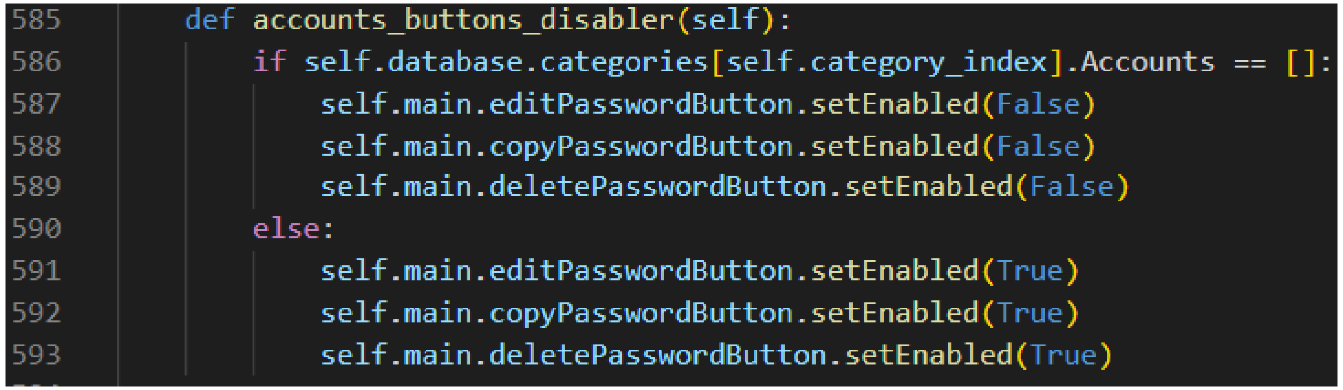
```python
    def edit_account(self, category_index, old_tag, new_tag, username, password, url):
        account_index = self.get_account_index(category_index, old_tag)
        try:
            self.check_duplicate_tags(new_tag, category_index, account_index)
            self.categories[category_index].Accounts[account_index].Tag = new_tag
            self.categories[category_index].Accounts[account_index].Username = username
            self.append_to_passwords(password, category_index, account_index)
            self.categories[category_index].Accounts[account_index].Url = url
            print(self.categories)
        except:
            raise Exception()
```



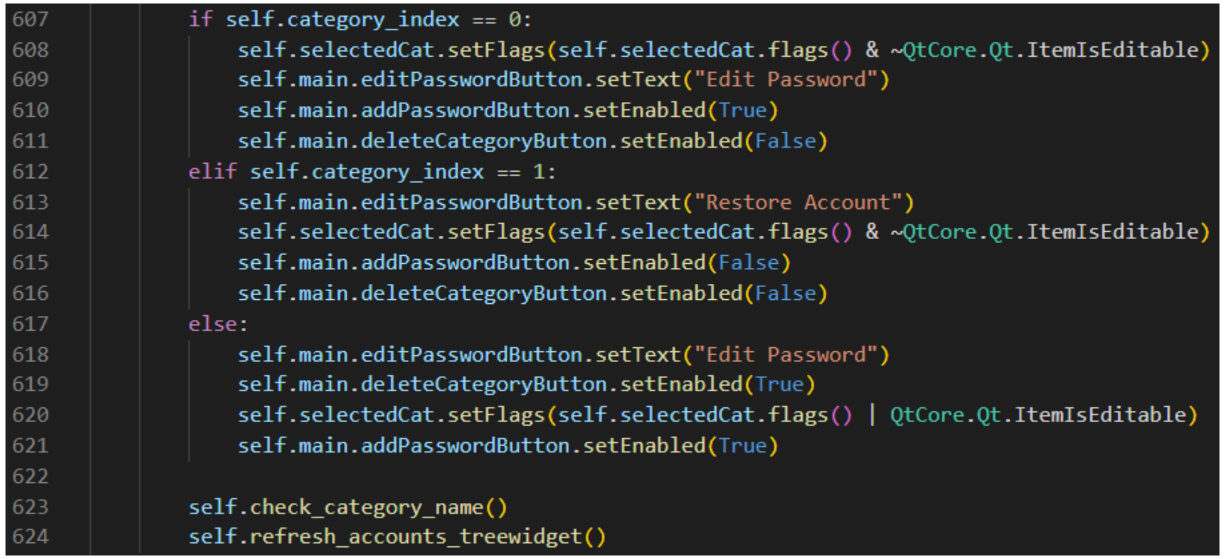
```python
            try:
                self.database.add_account(self.category_index, tag, username, password, URL)
                new_item = QtWidgets.QTreeWidgetItem()
                new_item.setText(0, tag)
                new_item.setText(1, username)
                new_item.setText(2, URL)
                new_item.setText(3, self.category_title)
                self.main.accountsTreeWidget.addTopLevelItem(new_item)
                self.addEditWindow.close()
            except:
                error_msg = QMessageBox()
                error_msg.setWindowTitle("Error")
                error_msg.setText("The tag must have a unique title")
                error_msg.setIcon(QMessageBox.Warning)
                error_msg.setStandardButtons(QMessageBox.Ok)
                error_msg.show()
                error_msg.exec_()
                error_msg.buttonClicked.connect(lambda: error_msg.close())
```

