Criterion B: Design Overview & Test Plan

[x] Appendix (Meeting) index.

Table of Contents

CRITERION B: DESIGN OVERVIEW & TEST PLAN 1

DESIGN OVERVIEW: 2

SOFTWARE INTERFACE: 2
PROGRAM PROCESS FLOW CHARTS: 9
- *Log-In/Sign-In 9*
- *Menu: 9*
- *In case the user chooses (Study Groups): 10*
- *In case the user chooses (Tutoring): 11*
- *In case the user chooses (Available Rooms): 12*

UMLS: 13
- *(UML) Controller Classes 13*
- *(UML) GUI Classes 14*

DATABASE DICTIONARY: 16
- *Tables 16*

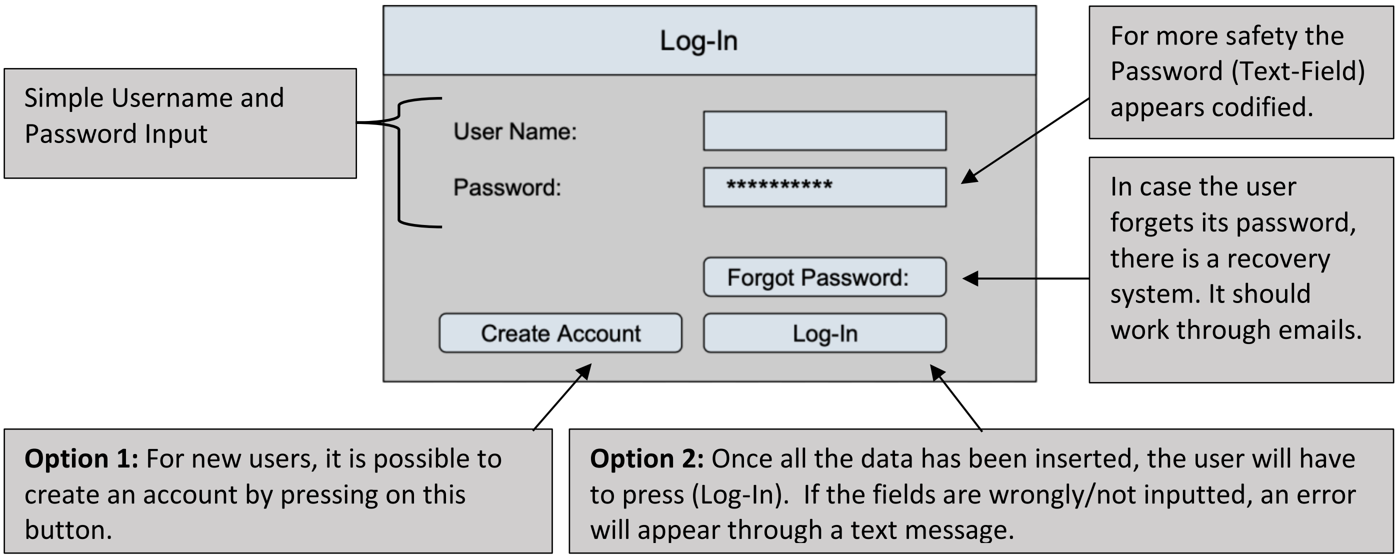
CLASSES: 18
MAIN ALGORITHMS: 19
TEST PLAN 22

Design Overview:

Software interface:

All designs are based on specific client requests - Refer to Appendix [A.1] & [B.2]

Design ***Diagram 1.***



Simple Username and Password Input

For more safety the Password (Text-Field) appears codified.

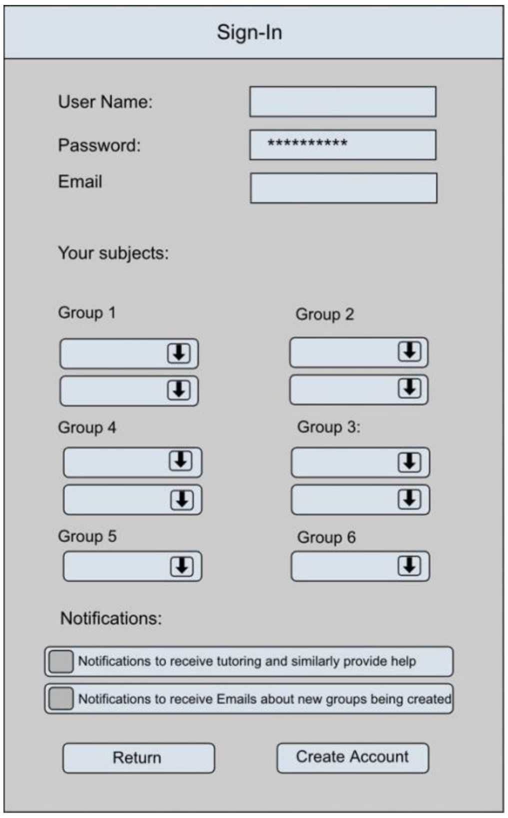
In case the user forgets its password, there is a recovery system. It should work through emails.

Option 1: For new users, it is possible to create an account by pressing on this button.

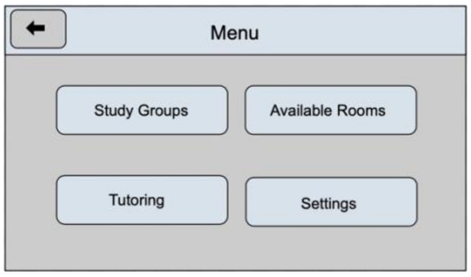
Option 2: Once all the data has been inserted, the user will have to press (Log-In). If the fields are wrongly/not inputted, an error will appear through a text message.

Option 1: Button (Create Account); leads to design **diagram 2** GUI
Option 2: Button (Log-In); leads to desing **diagram 3** GUI

Design ***Diagram 2***

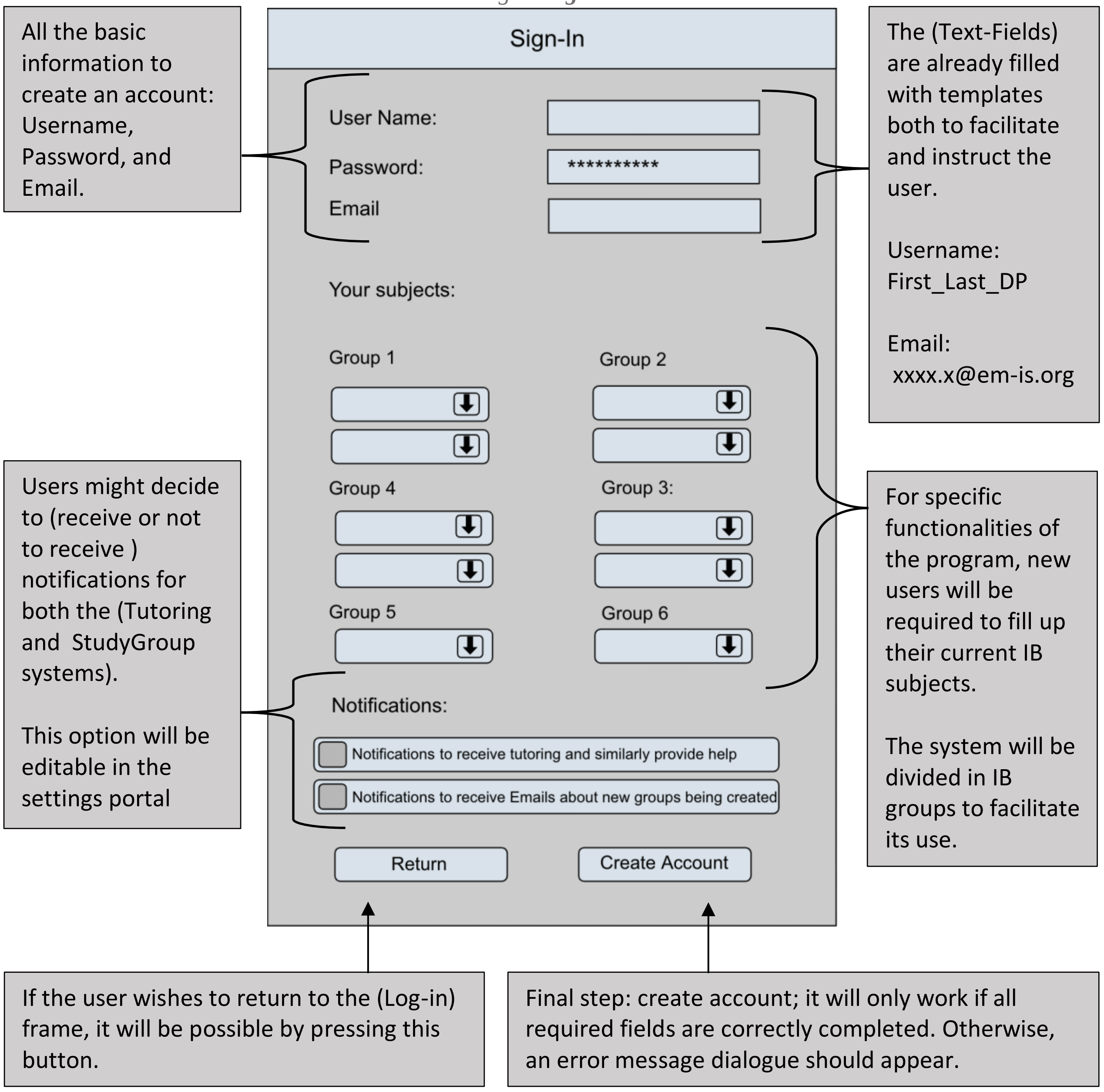


Design ***Diagram 3***



Option 1: Create Account

Design ***Diagram 2***



All the basic information to create an account: Username, Password, and Email.

The (Text-Fields) are already filled with templates both to facilitate and instruct the user.

Username: First_Last_DP

Email: xxxx.x@em-is.org

Users might decide to (receive or not to receive) notifications for both the (Tutoring and StudyGroup systems).

This option will be editable in the settings portal

For specific functionalities of the program, new users will be required to fill up their current IB subjects.

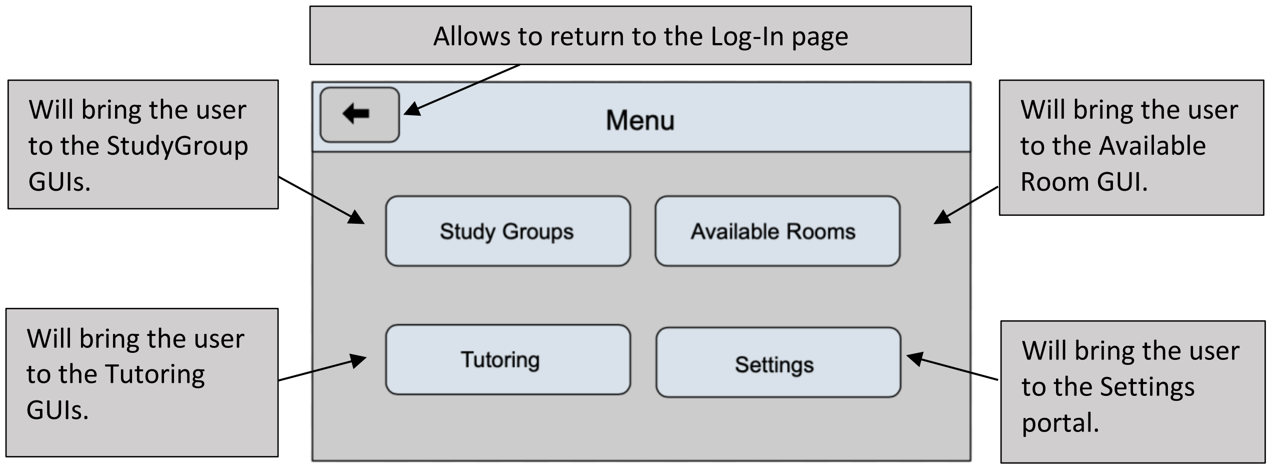
The system will be divided in IB groups to facilitate its use.

If the user wishes to return to the (Log-in) frame, it will be possible by pressing this button.

Final step: create account; it will only work if all required fields are correctly completed. Otherwise, an error message dialogue should appear.

Option 2: Log-In

Design ***Diagram 3***



Allows to return to the Log-In page

Will bring the user to the StudyGroup GUIs.

Will bring the user to the Available Room GUI.

Will bring the user to the Tutoring GUIs.

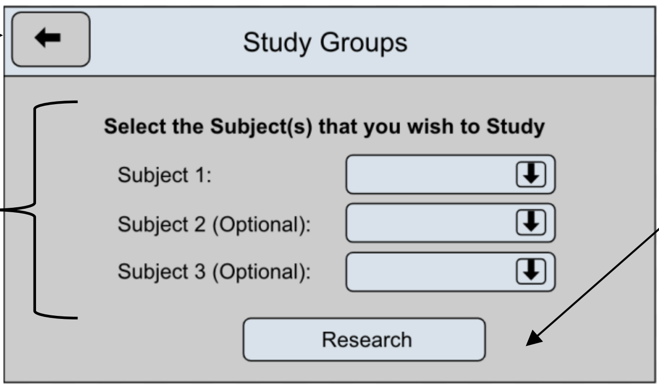
Will bring the user to the Settings portal.

