|  |  |
| --- | --- |
|  | MuReservation  Calites, John Vincent  Fradejas, Dayne N.  Reyest, Jean Tristan. |

1. Table of Contents

[A Objectives 1](#_Toc527343558)

[B Scope and Features 2](#_Toc527343559)

[B.1 Sign up 2](#_Toc527343560)

[B.2 Login 2](#_Toc527343561)

[B.3 Scheduling 2](#_Toc527343562)

[B.4 Time selection 2](#_Toc527343563)

[B.5 Room selection 2](#_Toc527343564)

[B.6 Reservation 2](#_Toc527343565)

[B.7 Remove accounts 2](#_Toc527343566)

[B.8 Remove reservation 2](#_Toc527343567)

[C Limitations 3](#_Toc527343568)

[C.1 Sign up 3](#_Toc527343569)

[C.2 Login 3](#_Toc527343570)

[C.3 Scheduling 3](#_Toc527343571)

[C.4 Time selection 3](#_Toc527343572)

[C.5 Room selection 3](#_Toc527343573)

[C.6 Reservation 3](#_Toc527343574)

[C.7 Remove accounts 3](#_Toc527343575)

[C.8 Remove reservation 3](#_Toc527343576)

[D System Requirements 4](#_Toc527343577)

[D.1 Hardware Requirements 4](#_Toc527343578)

[D.1.1 Processor 4](#_Toc527343579)

[D.1.2 Memory 4](#_Toc527343580)

[D.1.3 Storage 4](#_Toc527343581)

[D.2 Software Requirements 4](#_Toc527343582)

[D.2.1 Operating System 4](#_Toc527343583)

[D.2.2 Software Requirements 4](#_Toc527343584)

[E Class Diagram 5](#_Toc527343585)

[E.1 MuReservation Class Diagram 5](#_Toc527343586)

[E.2 Records Entity Relationship Diagram (ERD) 6](#_Toc527343587)

[E.3 MuReservation Flowchart 7](#_Toc527343588)

[E.4 Used Cased Diagram 10](#_Toc527343589)

[F Class Definition 11](#_Toc527343590)

[F.1 MuReservation Class Definition 11](#_Toc527343591)

[G Appendices 17](#_Toc527343592)

[G.1 Creation of Account 17](#_Toc527343593)

[G.2 Login to MuReservation 18](#_Toc527343594)

[G.3 Using Interfaces 20](#_Toc527343595)

[G.4 Show Accounts (Admin) 22](#_Toc527343596)

[H Analysis 27](#_Toc527343597)

[I Conclusion 27](#_Toc527343598)

List of Figures

[Figure E.1: Business Logic Class Diagram 5](#_Toc527343532)

[Figure E.2: Data Access (File Handling) Class Diagram 6](#_Toc527343533)

[Figure E.3: Records Database (ERD) 6](#_Toc527343534)

[Figure E.4: MuReservation Program Flowchart 7](#_Toc527343535)

[Figure E.5: Admin Interface Method Flowchart 8](#_Toc527343536)

[Figure E.6: User Interface Method Flowchart 9](#_Toc527343537)

[Figure E.7: MuReservation Used Cased Diagram 10](#_Toc527343538)

[Figure G.1: Account Creation 17](#_Toc527343539)

[Figure G.2: Signup Interface 17](#_Toc527343540)

[Figure G.3: Logging as User 18](#_Toc527343541)

[Figure G.4: Showing User Interface 18](#_Toc527343542)

[Figure G.5: Logging as Admin 19](#_Toc527343543)

[Figure G.6: Showing Admin Interface 19](#_Toc527343544)

[Figure G.7: Viewing Available Schedule 20](#_Toc527343545)

[Figure G.8: Viewing Available Schedule 20](#_Toc527343546)

[Figure G.9: Reserving a Schedule 21](#_Toc527343547)

[Figure G.10: Reserving a Schedule (Admin) 21](#_Toc527343548)

[Figure G.11: Information Search by Student Number 22](#_Toc527343549)

[Figure G.12: Edit Mode Activated Mode 22](#_Toc527343550)

[Figure G.13: Removing Account 23](#_Toc527343551)

[Figure G.14: Account Removed Successfully 23](#_Toc527343552)

[Figure H.1: Login MainWindow 24](#_Toc527343553)

[Figure H.2: Signup Dialog 24](#_Toc527343554)

[Figure H.3: Search Accounts (Admin) 25](#_Toc527343555)

[Figure H.4: Show Schedules (Admin) 25](#_Toc527343556)

[Figure H.5: Show Schedule (User) 26](#_Toc527343557)

# Objectives

To automate the reservation process of rooms for organizational events

To create an easier way of organizing events

To help avoid conflicts in scheduling the rooms

# Scope and Features

## Sign up

The users and admin is able to sign up for accounts in the program using their credentials, and it will be saved on the database of accounts

## Login

The users and admin is able to login to their created accounts, reflecting their credentials

## Scheduling

The users and admin can view the week schedule of reserved time slots for their selected date and room. The admin is able to view which organization reserved the time slots

## Time selection

The users and admin will be able to choose the time their reservation will start and end

## Room selection

The users and admin will be able to choose the room that is available for reservation

## Reservation

The users and admin will be able to reserve the room that they selected on the date and time that they want, and it will be saved on the database of reservations

## Remove accounts

The admin will be able to remove existing accounts from the database

## Remove reservation

The admin will be able to remove reserved rooms for specific time and date

# Limitations

## Sign up

There is no other security measure other than passwords.

## Login

There is no option to keep the user logged in

There is no option to remember passwords for users

There are no means to recover password other than to get them from the admin

## Scheduling

The users can only see that the rooms are reserved at a specific time, they cannot see what organization has the reservation.

## Time selection

The users can only select the available time shown on the interface.

