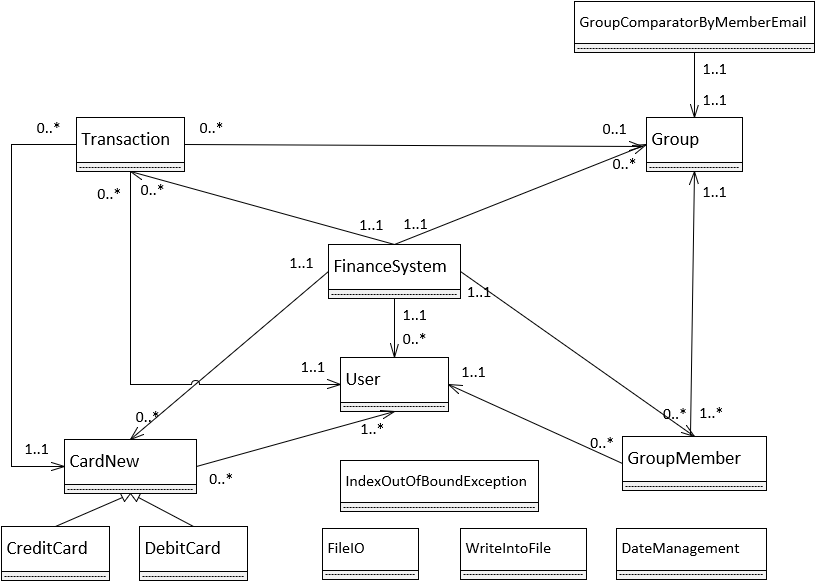
|  |  |
| --- | --- |
| **ISM 6259 Project** | Final Report  Project Name:  Finance Management Software  Group Members:  Emily Pfeffer  Yifei Sun  Shree Vidhya Sampath |

**Abstract**

Our project will revolve around a financial software information system, in which users can manage their personal finances for all transactions manually. People need an application to help them efficiently and easily manage their finances these days. With our Finance Management System, a general user can manage their personal finances, as well as group finances. For example, a couple may have joint cards, accounts, and transactions, so they may need to see each other’s purchases as a group. The Finance Management System lets each user record or modify their transactions, store their card information, create groups to share information with, and view summarized monthly reports of their transactions.

Before we walk through a demo of the Finance System, let’s delve into the structure.

**Class Diagram**

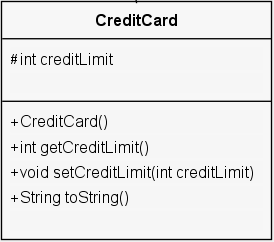
****

**Figure 1. (High-Level) Class Diagram of Finance System.**

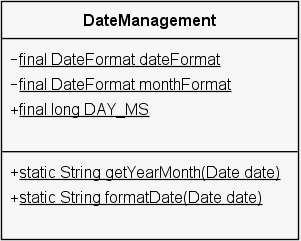
To easily view each class’ attributes and methods, please see Figures 2 through 18.

****

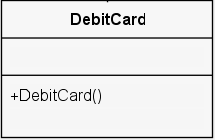
**Figure 2. CardNew class.**

****

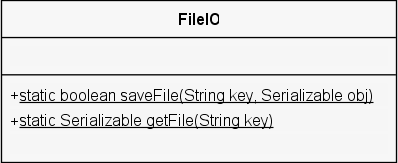
**Figure 3. CreditCard class, which is a subclass of the CardNew class.**