- First check
- Then change

ACCOUNT AND BUTTON ACCESSIBILITY



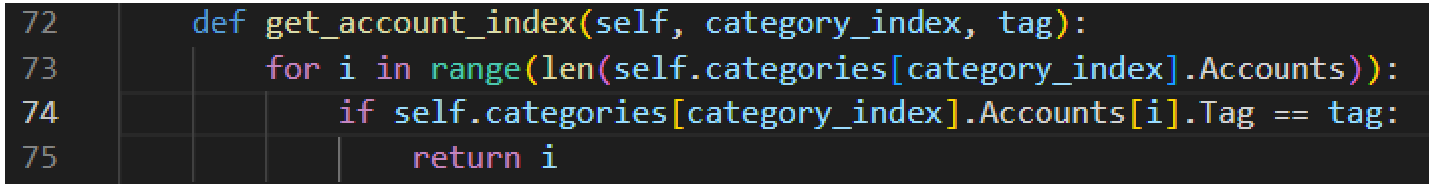
```python
    def accounts_buttons_disabler(self):
        if self.database.categories[self.category_index].Accounts == []:
            self.main.editPasswordButton.setEnabled(False)
            self.main.copyPasswordButton.setEnabled(False)
            self.main.deletePasswordButton.setEnabled(False)
        else:
            self.main.editPasswordButton.setEnabled(True)
            self.main.copyPasswordButton.setEnabled(True)
            self.main.deletePasswordButton.setEnabled(True)
```



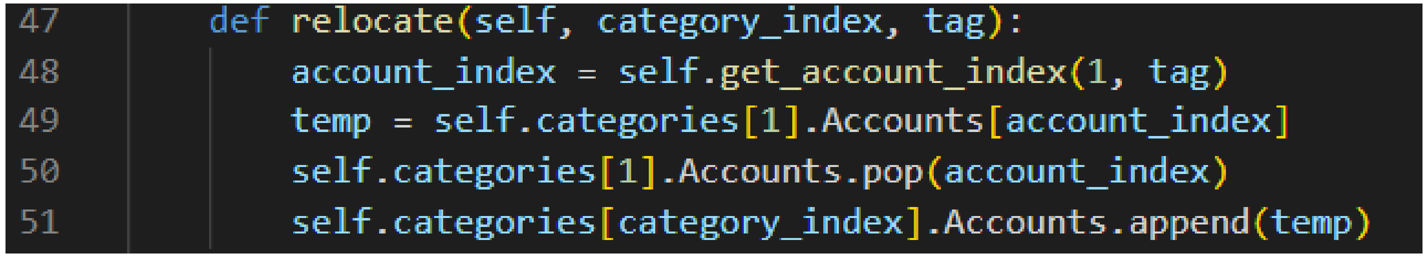
```python
        if self.category_index == 0:
            self.selectedCat.setFlags(self.selectedCat.flags() & ~QtCore.Qt.ItemIsEditable)
            self.main.editPasswordButton.setText("Edit Password")
            self.main.addPasswordButton.setEnabled(True)
            self.main.deleteCategoryButton.setEnabled(False)
        elif self.category_index == 1:
            self.main.editPasswordButton.setText("Restore Account")
            self.selectedCat.setFlags(self.selectedCat.flags() & ~QtCore.Qt.ItemIsEditable)
            self.main.addPasswordButton.setEnabled(False)
            self.main.deleteCategoryButton.setEnabled(False)
        else:
            self.main.editPasswordButton.setText("Edit Password")
            self.main.deleteCategoryButton.setEnabled(True)
            self.selectedCat.setFlags(self.selectedCat.flags() | QtCore.Qt.ItemIsEditable)
            self.main.addPasswordButton.setEnabled(True)

        self.check_category_name()
        self.refresh_accounts_treewidget()
```