## Room selection

The rooms that are available for reservation is limited

## Reservation

The users can only reserve the time slot shown at a given day and time.

## Remove accounts

The admin is the only one capable of removing accounts.

## Remove reservation

The admin is the only one capable of removing reservations.

# System Requirements

## Hardware Requirements

### Processor

Processor Core Count : 2

### Memory

Memory Capacity : 4 GB

### Storage

Disk Size : 200 GB

## Software Requirements

### Operating System

Operating System Name : Windows 10 Education

Architecture Type : 64-bit

Edition : Single Language

### Software Requirements

Software Application # 1 : Python 3.4

Software Application # 2 : Anaconda 4.5.11

# Class Diagram

## MuReservation Class Diagram



Figure E.1: Business Logic Class Diagram

In Figure F.1.a, it shows the class diagram for the business logic classes. The login interface is treated as the main window for the whole program. The LoginBL is used for logging in to the program which the user id is passed to UserBL and AccountsBL to be used for identification during processing. The ScheduleBL has all the function of UserBL which identifies the schedules available for the date given. In addition, the ScheduleBL has an exclusive function that the UserBL don’t have, which is the removing of schedule. Finally, the AccountsBL is responsible of removing and editing accounts found on the database.



Figure E.2: Data Access (File Handling) Class Diagram

This class is responsible for getting all the values need of each business logic. The diagram is not connected to the classes for business logic because it won’t fit the page. But, all classes has a value coming from the database.

## Records Entity Relationship Diagram (ERD)

Figure E.3: Records Database (ERD)

In the figure above, it shows that the user from the login userID, as it is passed from one database to another, the data from the user information can be accessed from the userID obtained from the login database.

## MuReservation Flowchart

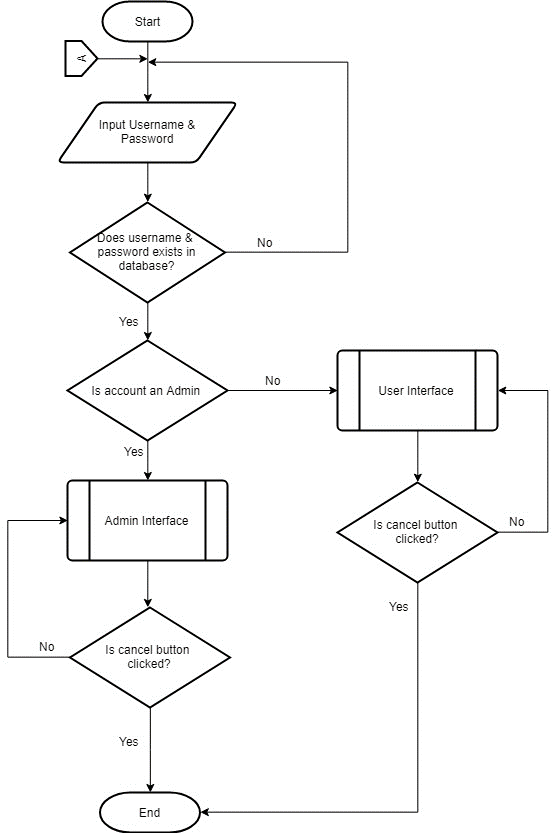


Figure E.4: MuReservation Program Flowchart

In figure E.3.a, it shows the program flowchart for the MuReservation. First, the user is prompt to enter a username and password, and the account is determined if it is an admin or not. If the account logged in is an admin, it will be redirected to the admin interface, if not, it will bre redirected to the ordinary user interface. Lastly, the program will continuously work until the cancel button is clicked.

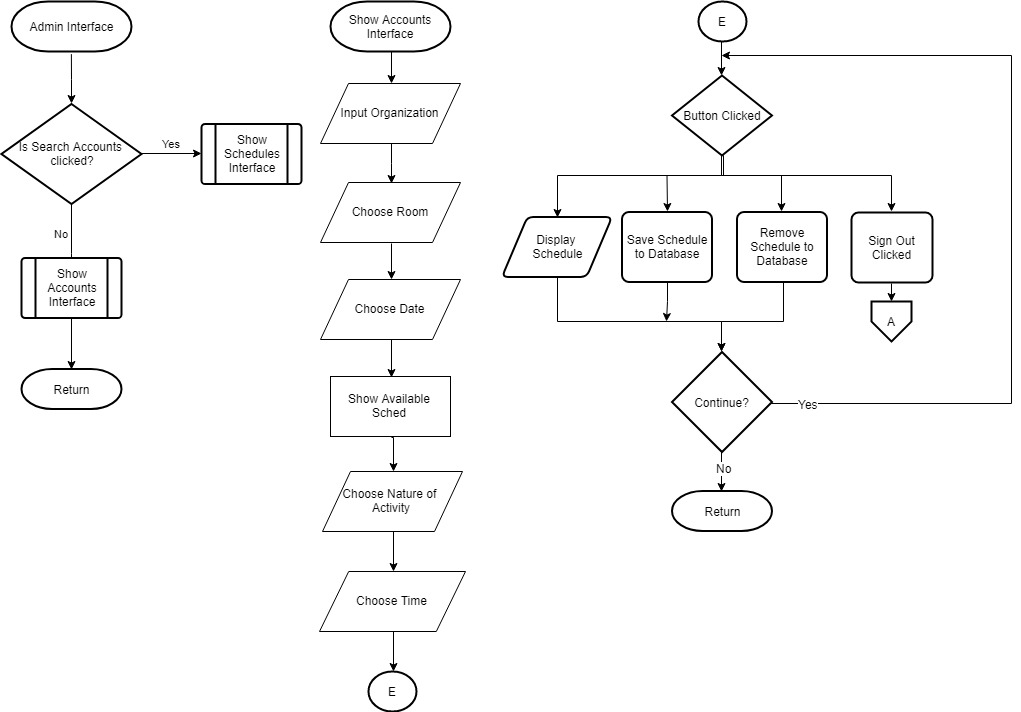


Figure E.5: Admin Interface Method Flowchart

On the admin interface, the admin has the capability of reserving, removing, and adding rooms to the database of the program. Also, the admin also has the capability of editing and removing an account existing on the software.

