Whitebox Systems

April 3, 2016

Final report

Members:

Jonathan Chan – Phase 1, editing previous phases, database and operations

Calvin Neethling – Phase 3, editing phase 2

Yan Liao – Phase 2, GUI

Hooseon Yim – Phase 3, editing phase 3

1.0 Introduction

Whitebox Systems is a local business that assembles custom computers according to the needs of their customers. The system they desire includes a database stored on Dropbox and a GUI. The database stores a customer's information and build specifications, while the GUI is used to access and alter the database.

2.0 Requirements

2.1 Functional Requirements

- Actors : privileges ((more/less?) privileges)

    - Administrators (1..*): Add, update, view, filter, and delete entries in the database.

    - Employees (1..*): View and filter entries in the database. Cannot view personal data.

    - Marketing (1..*): View parts and prices only.

    - Account database (1): Database to store account information of WhiteBox employees for log in.

    - Customer database (1): Database to store all information relating to the customer, like personal information and computer parts (name, price, warranty, and etc.)

- All fields to be initially in the database will be provided in a separate document by Whitebox systems on BaseCamp.

- Database operations depend on privilege level and are initiated through the GUI.

- User will access a GUI that will either be stored locally on chosen computers or on Dropbox.

- Account and Customer database will be stored on Dropbox

- Scalable GUI, if more operations are needed they are easy to implement.

- Scalable databases, if more columns and reference tables are needed it should be easy to implement.

- Log-in GUI: Allows an actor to log into the main GUI to perform operations on the database.

- Main GUI: Depending on which actor has logged in, different options will be presented to the actor.

    Administrator will have unlimited access to the Account and Customer databases. For Account Database, Administrators have the options to view/modify usernames,

passwords, and add/delete accounts. For Customer database, Administrators can perform operations like E-mail, View, New Customer, Update, Delete Customer, Add Column, and Delete Column.

Employees will have limited access to operations and information pertaining to the Customer database. The only operation employees have is to view customers (restricted from personal information) and computer parts.

Marketing will only have the operations to view computer parts and its associated prices.

2.2 Non-Functional Requirements

- GUI

    - Usability: GUI should be straight forward to operate and navigate

    - Safety: SQL injection for credentials

- Database

    - Scalability: Able to create additional fields and tables

    - Database stored on DropBox (covers capacity, reliability, ...)

    - Database backed up on multiple Droxbox accounts and on a hard drive

    - Secure code that eliminates SQL injection

    - Database Design: fast performance, no data loss

3.0 System Models

3.1 Scenarios

3.1.1 Scenario 1: Add Customer by uploading a text/excel/other file.

Actor

> Admin BasedGod

Summary

> Admin wants to add a new customer to the database by method of uploading a file with the correct template.

Description

1. Through the elicitation process with the customer, the file containing all required fields is created and filled out.

2. BasedGod logs in through “log in” GUI bringing him to the main menu. He selects the “add customer” option. Since he already has a populated customer form, BasedGod clicks on the “upload file” option and uploads his file.

3. The file is interpreted and previewed to BasedGod. He now has the option to “submit” or “cancel.” By clicking “submit” the corresponding columns are filled in the database.

4. BasedGod receives an acknowledgment of success.

3.1.2 Scenario 2: Employee wants to View/filter a customer’s build components

Actor

> Employee Kanye

Summary

> Employee wishes to view the components for a specific customer

Description

1. Kanye is tasked with building a PC for a specific customer. He logs in and chooses the “View/Filter” option on the main menu.

2. He is presented with a single drop down list that contains the fields Kanye has privilege to see. He first chooses “Customer ID” from the list and then fills the user input field with his customer’s ID.

3. From the new drop down list he selects “Build” and leaves the user input field empty.

4. When Kanye clicks the “View” option with no additional filers, the filtered information is returned to the GUI.

3.1.3 Scenario 3: Admin wants to view builds with specific parts

Actor

> Admin Vanic

Summary

> Admin wants to view previous PC builds

Description

1. Vanic logs in and chooses the “View/Filter” option.

2. In the first drop down list he chooses the “Builds” item from the list and leaves the user input empty. He clicks “More filters” for an additional drop down list.
3. Through the next drop down list Vanic chooses “processors” from the list and in the input field types “intel”.
4. He selects “Apply Filters” and the database returns all rows with “intel” in its “processor” column to the GUI.

3.1.4 Scenario 4: Admin wishes to delete a customer

Actor

> Admin AplFisher

Summary

> Data is already entered in the database and the customer wishes to cancel their order.

Description

1. AplFisher logs in and chooses the “delete customer” option
2. He types the customer ID into the input field and clicks the “preview” option. This returns the whole row associated with this customer.
3. He chooses to “Submit” which commits the operation to the database.
4. An acknowledgment is returned.

3.1.5 Scenario 5: Admin wants to add a new column to the database

Actor

> Admin Pacquiao

Summary

> Many customers ask for a PC accessory that is not included in the database, so the Admin would like to add a column for it to the database.

Description

1. Pacquiao logs in and then chooses the “database changes” option.
2. Then he selects the “Add a new column” option and is presented with drop down lists and input fields that are required to add a column to the desired place (such as “existing table”, “new table”, “column name”, etc).

3. Pacquiao selects and fills in the information and selects “preview,” this will return the table with the new addition.
4. After reviewing the change, he selects “Submit” to commit the change.
5. An acknowledgement is returned.

3.1.6 Scenario 6: Admin wishes to update a field for a customer

Actor

> Admin Gyllenhaal

Summary

> A customer's PC build information is already in the database, but they now want a model xyz for their graphics card.

Description

1. Gyllenhaal logs in and chooses the “update” option.

2. He is presented with a field for customer ID, he inputs a valid ID and then is presented with a drop down list that contain all database columns and an input field.