ACCESSING INDEX OF ITEM IN CATEGORY

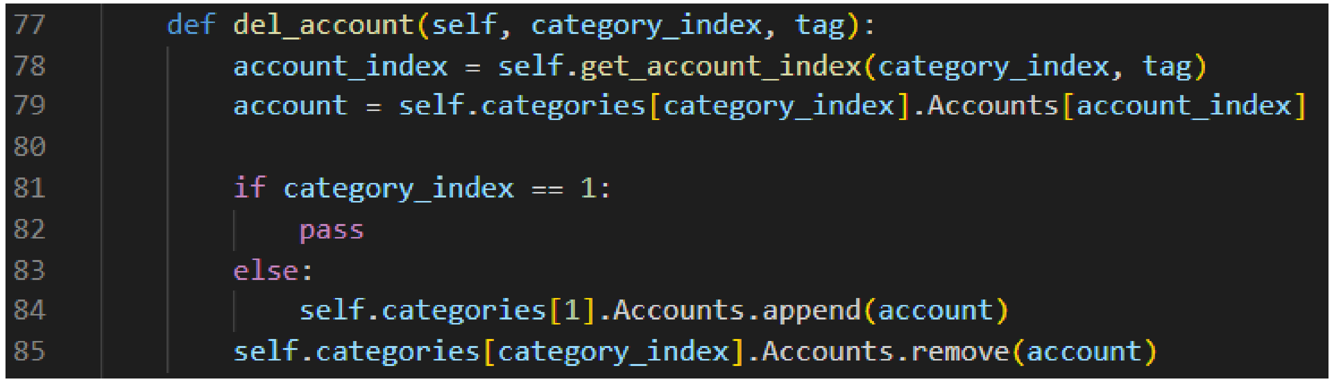
- Edit Account
- Del Account
- Copy Account



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



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

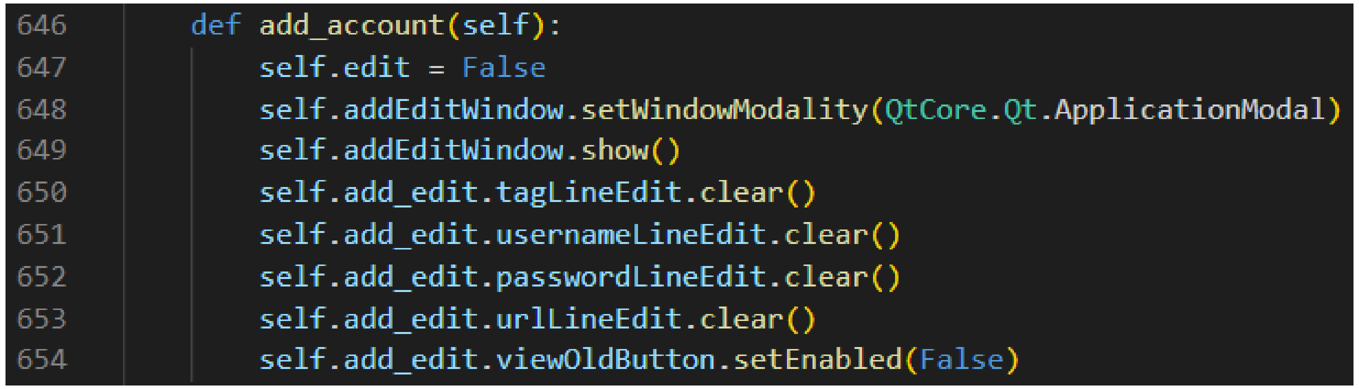


```
    def del_account(self, category_index, tag):
        account_index = self.get_account_index(category_index, tag)
        account = self.categories[category_index].Accounts[account_index]

        if category_index == 1:
            pass
        else:
            self.categories[1].Accounts.append(account)
        self.categories[category_index].Accounts.remove(account)
```