Option 2.1: Button (Return); leads to diagram **1 GUI**
Option 2.2: Button (Study Groups); leads to diagram **4 GUI**
Option 2.3: Button (Tutoring); leads to diagram **7 GUI**
Option 2.4: Button (Settings); leads to diagram **11 GUI**
Option 2.5: Button (Available Rooms); leads to diagram **16 GUI**

Option 2.2: StudyGroups

*Design **Diagram 4***

Return to Menu
+
The user will be able to input the subjects(s) that he/she wishes to study:

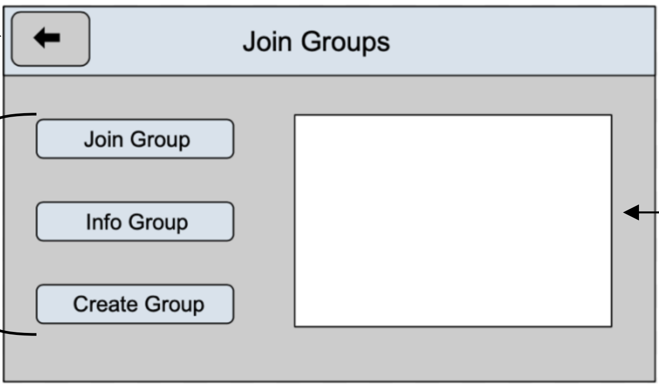


Button: The program will make a database search for an existing group with the required subject(s).

If the program finds a group already created with one of the subjects inserted it will display (Diagram 5) while on the other side if no groups were found (Diagram 6) will be displayed.

*Design **Diagram 5***

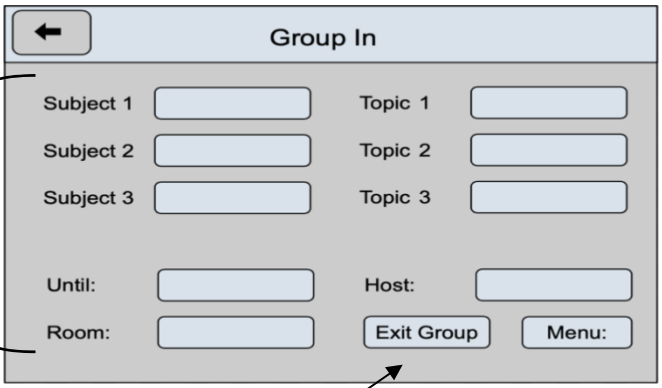
Return to diagram 4
+
Users will be able to use any of those buttons, providing different features.



List to display all StudyGroups found in the Database.

*Design **Diagram 6***

Return to Menu
+
The GUI should display all the relevant information of the group a user is currently in.



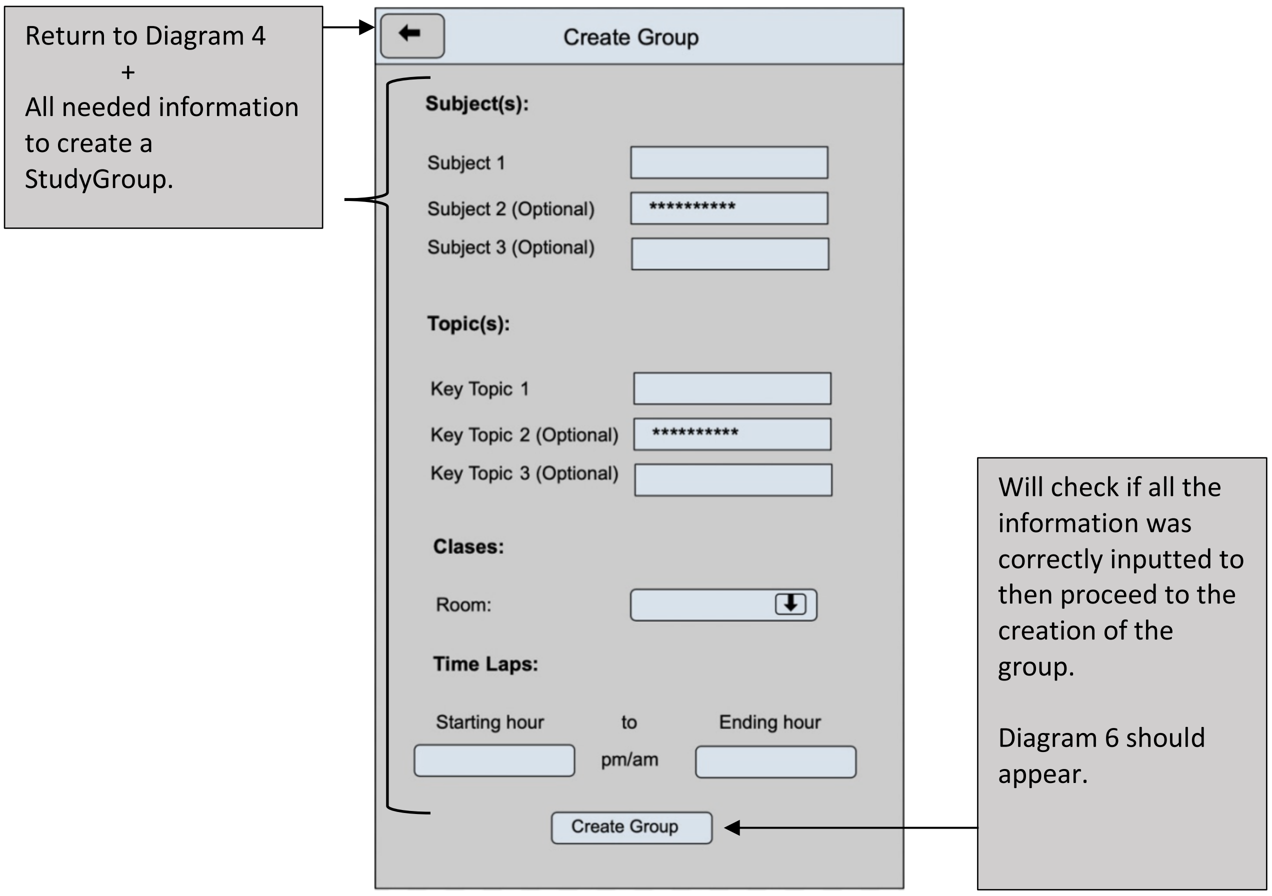
Menu: Will return to the Menu GUI.

Exit Group: Will allow the user to completely exit the group he/she is in.

Exit Group: If the user is a host the entire Group will be deleted for all members.

If a StudyGroup is not found (Diagram 4), or the user wishes to create a new StudyGroup (Diagram 5), the (Diagram 7) will be displayed:

*Design **Diagram 7***

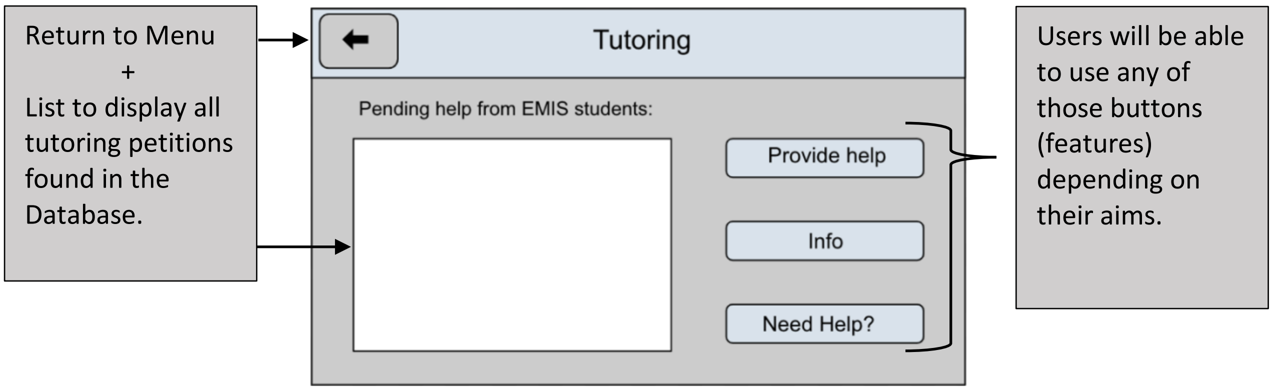

Return to Diagram 4
+
All needed information to create a StudyGroup.

Will check if all the information was correctly inputted to then proceed to the creation of the group.

Diagram 6 should appear.

Option 2.3: Tutoring

*Design **Diagram 8***

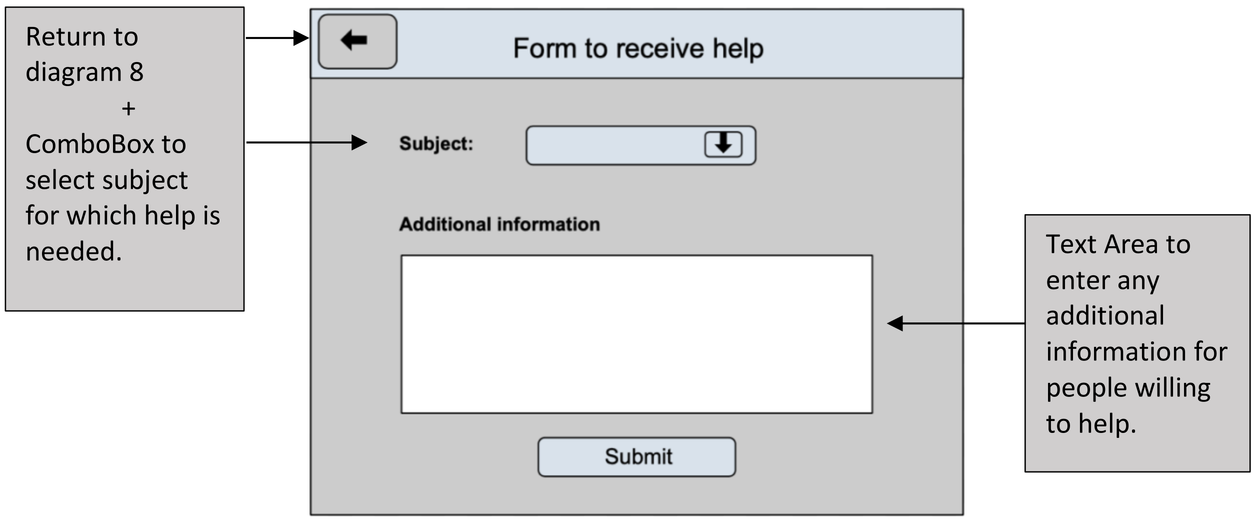

Return to Menu
+
List to display all tutoring petitions found in the Database.

Users will be able to use any of those buttons (features) depending on their aims.

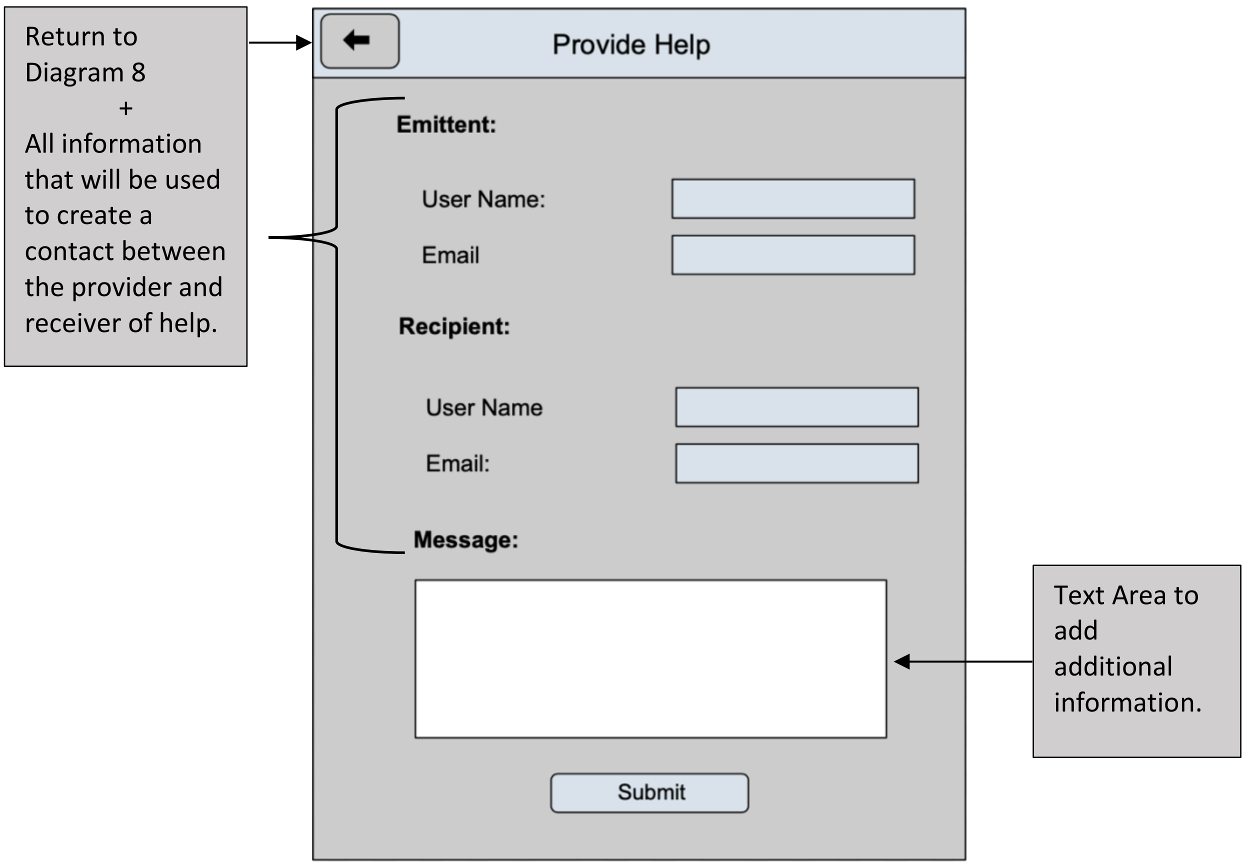
- User is facing **four** GUI options:

- If a user wishes to receive help, **(diagram 9)** will be displayed:
- If a user wishes to get info about a Tutoring petition, a message dialogue will be shown.
- If a user wishes to provide help, **(diagram 10)** will be displayed:

*Design **Diagram 9***

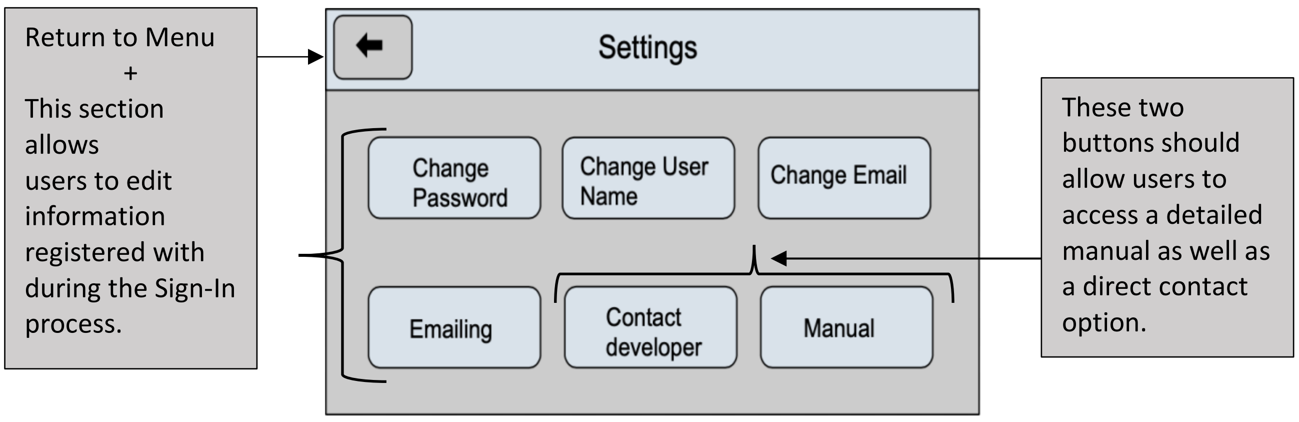

Return to diagram 8
+
ComboBox to select subject for which help is needed.
Form to receive help
Subject:
Additional information
Submit
Text Area to enter any additional information for people willing to help.