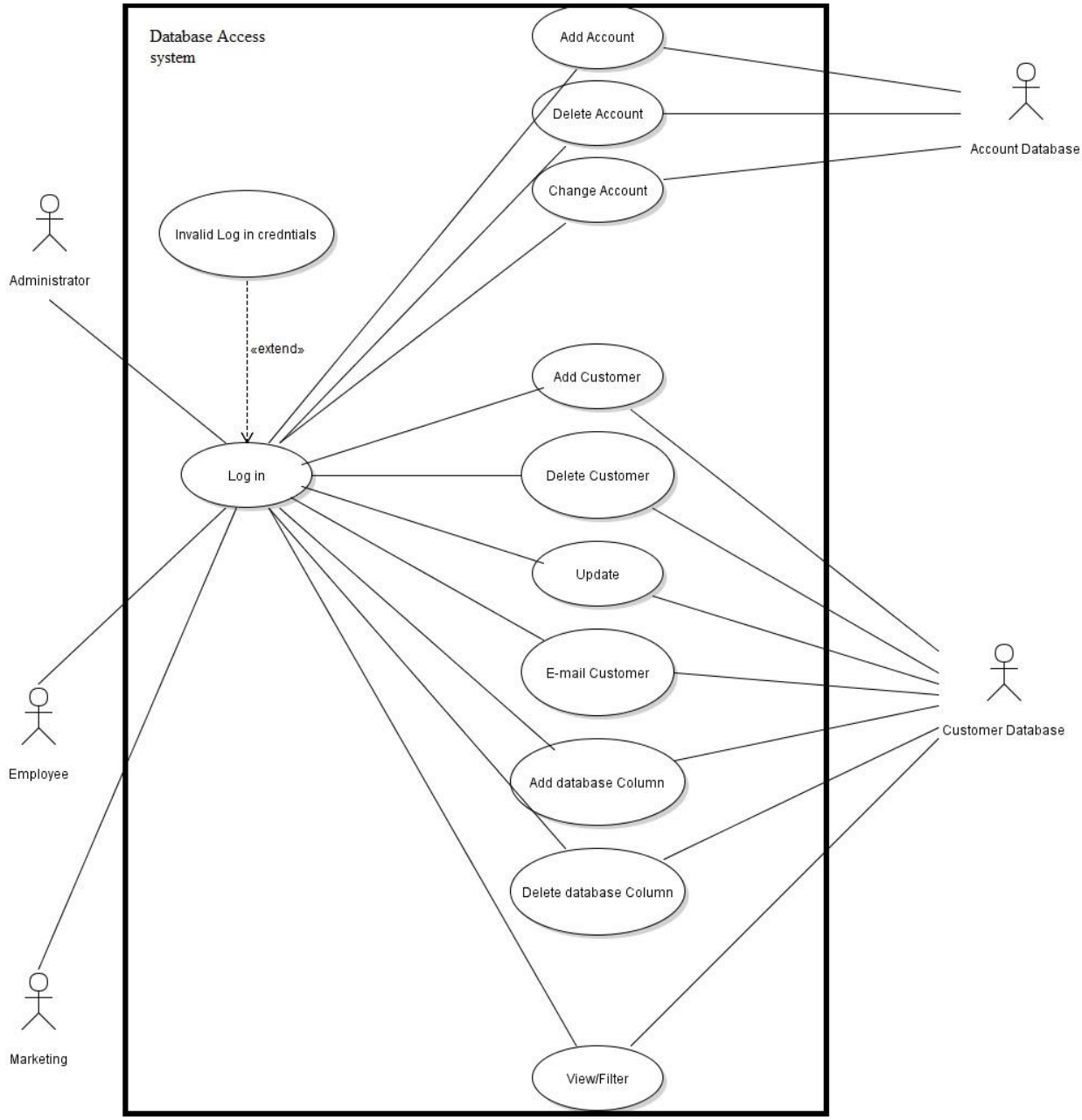
3. Gyllenhaal chooses GPU from the drop down list, enters xyz and selects “preview”. He is returned the edited column, and another drop down list and field if he wishes to update data that is referenced by “GPU”.

4. He is satisfied and clicks “Submit” to commit the change.

5. An acknowledgement it returned.

3.2 Use Case Diagram

FIGURE 1: Use case diagram for Whitebox systems



3.3 List of Actors

> Administrators: Has the highest privilege level. Access to all operations, able to view all information within the customer database and accounts database. Probably restricted from seeing other Admin information, unless initiated through a "forgot password" method.

> Employees: Privileges restricted to non-modify options, view/filter, and restricted to personal information about customers.

> Marketing: Privileges restricted to non-modify options, view/filter, and only able to view information on builds, such as parts and prices.

> Account database: Stores information for WhiteBox employees for the Log in GUI, which then determines which operations the actor has access to.

> Customer database: Stores information relating to the customers, like personal information and information relating to the computer parts.

3.4 Use Cases

3.4.1 Use Case 1: Login

> Brief description: Log into the main GUI

> Primary Actor:

> Administrator, Employee, or Marketing.

> Precondition:

> Administrator, Employee, or Marketing has access to the internet

> Account and Customer database is available

> Flow of events:

1. Administrator, Employee, or Marketing opens the log in GUI
2. Administrator, Employee, or Marketing enters their credentials
3. The system checks that the credentials are valid
4. The system initiates appropriate next GUI

> Post-condition

Administrator, Employee, or Marketing is logged in

> Bounded alternative flow: FOE 3

1. The system displays an error message

> Post-Condition:

No access granted.

> Exceptions:

> Administrator, Employee, or Marketing is notified if the connection to the Account database is unavailable.

> Non-functional requirements:

> Quick log in

> Administrator, Employee, or Marketing should gain their associated privileges.

3.4.2 Use case 2: AddCustomer with template file

> Brief description: Administrator has filled out a template file with customer information and wishes to upload it to the database to enter as a new customer.

> Primary Actor:

> Administrator

> Precondition:

> Administrator has access to the internet.

> Customer Database is available

> Flow of Events:

1. INCLUDE USE CASE Login
2. Administrator fills out the template for a customer
3. Administrator selects “Upload File”
4. The system responds by supplying a field for the file to be uploaded
5. Administrator chooses the file to upload and selects preview
6. The system checks that the file type is valid.
7. The system checks that the customer ID field in the file is valid
8. The system checks that the required fields in the file are filled
9. The system returns the information for the preview
10. Administrator selects “submit”
11. The system commits the change to the database
12. The system returns an acknowledgment
13. The system waits for Administrator's next action

> Post-condition:

Customer Database has been updated

> Bounded alternative flow: FOE 6-8

1. The system displays the appropriate error message

> Post-Condition:

No changes made to the database.

> Exceptions:

> The Administrator is notified if the connection or actions to the Customer database is unavailable.

> Non-functional requirements:

> Quick response

> Quick and accurate database update

3.4.3 Use Case 3: View/Filter

> Brief description: Administrator, Employee, or Marketing wishes to view certain items in the database. Depending on privilege level, certain items can be viewed.

> Primary Actor:

> Administrator, Employee, or Marketing.

> Precondition:

> Administrator, Employee, or Marketing has access to the internet

> Customer Database is available

> Flow of events:

1. INCLUDE USE CASE Login

2. Administrator, Employee, or Marketing selects “View/Filter”

3. The system provides fields to be filled

4. Administrator, Employee, or Marketing selects their desired filters

5. The system provides information based on applied filters

6. The system waits for Administrator, Employee, or Marketing next action

> Postcondition:

> Administrator, Employee, or Marketing views the information, filtered accordingly

> Post-condition

Specified data is viewable on the GUI

> Exceptions:

> Administrator, Employee, or Marketing is notified if the connection to the Customer database is unavailable

> Non-functional requirements:

> Quick and accurate return of specified data

> Privilege level restricts certain information

3.4.4 Use case 4: Update

> Brief description: Admin wishes to change a field entry for a specific customer.

> Primary Actor:

> Administrator

> Precondition:

> Administrator has access to the internet

> Customer Database is available

> Flow of events:

1. INCLUDE USE CASE Login
2. Administrator selects “Update”
3. The system provides fields to be filled
4. Administrator inputs desired changes
5. The system provides a preview of the change(s)
6. Administrator selects “submit”
7. The system commits the change(s) to the database
8. The system returns an acknowledgment

9. The system waits for Administrator's next action

> Post-condition:

> Database is updated with the change(s)

> Exceptions:

> Administrator is notified if the connection to the customer database is unavailable

> Non-functional requirements:

> Change applied quickly and correctly

3.4.5 Use case 5: Delete a customer

> Brief description: Administrator wishes to remove a customer from the database

> Primary Actors

> Administrator

> Precondition

> Administrator has access to the internet

> Customer Database is available

> Flow of events:

1. INCLUDE USE CASE Login
2. Administrator selects “delete customer”
3. Administrator provides the appropriate information
4. The system checks that the information is valid
5. The system previews the corresponding data
6. Administrator selects “confirm”
7. The system commits the change to the database
9. The system returns an acknowledgment
10. The system waits for Administrator's next action

> Post-condition:

Customer Database has been updated

> Bounded alternative flow: FOE 4

1. The system displays the appropriate error message

> Post-Condition:

No changes made to the database.