****

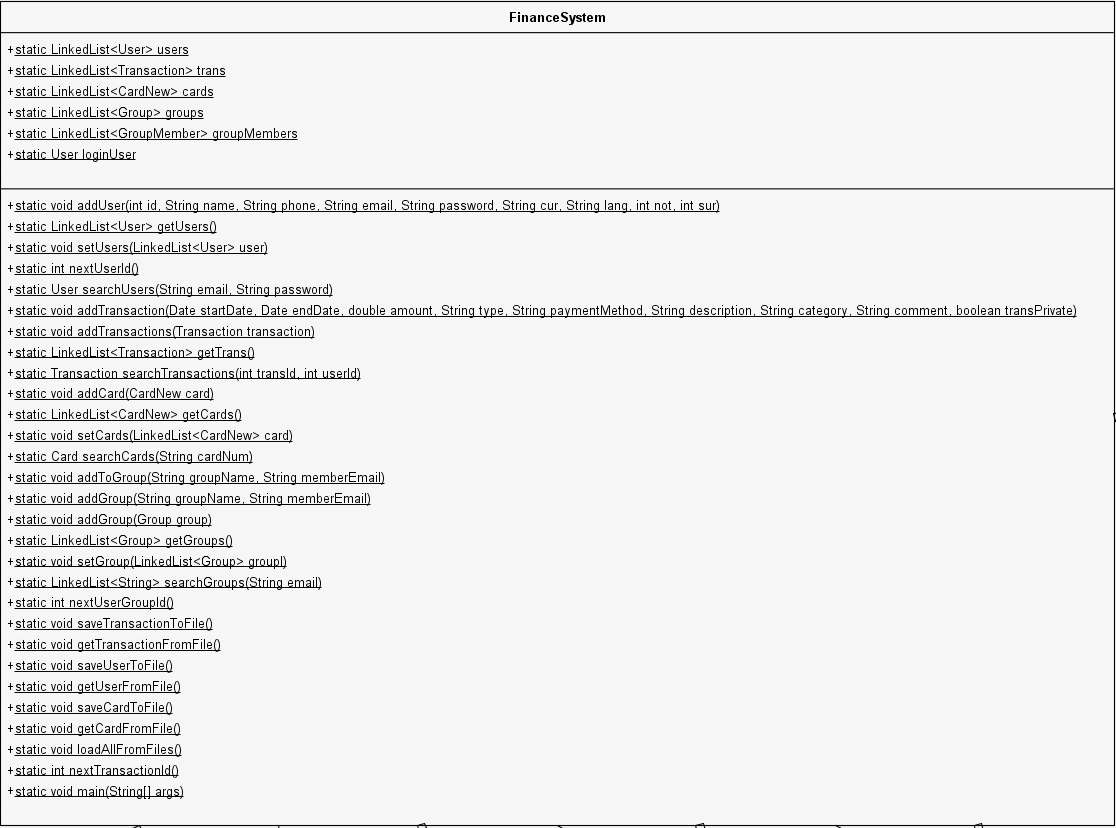
**Figure 4. DateManagement class, which is used throughout the Finance System.**

****

**Figure 5. DebitCard class, which is another subclass of the CardNew class.**

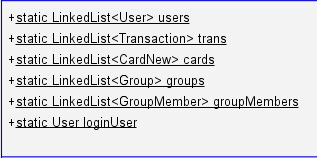
****

**Figure 6. FileIO class.**

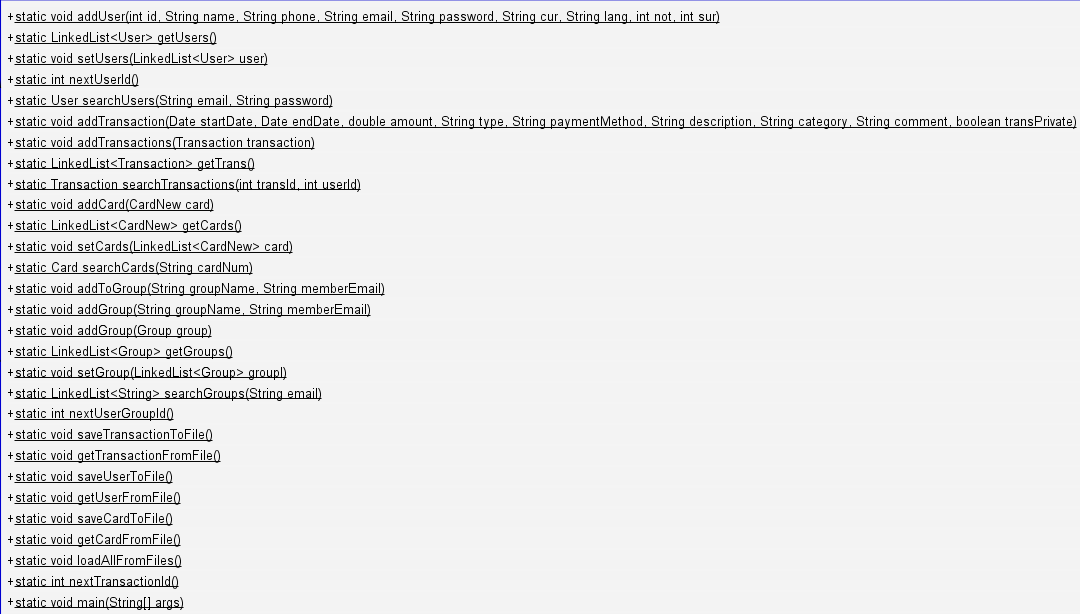
****

**Figure 7. FinanceSystem class (overall).**

For a better view of the FinanceSystem class’ attributes and methods, please see Figures 8 and 9, respectively.