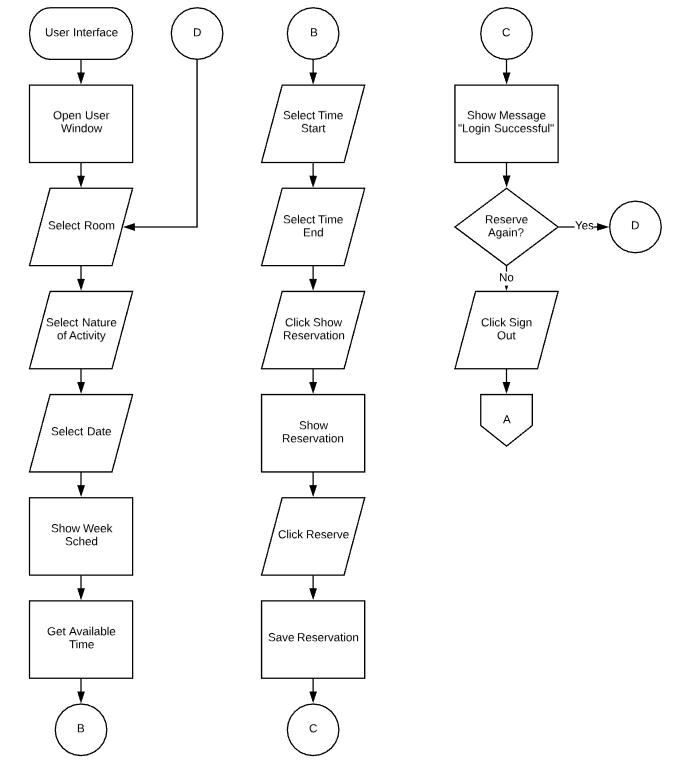


Figure E.6: User Interface Method Flowchart

In the user interface, unlike the admin, the user is only capable of reserving a room which a user doesn’t have the capability of editing his/her account as well the controlling the availability of the rooms to be reserved.

## https://scontent.fmnl3-1.fna.fbcdn.net/v/t1.15752-9/44027715_310981663032932_6579715126662791168_n.png?_nc_cat=100&oh=cdaf2343fb6b04fbf48bd332d3b74efa&oe=5C5A1CB6Used Cased Diagram

Figure E.7: MuReservation Used Cased Diagram

Actors are usually the individuals involved with the system defined according to their roles. The actor in this case diagram are the user, admin, and database. A use case describes how actors uses a system to accomplish a particular goal. Use cases are typically initiated by a user to fulfill goals describing the activities and variants involved in attaining the goal. The relationships in this figure are the lines the connect the actors and the use case. The huge rectangle in the figure is the system boundary. The system boundary defines the system of interest in relation to the world around it.

# Class Definition

## MuReservation Class Definition

|  |  |
| --- | --- |
| **Name:** | LoginBL |
| **Type:** | Class |
| **Description:** | Used to perform the operations for login interface |
| **Fields:** | userName: string |
| Password: string |
| **Methods:** | checkAccount(string userNameUi,string passwordUi): string |
| loginState(string userNameUi, stringpasswordUi): string |
| getAccountType(string userNameUi,string passwordUi) : string |
| GenerateID(string userType): string |

|  |  |
| --- | --- |
| **Name:** | LoginBL |
| **Member Name:** | checkAccount |
| **Type:** | Method |
| **Description:** | Used to return bool if account exists |
| **Parameters:** | string userNameUi |
| string passwordUi |
| **Return Value:** | Existence |
| **Return Type:** | bool |

|  |  |
| --- | --- |
| **Name:** | LoginBL |
| **Member Name:** | loginState |
| **Type:** | Method |
| **Description:** | Used to return a message on the state of the login interface |
| **Parameters:** | string userNameUi |
| string passwordUi |
| **Return Value:** | Message |
| **Return Type:** | string |

|  |  |
| --- | --- |
| **Name:** | LoginBL |
| **Member Name:** | GenerateID |
| **Type:** | Method |
| **Description:** | Used to return the ID of the user logging in |
| **Parameters:** | string userType |
| **Return Value:** | ID |
| **Return Type:** | string |

|  |  |
| --- | --- |
| **Name:** | LoginBL |
| **Member Name:** | getAccountType |
| **Type:** | Method |
| **Description:** | Used to return string if the user is an Admin or a User |
| **Parameters:** | string userName |
| **Return Value:** | Account Type |
| **Return Type:** | string |

|  |  |
| --- | --- |
| **Name:** | LoginBL |
| **Member Name:** | getEmailByUsername |
| **Type:** | Method |
| **Description:** | Returns the email with the username given |
| **Parameters:** | string userName |
| **Return Value:** | Email |
| **Return Type:** | string |

|  |  |
| --- | --- |
| **Name:** | SignupBL |
| **Type:** | Class |
| **Description:** | A class that handles the processes for signup interface |
| **Fields:** | String lastName |
| String givenName |
| String middleName |
| String userName |
| String emailAddress |
| String password |
| String organization |
| String studentNumber |
| String contactNumber |
| String userType |
| **Properties:** | None |
| **Methods:** | Check\_Password() : bool |
| Check\_StudentNumber() : bool |
| isComplete() : bool |
| StoreInfo() |
| redunduncyState(): bool |
| checkRedunduncy(string username, string emailAddress, string givenName) : string |