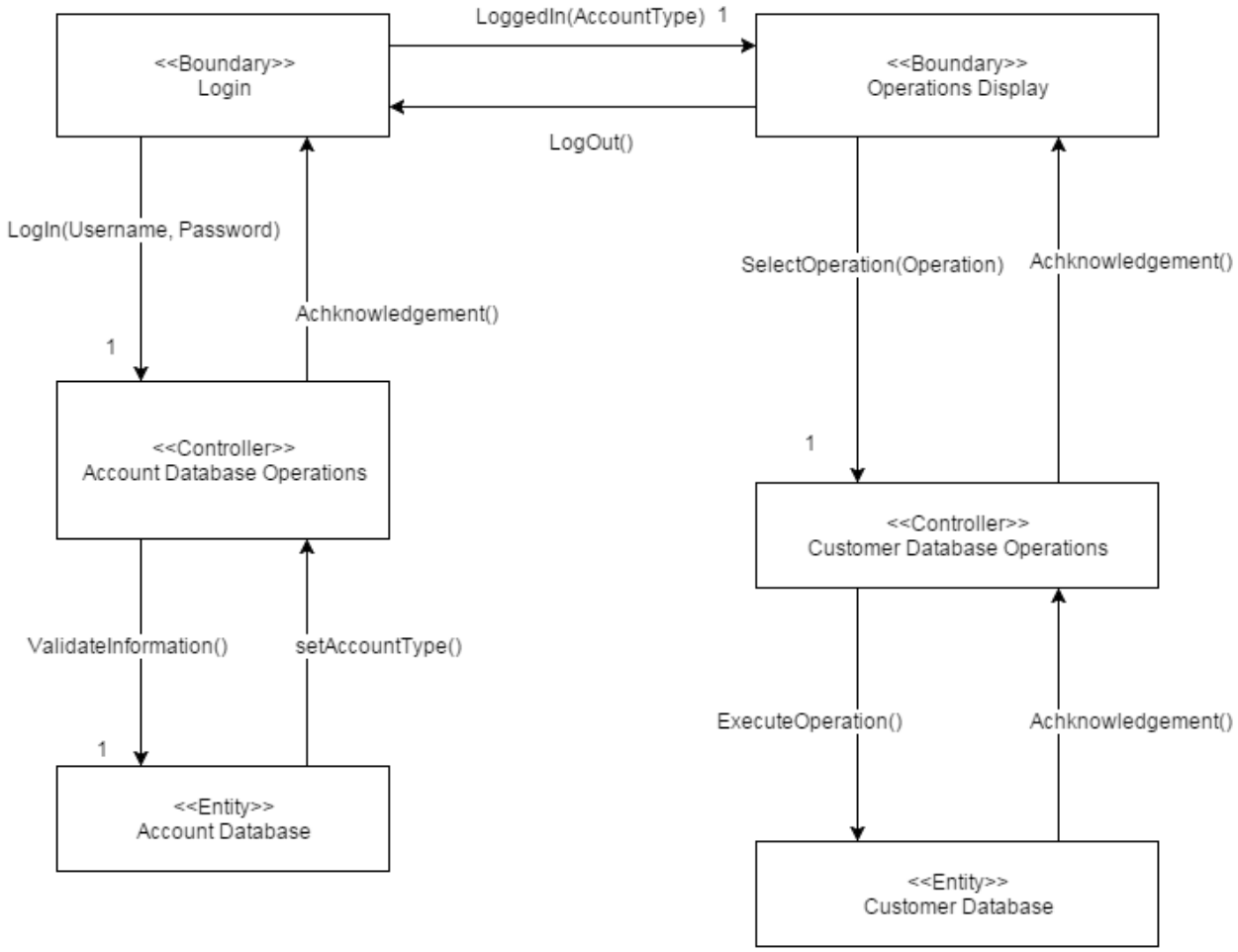
> Exceptions:

> Administrator is notified if the connection to the customer database is unavailable

> Non-functional requirements:

