Splitwise++

Prateek Pisat
Computer Science
Northeastern University
Boston, MA, United States
pisat.p@husky.neu.edu

***Abstract* — Splitwise is a free tool for friends and roommates to track bills and other shared expenses. Although the app is incredibly useful, it has few bugs that make the app unreliable. And hence, I intend to implement an improved version of the app. Essentially the app will be work exactly as Splitwise with common and annoying bugs fixed and some added features.**

I. Motivation

The primary motivation towards building this app was to fix certain annoying bugs that affect the flexibility and the reliability of Splitwise.

Problems with Splitwise:

- The balance bug: The app has this weird bug, where it fails to calculate the correct balance in certain situations. Consider x and y to be two users. Let x owe y some money (or vice-versa). Let this transaction be a part of some group g. Let y owe x some money (or vice-versa) and this transaction is not a part of any group. In this case, the app incorrectly calculates the balance x owes y. However, this case is not common, and may or may not always occur. This leads me to believe that, the bug may occur due to a random crash OR there might be some over-complication in how the transactions are represented.

- Participation Problems: For any set of items bought, it is not the case that everyone necessarily wants each item. Certain people may want certain items, others may not. However, the app assumes that all the people (in a group/ household) want each of the items bought. So, users need to manually (not a part of the app) specify what items they want to be included in. This leads to one bill being split into various sets of items, depending on what items, of one bill, people want to be included in. This leads to a huge added overhead, especially, if the bill contains a large number of items.

- No Concrete Record of Bills: Another problem with the above-mentioned complication is, since one bill is divided into separate, independent and disconnected pieces, there is no reference to the entire bill, with all the items, only a list of micro-transactions, that if joined together, will form the complete bill. However, the App doesn't provide any such features.

I intend to fix these common issues/ bugs. Along with fixing these problems I have added a few new features like a budget-tracking system.

Each user will be allowed to set a monthly budget. The app will keep track of how close the user is to achieving that budget. If the user's expenses surpass the set budget the app will identify certain expensive items that the user can avoid the next month.

II. Significance

To fix the previously mentioned bugs, I resort to the following techniques:

1. Having a Simpler Database Design:

Simplifying the database helps to eliminate the Balance Bug. Also, certain extraneous features such as groups can be eliminated, since they do not provide much functionality. Furthermore, having more indexes improves the speed to fetch operations and hence improves the overall performance.

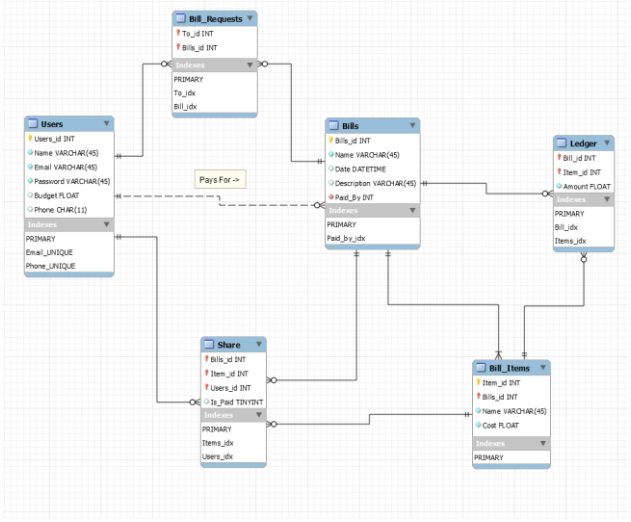
2. An Improved Way of Sharing a Bill:

The app allows each user to select what item they want to pay for. This functionality is included right within the app, otherwise, previously, users needed to specify this manually, outside the scope of the app. This fixes the "Participation Problem". This also fixes the third problem. Now, since the user can specify their participation right within the app, the bill need not be split into a set of individual items. Hence, keeping the bills intact and allowing a record of bills to exist.