TESTING

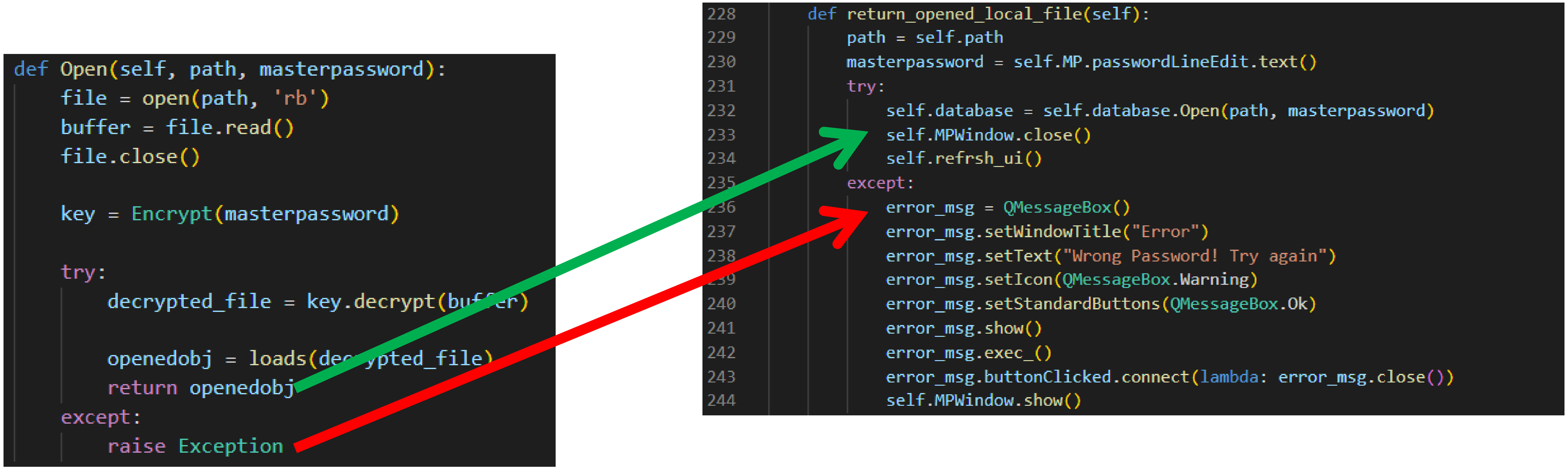
SET WINDOW MODALITY

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



```
    def add_account(self):
        self.edit = False
        self.addEditWindow.setWindowModality(QtCore.Qt.ApplicationModal)
        self.addEditWindow.show()
        self.add_edit.tagLineEdit.clear()
        self.add_edit.usernameLineEdit.clear()
        self.add_edit.passwordLineEdit.clear()
        self.add_edit.urlLineEdit.clear()
        self.add_edit.viewOldButton.setEnabled(False)
```