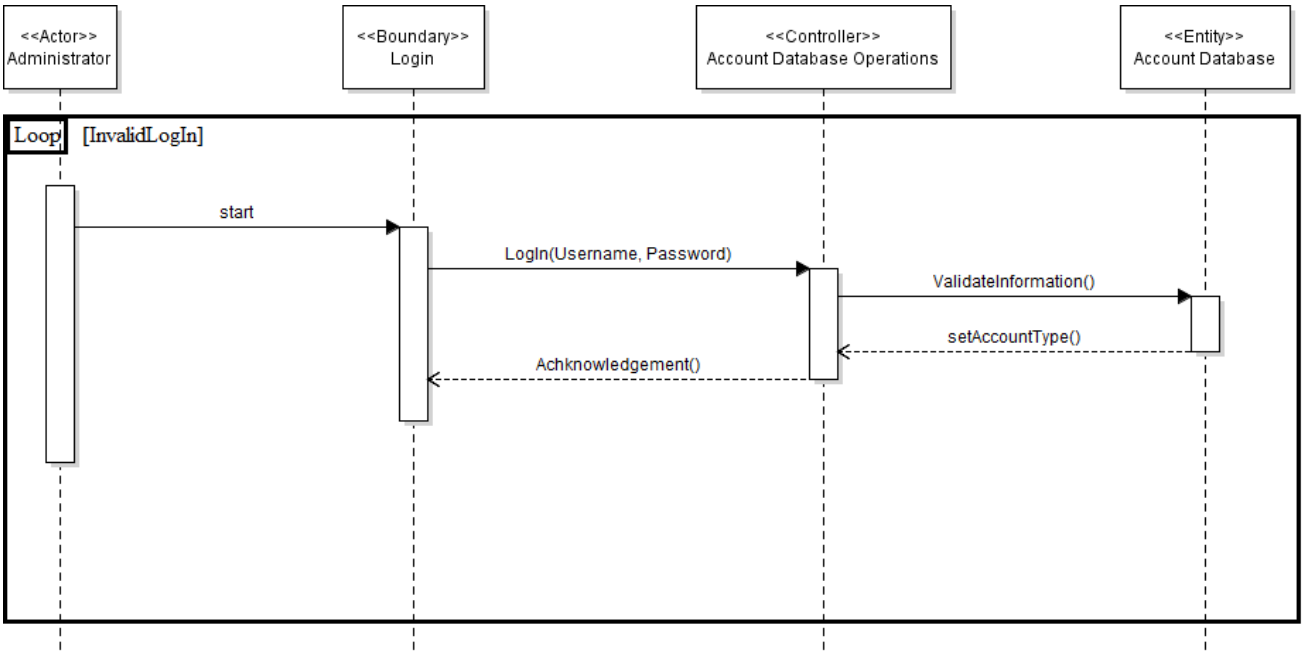
> Change applied quickly and correctly

3.5 Class Diagram (starting point)

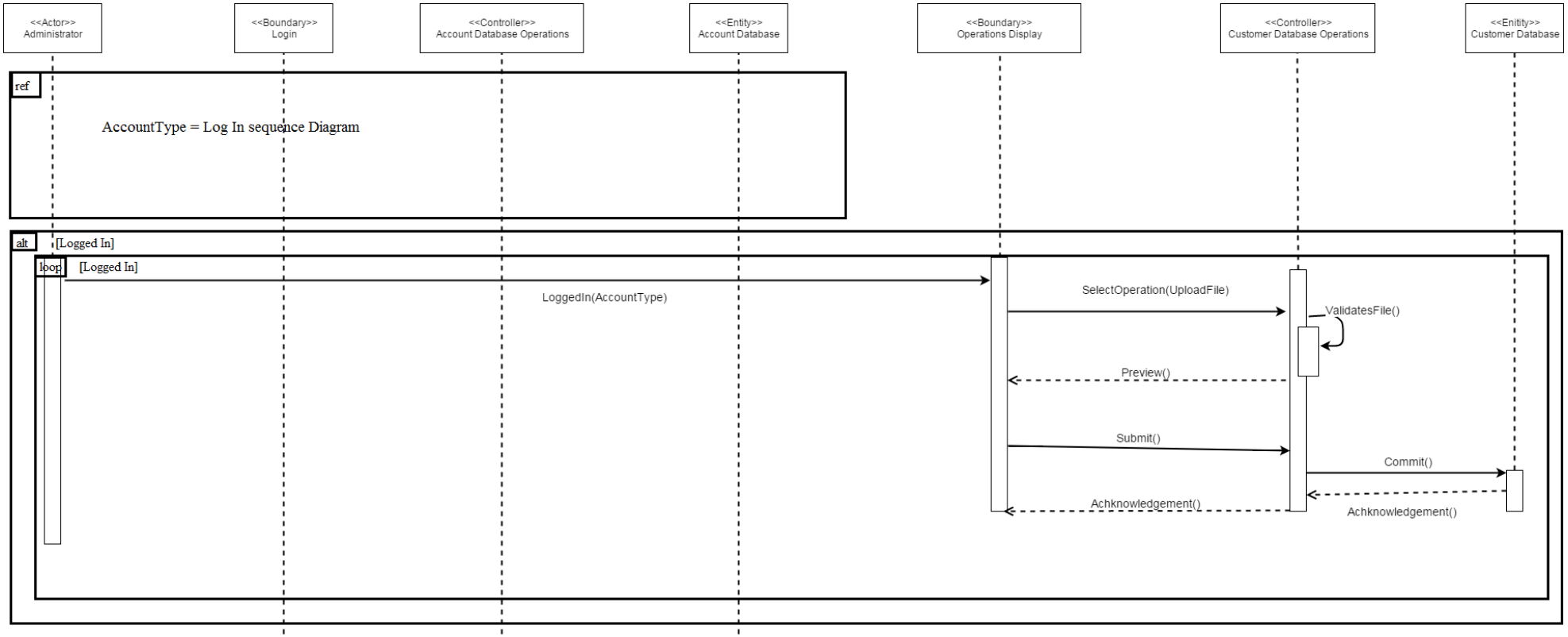


3.6 Sequence Diagrams

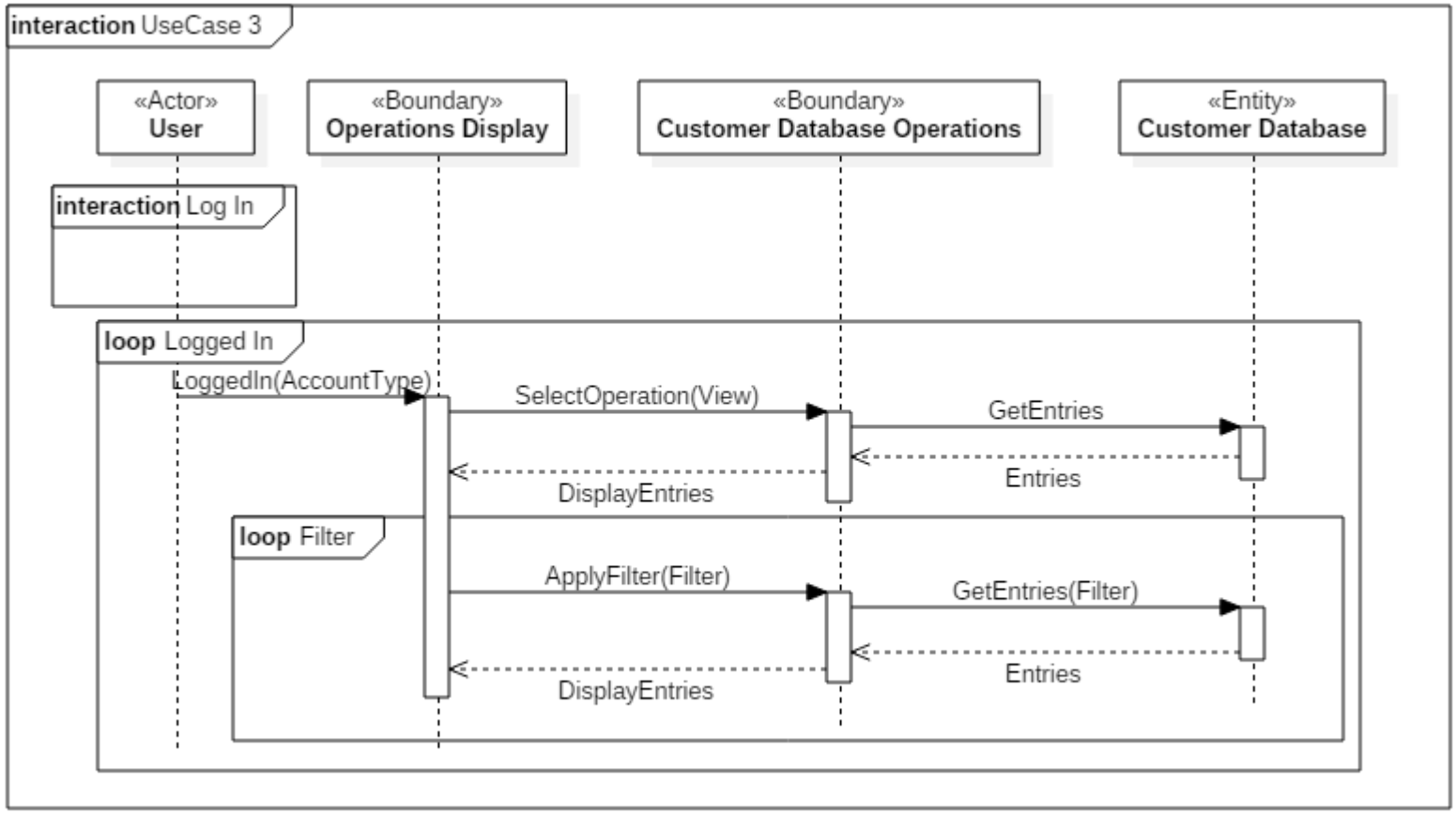
3.6.1 Log In Sequence Diagram (Use Case 1)