III. Problem Statement/ Project Goals

The app primarily consists of Users and Bills. Each user needs to register before they can use the app's features. Each user is associated with a Name, Email, Password, Phone Number and a Budget. Registered users can add Bills. Each bill is associated with a Name, an optional Description, a date, and the user who paid the bill. Each bill consists of Items. Each item belongs to a bill and is associated with the name of the item and cost. After adding a bill and items to that Bill the user will have to identify, which other users will they be sharing the bill with. So, once a complete bill is created the creator must send requests to other users, of the creator's choice, to participate in the bill, to identify what items do they want and should be charged appropriately. Each user may set a budget. If the spending for the user in that month goes over that month's budget, then the user should be notified regarding the same.

IV. E.R DIAGRAM



V. DATA ACCUSATION

I have provided 2 SQL Scripts, one for creating the Database Schema (called Create) and another to insert some test data into the database (called Insert).

I have not used any dataset. All the data, using which the app was tested was generated using real transactions, such as grocery bills. Also, in the Insert script, I have commented majority of the code (Everything except inserting users). It is recommended to run the script as it is. Uncommenting the rest of the code may make it difficult for the tester to test the accuracy/correctness of the app.

It is recommended that the user/tester uses the app UI to add data, instead of running SQL scripts, to test the correctness/accuracy.

VI. THE COMPONENTS/MODULES

The app uses the following modules/ components:

1. Login/ Registration:

To use the app, all users need to register. To register, click on the "Register" button on the Login Screen. After providing all the information on the Registration screen, click "Register".

Once a user has registered, they can proceed to login to the app by entering their email-id and password and clicking "Login".

Users can later update their details using the "Edit Profile" button provided on the home screen or delete their profile if they wish to, using the "Delete Profile" button provided on the home screen. The screen-shots are provided below.

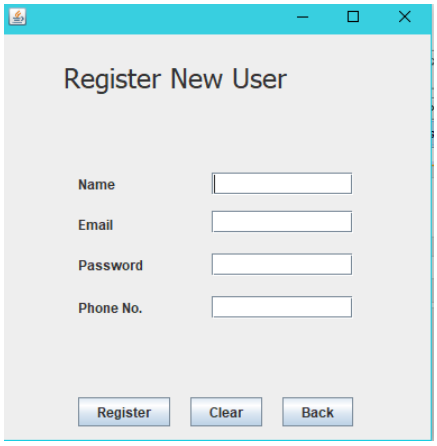


Fig.1. Registration

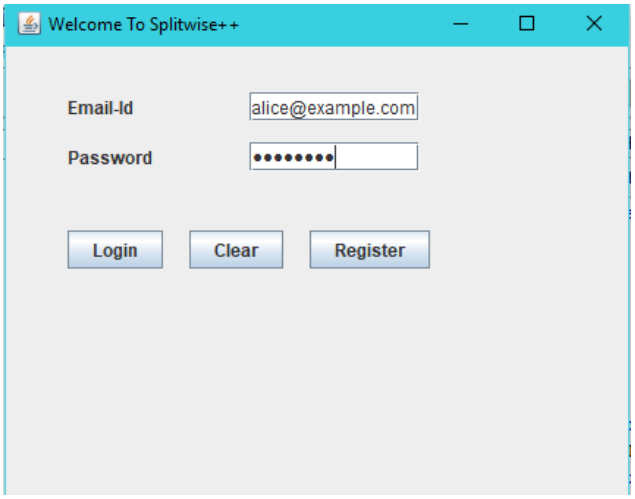


Fig.2. Login

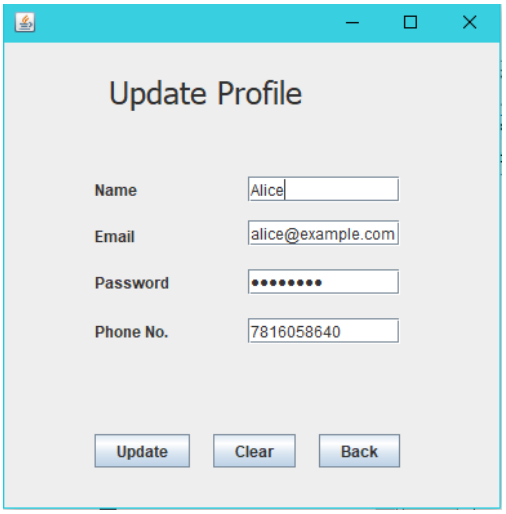


Fig.3. Update Profile.

