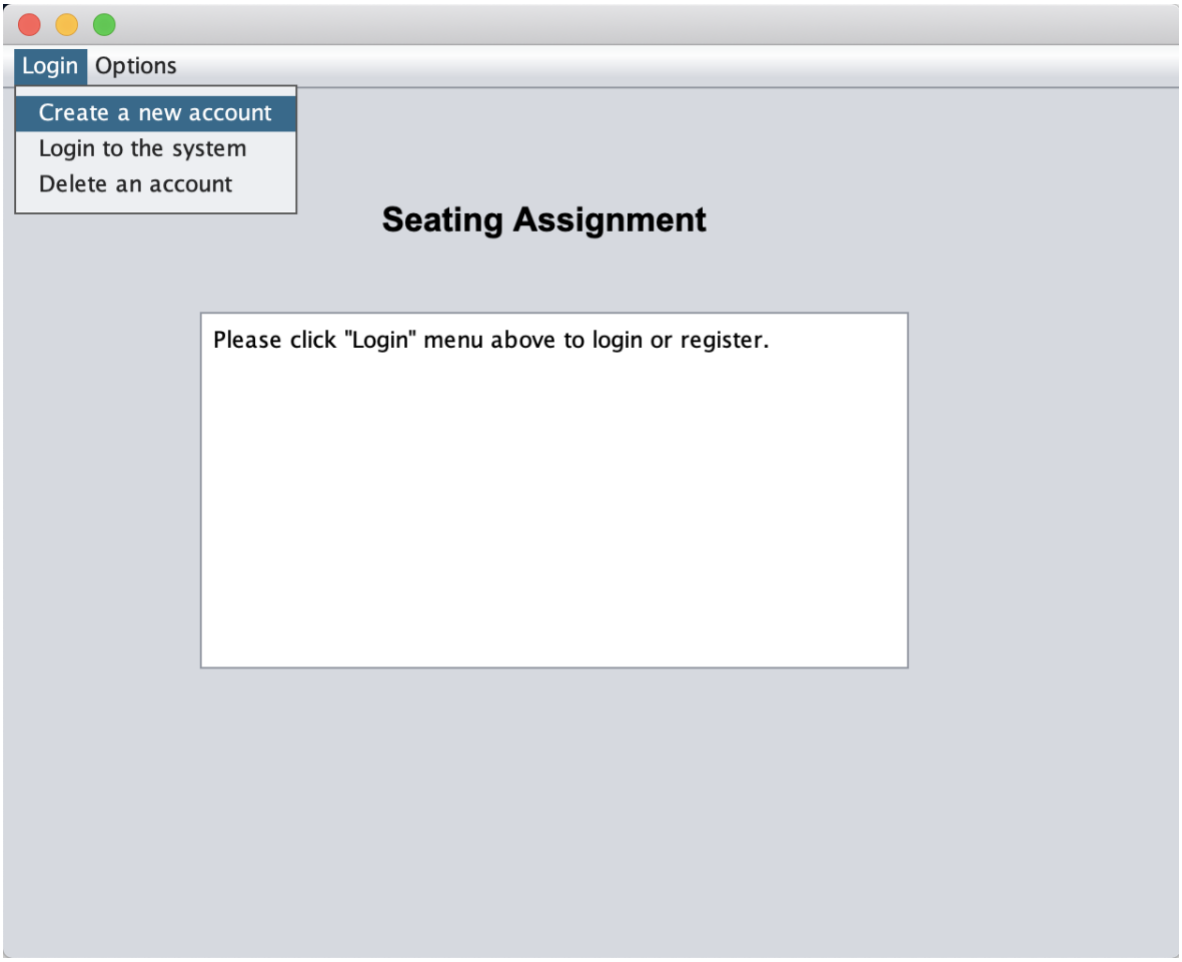
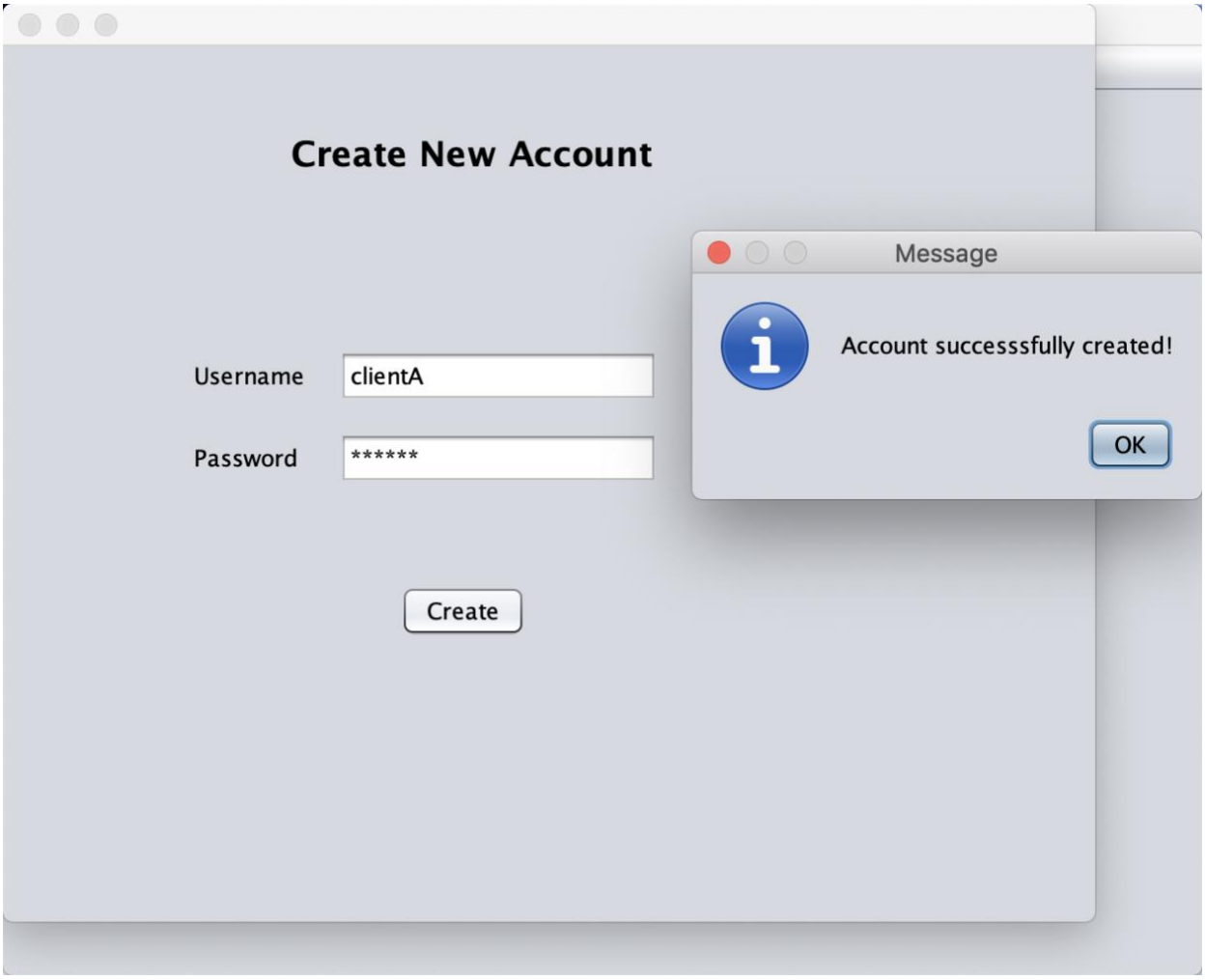
CRITERION B: DESIGN