|  |  |
| --- | --- |
| **Name:** | SignupBL |
| **Member Name:** | Check\_Password |
| **Type:** | Method |
| **Description:** | Checks if the password is acceptable |
| **Parameters:** | None |
| **Return Value:** | State |
| **Return Type:** | bool |

|  |  |
| --- | --- |
| **Name:** | SignupBL |
| **Member Name:** | Check\_StudentNumber |
| **Type:** | Method |
| **Description:** | Used to check if the student number format is valid or not |
| **Parameters:** | None |
| **Return Value:** | State |
| **Return Type:** | bool |

|  |  |
| --- | --- |
| **Name:** | SignupBL |
| **Member Name:** | StoreInfo |
| **Type:** | Method |
| **Description:** | Used to store account information to database |
| **Parameters:** | None |
| **Return Value:** | None |
| **Return Type:** | None |

|  |  |
| --- | --- |
| **Name:** | SignupBL |
| **Member Name:** | redunduncyState |
| **Type:** | Method |
| **Description:** | Checks if password, email, or student number is already existing |
| **Parameters:** | String username |
| String studentnumber |
| String emailAddress |
| **Return Value:** | State |
| **Return Type:** | bool |

|  |  |
| --- | --- |
| **Name:** | SignupBL |
| **Member Name:** | checkReduduncy |
| **Type:** | Method |
| **Description:** | Returns the message for redunduncy |
| **Parameters:** | String username |
| String studentnumber |
| String emailAddress |
| **Return Value:** | Message |
| **Return Type:** | String |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Type:** | Class |
| **Description:** | A class that handles the processes for accounts interface |
| **Fields:** | String lastName |
| String givenName |
| String middleName |
| String userName |
| String emailAddress |
| String password |
| String organization |
| String studentNumber |
| String contactNumber |
| String userID |
| **Properties:** | None |
| **Methods:** | initializeVariables () : AccountsBL |
| Check\_StudentNumber() |
| isComplete() |
| StoreInfo() |
| redunduncyState() : bool |
| checkRedunduncy(string username,string emailAddress,string givenName):string |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | intializeVariables |
| **Type:** | Contructor |
| **Description:** | Used to initialize all the variables in AccountsBL |
| **Parameters:** | String lastName |
| String givenName |
| String middleName |
| String userName |
| String emailAddress |
| String password |
| String organization |
| String studentNumber |
| String contactNumber |
| String userID |
| **Return Value:** | None |
| **Return Type:** | None |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | getDataByStudentNumber |
| **Type:** | Method |
| **Description:** | Get the list of data from a student number |
| **Parameters:** | String studentNumber |
| **Return Value:** | List |
| **Return Type:** | String |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | studentNumberExists |
| **Type:** | Method |
| **Description:** | Check if the student number is exsisting |
| **Parameters:** | String studentNumber |
| **Return Value:** | State |
| **Return Type:** | Bool |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | showStateOfStudentNumber |
| **Type:** | Method |
| **Description:** | Returns the message for the state of the password |
| **Parameters:** | String studentNumber |
| **Return Value:** | Message |
| **Return Type:** | String |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | getPasswordByStudentNumber |
| **Type:** | Method |
| **Description:** | Returns the password of the user with a specific student number |
| **Parameters:** | String studentNumber |
| **Return Value:** | Student number |
| **Return Type:** | String |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | updateInfoStatus |
| **Type:** | Method |
| **Description:** | Returns a message if data on the data base has been updated |
| **Parameters:** | None |
| **Return Value:** | Message |
| **Return Type:** | String |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | checkDataState |
| **Type:** | Method |
| **Description:** | Returns the state if data on the data base has been updated |
| **Parameters:** | None |
| **Return Value:** | State |
| **Return Type:** | Bool |

|  |  |
| --- | --- |
| **Name:** | AccountsBL |
| **Member Name:** | removeAccount |
| **Type:** | Method |
| **Description:** | Returns a message if the account has been successfully deleted |
| **Parameters:** | String userID |
| **Return Value:** | Message |
| **Return Type:** | String |