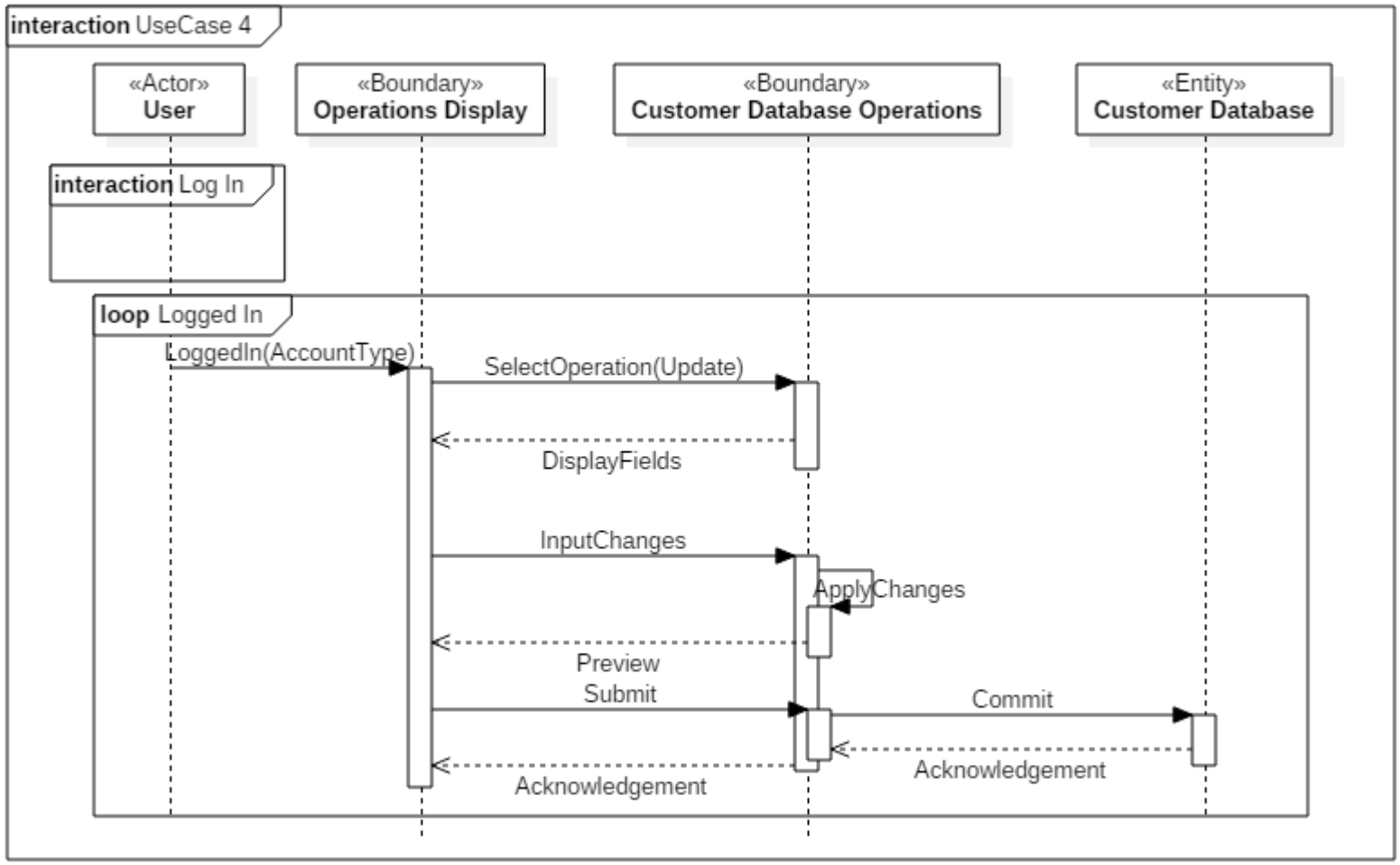
3.6.2 Add Customer Sequence Diagram(Use Case 3)



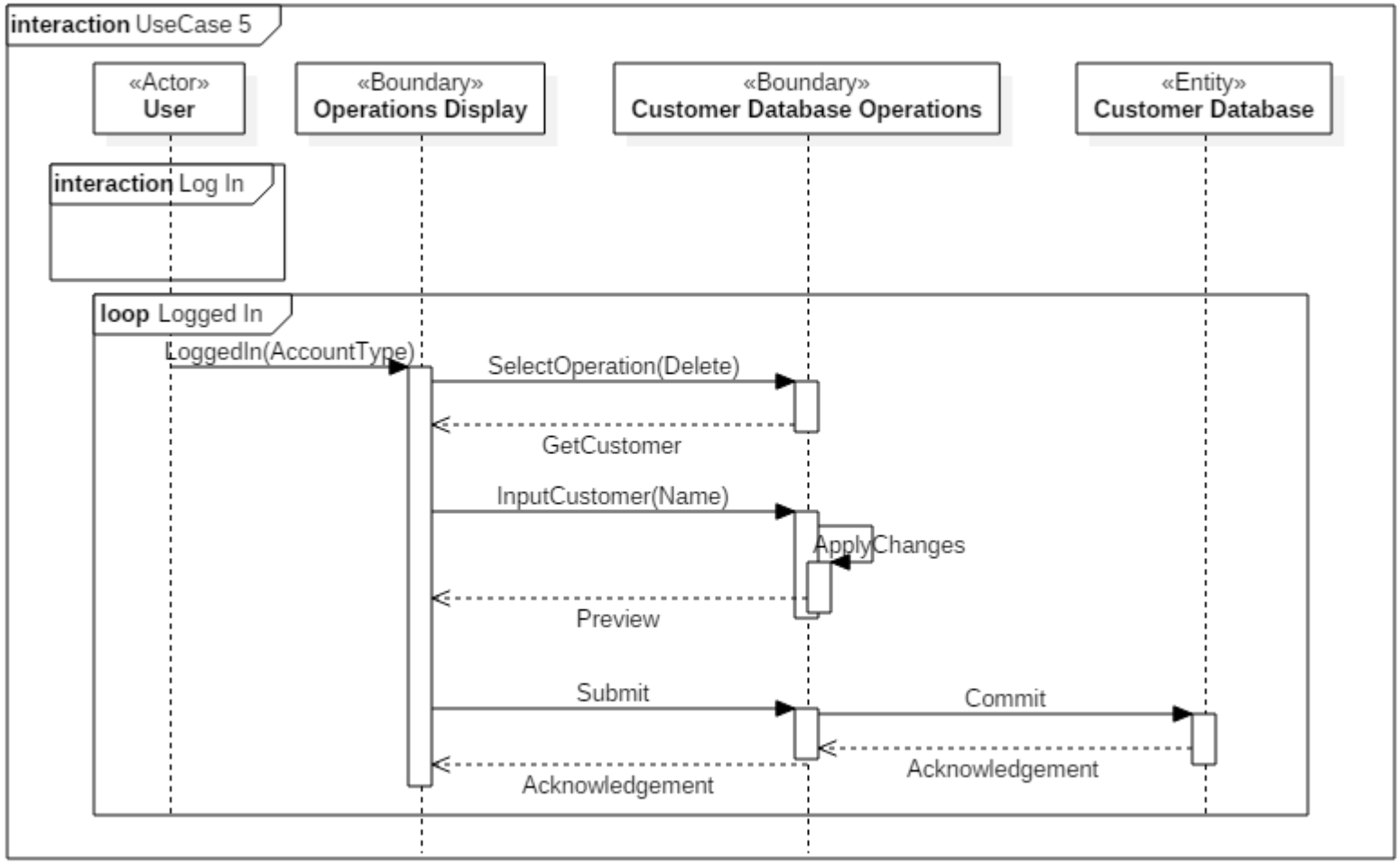
3.6.3 View/Filter Sequence Diagram (Use Case 3)