Design Diagram 10

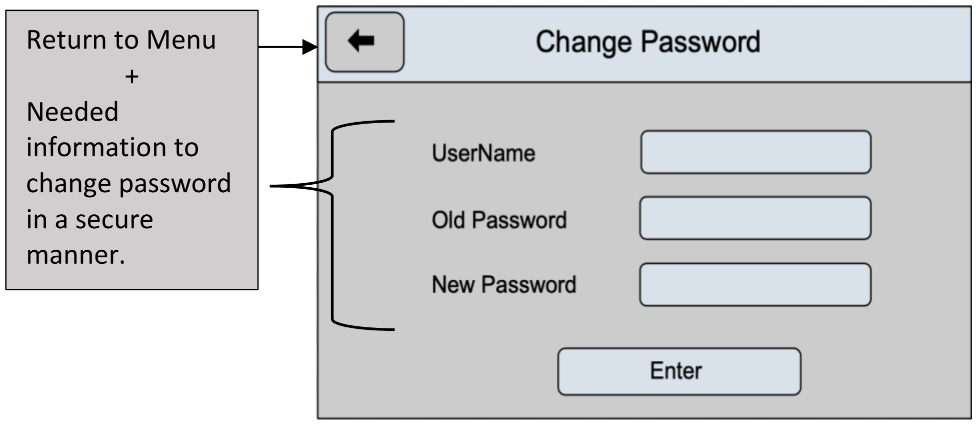

Return to Diagram 8
+
All information that will be used to create a contact between the provider and receiver of help.
Provide Help
Emittent:
User Name:
Email
Recipient:
User Name
Email:
Message:
Submit
Text Area to add additional information.

Option 2.4: Settings

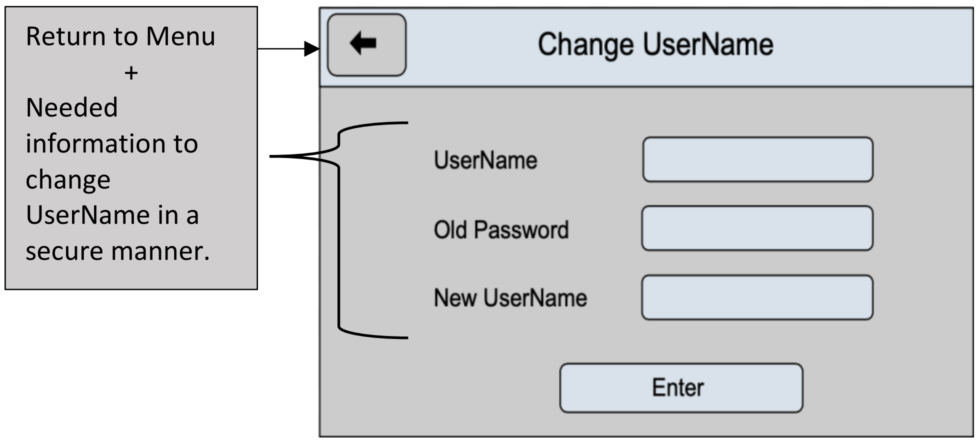
Design Diagram 11


Return to Menu
+
This section allows users to edit information registered with during the Sign-In process.
Settings
Change Password
Change User Name
Change Email
Emailing
Contact developer
Manual
These two buttons should allow users to access a detailed manual as well as a direct contact option.

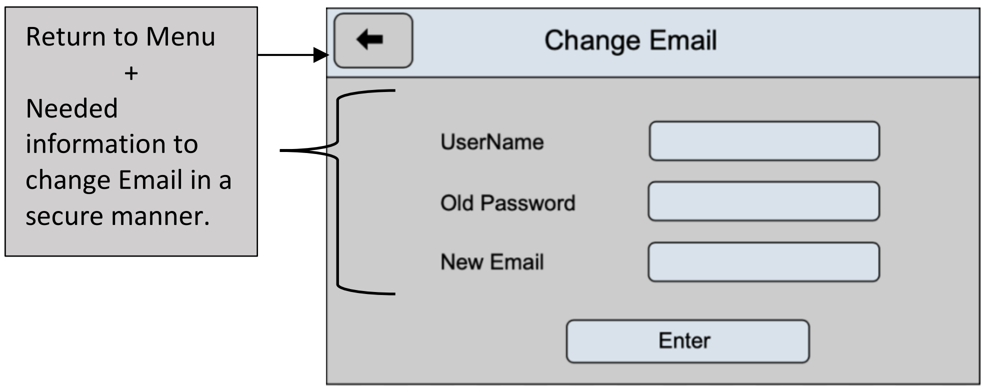
***Design Diagram 12** Button: (Change Password)*



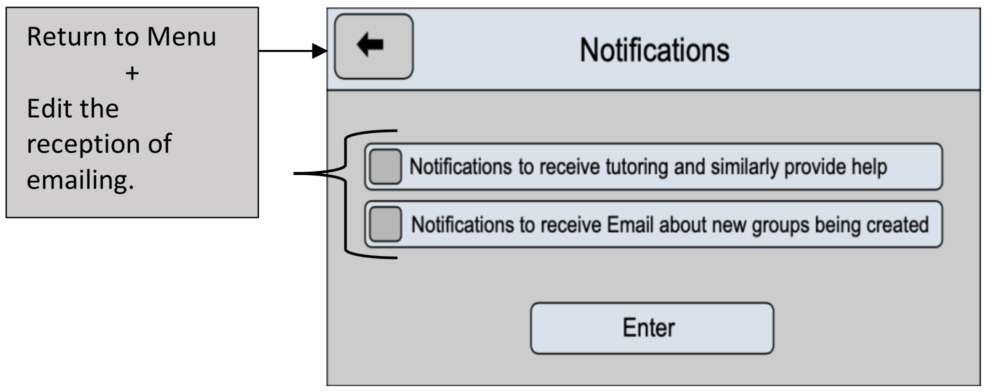
***Design Diagram 13** Button: (Change UserName)*



***Design Diagram 14** Button: (Change Email)*

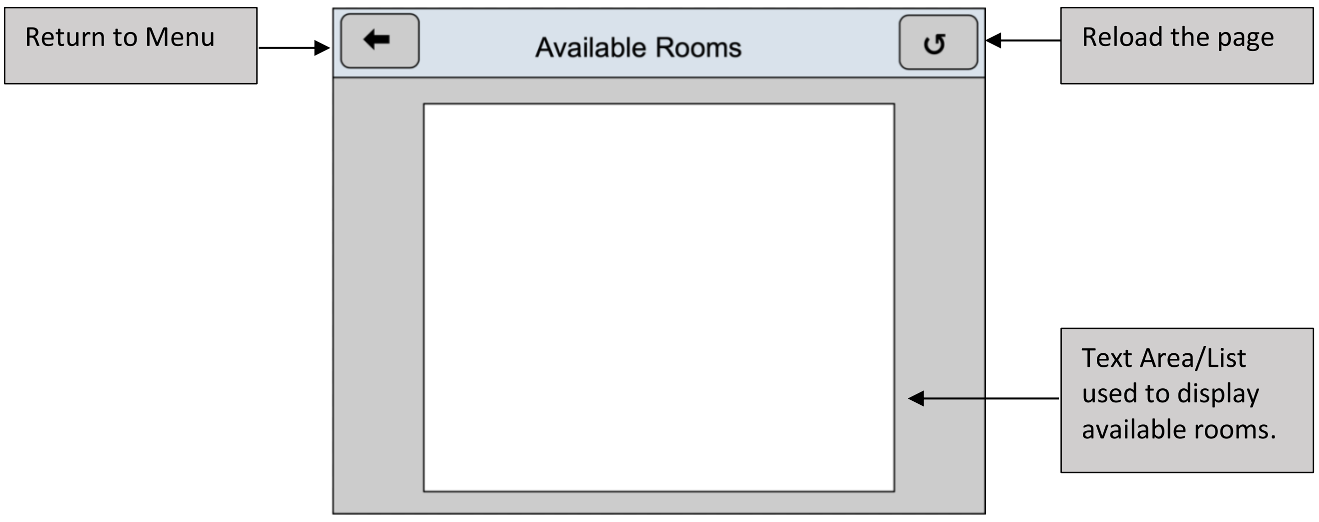


Design Diagram 15 *Button: (Emailing)*


Return to Menu
+
Edit the reception of emailing.
Notifications
Notifications to receive tutoring and similarly provide help
Notifications to receive Email about new groups being created
Enter

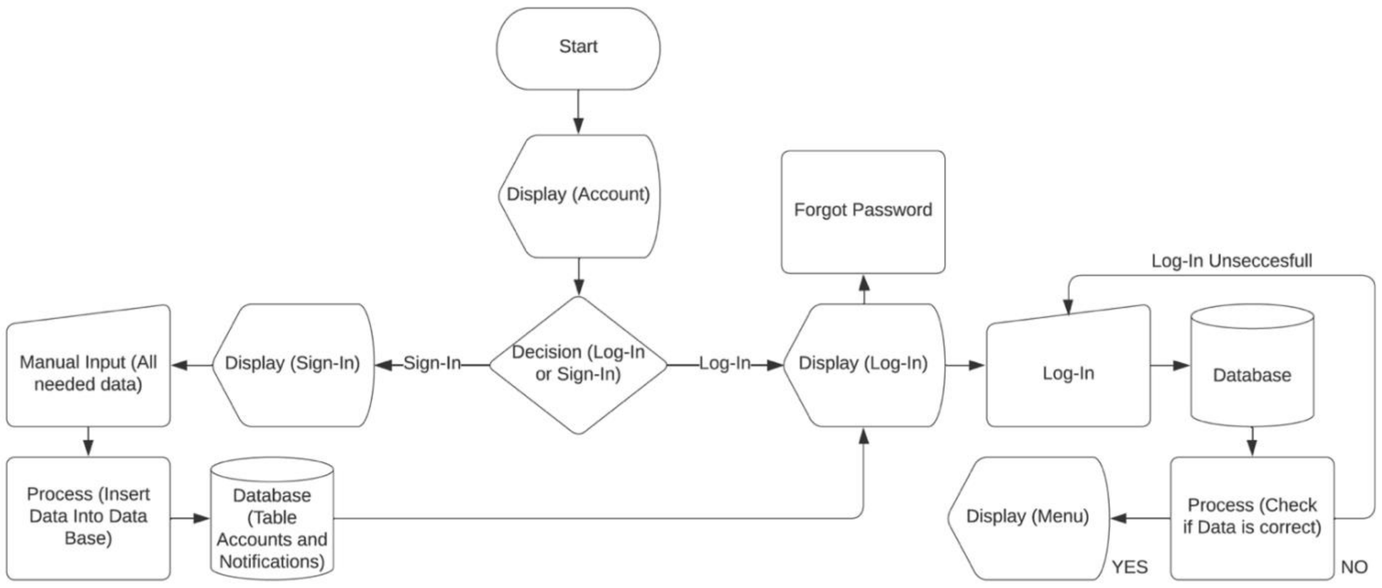
Option 2.3: Available Rooms

Design Diagram 16



Program process flow charts:

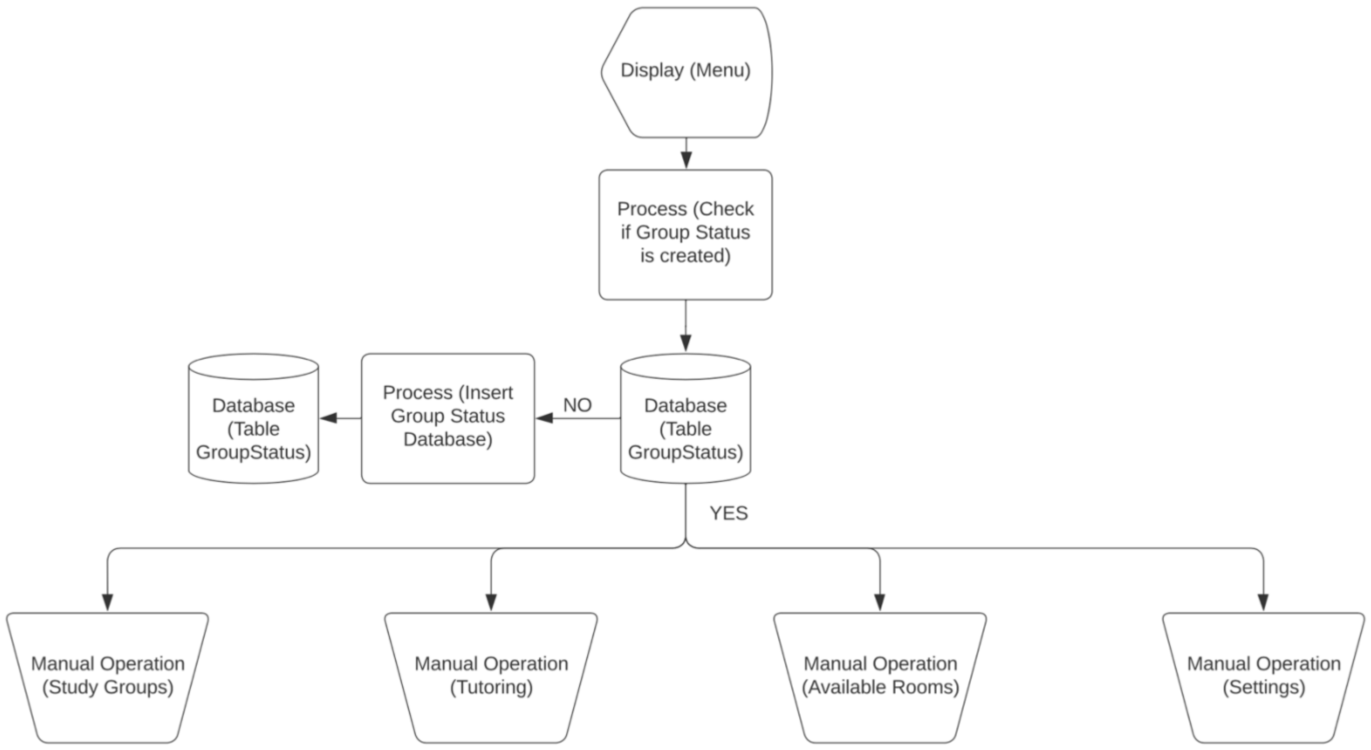
- Log-In/Sign-In



Start
Display (Account)
Decision (Log-In or Sign-In)
Sign-In
Log-In
Display (Sign-In)
Manual Input (All needed data)
Process (Insert Data Into Data Base)
Database (Table Accounts and Notifications)
Display (Log-In)
Forgot Password
Log-In
Log-In Unseccesfull
Database
Process (Check if Data is correct)
Display (Menu)
YES
NO

Flow chart 1. Log-In, Sign-In, and Menu entry.

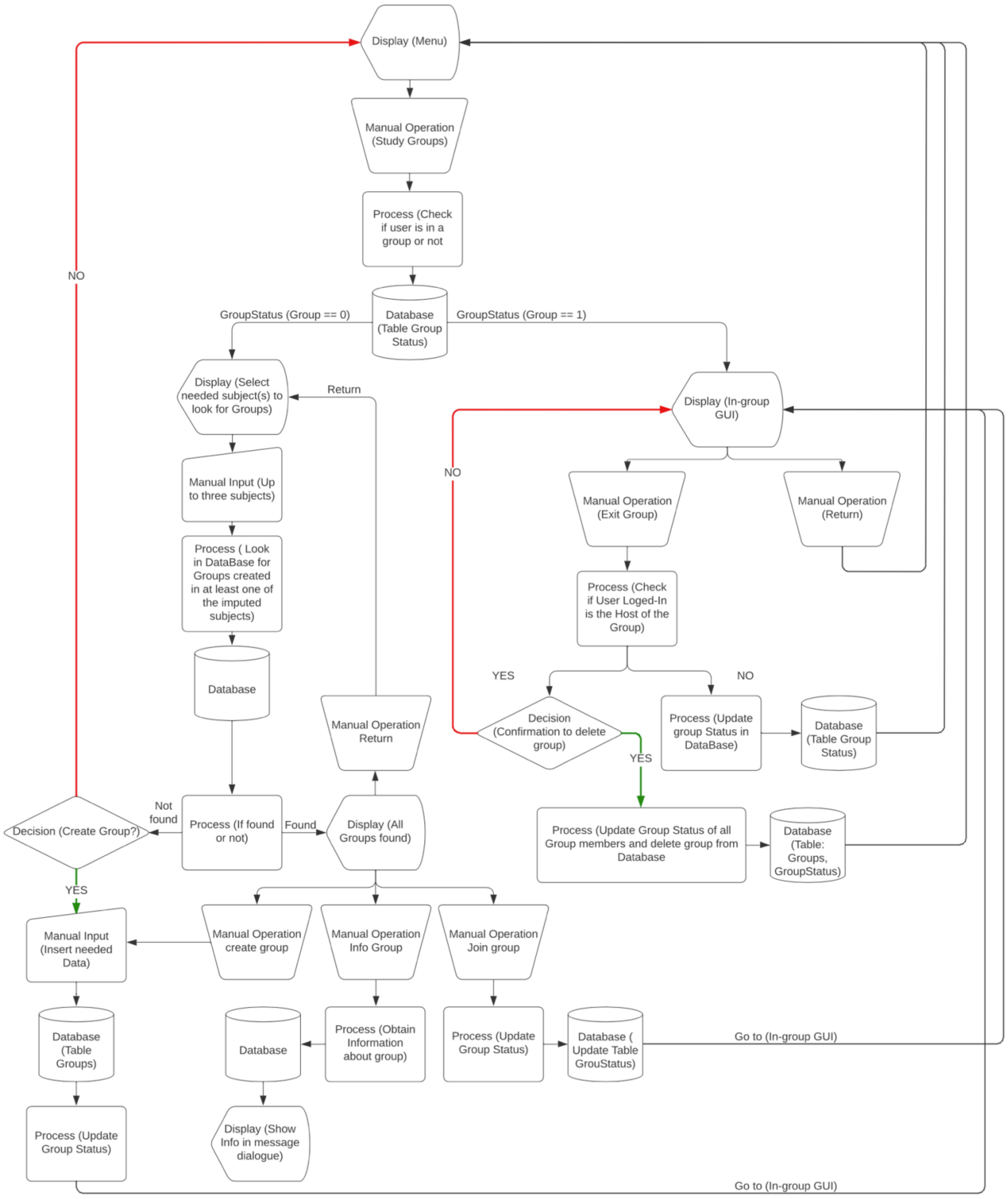
- Menu:



Display (Menu)
Process (Check if Group Status is created)
Database (Table GroupStatus)
NO
Process (Insert Group Status Database)
Database (Table GroupStatus)
YES
Manual Operation (Study Groups)
Manual Operation (Tutoring)
Manual Operation (Available Rooms)
Manual Operation (Settings)

Flow chart 2. Menu entry, Group Status Init Check, Menu Options.

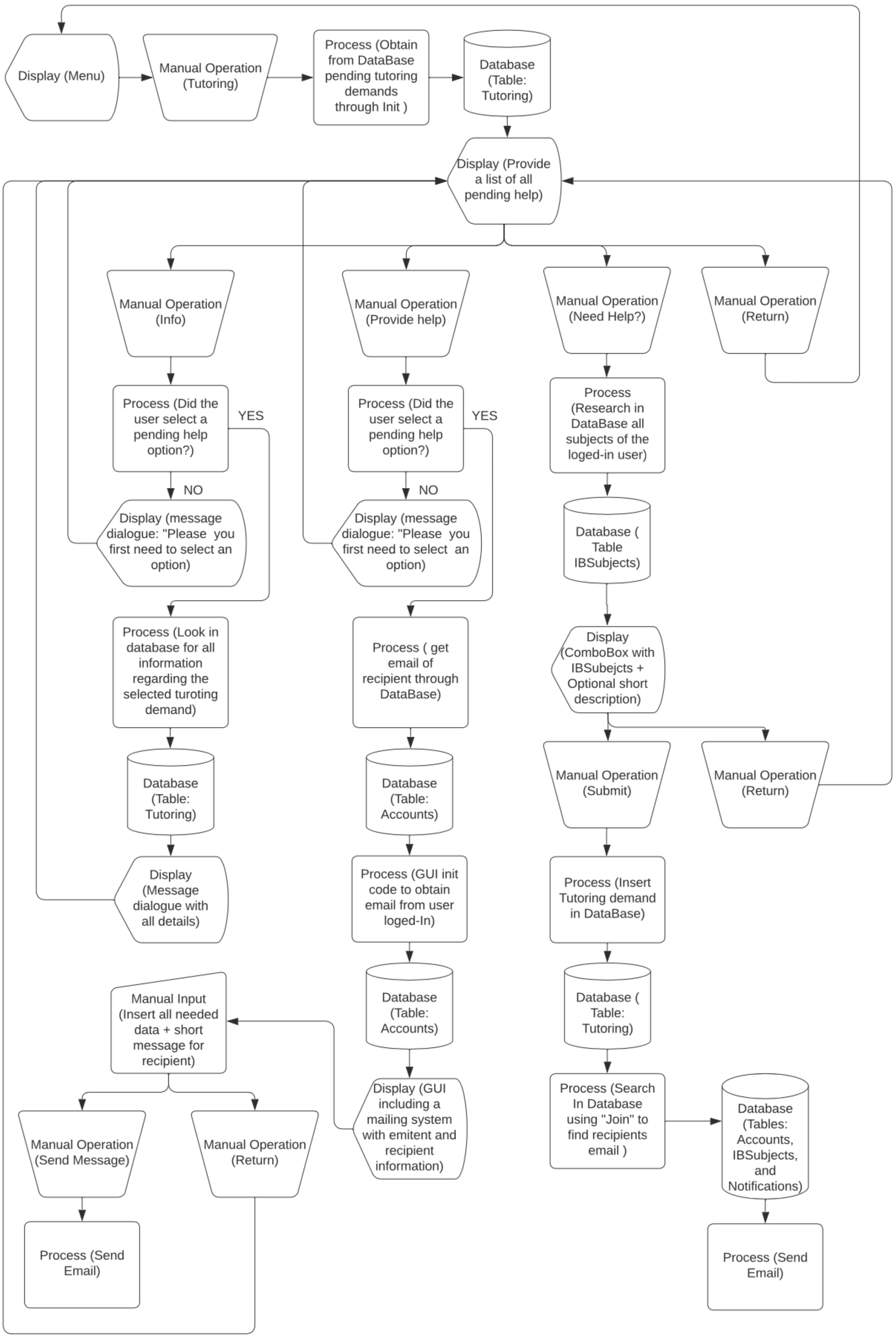
- In case the user chooses (Study Groups):



Display (Menu)
Manual Operation (Study Groups)
Process (Check if user is in a group or not
Database (Table Group Status)
GroupStatus (Group == 0)
GroupStatus (Group == 1)
NO
Display (Select needed subject(s) to look for Groups)
Return
Manual Input (Up to three subjects)
Process (Look in DataBase for Groups created in at least one of the imputed subjects)
Database
Manual Operation Return
Display (In-group GUI)
NO
Manual Operation (Exit Group)
Manual Operation (Return)
Process (Check if User Loged-In is the Host of the Group)
YES
NO
Decision (Confirmation to delete group)
YES
Process (Update group Status in DataBase)
Database (Table Group Status)
Process (Update Group Status of all Group members and delete group from Database
Database (Table: Groups, GroupStatus)
Decision (Create Group?)
Not found
Process (If found or not)
Found
Display (All Groups found)
YES
Manual Input (Insert needed Data)
Manual Operation create group
Manual Operation Info Group
Manual Operation Join group
Database (Table Groups)
Database
Process (Obtain Information about group)
Process (Update Group Status)
Database (Update Table GrouStatus)
Go to (In-group GUI)
Process (Update Group Status)
Display (Show Info in message dialogue)
Go to (In-group GUI)

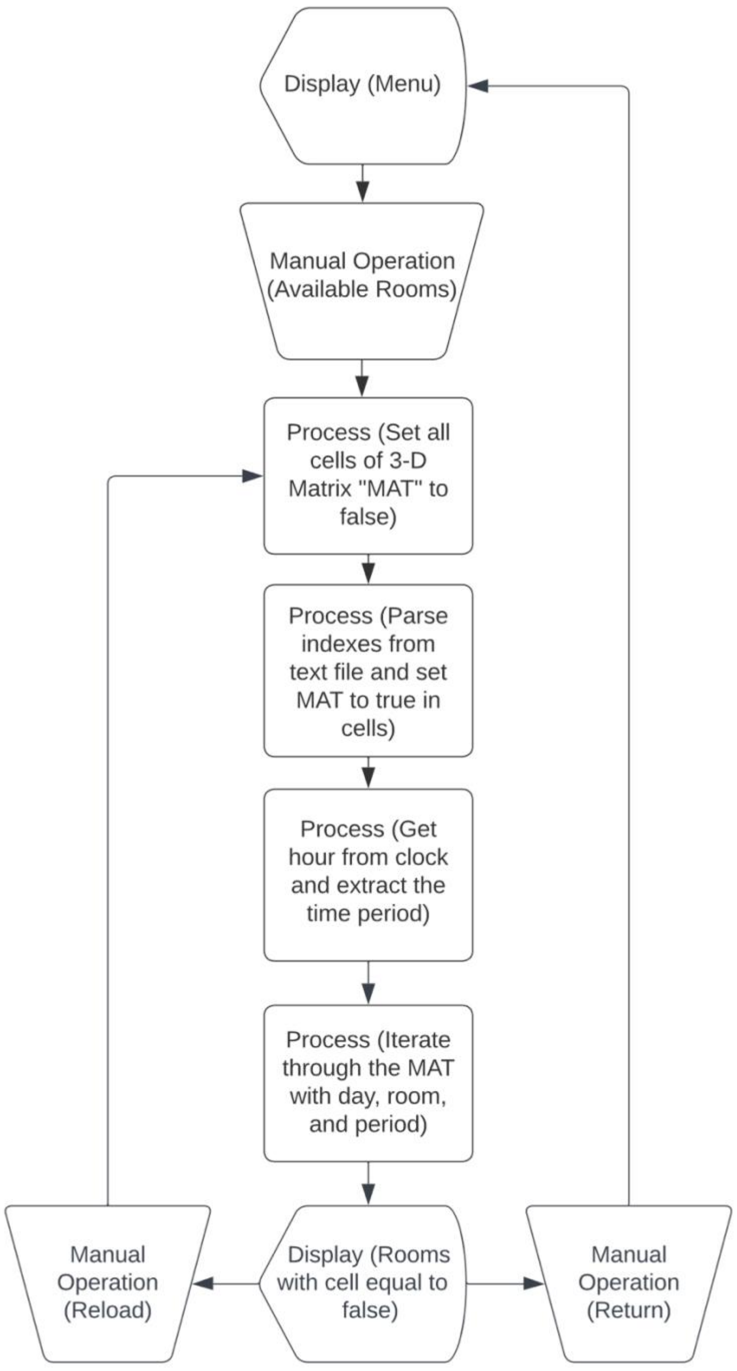
Flow chart 2.1. Study groups.