2. Adding a New Bill:

Creating a bill is divided into 3 steps, adding a bill, adding items to that bill, and sending the bill to the users the creator wishes to share the bill with. To add a new Bill, click on the "New Bill" button on the home screen. The following form should appear. After filling in the appropriate information click on "Create Bill".

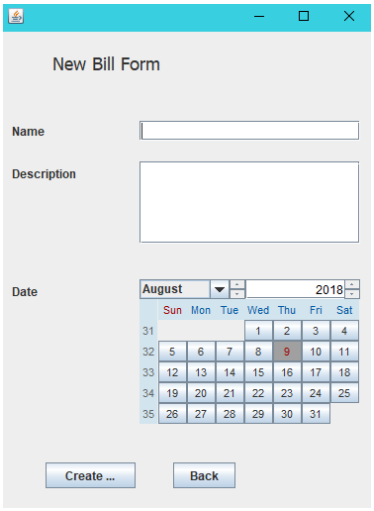


Fig.4. New Bill

Now the app will load a screen to add items to that bill. To add an item to a bill, mention the name and cost of the said item, then click “Next”. Continue this for each item. After adding the last item, click on “Done”. NOTE: The Item Cost field expects a numeric value, and would generate an error, if a non-numeric value is provided.

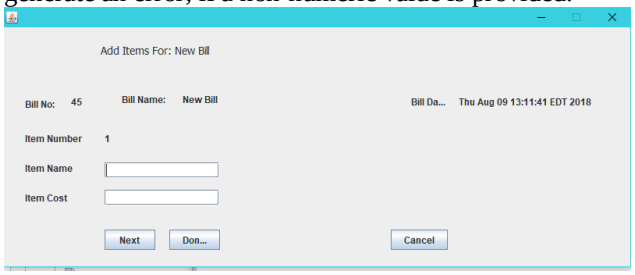


Fig.5. Add Items

This will load a page where the creator can send the bill to specific users, to share the bill with them. To send the bill to a specific person, select the person from the drop-down menu, and then click “Send”. Continue this for each person, the creator wishes to send that bill to. After sending a request to the last person, click on “Done”.

Clicking on “Cancel” at any of the three stages will delete the bill, hence deleting all the items associated with the bill, and any sent bill request.

Fig.6. Send Bill

3. Fill a Bill Request:

If the user has certain bills sent to them, then the “Pending Bill Requests” label on the home page will indicate the number of bill requests the person has yet to fill. To fill a bill-request the user need to click on “View Requests” on the home screen. This will show all the pending bill requests. If the user chooses to participate in a bill, they should click on “Participate”, else “delete” to delete the bill-request.

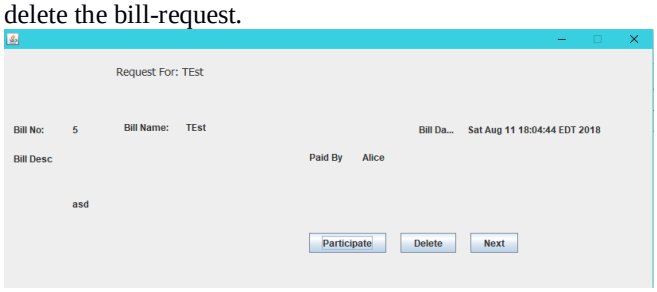


Fig.7. Choose to Participate.

If the user chooses to participate in the bill, then the following screen will load, which will display all the items in that bill, allowing the user to choose, which items they wish to pay for.