# Appendices

**User Manual**

## Creation of Account

Create Account for MuReservation

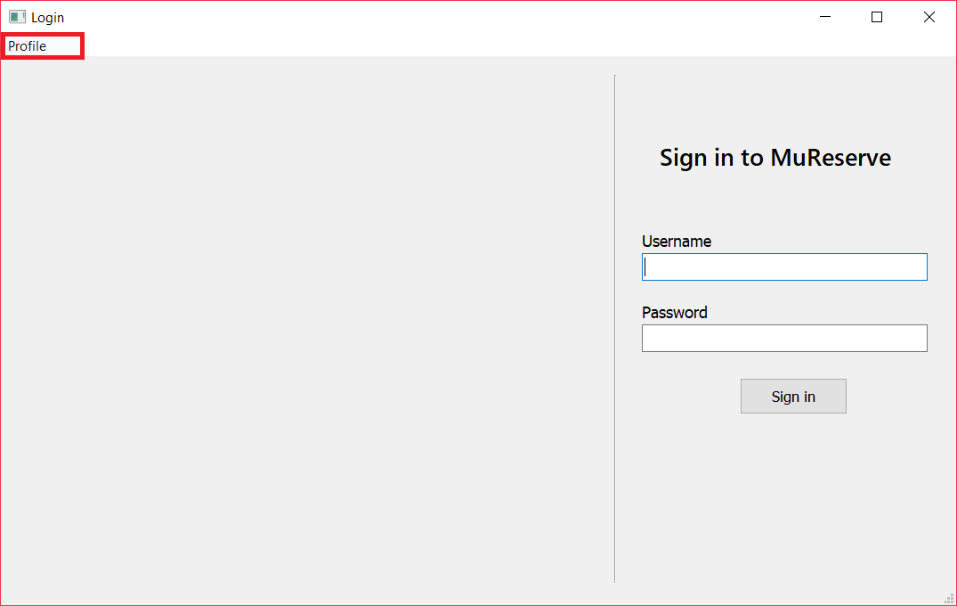


Figure G.1: Account Creation

Create Account for MuReservation

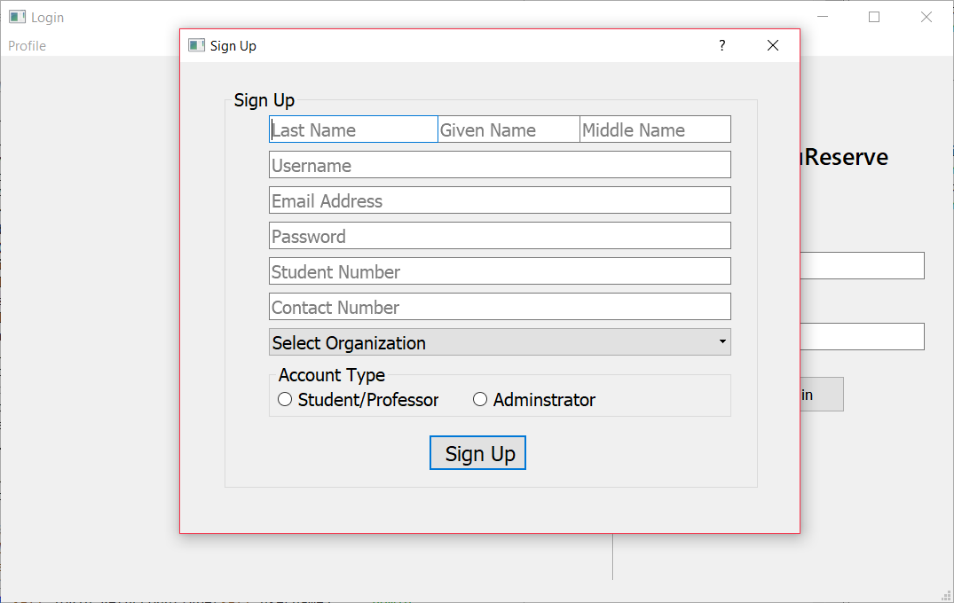


Figure G.2: Signup Interface

By completing the fields for signup, the data will automatically be saved to the database of the program.

## 

## Login to MuReservation

Login as User to MuReservation

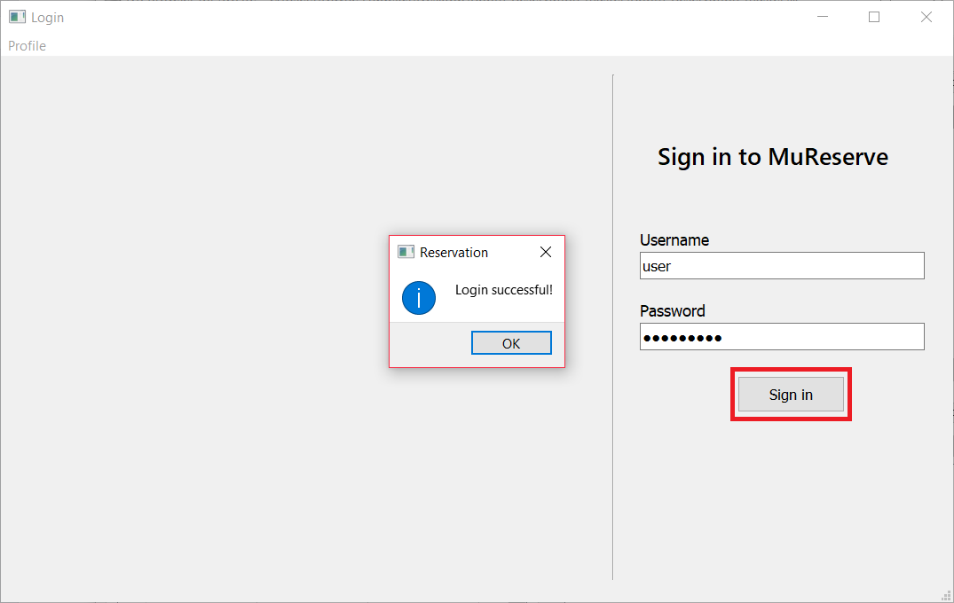


Figure G.3: Logging as User

User Interface

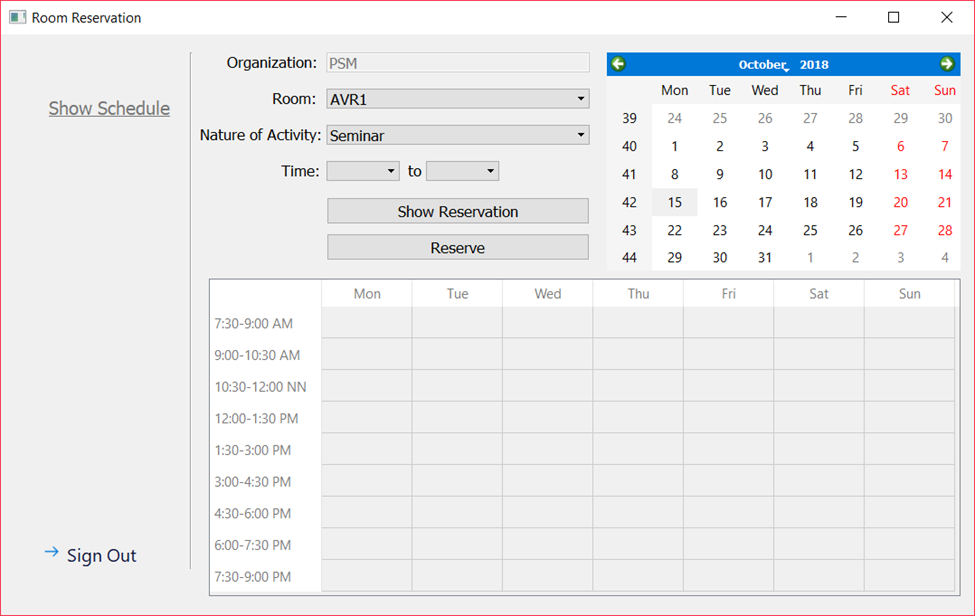


Figure G.4: Showing User Interface

In figure H.2.b, it shows the user interface since the account logged on the server is registered as an ordinary user.