- In case the user chooses (Tutoring):


Display (Menu)
Manual Operation (Tutoring)
Process (Obtain from DataBase pending tutoring demands through Init)
Database (Table: Tutoring)
Display (Provide a list of all pending help)
Manual Operation (Info)
Manual Operation (Provide help)
Manual Operation (Need Help?)
Manual Operation (Return)
Process (Did the user select a pending help option?)
YES
NO
Display (message dialogue: "Please you first need to select an option)
Process (Did the user select a pending help option?)
YES
NO
Display (message dialogue: "Please you first need to select an option)
Process (Research in DataBase all subjects of the loged-in user)
Database (Table IBSubjects)
Display (ComboBox with IBSubejcts + Optional short description)
Process (Look in database for all information regarding the selected turoting demand)
Database (Table: Tutoring)
Display (Message dialogue with all details)
Process (get email of recipient through DataBase)
Database (Table: Accounts)
Process (GUI init code to obtain email from user loged-In)
Database (Table: Accounts)
Display (GUI including a mailing system with emitent and recipient information)
Manual Operation (Submit)
Manual Operation (Return)
Process (Insert Tutoring demand in DataBase)
Database (Table: Tutoring)
Process (Search In Database using "Join" to find recipients email)
Database (Tables: Accounts, IBSubjects, and Notifications)
Process (Send Email)
Manual Input (Insert all needed data + short message for recipient)
Manual Operation (Send Message)
Manual Operation (Return)
Process (Send Email)

Flow chart 2.2. Tutoring.

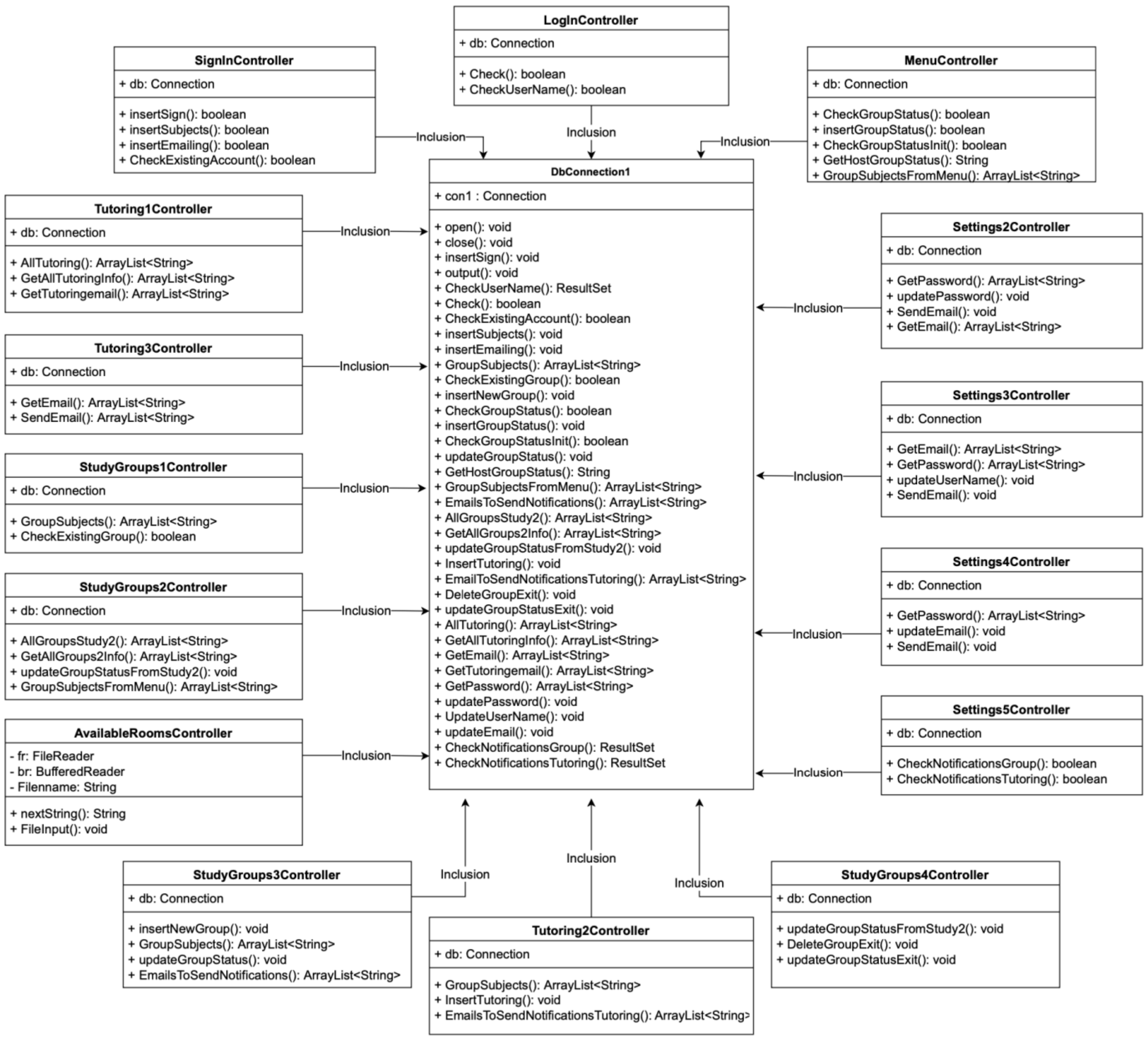
- In case the user chooses (Available Rooms):


Display (Menu)
Manual Operation (Available Rooms)
Process (Set all cells of 3-D Matrix "MAT" to false)
Process (Parse indexes from text file and set MAT to true in cells)
Process (Get hour from clock and extract the time period)
Process (Iterate through the MAT with day, room, and period)
Manual Operation (Reload)
Display (Rooms with cell equal to false)
Manual Operation (Return)

Flow chart 2.3. Available Rooms.

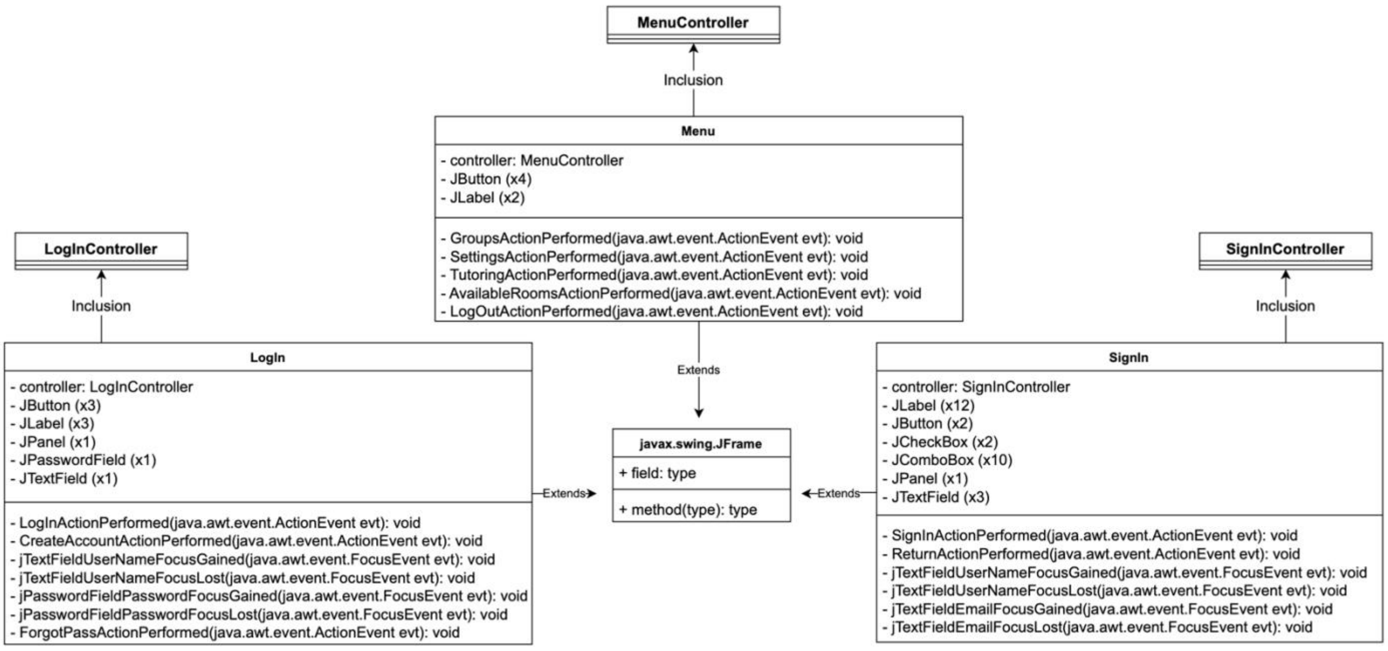
UMLs:

- (UML) Controller Classes

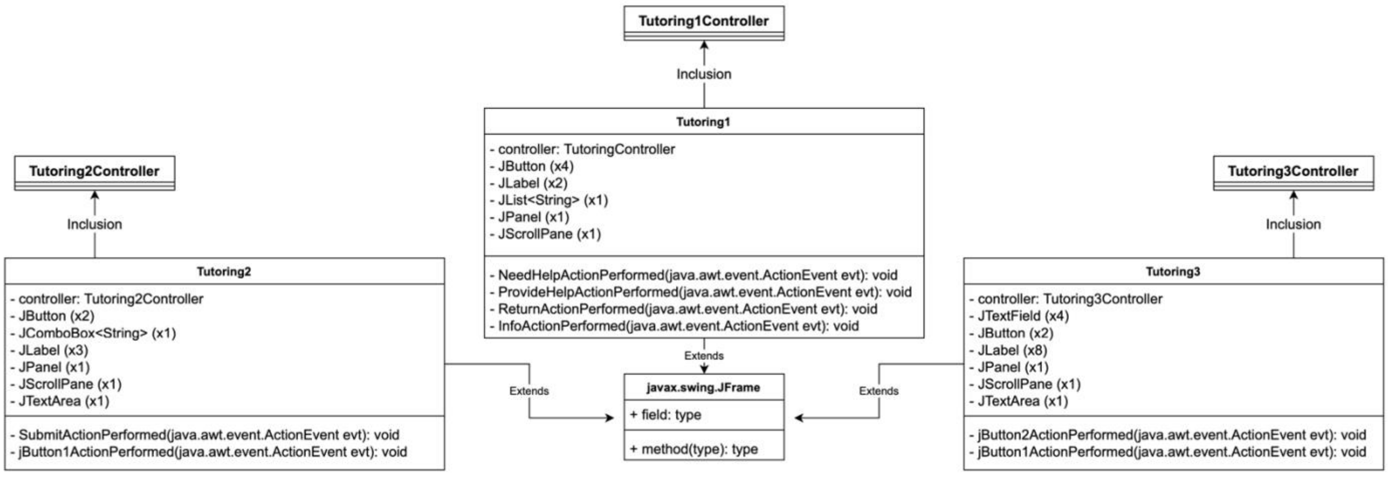

LogInController
+ db: Connection
+ Check(): boolean
+ CheckUserName(): boolean
SignInController
+ db: Connection
+ insertSign(): boolean
+ insertSubjects(): boolean
+ insertEmailing(): boolean
+ CheckExistingAccount(): boolean
MenuController
+ db: Connection
+ CheckGroupStatus(): boolean
+ insertGroupStatus(): boolean
+ CheckGroupStatusInit(): boolean
+ GetHostGroupStatus(): String
+ GroupSubjectsFromMenu(): ArrayList<String>
DbConnection1
+ con1 : Connection
+ open(): void
+ close(): void
+ insertSign(): void
+ output(): void
+ CheckUserName(): ResultSet
+ Check(): boolean
+ CheckExistingAccount(): boolean
+ insertSubjects(): void
+ insertEmailing(): void
+ GroupSubjects(): ArrayList<String>
+ CheckExistingGroup(): boolean
+ insertNewGroup(): void
+ CheckGroupStatus(): boolean
+ insertGroupStatus(): void
+ CheckGroupStatusInit(): boolean
+ updateGroupStatus(): void
+ GetHostGroupStatus(): String
+ GroupSubjectsFromMenu(): ArrayList<String>
+ EmailsToSendNotifications(): ArrayList<String>
+ AllGroupsStudy2(): ArrayList<String>
+ GetAllGroups2Info(): ArrayList<String>
+ updateGroupStatusFromStudy2(): void
+ InsertTutoring(): void
+ EmailToSendNotificationsTutoring(): ArrayList<String>
+ DeleteGroupExit(): void
+ updateGroupStatusExit(): void
+ AllTutoring(): ArrayList<String>
+ GetAllTutoringInfo(): ArrayList<String>
+ GetEmail(): ArrayList<String>
+ GetTutoringemail(): ArrayList<String>
+ GetPassword(): ArrayList<String>
+ updatePassword(): void
+ UpdateUserName(): void
+ updateEmail(): void
+ CheckNotificationsGroup(): ResultSet
+ CheckNotificationsTutoring(): ResultSet
Tutoring1Controller
+ db: Connection
+ AllTutoring(): ArrayList<String>
+ GetAllTutoringInfo(): ArrayList<String>
+ GetTutoringemail(): ArrayList<String>
Tutoring3Controller
+ db: Connection
+ GetEmail(): ArrayList<String>
+ SendEmail(): ArrayList<String>
StudyGroups1Controller
+ db: Connection
+ GroupSubjects(): ArrayList<String>
+ CheckExistingGroup(): boolean
StudyGroups2Controller
+ db: Connection
+ AllGroupsStudy2(): ArrayList<String>
+ GetAllGroups2Info(): ArrayList<String>
+ updateGroupStatusFromStudy2(): void
+ GroupSubjectsFromMenu(): ArrayList<String>
AvailableRoomsController
- fr: FileReader
- br: BufferedReader
- Filename: String
+ nextString(): String
+ FileInput(): void
Settings2Controller
+ db: Connection
+ GetPassword(): ArrayList<String>
+ updatePassword(): void
+ SendEmail(): void
+ GetEmail(): ArrayList<String>
Settings3Controller
+ db: Connection
+ GetEmail(): ArrayList<String>
+ GetPassword(): ArrayList<String>
+ updateUserName(): void
+ SendEmail(): void
Settings4Controller
+ db: Connection
+ GetPassword(): ArrayList<String>
+ updateEmail(): void
+ SendEmail(): void
Settings5Controller
+ db: Connection
+ CheckNotificationsGroup(): boolean
+ CheckNotificationsTutoring(): boolean
StudyGroups3Controller
+ db: Connection
+ insertNewGroup(): void
+ GroupSubjects(): ArrayList<String>
+ updateGroupStatus(): void
+ EmailsToSendNotifications(): ArrayList<String>
Tutoring2Controller
+ db: Connection
+ GroupSubjects(): ArrayList<String>
+ InsertTutoring(): void
+ EmailsToSendNotificationsTutoring(): ArrayList<String>
StudyGroups4Controller
+ db: Connection
+ updateGroupStatusFromStudy2(): void
+ DeleteGroupExit(): void
+ updateGroupStatusExit(): void
Inclusion

- (UML) GUI Classes

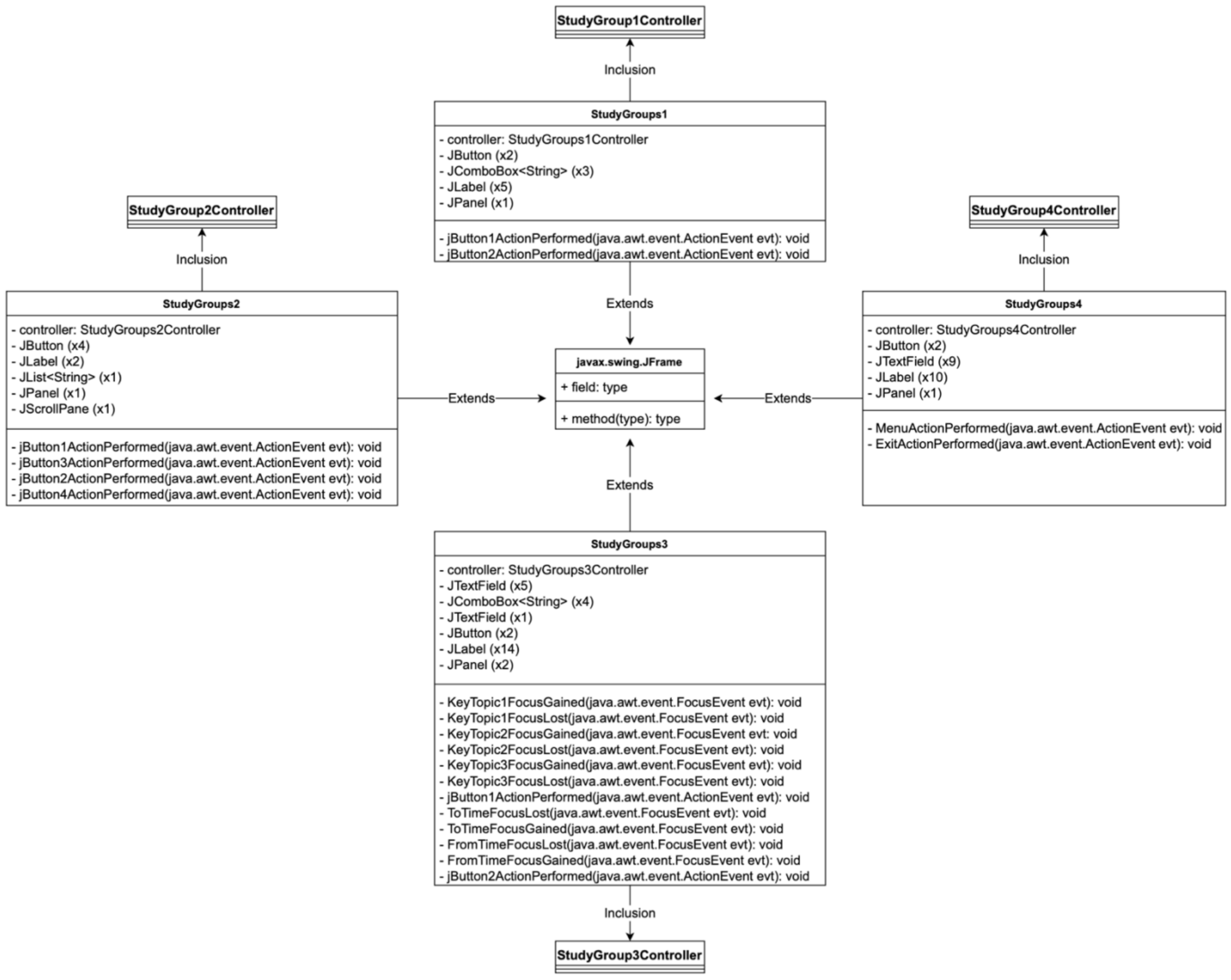
  - *(UML) Log-In, Sign-In, & Menu:*


MenuController
Inclusion
Menu
- controller: MenuController
- JButton (x4)
- JLabel (x2)
- GroupsActionPerformed(java.awt.event.ActionEvent evt): void
- SettingsActionPerformed(java.awt.event.ActionEvent evt): void
- TutoringActionPerformed(java.awt.event.ActionEvent evt): void
- AvailableRoomsActionPerformed(java.awt.event.ActionEvent evt): void
- LogOutActionPerformed(java.awt.event.ActionEvent evt): void
LogInController
Inclusion
LogIn
- controller: LogInController
- JButton (x3)
- JLabel (x3)
- JPanel (x1)
- JPasswordField (x1)
- JTextField (x1)
- LogInActionPerformed(java.awt.event.ActionEvent evt): void
- CreateAccountActionPerformed(java.awt.event.ActionEvent evt): void
- jTextFieldUserNameFocusGained(java.awt.event.FocusEvent evt): void
- jTextFieldUserNameFocusLost(java.awt.event.FocusEvent evt): void
- jPasswordFieldPasswordFocusGained(java.awt.event.FocusEvent evt): void
- jPasswordFieldPasswordFocusLost(java.awt.event.FocusEvent evt): void
- ForgotPassActionPerformed(java.awt.event.ActionEvent evt): void
Extends
Extends
javax.swing.JFrame
+ field: type
+ method(type): type
Extends
SignInController
Inclusion
SignIn
- controller: SignInController
- JLabel (x12)
- JButton (x2)
- JCheckBox (x2)
- JComboBox (x10)
- JPanel (x1)
- JTextField (x3)
- SignInActionPerformed(java.awt.event.ActionEvent evt): void
- ReturnActionPerformed(java.awt.event.ActionEvent evt): void
- jTextFieldUserNameFocusGained(java.awt.event.FocusEvent evt): void
- jTextFieldUserNameFocusLost(java.awt.event.FocusEvent evt): void
- jTextFieldEmailFocusGained(java.awt.event.FocusEvent evt): void
- jTextFieldEmailFocusLost(java.awt.event.FocusEvent evt): void

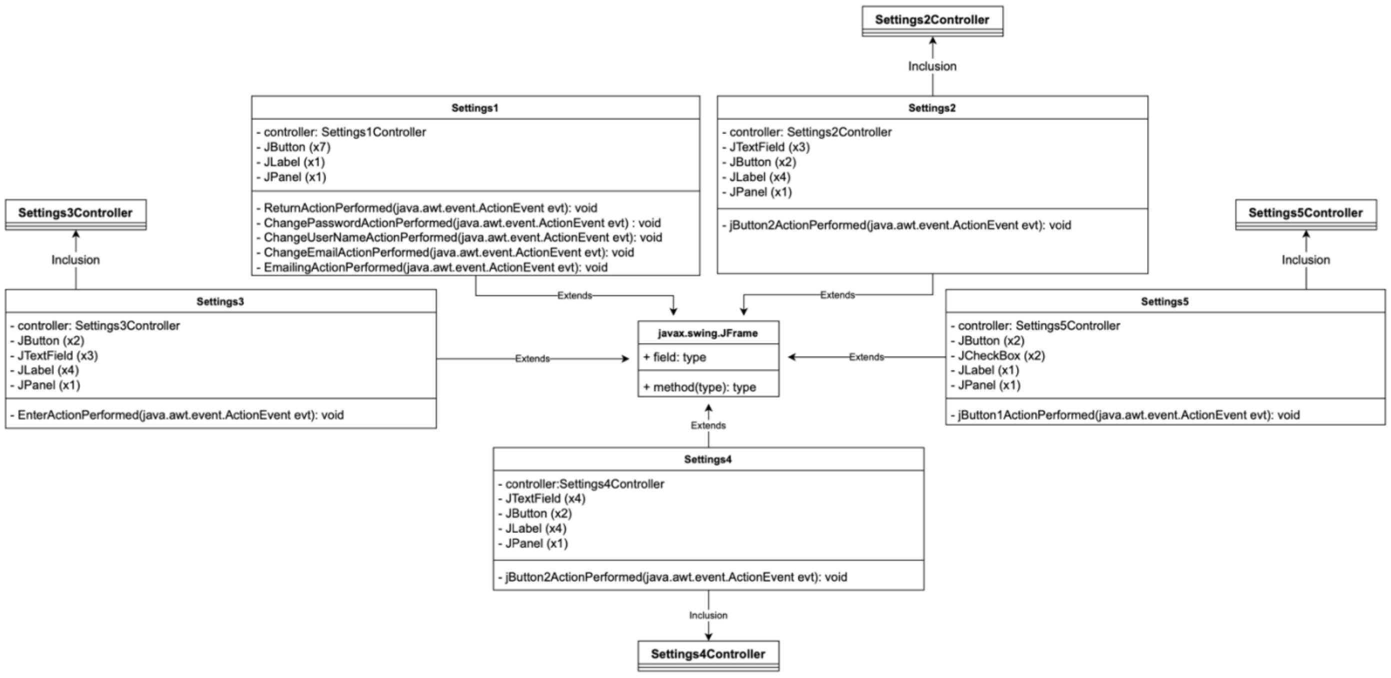
  - *(UML) Tutoring:*


Tutoring1Controller
Inclusion
Tutoring1
- controller: TutoringController
- JButton (x4)
- JLabel (x2)
- JList<String> (x1)
- JPanel (x1)
- JScrollPane (x1)
- NeedHelpActionPerformed(java.awt.event.ActionEvent evt): void
- ProvideHelpActionPerformed(java.awt.event.ActionEvent evt): void
- ReturnActionPerformed(java.awt.event.ActionEvent evt): void
- InfoActionPerformed(java.awt.event.ActionEvent evt): void
Tutoring2Controller
Inclusion
Tutoring2
- controller: Tutoring2Controller
- JButton (x2)
- JComboBox<String> (x1)
- JLabel (x3)
- JPanel (x1)
- JScrollPane (x1)
- JTextArea (x1)
- SubmitActionPerformed(java.awt.event.ActionEvent evt): void
- jButton1ActionPerformed(java.awt.event.ActionEvent evt): void
Extends
Extends
javax.swing.JFrame
+ field: type
+ method(type): type
Extends
Tutoring3Controller
Inclusion
Tutoring3
- controller: Tutoring3Controller
- JTextField (x4)
- JButton (x2)
- JLabel (x8)
- JPanel (x1)
- JScrollPane (x1)
- JTextArea (x1)
- jButton2ActionPerformed(java.awt.event.ActionEvent evt): void
- jButton1ActionPerformed(java.awt.event.ActionEvent evt): void