****

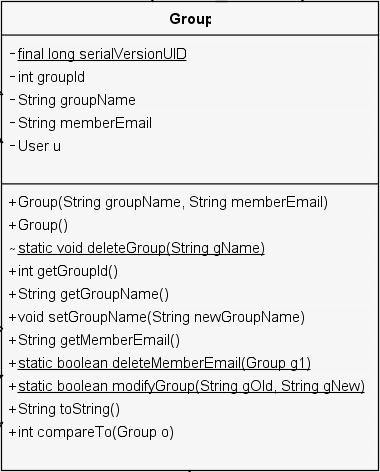
**Figure 8. FinanceSystem class attributes.**

****

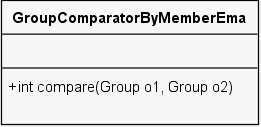
**Figure 9. FinanceSystem class methods.**

In Figure 9, the cutoff method is as follows:

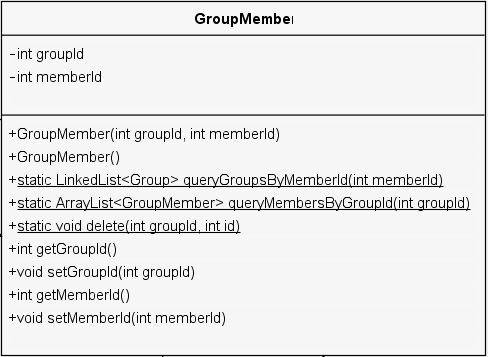
+static void addTransaction(Date startDate, Date endDate, double amount, String type, String paymentMethod, String description, String category, String comment, boolean transPrivate)

****

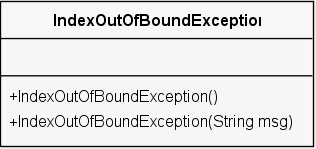
**Figure 10. Group class.**

****

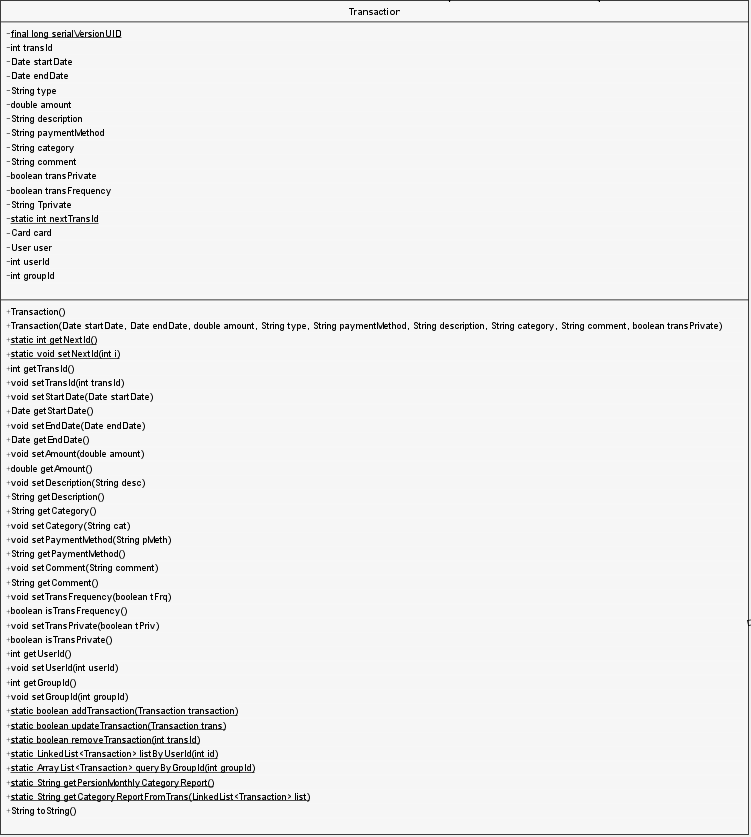
**Figure 11. GroupComparatorByMemberEmail class.**