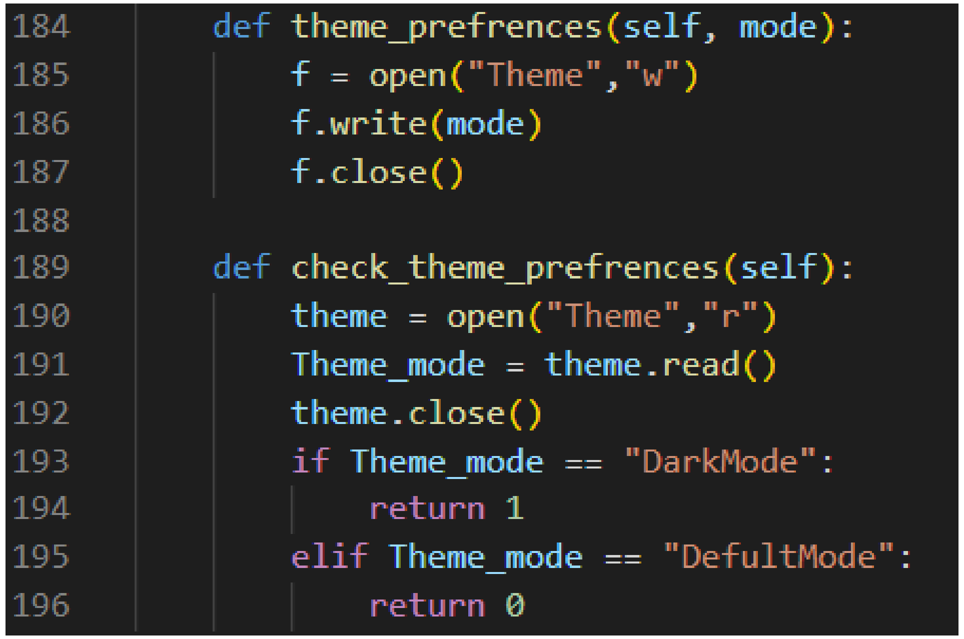
EXPECTATION HANDLING



```python
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
```

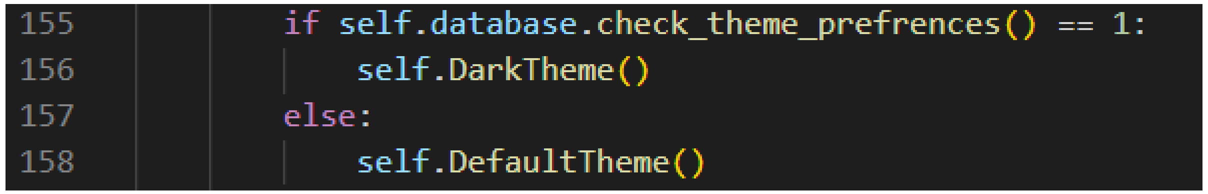
```python
    def return_opened_local_file(self):
        path = self.path
        masterpassword = self.MP.passwordLineEdit.text()
        try:
            self.database = self.database.Open(path, masterpassword)
            self.MPWindow.close()
            self.refrsh_ui()
        except:
            error_msg = QMessageBox()
            error_msg.setWindowTitle("Error")
            error_msg.setText("Wrong Password! Try again")
            error_msg.setIcon(QMessageBox.Warning)
            error_msg.setStandardButtons(QMessageBox.Ok)
            error_msg.show()
            error_msg.exec_()
            error_msg.buttonClicked.connect(lambda: error_msg.close())
            self.MPWindow.show()
```

THEME PREFERENCES



```
    def theme_prefrences(self, mode):
        f = open("Theme","w")
        f.write(mode)
        f.close()

    def check_theme_prefrences(self):
        theme = open("Theme","r")
        Theme_mode = theme.read()
        theme.close()
        if Theme_mode == "DarkMode":
            return 1
        elif Theme_mode == "DefultMode":
            return 0
```

__INIT__



```
            if self.database.check_theme_prefrences() == 1:
                self.DarkTheme()
            else:
                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%}	✓	✓	x
UI Creation {5%}	✓	✓	x
Data Structure {20%}	✓	✓	x
UI with triggers {10%}	✓	✓	x
Open/Save files {10%}	✓	✓	x
Open URL {5%}	✓	✓	x
Encryption {5%}	✓	✓	x
Packaging {20%}	✓	✓	x
Extra Features {20%}	✓	✓	x
TOTAL Contribution:	51%	49%	0%

ANY QUESTIONS