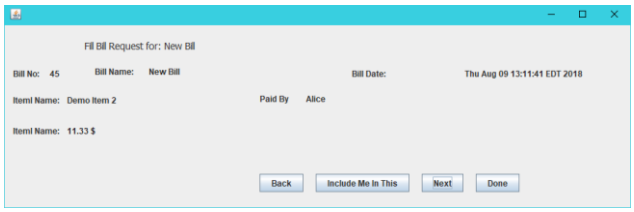


Fig.8. Item Selection

After the user is done selecting what items they wish to pay for, they must click “Done”, this will load the home screen, with the updated balance, i.e. how much they owe someone or how much someone owes them. Consider the following screen-shot as an example:

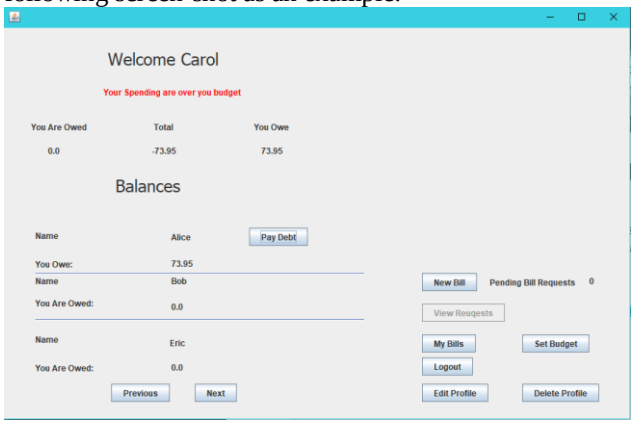


Fig.9. User Homepage.

4. Paying a Debt:

The user(debtor) may choose to pay the debt they owe to some other user(creditor); however, they can’t pay unless all the bill requests for the bills that were shared amongst the creditor and the debtor have been filled by all the users to who the request for those bills was sent. This ensures that the cost of the items is distributed fairly, and the person paying first doesn’t have to pay more than the person paying later.

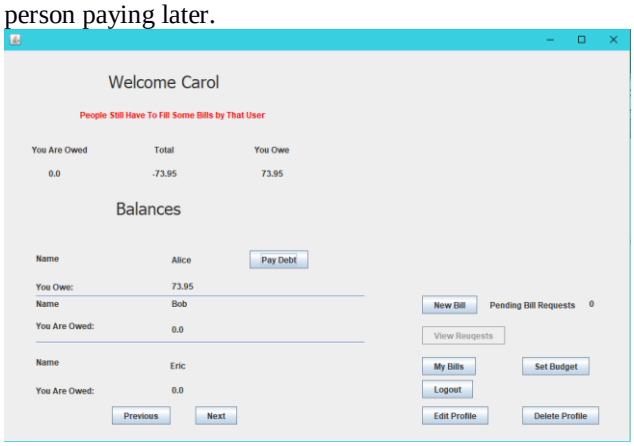


Fig.10. Requests are remaining.

Once all the bill-requests, for a bill, have been filled, a debtor can then click the “pay-debt” button and the debt will be cleared. Let A owe B some amount x, and B owe A some amount y. If x > y then A will see the “Pay Debt” button enabled on their home screen, however, B won’t see a “Pay Debt” button, but they will see the amount A owe them and vice-versa. This is shown in the following screen-shots.