Login as Admin to MuReservation

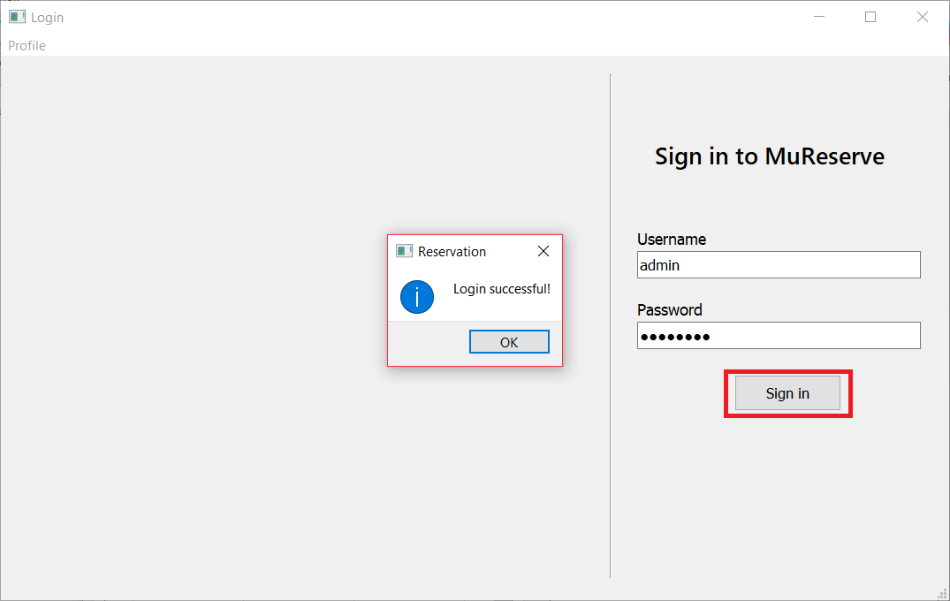


Figure G.5: Logging as Admin

Admin Interface

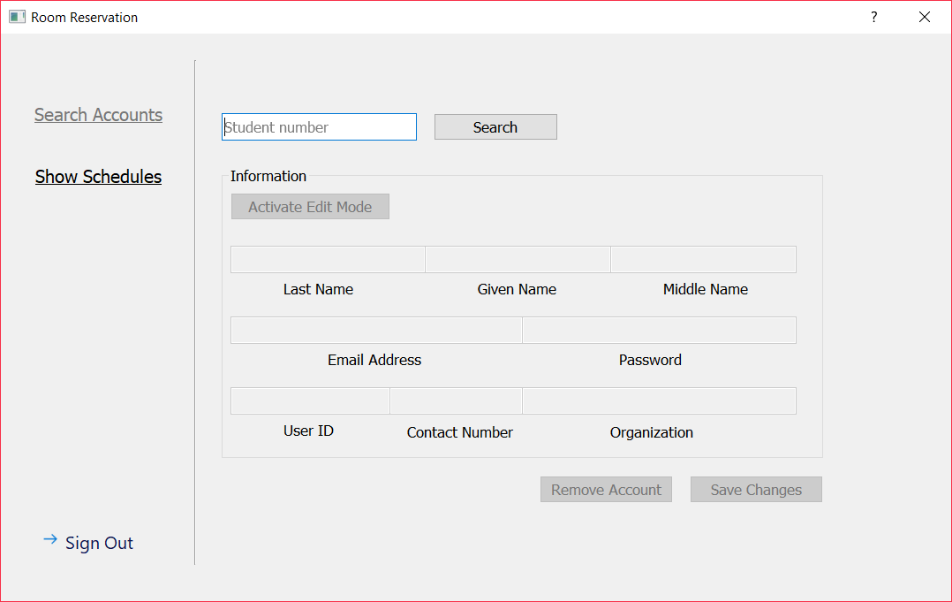


Figure G.6: Showing Admin Interface

In figure H.2.d, it shows the interface for admin users.