Object Design

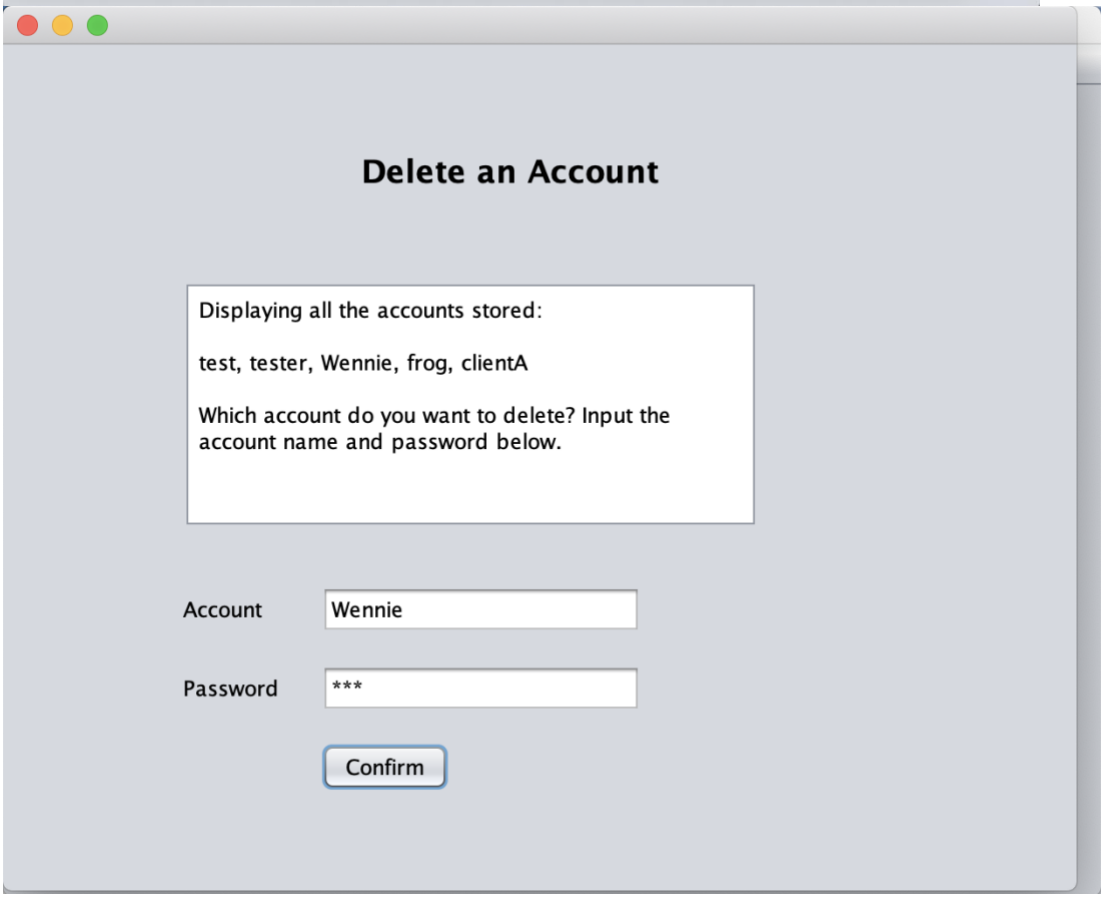
This program will have several user interfaces. When starting the program, the program displays a message to tell the user to login to the system. By clicking the “Login” menu above, the user can choose to create a new account, login to the system, or delete an existing account.



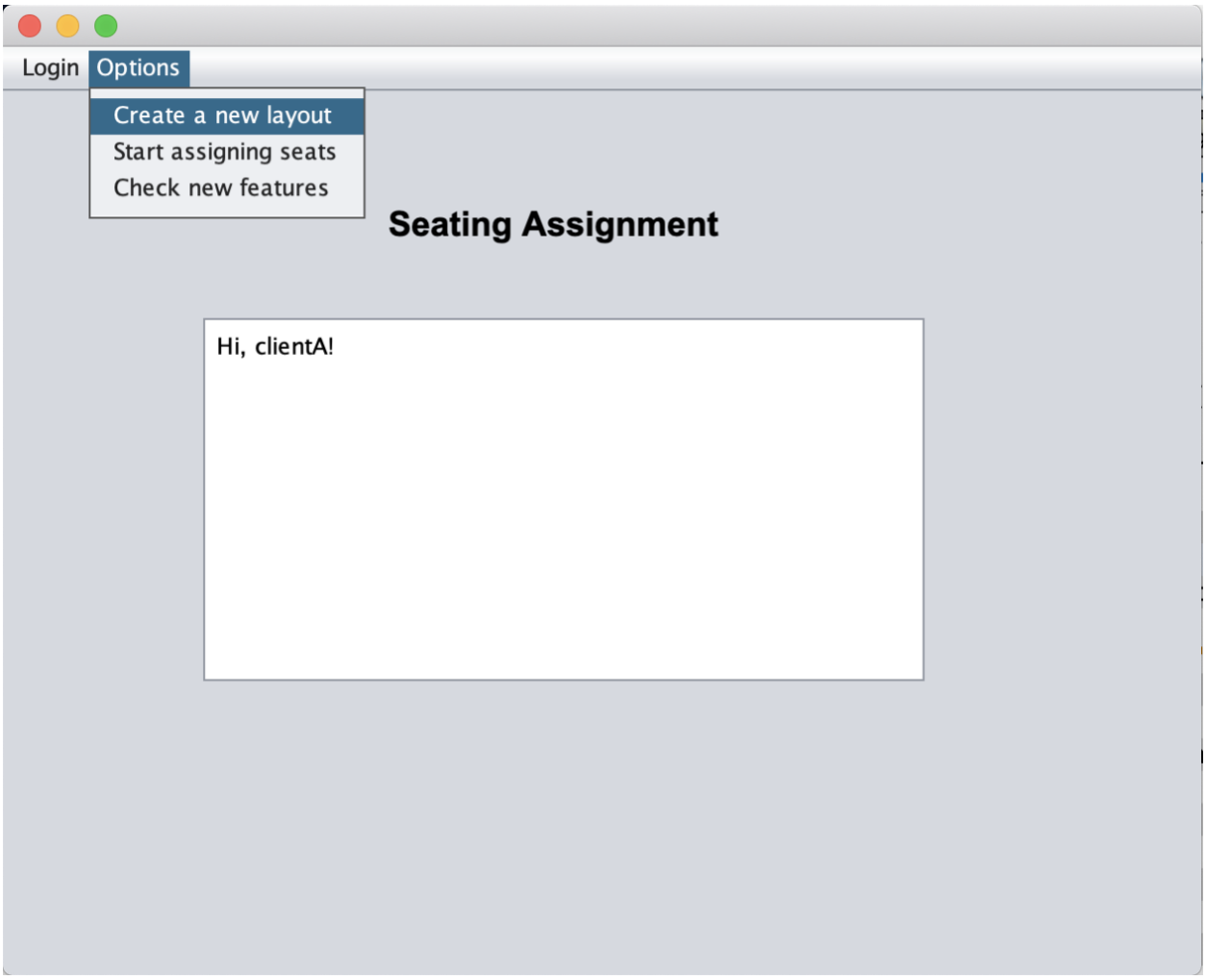
Each choice under “Login” menu would have a corresponding frame popping out.