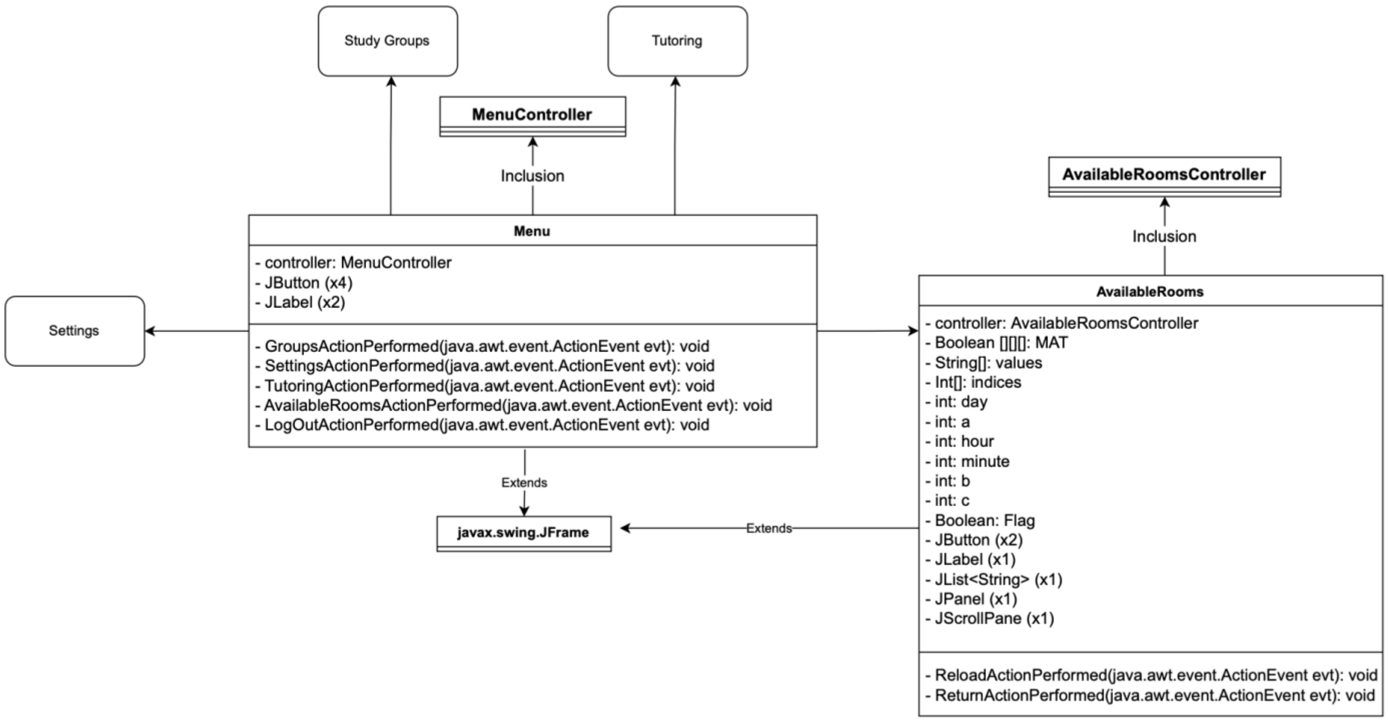
- *(UML) StudyGroups:*


StudyGroup1Controller
Inclusion
StudyGroups1
- controller: StudyGroups1Controller
- JButton (x2)
- JComboBox<String> (x3)
- JLabel (x5)
- JPanel (x1)
- jButton1ActionPerformed(java.awt.event.ActionEvent evt): void
- jButton2ActionPerformed(java.awt.event.ActionEvent evt): void
Extends
StudyGroup2Controller
Inclusion
StudyGroups2
- controller: StudyGroups2Controller
- JButton (x4)
- JLabel (x2)
- JList<String> (x1)
- JPanel (x1)
- JScrollPane (x1)
- jButton1ActionPerformed(java.awt.event.ActionEvent evt): void
- jButton3ActionPerformed(java.awt.event.ActionEvent evt): void
- jButton2ActionPerformed(java.awt.event.ActionEvent evt): void
- jButton4ActionPerformed(java.awt.event.ActionEvent evt): void
Extends
javax.swing.JFrame
+ field: type
+ method(type): type
Extends
StudyGroup4Controller
Inclusion
StudyGroups4
- controller: StudyGroups4Controller
- JButton (x2)
- JTextField (x9)
- JLabel (x10)
- JPanel (x1)
- MenuActionPerformed(java.awt.event.ActionEvent evt): void
- ExitActionPerformed(java.awt.event.ActionEvent evt): void
Extends
StudyGroups3
- controller: StudyGroups3Controller
- JTextField (x5)
- JComboBox<String> (x4)
- JTextField (x1)
- JButton (x2)
- JLabel (x14)
- JPanel (x2)
- KeyTopic1FocusGained(java.awt.event.FocusEvent evt): void
- KeyTopic1FocusLost(java.awt.event.FocusEvent evt): void
- KeyTopic2FocusGained(java.awt.event.FocusEvent evt: void
- KeyTopic2FocusLost(java.awt.event.FocusEvent evt): void
- KeyTopic3FocusGained(java.awt.event.FocusEvent evt): void
- KeyTopic3FocusLost(java.awt.event.FocusEvent evt): void
- jButton1ActionPerformed(java.awt.event.ActionEvent evt): void
- ToTimeFocusLost(java.awt.event.FocusEvent evt): void
- ToTimeFocusGained(java.awt.event.FocusEvent evt): void
- FromTimeFocusLost(java.awt.event.FocusEvent evt): void
- FromTimeFocusGained(java.awt.event.FocusEvent evt): void
- jButton2ActionPerformed(java.awt.event.ActionEvent evt): void
Inclusion
StudyGroup3Controller

- *UML: Settings:*


Settings2Controller
Inclusion
Settings1
- controller: Settings1Controller
- JButton (x7)
- JLabel (x1)
- JPanel (x1)
- ReturnActionPerformed(java.awt.event.ActionEvent evt): void
- ChangePasswordActionPerformed(java.awt.event.ActionEvent evt) : void
- ChangeUserNameActionPerformed(java.awt.event.ActionEvent evt): void
- ChangeEmailActionPerformed(java.awt.event.ActionEvent evt): void
- EmailingActionPerformed(java.awt.event.ActionEvent evt): void
Settings2
- controller: Settings2Controller
- JTextField (x3)
- JButton (x2)
- JLabel (x4)
- JPanel (x1)
- jButton2ActionPerformed(java.awt.event.ActionEvent evt): void
Settings3Controller
Inclusion
Settings5Controller
Inclusion
Extends
Extends
Settings3
- controller: Settings3Controller
- JButton (x2)
- JTextField (x3)
- JLabel (x4)
- JPanel (x1)
- EnterActionPerformed(java.awt.event.ActionEvent evt): void
Extends
javax.swing.JFrame
+ field: type
+ method(type): type
Extends
Settings5
- controller: Settings5Controller
- JButton (x2)
- JCheckBox (x2)
- JLabel (x1)
- JPanel (x1)
- jButton1ActionPerformed(java.awt.event.ActionEvent evt): void
Extends
Settings4
- controller:Settings4Controller
- JTextField (x4)
- JButton (x2)
- JLabel (x4)
- JPanel (x1)
- jButton2ActionPerformed(java.awt.event.ActionEvent evt): void
Inclusion
Settings4Controller

- *(UML) Available Rooms:*


Study Groups
Tutoring
MenuController
Inclusion
AvailableRoomsController
Inclusion
Menu
- controller: MenuController
- JButton (x4)
- JLabel (x2)
- GroupsActionPerformed(java.awt.event.ActionEvent evt): void
- SettingsActionPerformed(java.awt.event.ActionEvent evt): void
- TutoringActionPerformed(java.awt.event.ActionEvent evt): void
- AvailableRoomsActionPerformed(java.awt.event.ActionEvent evt): void
- LogOutActionPerformed(java.awt.event.ActionEvent evt): void
Settings
AvailableRooms
- controller: AvailableRoomsController
- Boolean [][][]: MAT
- String[]: values
- Int[]: indices
- int: day
- int: a
- int: hour
- int: minute
- int: b
- int: c
- Boolean: Flag
- JButton (x2)
- JLabel (x1)
- JList<String> (x1)
- JPanel (x1)
- JScrollPane (x1)
- ReloadActionPerformed(java.awt.event.ActionEvent evt): void
- ReturnActionPerformed(java.awt.event.ActionEvent evt): void
Extends
javax.swing.JFrame
Extends

Database Dictionary:

Storage System:	SQLite Databases
Number of Databases used:	1
Number of tables used:	6
Format Database:	.db
Database facilitator:	DB Browser

- Tables

Table 1. Data Dictionary for 'Accounts' table

Name	Type	Not null	Unique	Primary Key	Notes
UserName	Text	Not null	Unique	Primary key	Used as ID element & (Log-In)
Password	Text	Not null	Unique		Henhance security (Log-In)
Email	Text	Not null	Unique		For emailing and password recovery

Table 2. Data Dictionary for 'GroupStatus' table

Name	Type	Not null	Unique	Primary Key	Notes
UserName	Text	Not null	Unique		Used as ID element & (Log-In)
Status	Integer				Check if a specific user is in a group
Host	Text	Not null			Creator of study group

Table 3. Data Dictionary for 'Groups' table

Name	Type	Not null	Unique	Primary Key	Notes
UserName	Text	Not null	Unique		Used as ID element & (Log-In)
Subject1	Text				Study group data
Subject2	Text				Study group data
Subject3	Text				Study group data
Topic1	Text				Study group data
Topic2	Text				Study group data
Topic3	Text				Study group data
Room	Text				Study group data
FromT	Integer				Study group data
ToT	Integer				Study group data

Table 4. Data Dictionary for 'IBSubjects' table

Name	Type	Not null	Unique	Primary Key	Notes
UserName	Text	Not null	Unique	Primary key	Used as ID element & (Log-In)
Group1	Text	Not null			Subject of user
Group1Entra	Text				Subject of user
Group2	Text	Not null			Subject of user
Group2Extra	Text				Subject of user
Group 3	Text	Not null			Subject of user
Group 3Extra	Text				Subject of user
Group 4	Text	Not null			Subject of user
Subject4Extra	Text				Subject of user
Group5	Text	Not null			Subject of user
Group6	Text	Not null			Subject of user

Table 5. Data Dictionary for 'Notifications' table

Name	Type	Not null	Unique	Primary Key	Notes
UserName	Text	Not null	Unique	Primary key	Used as ID element & (Log-In)
Tutoring	Integer	Not null			Emailing for tutoring (user agreement)
Groups	Integer	Not null			Emailing for groups (user agreement)

Table 6. Data Dictionary for 'Tutoring' table

Name	Type	Not null	Unique	Primary Key	Notes
UserName	Text	Not null		Primary key	Used as ID element & (Log-In)
Subject	Text	Not null			Subject help petition storage
Description	Text				Store details of petition

Classes:

Class	Purpose
LogIn	Entry portal for users; Password retrieval; SignIn access.
LogInController	Stores the functions of the “LogIn” class.
SignIn	Allows the creation of new accounts.
SignInController	Stores the functions of the “SignIn” class.
Menu	Student portal that gives access to the project’s features.
MenuController	Stores the functions of the “Menu” class.
StudyGroups1	Allows to research for StudyGroups.
StudyGroups1Controller	Stores the functions of the “StudyGroup1” class.
StudyGroups2	Shows matching StudyGroups + Information.
StudyGroups2Controller	Stores the functions of the “StudyGroup2” class.
StudyGroups3	Allows the creation of new StudyGroups.
StudyGroups3Controller	Stores the functions of the “StudyGroup3” class.
StudyGroups4	Portal of a student when a StudyGroup is joined.
StudyGroups4Controller	Stores the functions of the “StudyGroup4” class.
Tutoring1	Portal that gives access to all the tutoring petitions.
Tutoring1Controller	Stores the functions of the “Tutoring1” class.
Tutoring2	Form to create a tutoring petition.
Tutoring2Controller	Stores the functions of the “Tutoring2” class.
Tutoring3	Form to provide help to a tutoring petition.
Tutoring3Controller	Stores the functions of the “Tutoring3” class.
Settings1	Portal of all settings.
Settings2	Form to change password.
Settings2Controller	Stores the functions of the “Seetings2” class.
Settings3	Form to change username.
Settings3Controller	Stores the functions of the “Seetings3” class.

Settings4	Form to change email.
Settings4Controller	Stores the functions of the “Seetings4” class.
Settings5	Form to activate/desactivate the receival of notifications.
Settings5Controller	Stores the functions of the “Seetings5” class.
AvailableRooms	Dynamic display that shows avaliable study rooms.
AvailableRoomsController	Stores the functions of the “AvailiableRooms” class.
DbConnection1	Stores functions that (connect, interact, open/close) the Databse.
MailUtil	Stores fuctions that allow the sending of emails.
Main	Starting point of the code. First class to run.

Main algorithms:

Class	Function	Pseudocode	Explanation

	Insert…(..)	PreparedStatement ps
		try
		  Query: "insert into table (x) values (?)”
		  ps = con.prepareStatement(Query)
		  ps.setString(1, x)
		  ps.execute()
		catch
		  print error
			Inserts the function’s parameters into a specific table. It can also be done into a specific column by adding a “Where” operator. This technique is to be used when data has to be inserted.

	Retrieve…()	PreparedStatement ps
		try
		  Query: “select * from table"
		  ps = con.prepareStatement(Query)
		  ResultSet variable (rs) = ps.executeQuery()
		  Loop while (rs.next())
		    rs.getString(0)
		    rs.getString(..)
		    rs.getString(…)
		catch
		  print error
			Retrieves data from a specific table in the database. More complexity and precision can be added by adding conditional or logical SQL operators.