## Using Interfaces

Reserving to User Interface

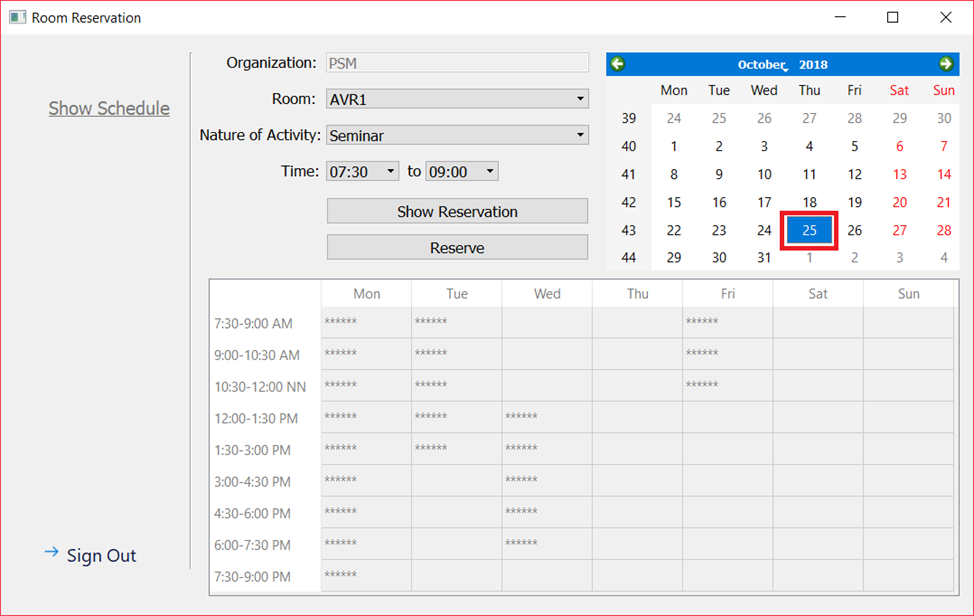


Figure G.7: Viewing Available Schedule

On figure H.2.d, when a date is clicked on the calendar, the program will automatically show the schedules that are already reserved for the whole week.

Showing Reservation Schedule

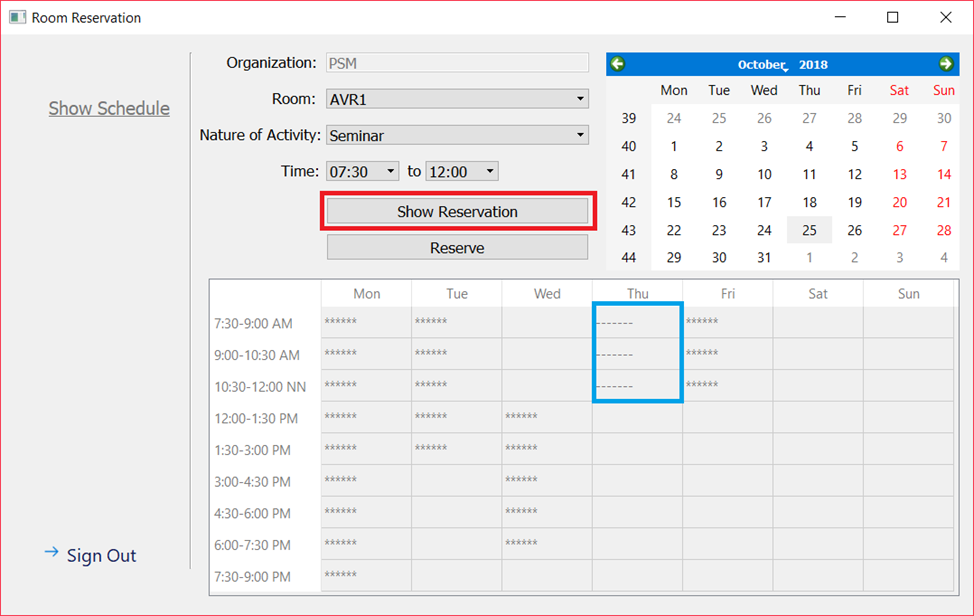


Figure G.8: Viewing Available Schedule

On figure G.3.b, it when show reservation is clicked, the program will show the schedule reserved.

Reserving a Schedule

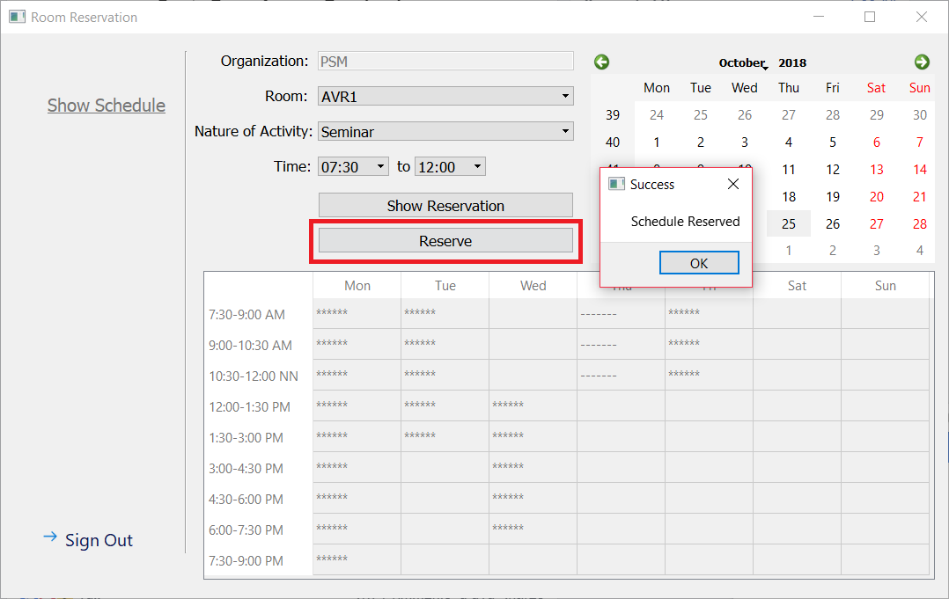


Figure G.9: Reserving a Schedule

After clicking the reserve button, the program will automatically save the schedule to its database. Also, when sign out is clicked, it will redirect the user to login window.