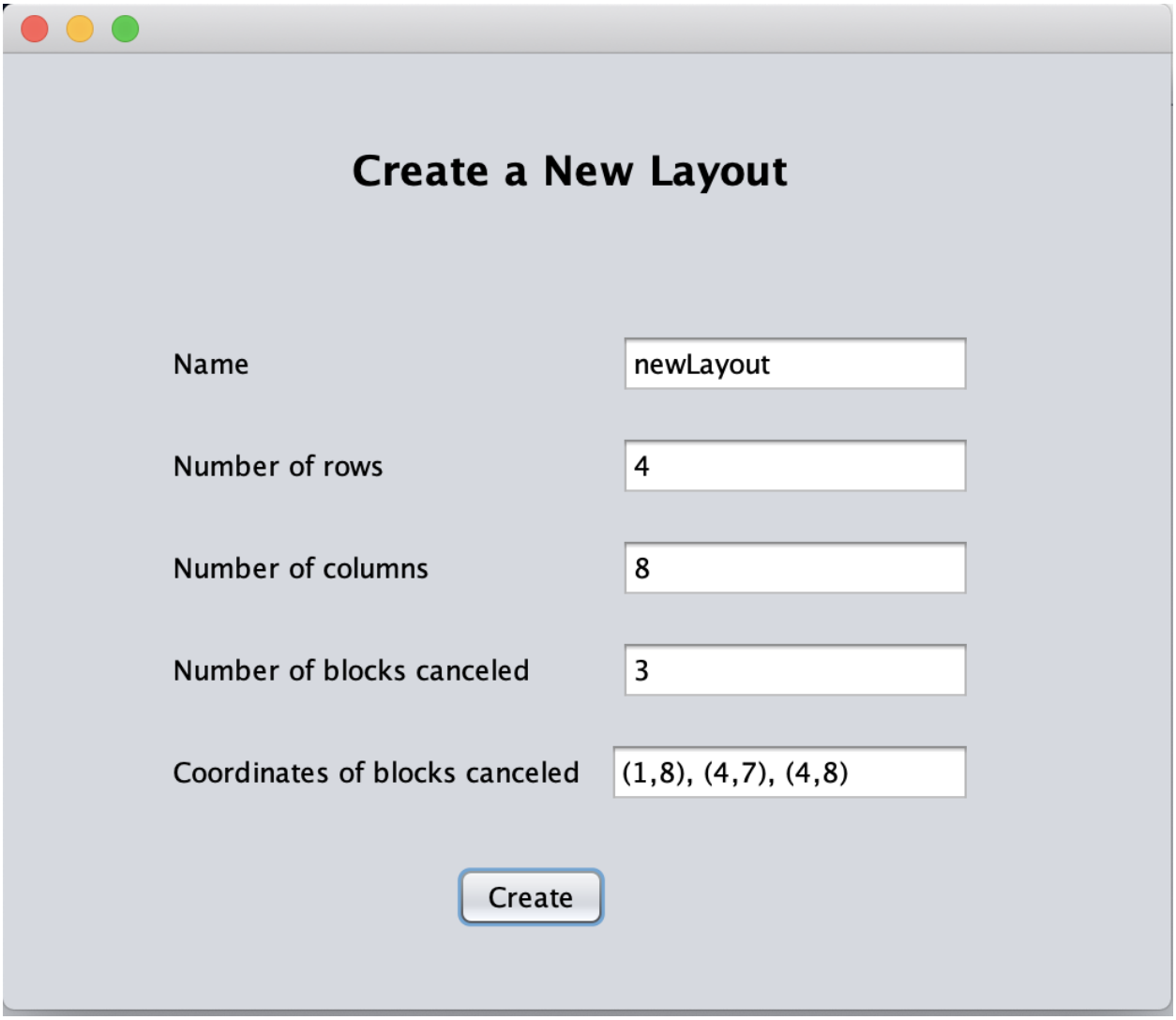

Create New Account
Username
clientA
Password
******
Create
Message
Account successsfully created!
OK