3.6.4 Update (Use Case 4)

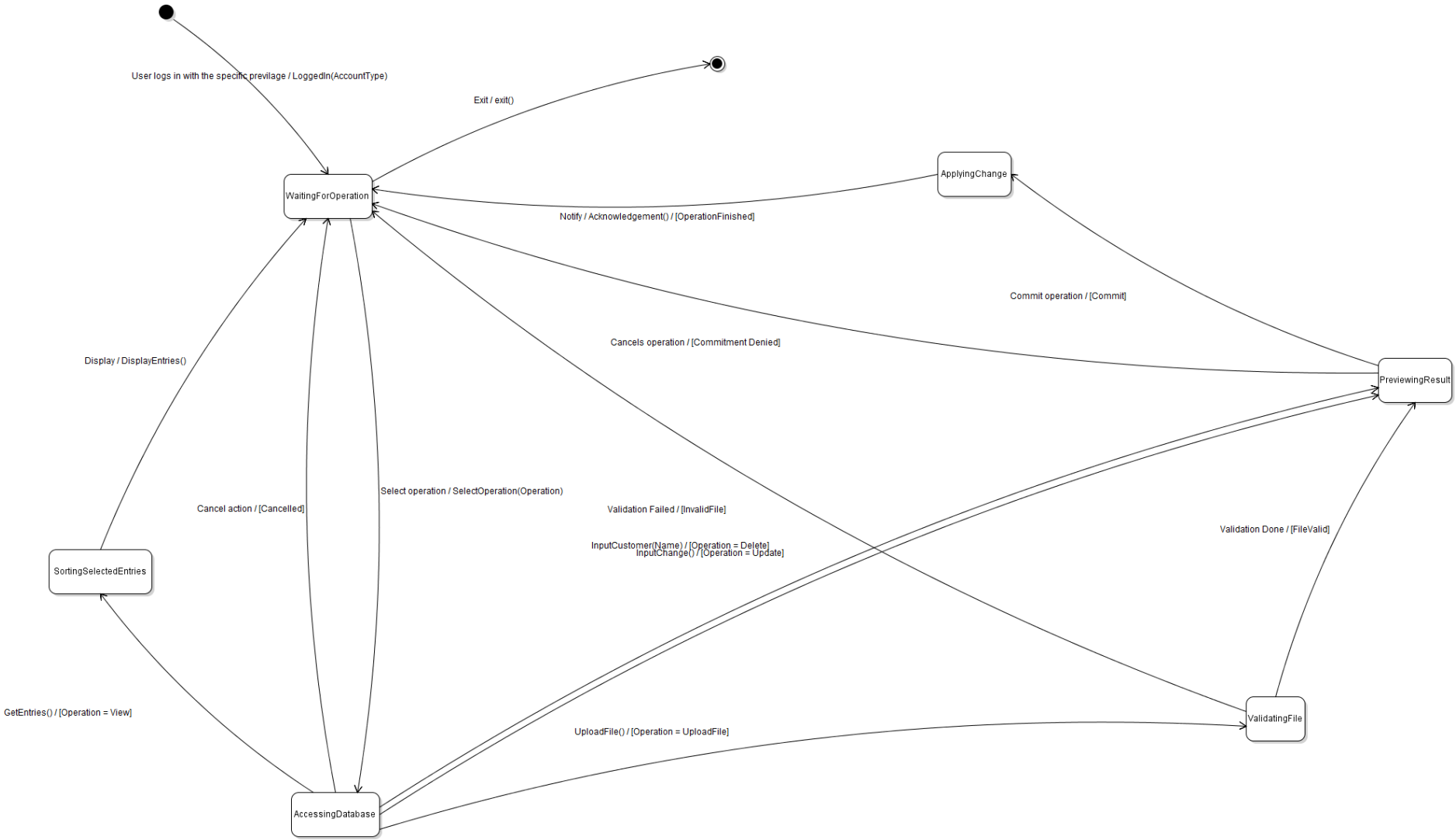


3.6.5 Delete (Use Case 5)

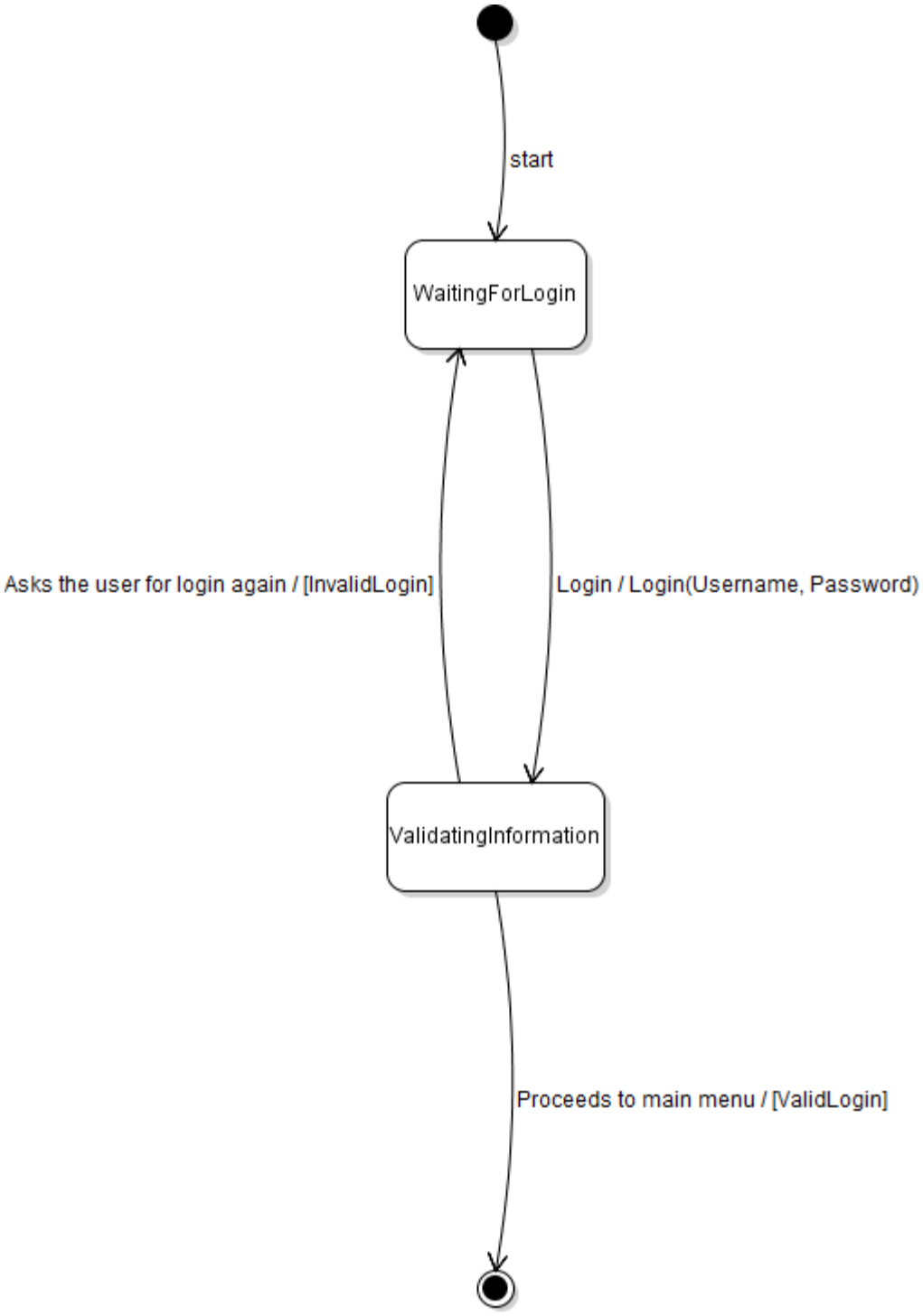


3.7 Statehart Diagram

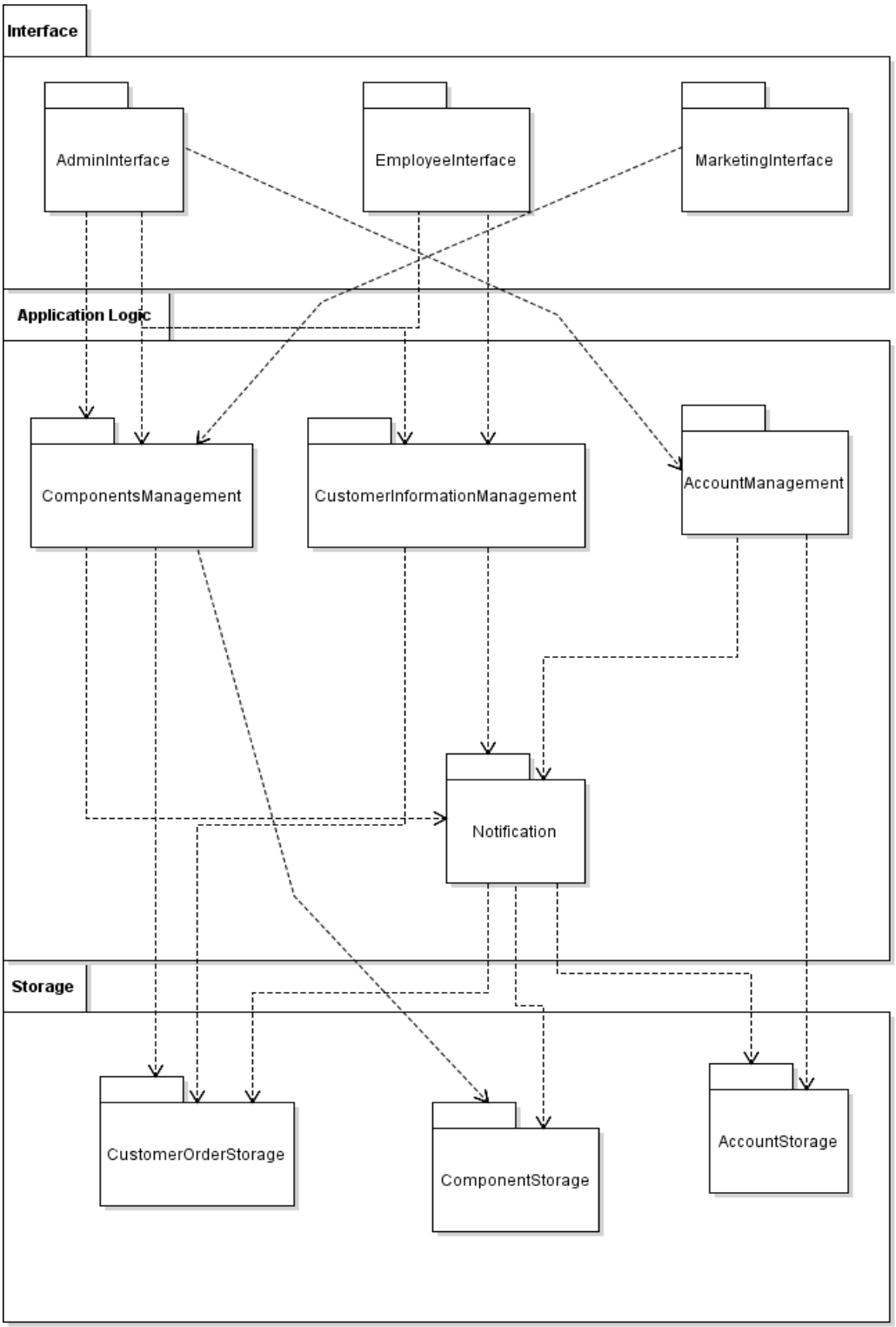
3.7.1 Entity Class: Customer Database



3.7.2 Control Class: Account Database Operations



3.8 Design: Subsystem Decomposition, Interfaces, and Storage Choice



3.8.1 Design Interface Contracts

> Pre-conditions:

> Administrator, Employee, or Marketing has internet access.

> Administrator, Employee, or Marketing has a valid account to access the system.

> Post-conditions:

> If Administrator makes change to account information, the data in AccountStorage gets updated accordingly.

> If Administrator or Employee makes change to customers' information, the data in CustomerOrderStorage gets updated accordingly

> If Administrator, Employee, or Marketing changes information on components, the data in ComponentsStorage and CustomerOrderStorage gets updated accordingly.

>When changes are made, the system sends out the notification.

>Method Signatures:

>login(username, password): This method allows a user to enter his/her username and password

>verifyLogin(username, password): It goes through the database that the user tries to access, and return warning message if the user enters wrong username or password. If the username and the password are verified, the user gains the access to the system.

>exit(): It allows the user to log out and exits the program.

>editEntry(entry, newValue): It is called when the user wants to edit the customer, component, or account information.

>deleteEntry(entry): It is called when the user wants to delete customer, component, or account information.

>addEntry(newValue): It is called when the user wants to add customer, component, or account information.

>viewEntry(entry): It shows the user the specific entry.

>notify(): Whenever the user chooses an action, it notifies the user that the system completed the specific task.