	Check…(..)	PreparedStatement ps
		try
		  Query: "select * from table where x = y"
		  ps = con.prepareStatement(Query)
		  ResultSet variable (rs) = ps.executeQuery()
		  if (rs.next())
		    return rs OR Boolean true/false
		catch
		  print error
			This function is just a commonly used adaptation of the retrieve feature. It will allow to check if a given parameter was found in the table. The precision of the call will depend on the query.

DBConnecton	Update…(..)	PreparedStatement ps try Query: "update table SET x = ?, WHERE y = ?" ps = con.prepareStatement(Query) ps.setString(1, x) ps.executeUpdate() catch print error	The update function uses a specific command to replace a value of a cell in a table. The specific cell to replace is established through the "Where" operator.
	Delete…(..)	PreparedStatement ps try Query: "Delete from table" ps = con.prepareStatement(Query) ps.executeUpdate() catch print error	The Delete function uses the "delete" command to eliminate a given place stated by the query.
	List of emails to send notifications	PreparedStatement ps Try Query = "select Email from Notifications N join Accounts A on N.UserName = A.UserName join IBSubjects I on I.UserName = A.UserName WHERE N.Groups = 1 and I.Group1 == '" +Subject1+ "' or I.Group1 == '" +Subject2+ "' or I.Group1 == '" +Subject3+ "' " …….. Very Long ……… ps = con.prepareStatement(Query) ResultSet variable (rs) = ps.executeQuery() ArrayList of Strings named Emails loop while (rs.next()) Emails.add(rs.getString(1)) return Emails catch print error	This is used to recollect the emails of users that should receive notifications when a StudyGroup is created. The function is in charge of returning an Array List containing all emails of users which have authorized the receival of notifications and that have at least one of the subjects with the StudyGroup, in common. The function has required a set of commands and operators such as "Join" to fit the complex query and its objective.
AvailableRooms Controller	nextString()	Private BufferedReader br try BufferedReader br br.readLine() catch print error	This function uses the BufferedReader class to read characters from the input stream (txt file). It is used to read each line from the text file containing the three indexes of the Available rooms Matrix.
	FileInput(..)	FileReader named (fr) BufferedReader named (br) filename = function parameter (file) try fr = new FileReader(filename) br = new BufferedReader(fr) catch print error	This function uses the FileReader and BufferedReader classes in order to process and select the text file. It will be used by the File parsing function to select a file.

AvailableRooms	File parsing	try
		  For loop i from 0 to rows of text file
		    String txt = nextString
		    Array of strings “values” = split text with comma “,”
		    Array of ints “indices” = new array [3]
		    For loop k from 0 to 3
		      indices[k] = Integer parse values[k]
		      Matrix [indices[0]] [indices[1]] [indices[2]] set true
		catch
		  print error
			This function will be essential to retrieve the indexes of the available rooms Matrix. If successful, it should extract the indexes from the text file while simultaneously setting the matrix to true at the cells of given indexes. This was the most efficient option in order to not store the data in the code.

	Time period classification	Hour = Hour_Of_Day
		Minute = Minute_Of_Day
		Total_min = (Hour*60) + Minute
		
		  if (b >= 510 and b <= 570)
		    Period = Period 1
		
		  else if (b >= 570 and b <= 635)
		    Period = Period 2
		
		  else if (b >= 650 and b <= 710)
		    Period = Period 3
		
		  else if (b >= 710 and b <= 775)
		    Period = Period 4
		
		  else if (b >= 820 and b <= 880)
		    Period = Period 5
		
		  else if (b >= 880 and b <= 945)
		    Period = Period 6
		
		  else if (b >= 945 and b <= 1010)
		    Period = Period 7
		
		  else if (b >= 1010 and b <= 1100)
		    Period = Period 8
		
		  else
		    Period = free
			This simple algorithm will require the use of the java “Calendar” class to obtain the exact time in terms of hours and minutes. Once obtained, the algorithm will derive the exact period.
			
			Period 1
			8:30 am = 510 min
			9:30 am = 570 min
			
			Period 2
			9:35 am = 575 min
			10:35 am = 635 min
			
			Period 3
			10:50 am = 650 min
			11:50 am = 710 min
			
			Period 4
			11:55 am = 715 min
			12:55 pm = 775 min
			
			Period 5
			13:40 pm = 820 min
			14:40 pm = 880 min
			
			Period 6
			14:45 pm = 885 min
			15:45 pm = 945 min
			
			Period 7
			15:50 pm = 950 min
			16:50 pm = 1010 min
			
			Period 8
			17:00 pm = 1020 min
			18:20 pm = 1100 min

Test Plan

Test	Success criteria	Expected result	Methods of testing
1	**(a) High Priority:** The system *must* be able to provide a clear dynamic list of available rooms at any moment.	When the user opens the AvailableRooms GUI (*Design Diagram 16*), a list of available rooms should appear. This list should also be clear and optimally contain an update button in the case a user wishes to refresh the list.	**1°** Create a temporary user account. **2°** Access the "Available Rooms" GUI. **3°** Compare the results with the interactive timetable of the School.
2	**(b) High Priority:** The "StudyGroup" subsystem *must* propose to users matching groups. It is assumed that a "matching" happens when a StudyGroup is set with the same subject as the user's filter search.	When a user enters in the StudyGroups GUI (*Design Diagram 4*), an option to input desired subjects should appear. After processing this demand, the system should automatically provide matching StudyGroups if existent.	**1°** Create 2 temporary user accounts (a,b) **2°** Access the "StudyGroups" GUI with account (a). **3°** Create a StudyGroup. → Log-out. **4°** Log-In with account (b). **5°** Access the "StudyGroups" GUI. **6°** Input desired subject(s). **7°** Check how the system reacts.
3	**(c) High Priority:** The system *must* allow the creation of study groups, both in the case that non were found, and that the user wishes to create one.	When a user enters in the StudyGroups GUIs (*Design Diagrams 4,5,6,7*), the subsystem should allow for the creation of a StudyGroup. This must be valid in both the cases the matching StudyGroups were and were not found.	**1°** Create a temporary user account. **2°** Access the "StudyGroups" GUI. **3°** Input random subjects **4°** Check if the system allows to create a StudyGroup when a matching group is found. (pass/fail) **5°** Check if the system allows to create a StudyGroup when a matching group is not found. (pass/fail)
4	**(d) High Priority:** The program *has* to be able to send emails when new StudyGroups are created. The recipients *must* share a subject with the StudyGroup and have agreed on receiving notifications.	An email should be received from every user that agreed on incoming notifications and that studies at least one of the subjects used to create the StudyGroup. The email should also indicate witch subjects are in common.	**1°** Create 3 temporary user accounts with similar subjects (a,b,c) **2°** Use one of the accounts (a) to create a StudyGroup. **3°** Check if the other accounts (b,c) received notifications through email. **4°** Test different combinations of subjects.
5	**(e) High Priority:** The program *has* to be able to send emails when new Tutoring petitions have been created. (Only for users that agreed on receiving Tutoring notifications and study the subject being asked for)	An email should be received from every user that agreed on incoming notifications for Tutoring and that studies the subject being asked for. The email should also contain a short, personalized message from the user asking for help.	**1°** Create 3 temporary user accounts with similar subjects (a,b,c) **2°** Use one of the accounts (a) to create a Tutoring petition. **3°** Check if the other accounts (b,c) received notifications through email. **4°** Test different combinations of subjects.

6	**(f) High Priority:** The program *has* to be able to send a retrieved password through email when the button "Forgot password?" is pressed.	This subsystem should send a retrieved password by email to the UserName account inputted in the Log-In GUI (*Design Diagram 1*). This should be activated by pressing a button. If the user is not found in the database an error message should popup.	**1°** Create a temporary user account. **2°** Access the "Log-In" GUI. **3°** Write the UserName. **4°** Press the retrieve password button. **5°** Await the email containing the password. (pass/fail)
7	**(g) High Priority:** Students *need to be able* to ask for help, similarly, provide Tutoring.	There should be a GUI and subsystem entirely designed for this function (*Design Diagrams 8,9,10*). Optimally, users should be able to create help petitions or help others that have asked for Tutoring. There could be three GUI classes, one for each feature: **1°** List of Tutoring petitions. (*Design Diagram 8*) **2°** Creation of Tutoring petition. (*Design Diagram 9*) **3°** Help a Tutoring petition. (*Design Diagram 10*)	**1°** Create a temporary user account. **2°** Access the "Tutoring" subsystem. **3°** Look for Tutoring petitions. (pass/fail) **4°** Try to help a Tutoring petition. (pass/fail) **5°** Try to create a Tutoring petition. (pass/fail)
8	**(h) High Priority:** If a host quits a StudyGroup, the group *must* be automatically deleted. Meaning that all users in it will be logged-out of the group.	When exiting a StudyGroup as a host (*Design Diagram 6*), a message should appear to confirm the deletion. Once deleted, all the users signed-up to this group will be automatically ejected, editing the database.	**1°** Create 2 temporary user accounts (a,b) with similar subjects. **2°** Use one of the accounts (a) to create a StudyGroup. **3°** Log-In with the other account (b) to join the StudyGroup. **4°** Log-In again with the Host account (a) and exit the StudyGroup. Confirm the deletion of the StudyGroup. **5°** Check the database for the (a,b) users. (pass/fail)
9	**(i) Medium Priority:** The System *must* allow users to obtain information from a StudyGroup before joining.	When matching StudyGroups are presented in the GUI (*Design Diagram 5*) there should be a button that allows the retrieval of information before having to join: E.g. (Host, Description, hours etc…)	**1°** Create 2 temporary user accounts (a,b) with similar subjects. **2°** Use one of the accounts (a) to create a StudyGroup. **3°** Log-In with the other account (b) to join a StudyGroup. **4°** Check with account (b) if the Info button works before joining. (pass/fail)
10	**(j) Medium Priority:** When creating a StudyGroup the user *should* have access to a place where to describe the main focus of the group.	In the process of creating a StudyGroup (*Design Diagram 7*) there should be a dedicated place where to fill up a quick description of the goals and main focus of the StudyGroup. Optimally, it could be a text Area.	**1°** Create a temporary user account. **2°** Access the "StudyGroups" subsystem. **3°** Access the StudyGroup (*Design Diagram 7)* GUI. **4°** Was a dedicated writing space achieved? Does it work? Is it clear and visible?

11	**(k) Medium Priority:** The program *must* allow the changing of UserName.	When accessing the settings portal (*Design Diagram 11*), there should be a subsystem to change the UserName (*Design Diagram 13*). This should be feasible by inputting the old and new usernames coupled with the password to enhance security.	**1°** Create a temporary user account. **2°** Access the "Settings" portal. **3°** Access the "Change UserName" GUI. **4°** Input old and new UserNames. **5°** Input password. **6°** Check database. (pass/fail)
12	**(l) Medium Priority:** StudyGroups *can only exist* for ten hours. After that the StudyGroup should be renovated or otherwise will be deleted from the main view and database.	When a StudyGroup lasts 10 hours from the time it was created it should be automatically deleted. Hence, ejecting all users including the host. There should also be an option to extend the StudyGroup. It may pop up on a message dialogue.	**1°** Create 2 temporary user accounts (a,b) with similar subjects. **2°** Use one of the accounts (a) to create a StudyGroup. **3°** Log-In with the other account (b) to join the StudyGroup. **4°** Wait 10 hours. (Nighttime) **5°** Check if StudyGroup was deleted or not. (pass/fail) **6°** Reiterate the experiment but this time extending the lifetime of the StudyGroup. (pass/fail)
13	**(m) Medium Priority:** *Option* to receive or not to receive emails for Tutoring or new StudyGroups created.	When accessing the "Settings portal" (*Design Diagram 11*), there should be a subsystem to change the setting to receive or not to receive notifications. This must be feasible for both (StudyGroups and Tutoring) independently.	**1°** Create a temporary user account. **2°** Access the "Settings" portal. **3°** Access the "Notifications" GUI (*Design Diagram 15)*. **4°** Change the settings. (pass/fail) **5°** Check in database (pass/fail) **6°** Repeat with different combinations. (pass/fail)
14	**(n) Medium Priority:** Users *should* be able to research/create interdisciplinary StudyGroups. Meaning StudyGroups that were/are registered with more than one subject.	When researching or creating matching StudyGroups with inputted subjects, there should be an option to add several subjects in the filter list. In such case a StudyGroup could be simultaneously working in two subjects at the same time.	**1°** Create a temporary user account. **2°** Access the "StudyGroups" GUI. **3°** Search for StudyGroups with more than one subject. (pass/fail) **4°** Create StudyGroup with more than one subject. (Pass/fail)
15	**(o) Low Priority:** When the "Password" or "Email" of a user is being edited on the settings category, the system *has* to be able to inform the user by email.	Every time a user decides to change the "Password" or "Email" a notification will be sent in order to inform/remind the user of the new account information.	**1°** Create a temporary user account. **2°** Access the "Settings" portal. **3°** Change password and email. **4°** Check if emails were received to inform of the new account information. (Pass/fail)
16	**(p) Low Priority:** The program *has* to be able to allow students to exceptionally book specific study rooms for given reasons (Standardized tests/Exams). This will be done by sending a common email to all users .	Whenever a user books a specific room, all users of the system should be emailed, giving all the necessary information: "What is happening" "From when to when" "Where" "Why"	**1°** Create a temporary user account. **2°** Access the "AvailableRooms" GUI. **3°** Make a petition through the "Booking" GUI. **4°** Check emails of all test users.

17	**(q) Low Priority:** When creating StudyGroups that exceeds the curfew hour, there *should* be a notice to remind students of asking for extended curfew.	If a user attempts to create a StudyGroup that exceeds the curfew time, a pop up window should alert and remind the user to ask for extended curfew or leave by 10:30 pm.	**1°** Create a temporary user account. **2°** Access the "StudyGroups" GUI. **3°** Try to create a StudyGroup that exceeds curfew time. **4°** Check system response (pass/fail).

Word count: 0