Login
Username
clientA
Password
******
Confirm
Message
Login successful!
OK

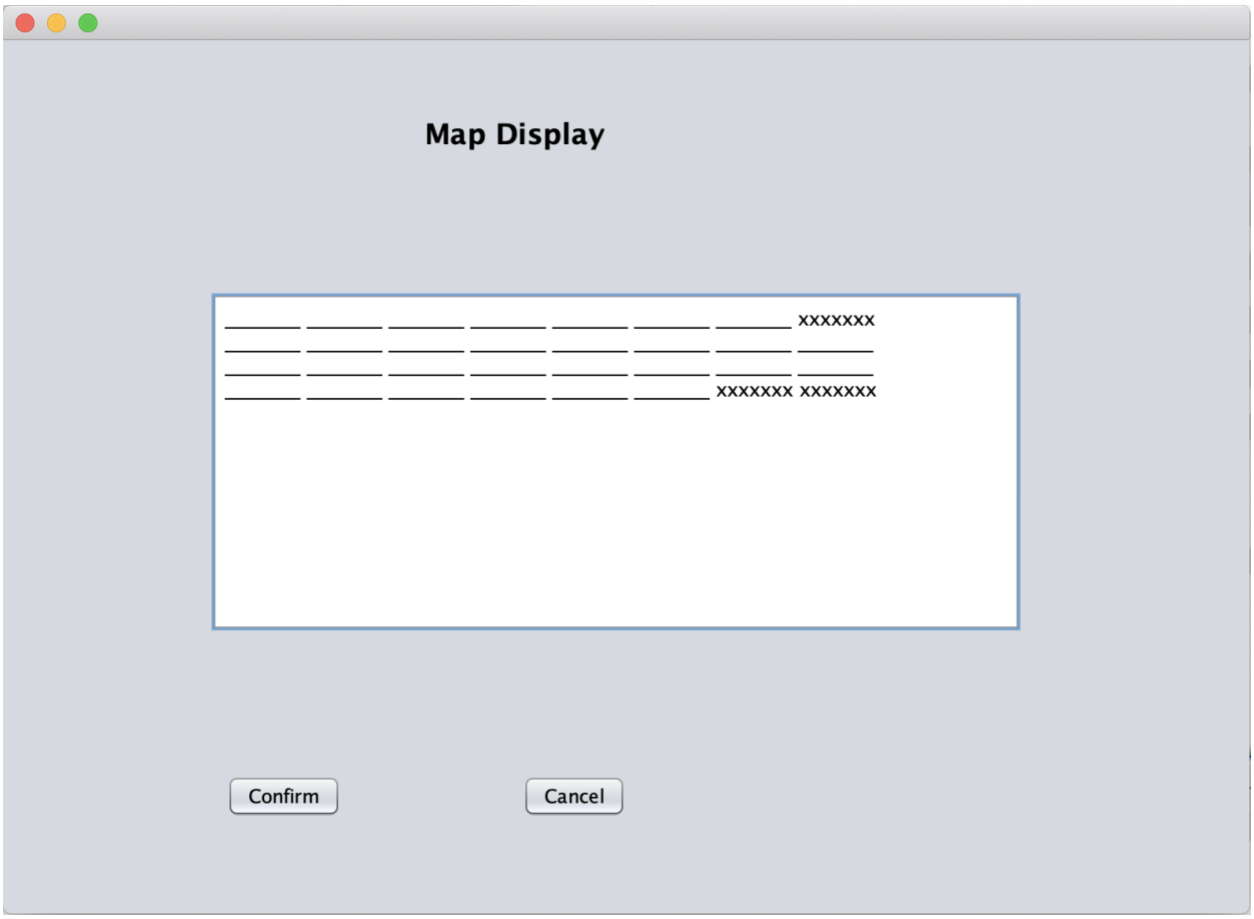

Delete an Account
Displaying all the accounts stored:
test, tester, Wennie, frog, clientA
Which account do you want to delete? Input the account name and password below.
Account
Wennie
Password
***
Confirm

After the client login to the system, the program displays the greeting message. The client can then select “Options” menu. Three choices under this menu would correspond to three different frame pops out. Choices “Start assigning seats” and “Check new features” are not implemented yet. Clicking “Start assigning seats” would currently display the seating table in the terminal. Clicking “Check new features” would simply tell the client that the only seating feature is the random seat assignment.



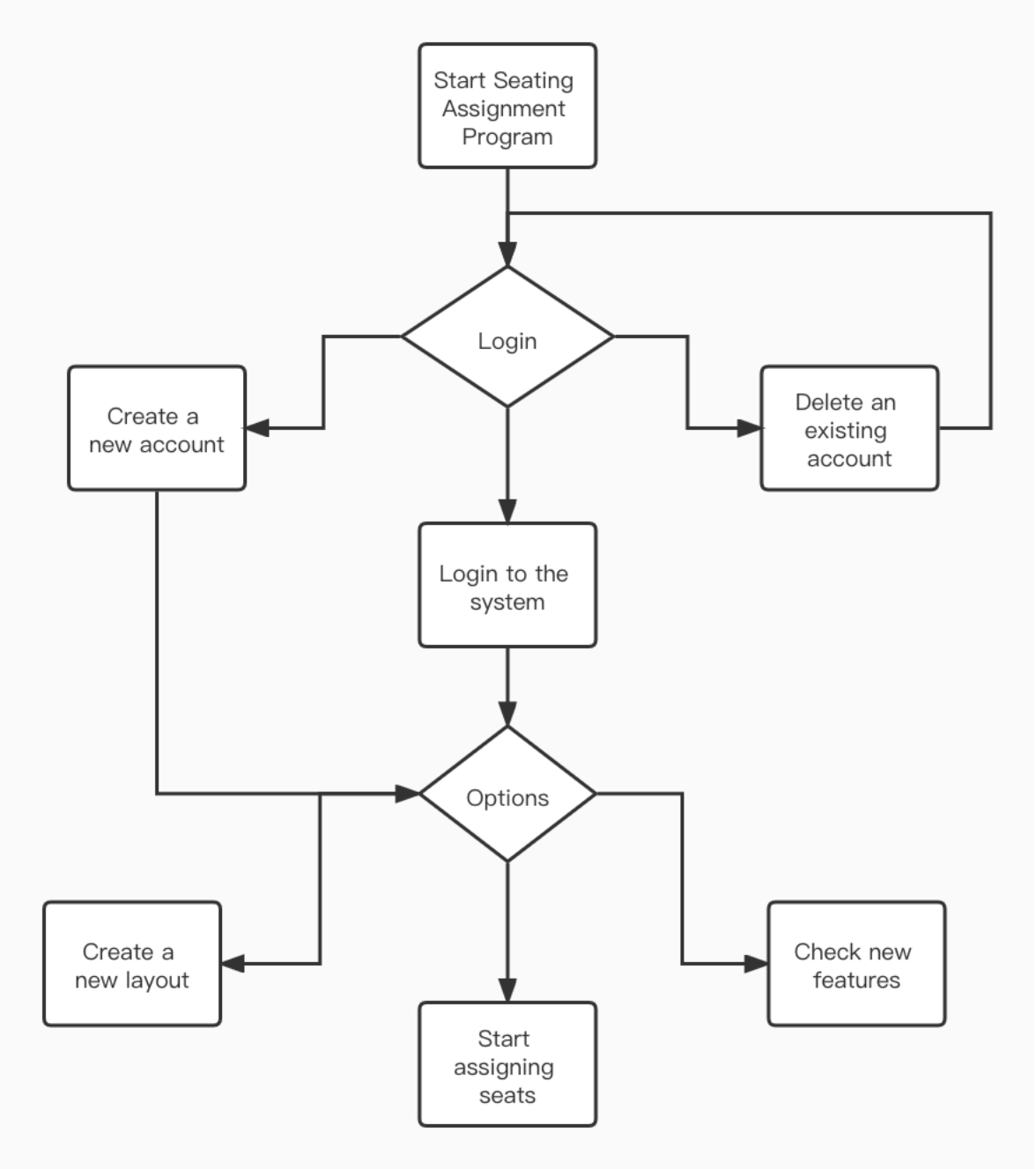


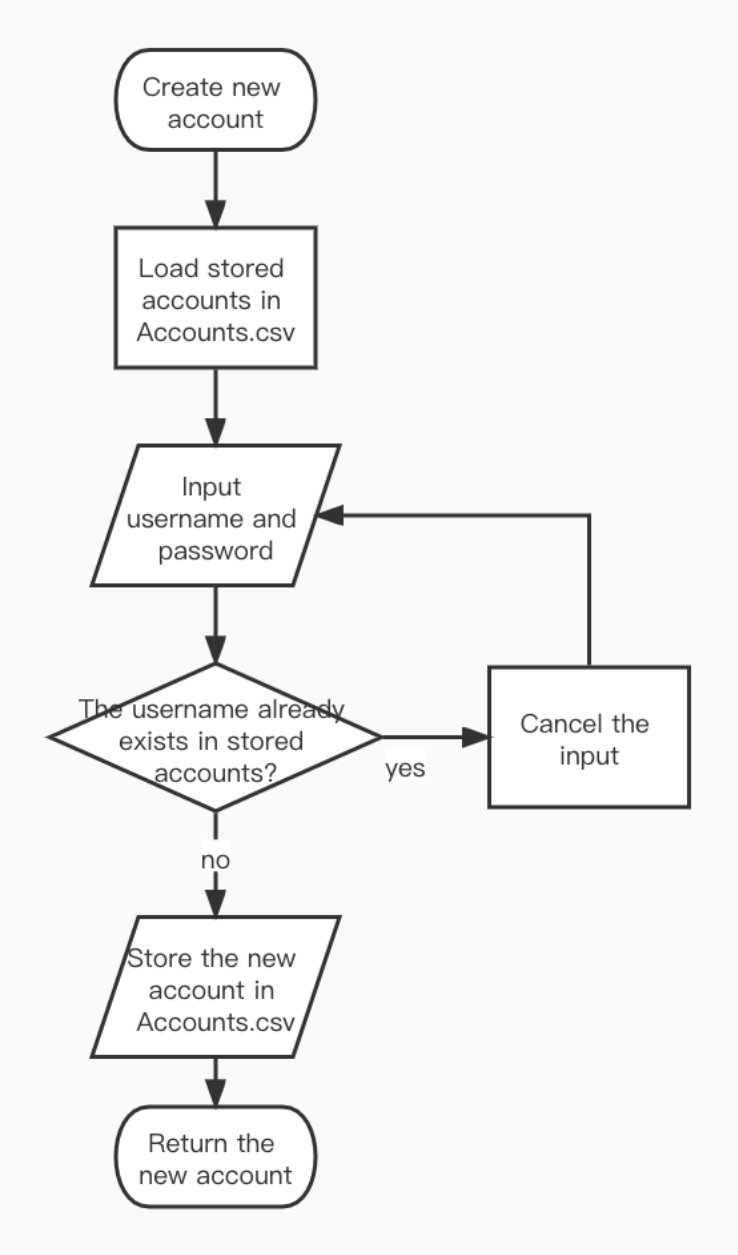
After the client input data for the new layout, the program displays the map, and the client can choose whether to conform and save the layout or to cancel and create the layout again.