****

**Figure 12. GroupMember class.**

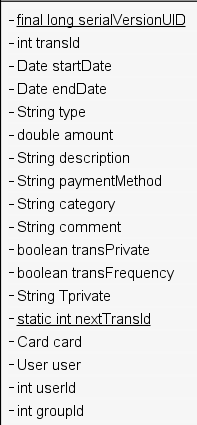
****

**Figure 13. IndexOutOfBoundException class.**

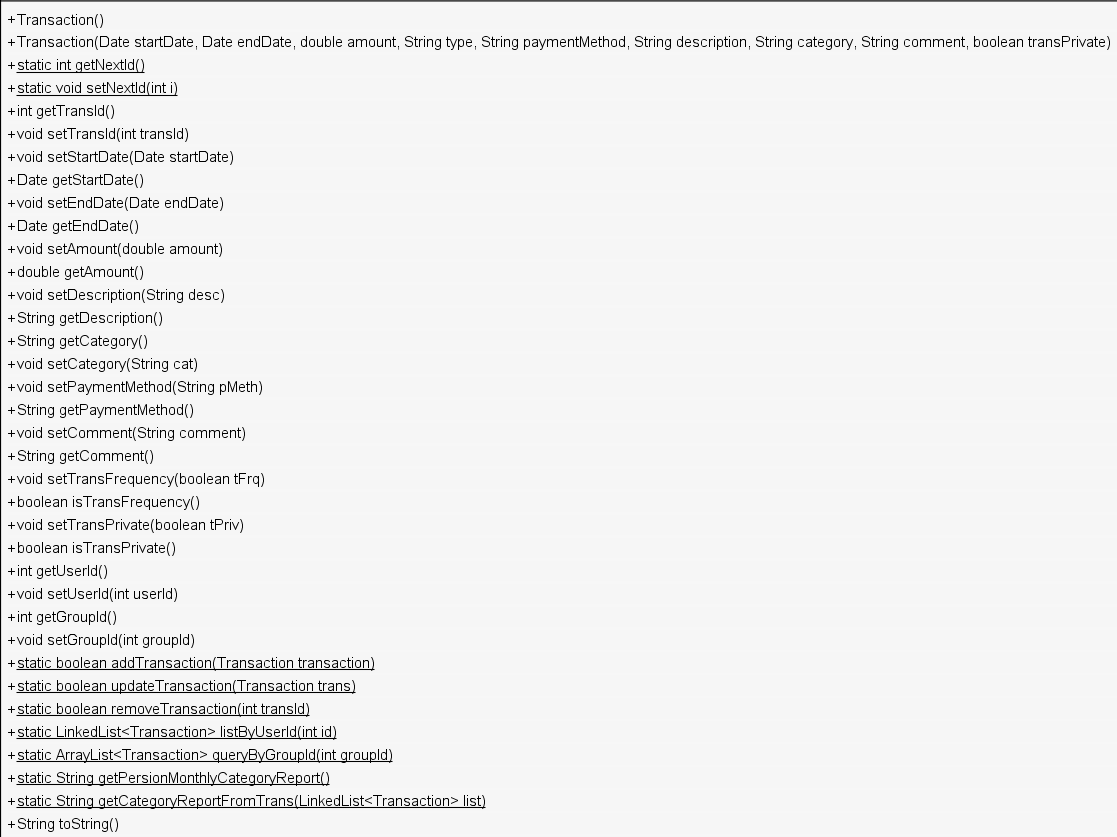
****

**Figure 14. Transaction class.**

For easier visibility of the Transaction class, please see Figures 15 and 16 for the Transaction class’ attributes and methods, respectively.

****

**Figure 15. Transaction class attributes.**

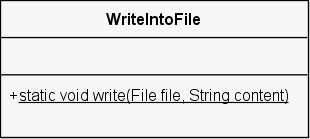
****

**Figure 16. Transaction class methods.**

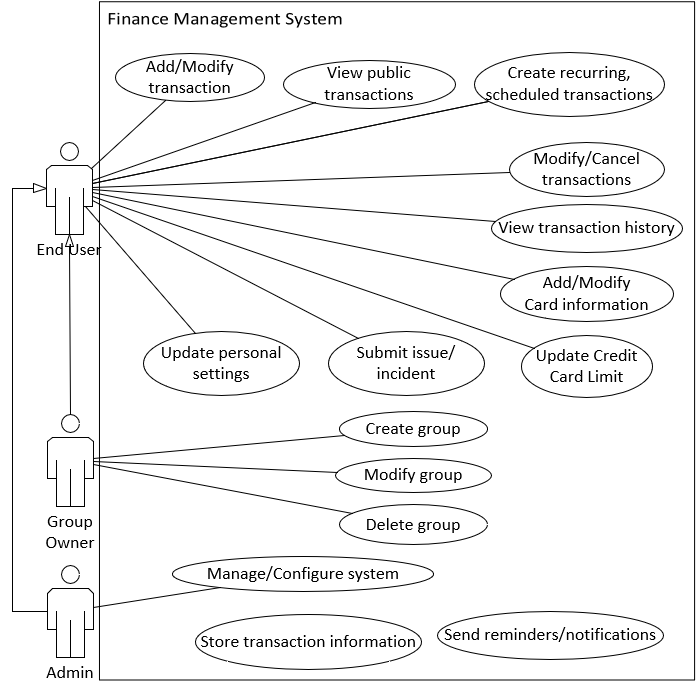
Figure 16’s cutoff method reads as follows:

+Transaction(Date startDate, Date endDate, double amount, String type, String paymentMethod, String description, String category, String comment, boolean transPrivate)

**Figure 17. User class.**

****

**Figure 18. WriteIntoFile class.**

**Use Case Diagram:**

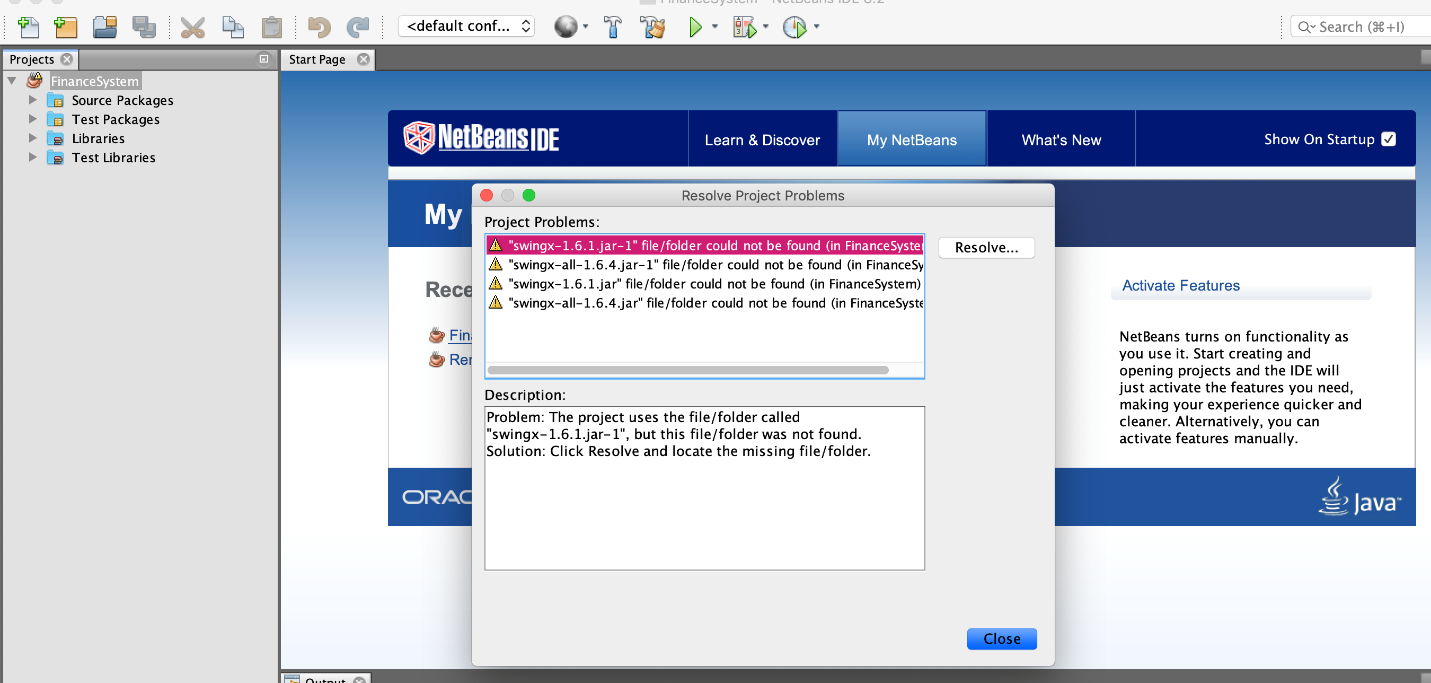
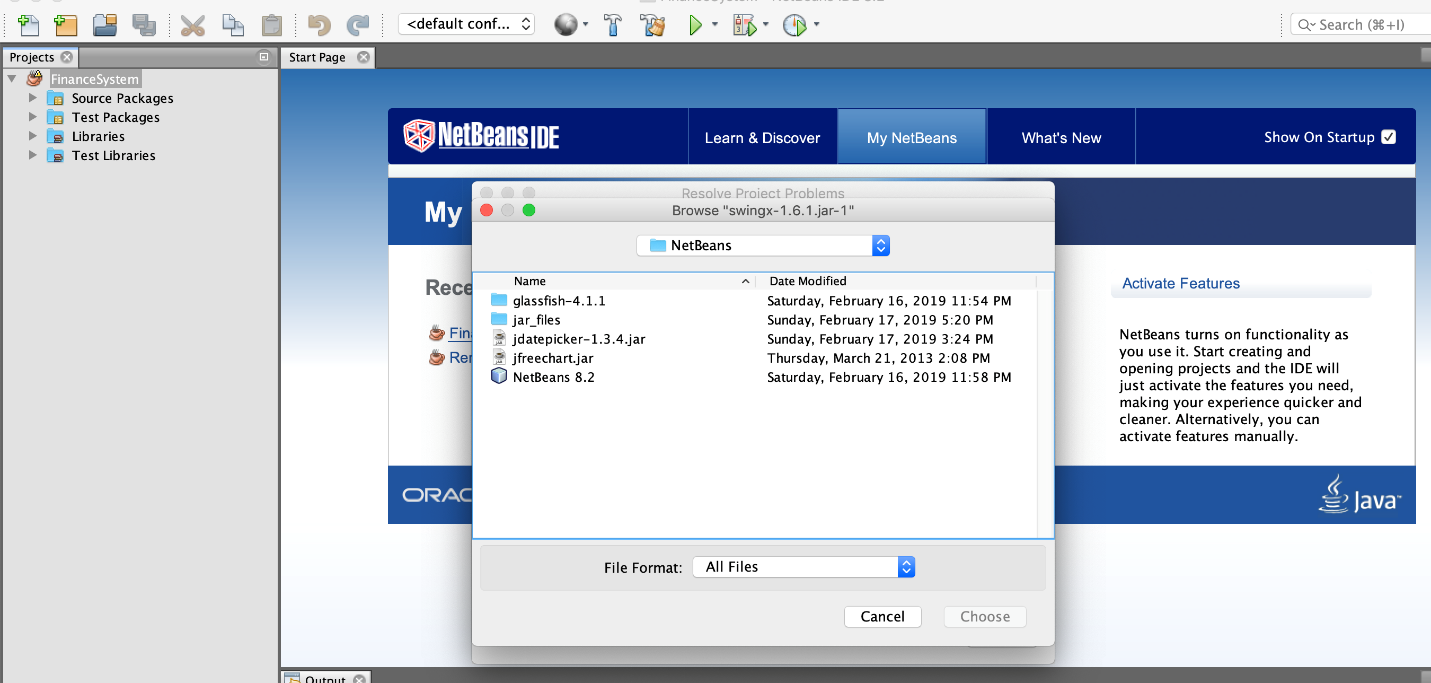
**Figure 19. Use Case Diagram.**