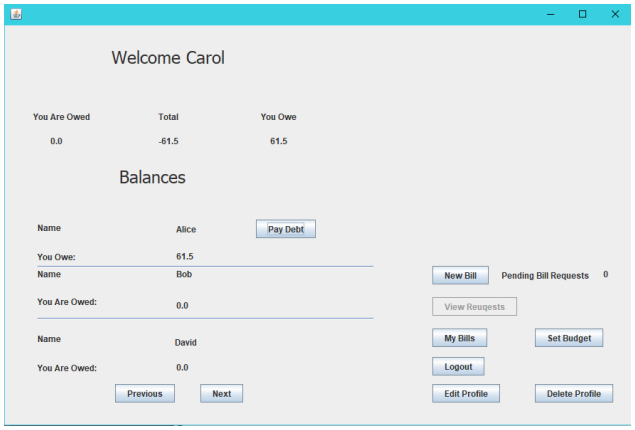


Fig.11. Carol owes Alice

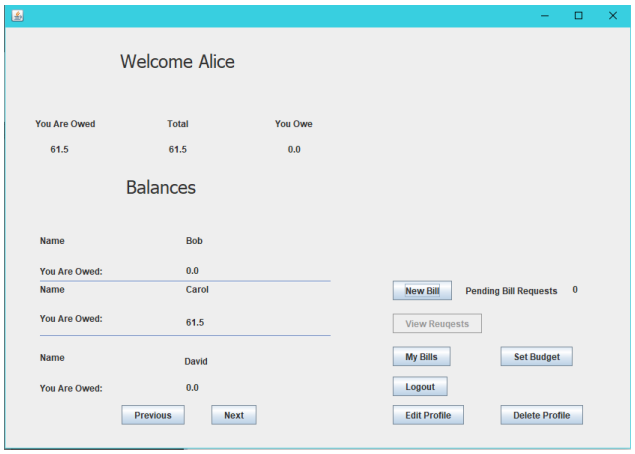


Fig.12. Alice is Owed by Carol

On clicking the "Clear Debt" button, all the debts amongst A and B will be settled. This is done to simplify payments to one another. Instead of having separate records, of A owing some money to B and B owing some money to A, the two records are condensed into one. And the one who owes more, pays the other, settling everything.

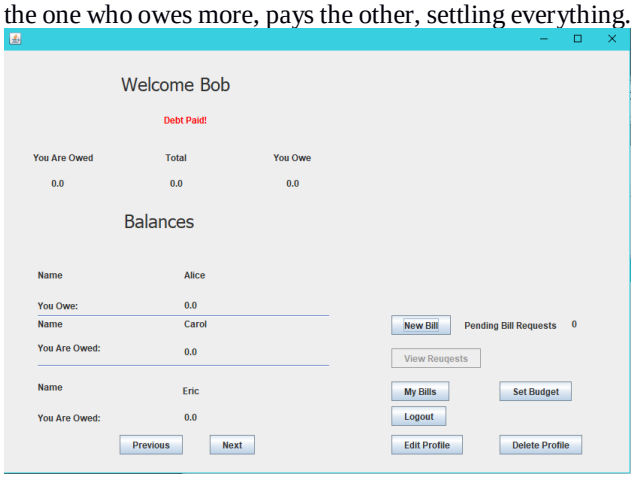


Fig. 13. After Paying a Debt

5. Managing Bills

Bills are primarily divided into two categories; the bills that a user created, and the ones that the user participated in. All these bills can be viewed or edited by clicking on the "My Bills" button on the user homepage, which loads the following screen. NOTE: if the user has not created any bills or is not included in any created bill, the "My Bills" button will be disabled.

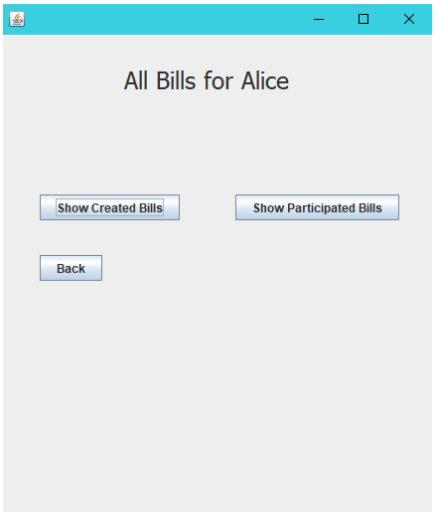


Fig.14. My Bills

Now the user has the option of viewing/editing the bill that they created, or the bill that they participated in. Either of which can be done by clicking on the appropriate button.

If the user selects "Show Created Bills" then they can see all the bills they created. And hence, they will be allowed to update the name, date, description of each bill, as well as the items and their cost, and update the users with who the bill was shared with.