4.0 Code link

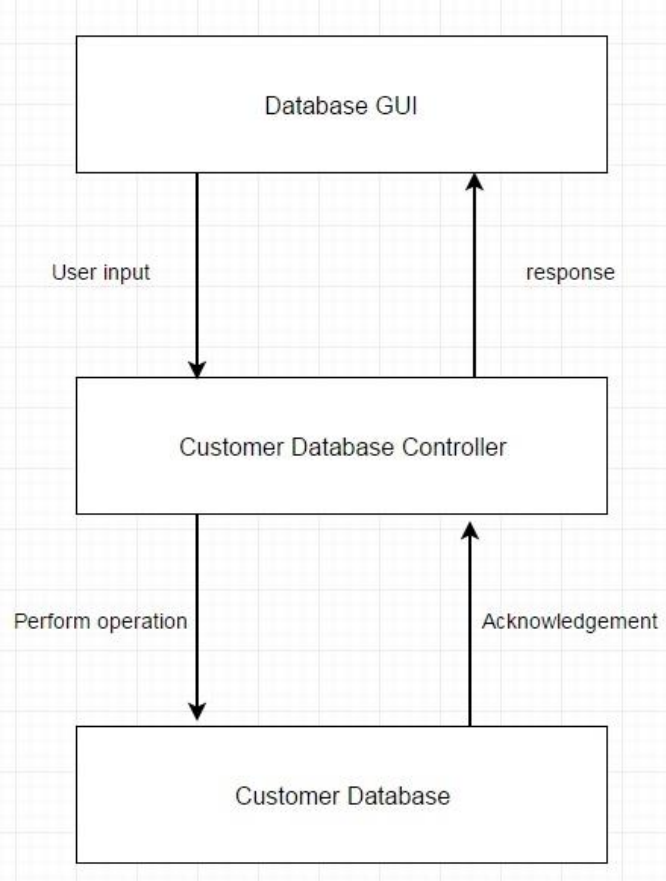
https://github.com/SetzerBULKBARN/whitebox

We implemented the filter function for the customer database.

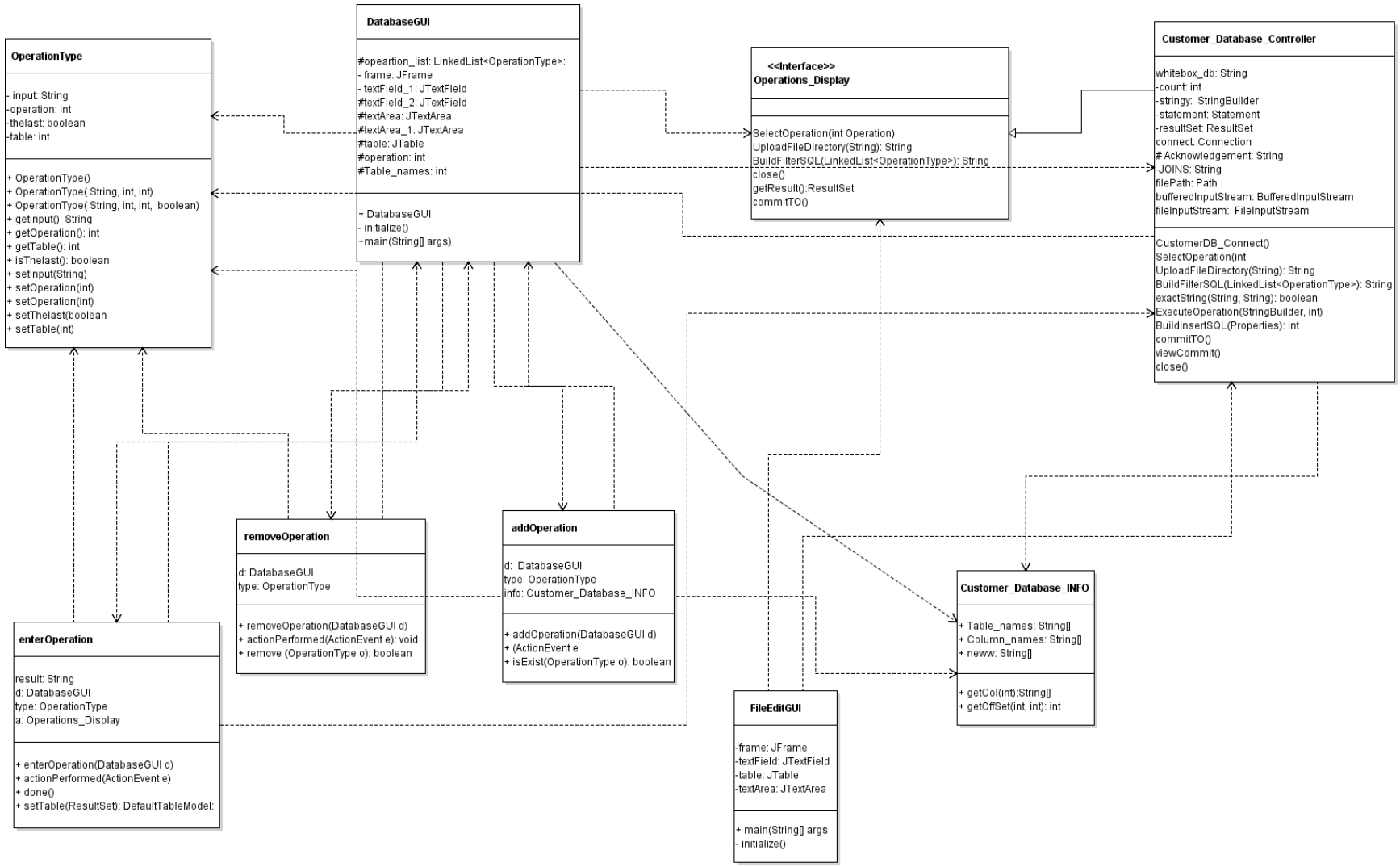
Attempted to do insert as well, but was not successful.

4.1 System Design

We tried to implement the 3 layer architecture where the presentation is the graphical user interface, the logic layer is the class that performs the operations on the database, and the storage layer is a database that is done with MySQL.



4.2 Full Class Diagram



4.3 Test plan

Method for testing the GUIs was using the stub method, with whatever the user input in the GUI the output would be outputted and the stub would respond. Main method in testing the operations to be performed on the database was manually executing the SQL query in MySQL then inputting the same specifications in the GUI and matching both output result sets and queries constructed.

Ran FindBugs for static analysis and multiple markers were at a method, in OperationType.java:

```
public void setTable(){
        this.table = table;
    }
```

The bug description was: Self assignment of field table in OperationType.setTable(),

Such assignments are useless, and may indicate a logic error or typo.

Rank: Scariest (3), **confidence**: Normal
Pattern: SA_FIELD_SELF_ASSIGNMENT
Type: SA, **Category**: CORRECTNESS (Correctness)

Action was to get rid of this method since it was not being used by the GUI.

4.4 Code Coverage Report (used Emma)

Coverage 56.6 %; Covered Instructions: 1588; Missed Instructions: 1217;

Total Instructions: 2805

A lot of instructions missed because we did not complete the other functionality, which is inserting into the database. It includes its own GUI and methods in the operations layer.

5.0 Risks/Open Issues

- Risks

- Design Flaws: unauthorized privilege escalation, slow performance, data loss
- Data Corruption: Human Error
- Depends on Dropbox availability

    - Malware: Leakage of personal information
    - SQL injection on log in screen
    - Limited Security

- Open Issues
    - Non-dynamic code.