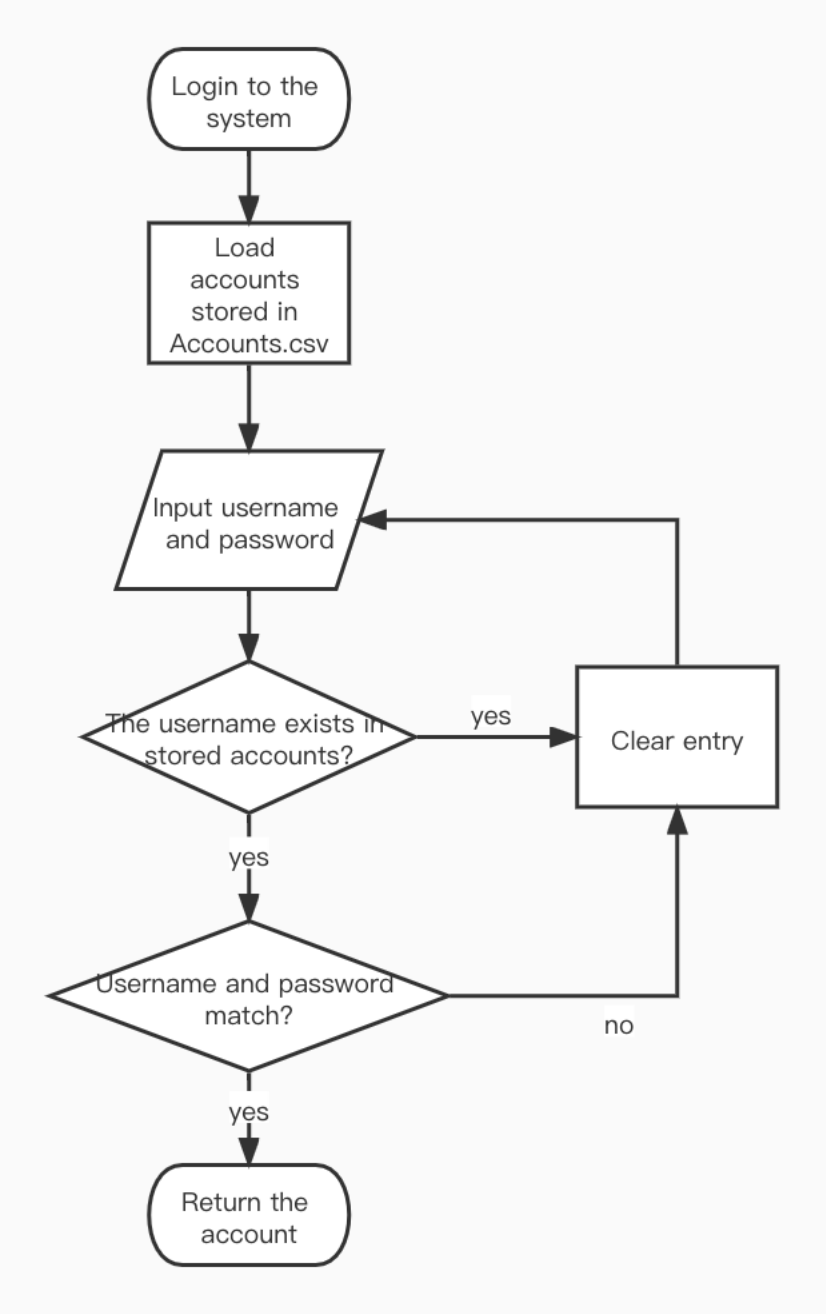


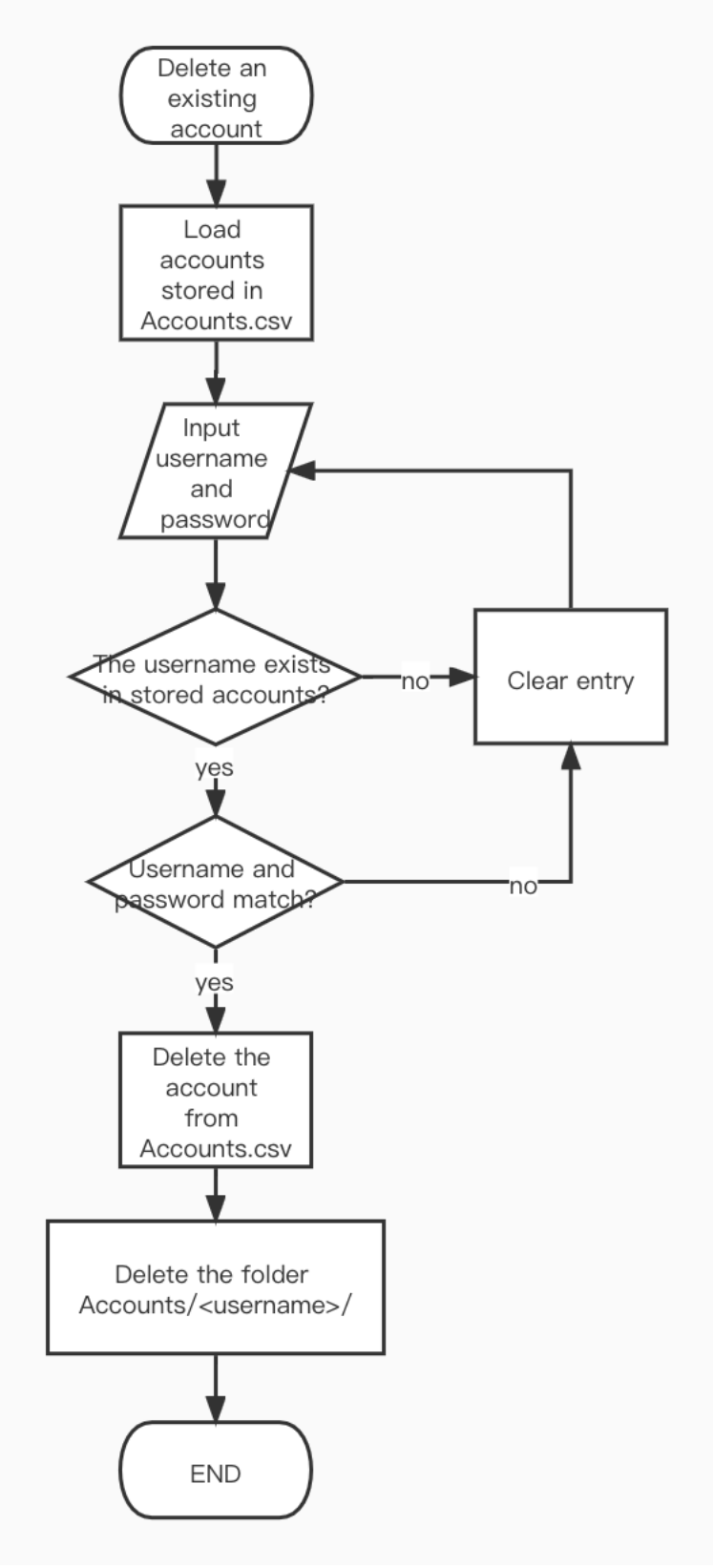
Processing Flowchart

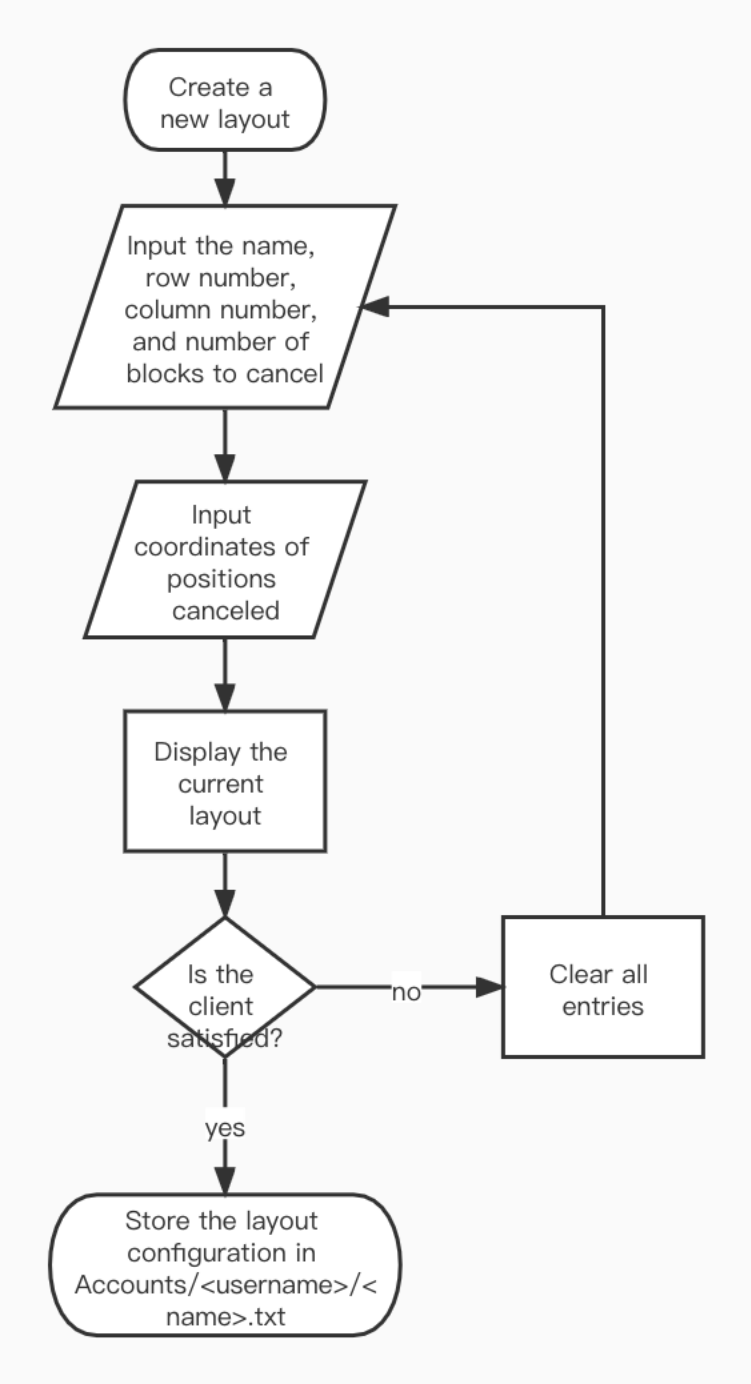
Below flowcharts outline the overall logic of the program.