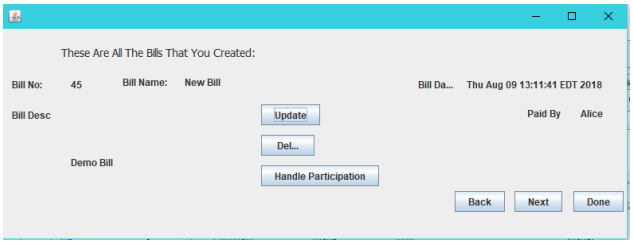


Fig.15. View/ Update Created Bills.

Clicking on the "Update" button will load the following screen where the user can update the name, description or the date of the bill.

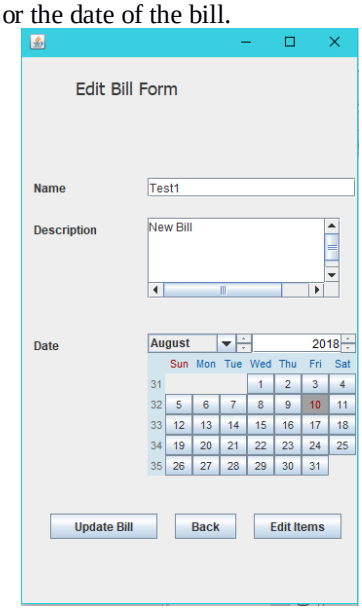


Fig.16. Update Bill.

Clicking on the "Edit Items" loads the following screen where the user can edit the items associated with a bill, for example, their name and cost.

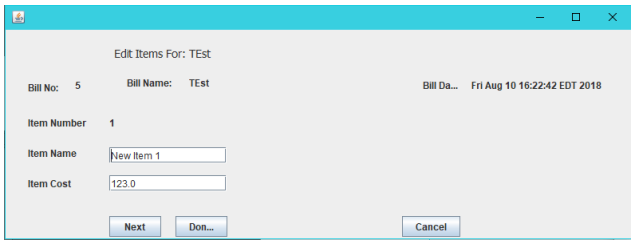


Fig.17. Update Bill Items.

On clicking "Update Participation", the flowing screen will load, and the creator can then update the bill requests that were sent, allowing them to either sent more requests, or delete previously sent requests.

Fig.18. Update Bill Participation (The users the bill was shared with)

If the user selects "Show Participated Bill", then the user can see all the bills that they chose to participate in, and hence can update their participation in each of those bills. To update the participation in a bill, click on the "Update Participation" button.

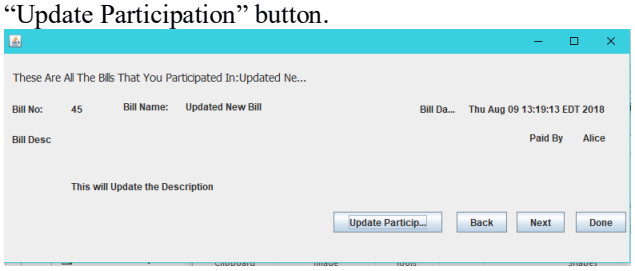


Fig.19. View/ Update Participation

The user can now update what item they want to be included in. Once they are done with the updates they can click on the "Done" button.

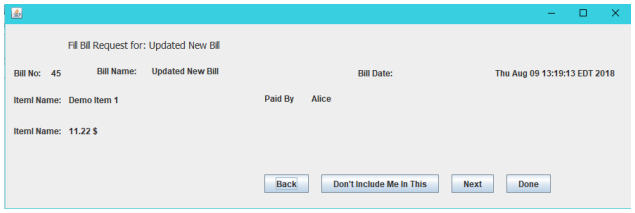


Fig.20. Update Item Participation.

6. Setting a Budget:

Each user can set a budget. If the spending of the user in that month, exceeds the budget set by the user, then the user's home page will display a message indicating that user's spending is more than what they intended to.

To set or update the budget, the user needs to click on the "Set Budget" button on the home page. The user will see the following screen.