**Demo Transcript**

***List of Usernames and Passwords***

|  |  |  |
| --- | --- | --- |
| **Username/Email** | **Password** | **Notes** |
| newuser@ufl.edu | new | Used for demo |
| emily@ufl.edu | emily | Existing information populated |

To begin, please open the FinanceSystem project in NetBeans. Our group opted to use some additional palette features of NetBeans, including a pie chart and date selectors. Although the associated jar files are stored within the project folder, NetBeans may bark after migrating the FinanceSystem to a new location. If you experience an issue, simply right-click on the project within NetBeans, and select the Resolve Project Issues option. Then follow the steps below:

1. Click “Resolve Problems” button.
2. Select a project problem and click the “Resolve…” button.
3. Navigate to the Finance System folder location on your machine and click the “Choose” button.
4. Repeat steps 1-3 for each problem.
5. Once all of the problems are resolved, close the window.

***User Login***

When entering the Finance System, the user is taken to a LoginGUI page. For our demo, we would like to start with the initial user creation and login.

1. User clicks “Create New User” button.
2. User clicks “Save” button (with empty fields).
   1. System displays error message.
3. User fills in Name, Phone, Email, Password, and Confirm Password fields with different values for Password and Confirm Password.
   1. System displays error message.
4. Fill in all fields with matching/confirmed password.
   1. System displays success message.
5. User clicks “OK”.
   1. System displays LoginGUI screen.

At this point, the new user has been added to a User text file, essentially storing them in the system. Whenever the Finance System project is run again, they can now login with those credentials. In the real world, you would not have this, because then a user would have a full list of every user’s email and passwords. However, this is useful for the purposes of this project and demo.

1. Leaving the email and password empty, user clicks “Login”.
   1. System displays error message.
2. User fills in an existing email with the incorrect password.
   1. System displays error message.
3. User fills in their email and associated password.
4. User clicks “Login” button.
   1. System displays HomeGUI screen.

***Adding and Modifying Cards***

Upon entering the Finance System, the user will need to add their card information. Starting from the HomeGUI, here are the next steps.

1. User clicks “Cards” button in top of HomeGUI screen.
   1. System displays AddCardGUI screen.
2. User fills in card name, number, and expiration date.

To avoid confusion for the end user, we built in functionality on the type radio buttons. If the user selects that the card is a debit card, then the limitation information is not needed and gets removed.

1. User selects “Debit” radio button.
   1. System hides “Limitation” label and text field.

However, if the user selects that the card is a credit type, then the limitation information reappears.