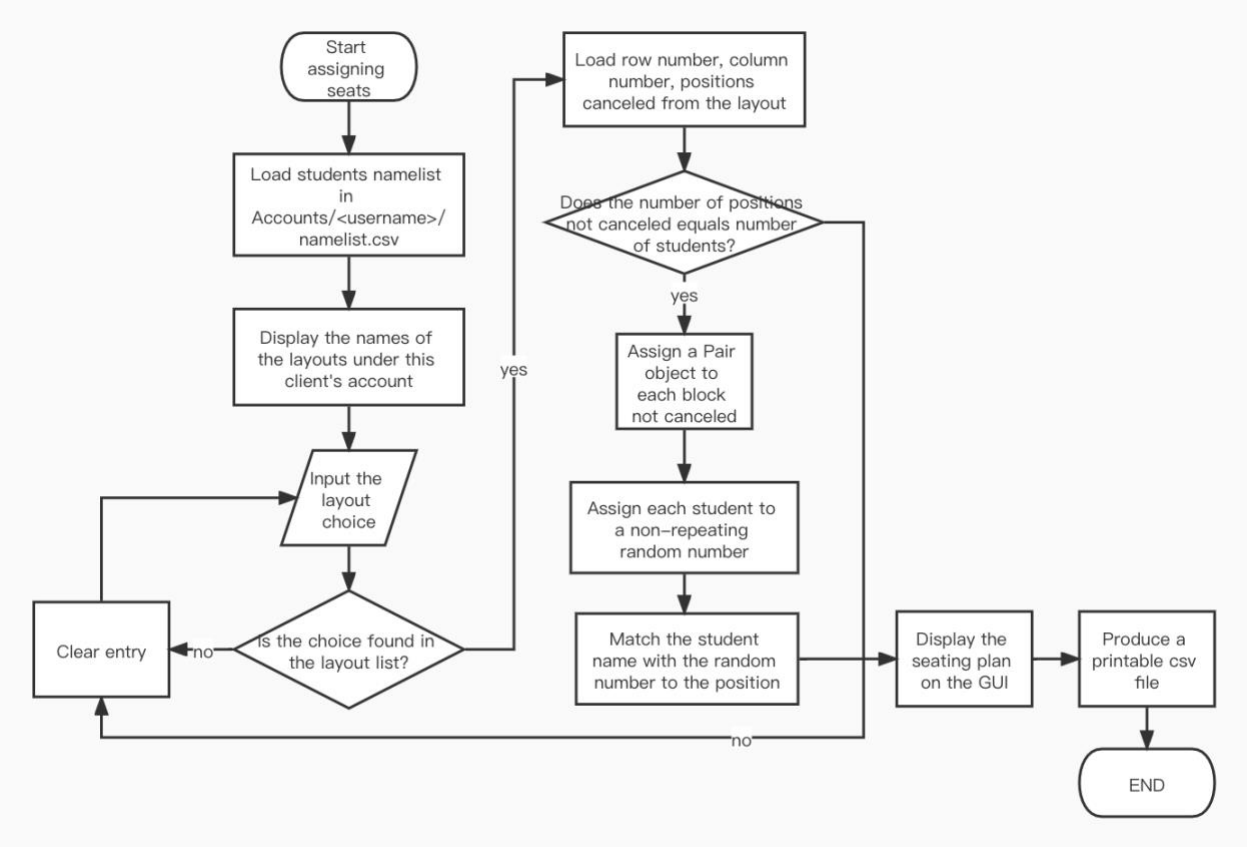

Start Seating Assignment Program
Login
Create a new account
Delete an existing account
Login to the system
Options
Create a new layout
Check new features
Start assigning seats