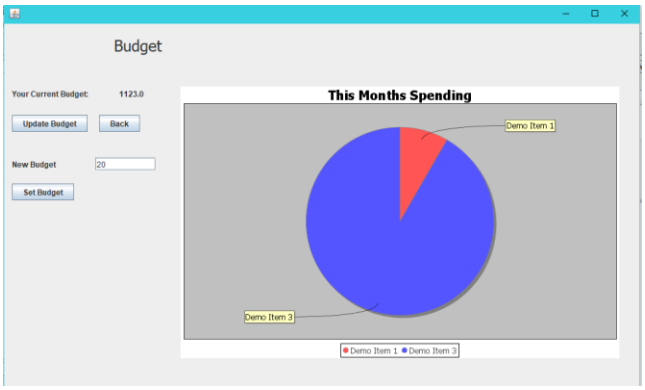


Fig.21. User Budget.

Here the user can set or update their previously set budget. Additionally, a Pie-Chart of the user's spending for that month is shown to the right. This provides a quick way for the user to identify what item are they spending more on.

VII. USE CASES

As mentioned in the description, this app is for friends/ roommates to keep track of expenses, and ensure that every bill is split equally, and everyone gets paid.

Additionally, this app can also be used for maintaining a budget, to keep track of the money spent that month, and on what items was the money spent on.

The app can also be used as a general record of all the bills that were generated by a user.

VIII. TECHNICAL SPECIFICATIONS

The app was developed using the NetBeans IDE[7] and is best viewed in the same. The app can be imported into IntelliJ Idea or Eclipse, however, based on my tests, IntelliJ has some issues with the GUI Forms built using NetBeans and hence throws some errors.

Eclipse, on the other hand, can import the project without any errors. In all the cases, you'll need to import some external libraries. All the external libraries are provided in 'src/lib' folder. The process to import the project to both the IDEs is provided below:

1. Eclipse:
   a. Start Eclipse.
   b. Create a new Java Project called 'Splitwise'.
   c. Copy all the files the 'src' folder of my project to the 'src' folder for the project that you just created.
   d. Also, copy the 'database.secret' file into the 'Splitwise' root directory.
   e. Run the project. The IDE will ask you for the main class, select Splitwise.java
   f. More information here[5].
2. IntelliJ Idea:
   a. Start IntelliJ.
   b. Go to File -> Import Project.
   c. Navigate and select my provided project folder.
   d. Open the Ant Build Tool window.
   e. Click on the '+' to add a build file.
   f. Now navigate to the 'build.xml' file in the project folder and click ok.
   g. Now click on the Build menu.
   h. There should be an option named 'Splitwise', click on it.
   i. Now select 'Run', from the list of options.
   j. More information here[6].

IX. Conclusion/Self-Assessment

The app successfully fixes all the bugs and errors previously mentioned. Users can now select the individual items that they would want to pay for and are hence charged appropriately. Since, the bill is no longer divided into records of individual items, a complete record of all the bill created is stored by the app and allows the user to view the bills they have participated in and additionally edit the bills that they have created.

The simple database design eliminates the 'balance bug' and proper indexing helps in improved performance. The Budget Tracking feature helps users to identify when their monthly spending is above their predefined budget. This also shows the user a Pie Chart of how the user has been spending their money, to help them identify items that they could avoid.

Given more time, I would have added more components such as 'Friends', so, instead of traversing a list of all the users, while select users to share bills with, the user can traverse a possibly smaller list of friends. Additionally, I would also like to add a messaging system amongst the users. These messages would allow the bill creators to remind users to either pay their remaining debt or to fill a previously sent bill request.

The entire app along with full-sized screenshots and diagrams can be found here[8].

X. References

These are the research papers and websites that I used as a reference.

[1] https://dev.mysql.com/doc/
[2] https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/
[3] http://www.jfree.org/jfreechart/api/javadoc/index.html
[4] https://secure.splitwise.com
[5] https://stackoverflow.com/questions/21535023/how-to-get-your-netbeans-project-into-eclipse
[6] https://www.youtube.com/watch?v=0hDxv7jjlPc
[7] https://netbeans.org/downloads/start.html?platform=windows&lang=en&option=all
[8] https://github.com/PrateekPisat/CS5200/tree/master/Project