1. User selects “Credit” radio button.
   1. System displays “Limitation” label and text field.
2. User fills in Limitation text field value.
3. User clicks “Save” button.
   1. System displays success message.

Now that a card has been added, it will update the Existing Cards jList on the AddCardGUI.

1. User clicks “OK” button.
   1. System refreshes AddCardGUI, which populates the Existing Cards jList.

Of course, a user can have multiple cards, so we’ll show what happens each time a card is created by making an additional two cards.

1. User fills in a second cards information including name, number, and expiration date.
2. User selected “Debit” radio button.
   1. System hides “Limitation” label and text field.
3. User clicks “Save” button.
   1. System displays success message.
4. User clicks “OK” button.
   1. System refreshes AddCardGUI and appends the new card to the Existing Cards jList.

Notice that the credit card has a limit and the debit card does not.

1. User repeats steps 17 through 20 to create a third card.
2. User selects the first card from the Existing Cards.
3. User clicks “Modify” button.
   1. System populates AddCardGUI fields in top half with existing card information.
4. User modifies the card expiration date to be in the past.
   1. System displays an error message.

Since a card’s expiration date cannot be in the past and still be valid, the system recognizes the issue and refuses to create the card until fixed.

1. User clicks “OK” button.
2. User updates the expiration date to be in the future.
3. User clicks “Save” button.
   1. System refreshes AddCardGUI with updated card information.

Had the user also updated the card number, note that this would have added it as a new card instead of replacing the existing card. Cards should only be *modified* if the card number matches as the unique identifier of a card. So, suppose that the user wants to *delete* a card from their list.

1. User selects the second card from the Existing Card jList.
2. User clicks “Delete” button.
   1. System refreshes AddCardGUI with requested card removed.

***Creating, Modifying, and Deleting Groups***

After a user adds a card, they may decide to create a group with a friend or family member. Through groups, multiple users can view shared transactions. For example, if a pair of roommates wants to have their rental payments visible to both users, then they may want to create a group. As a new user, they would not have any groups, so a blank screen is expected initially. For the purposes of including read and write functionality into our project, the ViewGroupGUI screen will not display a user’s existing groups unless they request it.

1. User clicks “Groups” button.
   1. System displays ViewGroupGUI screen.
2. User clicks “Create New Group” button.
   1. System displays AddGroupGUI screen.
3. Without modifying the fields, user clicks “Save” button.
   1. System displays two error messages – one for required fields, and the other for an invalid email address.
4. User clicks “OK” button twice.
5. User fills in desired Group Name and leaves Member Email either blank or with the default text.
   1. System displays error message.
6. User clicks “OK” button.
7. User fills in email address with emily@ufl.edu.
   1. System displays success message.
8. User clicks “OK” button.
9. User fills in email address with yifei@ufl.edu.
10. User clicks “Save” button.
    1. System displays success message.
11. User clicks “OK” button.
12. User clicks “Done” button.
    1. System adds members to existing group.
    2. System displays ViewGroupGUI screen with updated jList.

For demonstration purposes, the user can save their groups into an Object file.

1. User clicks “Write into Object File” button.

Now whenever the user logs into the Finance System, then can easily pull their existing groups from the Object File.

1. User clicks “Read into Object File” button to import existing group records.
   1. System refreshes ViewGroupGUI screen.
2. User clicks “Sort by Email” button.
   1. System sorts groups by email.
3. User clicks “Sort by Group Name” button.
   1. System sorts groups by name.
4. User selects any of the groups.
5. User clicks “Modify Group” button.
   1. System displays ModifyGroupGUI screen.
6. User selects a group name to edit from the dropdown.
7. User clicks “Edit Group Name” button.
   1. System displays EditGroupGUI screen with the selected group name populated.
8. User modifies group name.
9. User clicks “Cancel” button.
   1. System displays ModifyGroupGUI screen without any changes.
10. User repeats steps 48 through 50.
11. User clicks “Save” button.
    1. System displays ViewGroupGUI with new group name for each group member.
12. User selects group.
13. User clicks “Modify Group” button.
    1. System displays ModifyGroupGUI screen.
14. User selects group.
    1. System refreshes group member list.
15. User selects group member from list.
16. User clicks “Delete Group Member” button.
17. User selects a different group member from the list.
18. User (again) clicks “Delete Group Member” button.
19. User clicks “Save” button.
    1. System displays ViewGroupGUI with those selected members removed.
20. User clicks “Write into Object file”
    1. System updates Object file on machine.

***Add Transaction***

1. User clicks “Transactions” button.
   1. System displays AddTransactionGUI screen.

The Payment Method and Group drop downs will be populated with the current user’s cards and groups to select from. Whenever a card or group is deleted by the user, the drop-down list will reflect the latest available options.