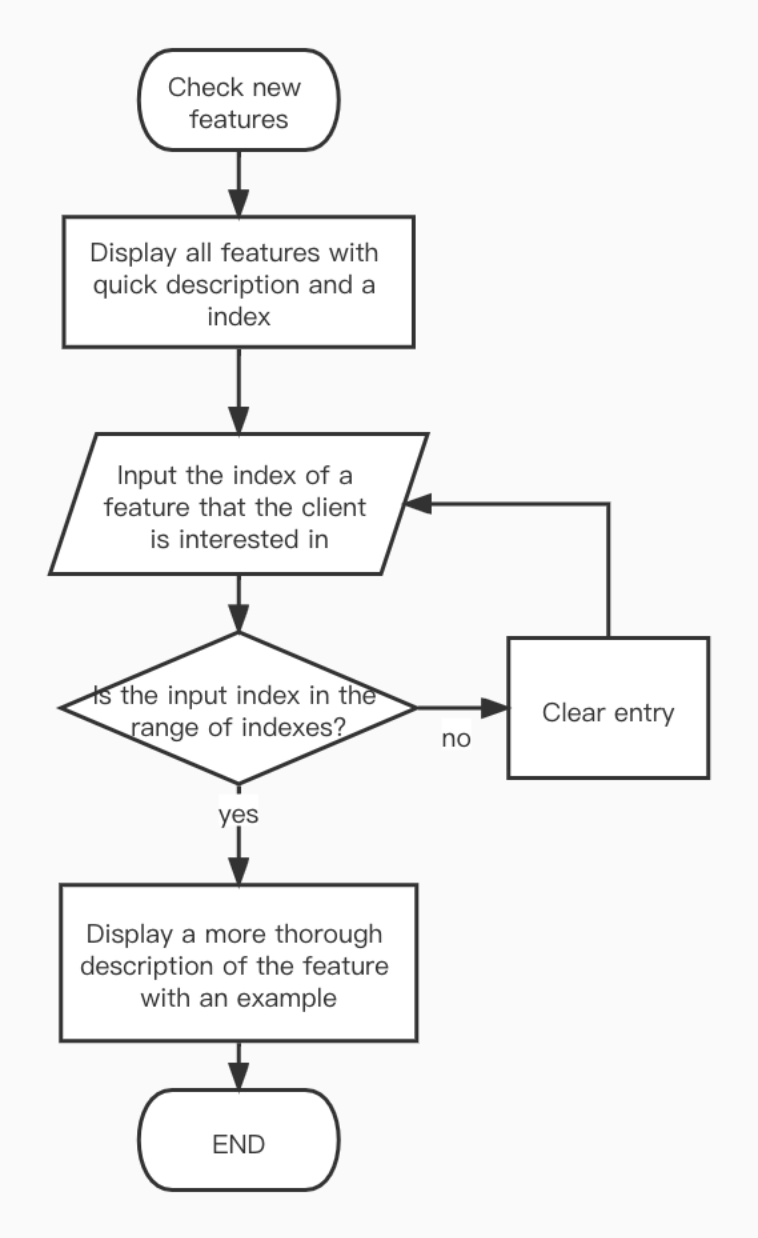

Create new account
Load stored accounts in Accounts.csv
Input username and password
The username already exists in stored accounts?
yes
Cancel the input
no
Store the new account in Accounts.csv
Return the new account


Login to the system
Load accounts stored in Accounts.csv
Input username and password
The username exists in stored accounts?
yes
Clear entry
yes
Username and password match?
no
yes
Return the account


Delete an existing account
Load accounts stored in Accounts.csv
Input username and password
The username exists in stored accounts?
no
Clear entry
yes
Username and password match?
no
yes
Delete the account from Accounts.csv
Delete the folder Accounts/<username>/
END