1. User (without modifying any fields) clicks “Save” button.
   1. System displays error message.
2. User clicks “OK” button.
3. User enters Start Date as 02/25/2019, End Date as 02/24/2019, and rest of fields.
4. User clicks “Save” button.
   1. System displays error message.
5. User sets End Date as “03/01/2019”.
6. User sets the amount as “test”.
7. User clicks “OK” button.
   1. System catches NumberFormatException and displays error message.
8. User clicks “OK” button.
9. User sets Amount as “1000”.
10. User clicks “OK” button.
    1. System creates transaction and displays success message.
11. User clicks “OK” button.

If the user wanted to modify the information before leaving, they could from here. If not, then they could click “Cancel” to clear the information.

1. User clicks “Cancel button.
   1. System clears AddTransactionGUI fields.
2. User repeats steps 66 through 74 with typically good information to create another transaction.

***View, Modify, and Delete Existing Transactions***

1. User clicks “Records” button in top.
   1. System displays RecordGUI screen with existing transactions.
2. User selects second transaction.
3. User clicks “Modify” button.
   1. System displays AddTransactionGUI with the selected information.
4. User changes amount to “5000.23”.
5. User clicks “Save” button.
   1. System displays RecordGUI with updated transaction information.
6. User selects a different transaction.
7. User clicks “Delete” button.
   1. System deletes transaction and refreshes list.

***Run a Report***

1. User clicks the “Reports” button at the top.
   1. System displays ReportGUI screen.
2. User clicks “Run Monthly Report” button.
   1. System display jFileChooser window.
3. User selects desired folder and file name.
4. User clicks “Save” button.
   1. System displays success message.
5. User clicks “OK” button.

At this point, the user can access the report on their machine. This will include monthly summary information based on each transaction’s category.

***View and Update User Settings***

1. User clicks “Settings” button in top.
   1. System displays SettingGUI screen.

The Email field is read only, so the user cannot modify it. However, they can update their name, phone, currency, and language preferences.

1. User updates phone number.
2. User clicks “Save” button.
   1. System displays success message.

***View Group Transactions***

1. User clicks “Groups” button in top.
   1. System displays ViewGroupGUI screen.
2. User clicks “View Group Transactions” button.
   1. System displays GroupTransactionGUI screen with group records.
3. User clicks “OK” button.
   1. System displays ViewGroupGUI screen.

***Home Page Updates***

Now that the user has been added to at least one group and created some transactions, some information will appear in the HomeGUI screen.

1. User clicks “Home” button in top.
   1. System displays HomeGUI screen.

Notice that the Notifications jList tells the user which groups they were added to. Also, the Debit Amount and Credit Amount fields show totals of the amount used. From the Home page, the user can also access a pie chart of their transactions.

1. User clicks “Show Pie Chart” button.
   1. System displays pie chart of transactions with percentages, and leaves HomeGUI visible in the background.
2. User exits pie chart.
   1. System closes pie chart.

***Submitting a Support Ticket***

Although the entirety of this functionality is not built in, we did want to consider the long-run and how getting help should work.

1. User clicks “Help” button.
   1. System displays HelpGUI screen.
2. User clicks “Submit” button without any information.
3. System displays error message.
4. User fills in description.

In the future with a fully functioning system, we would have support tickets sent to a Service Management ticketing system of some sort. For now, we simply use a JOptionPane to display a ticket submission message.

1. User clicks “Submit” button.
2. System displays success message.
3. User clicks “OK” button.
4. System displays HomeGUI screen.

***Logout***

A user can logout from either the HomeGUI or SettingGUI screens, or by simply exiting the window.

1. User clicks “Logout” button.
   1. System displays LoginGUI screen.

***Log Back in as Existing User***

After closing the project and re-running, an existing user should be able to view their previously entered information, with the exception of their Groups.

1. User enters existing email and respective password.
2. User clicks “Login”.
   1. System displays HomeGUI screen.

Notice that at this point, only the debit and credit amount information is populating on the HomeGUI. Notifications for group additions would not exist, since the user needs to re-populate their groups.

1. User clicks “Records” button.
   1. System displays RecordGUI screen.

The existing transactions are still visible in the system, thanks to the Transaction file that was written when originally creating the transactions. The same would apply to the user’s cards.

1. User clicks “Groups” button.
   1. System displays ViewGroupGUI screen.
2. User clicks “Read from Object File” button.
   1. System populates the group jList with group names and members.
3. User clicks “View Group Transactions” button.
   1. System displays all group transactions.

At this point, all of the functionality is available for the user. Thank you for watching our demo!