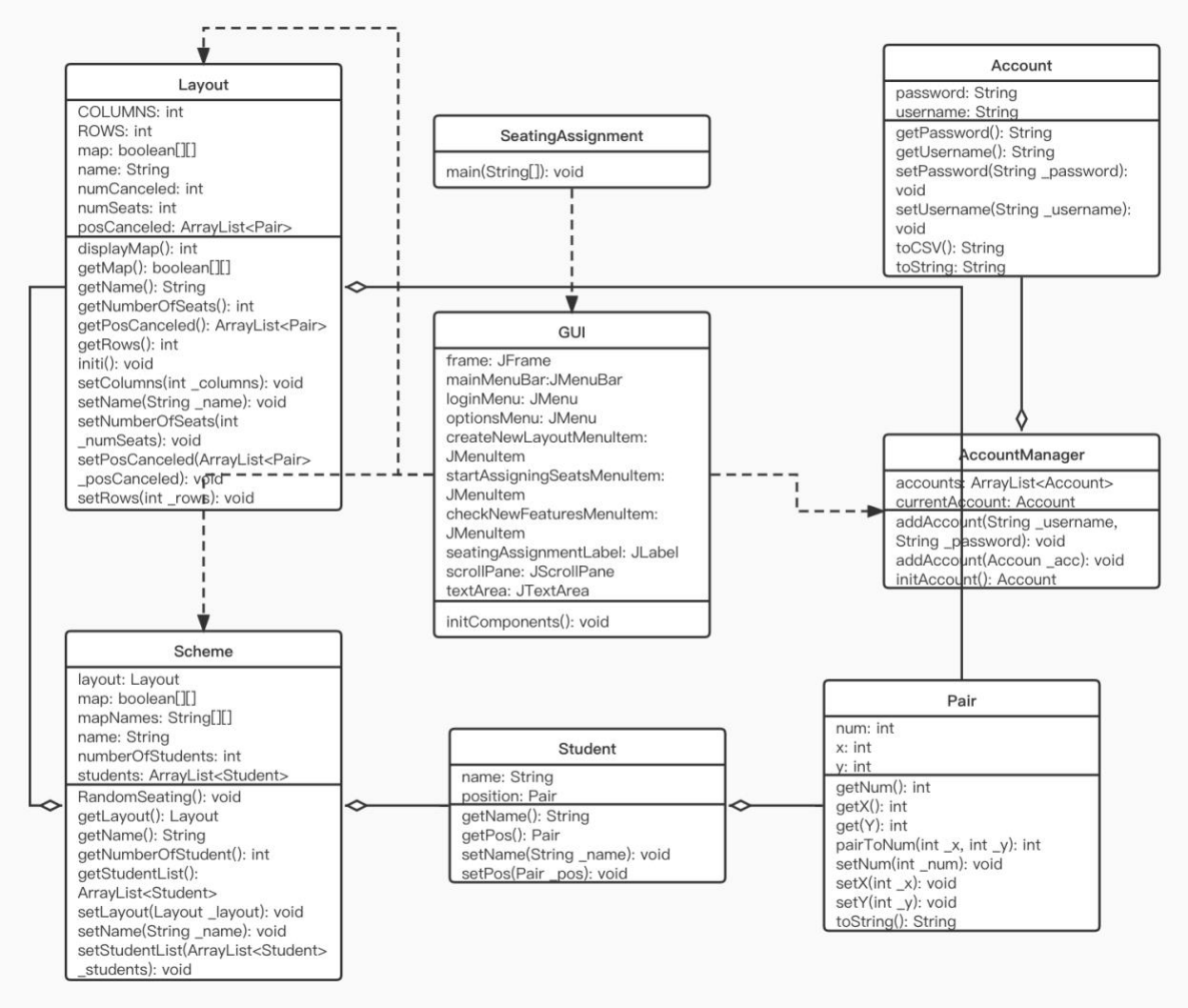

Create a new layout
Input the name, row number, column number, and number of blocks to cancel
Input coordinates of positions canceled
Display the current layout
Is the client satisfied?
no
Clear all entries
yes
Store the layout configuration in Accounts/<username>/<name>.txt


Start assigning seats
Load students namelist in Accounts/<username>/ namelist.csv
Display the names of the layouts under this client's account
Input the layout choice
Is the choice found in the layout list?
no
Clear entry
yes
Load row number, column number, positions canceled from the layout
Does the number of positions not canceled equals number of students?
yes
Assign a Pair object to each block not canceled
Assign each student to a non-repeating random number
Match the student name with the random number to the position
no
Display the seating plan on the GUI
Produce a printable csv file
END



Class Diagram

I will code the program using object-oriented programming, consist of 8 classes and each implements the use of encapsulation and polymorphism. This UML diagram shows the class components (instance variables and methods) and relationships between classes.



Pseudocode

I use pseudocode to help illustrate the functions of the main methods used in this program.

Create a new account:

```
    loop while true
    myUsername = input from username textField
    myPassword = input from password textField
    boolean ok = true
    loop for each acc in accounts
        if acc's username equals myUsername then
            output "Account already exist! Create your account again."
            ok = false
            break
        end if
    end loop
```

```
        if ok = false then
                clear username textField
                clear password textField
                continue
        end if
        output "Account successsfully created!"
        Account myAcc = new Account(myUsername, myPassword)
        add myAcc in AccountManager
        add myAcc to Accounts.csv
        add new directory Accounts/<myUsername>
    end loop


Login:
        if acounts is empty then
                throw Exception "No registered users!"
        end if
        loop while true
        myUsername = input from username textField
        myPassword = input from password textField
        boolean ok = true
        loop for each acc in accounts
                if acc's username equals myUsername and acc's password equals myPassword
then
                        output "Login successful!"
                        currentAccount = acc
                        return acc
                end if
        end loop
                output "Username and password don't match. Input again."
        clear username textField
        clear password textField
    end loop


Delete an account:
        Display all accounts stored
        loop while true
        myUsername = input from username textField
        myPassword = input from password textField
        boolean ok = false
        loop for each acc in accounts
                if acc's username equals myUsername and acc's password equals myPassword
then
                        output "Account "+myUsername+" is canceled"
                        remove the account from accounts
```

```
                update Accounts.csv
                delete Accounts/<myUsername> directory
                ok = true
            end if
        end loop
        if ok = true then
            break
        else
            output "Username and password don't match. Input again."
        end if
    end loop


Create a new layout:
        _name = input from name textField
        row = input from row textField
        col = input from column textField
        cancelNum = input from number of blocks to cancel textField
        loop for i from 1 to cancelNum
            _x = input from x textField
            _y = input from y textField
            add new Pair(_x,_y) to _posCanceled
    end loop
        Layout myLayout = new Layout(_name,row,col,cancelNum,_posCanceled)
        Display the current map. Ask if the client likes the layout
        opt = input from option textField
        if opt = 1 then
            Store the layout configuration in Accounts/<username>/<_name>.txt
            output "Your layout named "+<_name>.txt+" is successfully saved."
        else
            Restart this function
        end if


RandomSeating:
        username = current account's username
        Read the namelist from Accounts/+<username>+/namelist.csv. If the file is not found,
throws FileNotFoundException
        Add all names in namelist to students
        Add all names of layouts under the client's account in theNames
        Display theNames
        boolean ok = false
        loop while ok = false
            yourChoice = input from choice textField
            loop for each aName in theNames
                if yourChoice equals aName then
```

```
                    ok = true
                    break
            end loop
            output "This name not found! Input again!"
            Clear choice entry
    end loop
    Create layout from the choice
    Load map and numberOfStudents from layout
    if students's size not equals numberOfStudents then
            throw new Exception("Number of student does not match!")
    else
            mapNames = new String[layout's ROWS + 1][layout's COLUMNS + 1]
            ok = false
            filled = 0
ArrayList<String> seats = new ArrayList<String>();
HashMap<String,Pair> pos = new HashMap<String,Pair>();
HashMap<String,Integer> book = new HashMap<String,Integer>();
filled = 0
clear book
clear seats
clear pos
loop while filled < numberOfStudents
    randm = a random number in range [0,numberOfStudents - 1]
    if the randm'th student's name in book = null then
            add the randm'th student's name in book
            add the randm'th student's name in seats
            filled = filled + 1
    end if
end loop
Display the current map with student names
Export the map to seating.csv
    end if
```

Test Plan

These tests are necessary to prove that my program fulfills all the success criteria.

Action to test	Method of testing	Expected result
The client can set up the configuration of a new seating plan, including row and column number and the empty positions.	Click “Options”, then click “Create a new layout”, then enter the name, row number, column number, and positions to cancel.	A map with user-entered data is displayed on the GUI. The user can then choose whether to accept or decline the layout.

The client can select different seating scheme, such as random assignment, or moving by independent rows/columns.	Click “Options”, then click “Start assigning seats”. Multiple seating scheme is provided on the frame popped out.	The user can select which seating scheme to adopt.
The client can choose the type of layout used in the seating plan.	Click “Options”, then click “Start assigning seats” and select a scheme.	All layouts the client created previously will be displayed as options in the GUI.
The program can export the plan to a .csv file.	Click “Options”, then click “Start assigning seats” and select a scheme. Select a layout to be used.	A new file seating.csv is generated under Accounts/<username>/ directory.
The client can adjust the seating plan manually by directly editing the produced .csv file.	Go to Accounts/<username>/ directory, and open seating.csv.	The client can edit the contents in seating.csv. The client can also save changes.
The program can display the seating plan on a GUI.	Click “Options”, then click “Start assigning seats” and select a scheme. Select a layout to be used.	A frame will pop out, displaying the seating plan with student names and their positions.

(424 words)