Reserving a Schedule

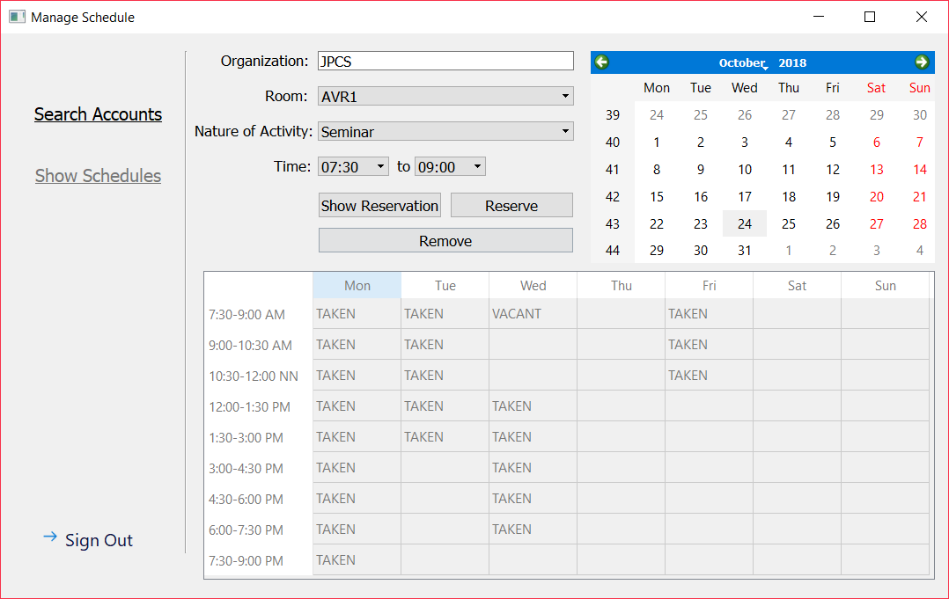


Figure G.10: Reserving a Schedule (Admin)

In figure G.3.d, it has the same reservation process, but the difference is, the admin has the capability to remove reserve and remove a room desired.

## Show Accounts (Admin)

Searching for Accounts

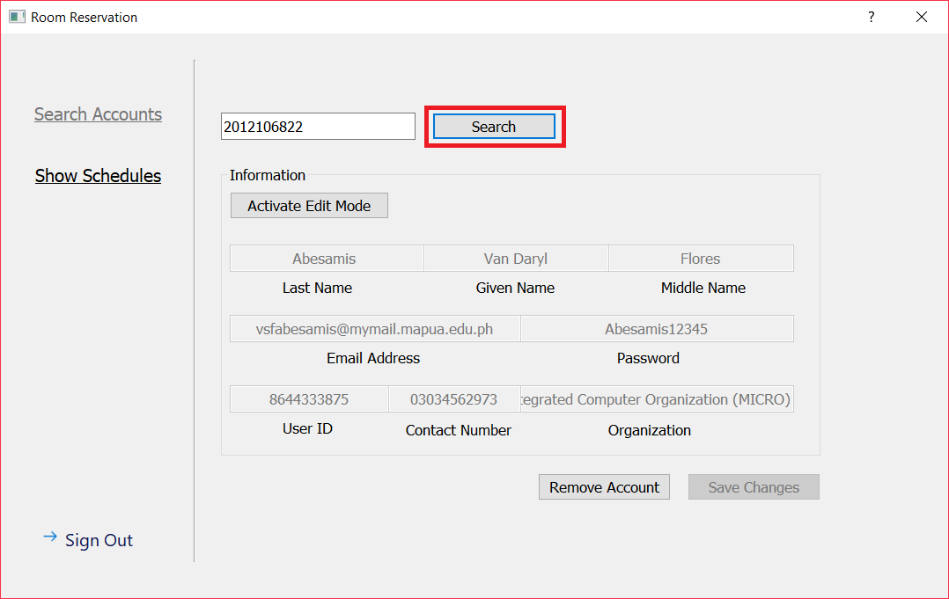


Figure G.11: Information Search by Student Number

In figure G.4.a, the figure shows when the search button is clicked with a valid student number, the program will automatically fill the table of information.

Activate Edit Mode

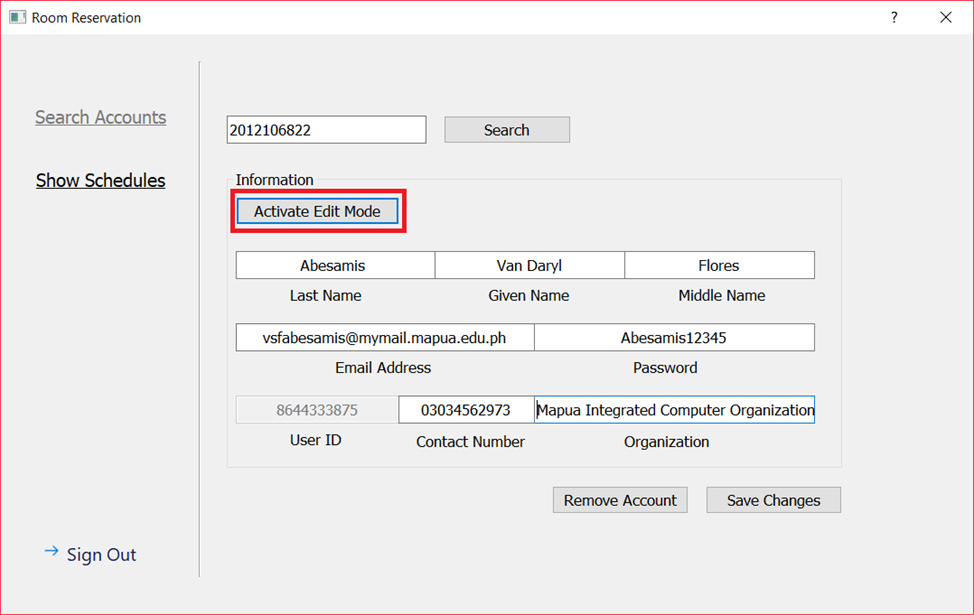


Figure G.12: Edit Mode Activated Mode

In figure G.4.b, when the activate edit mode is clicked, it allows the admin to edit, remove account, and save changes when necessary.

Remove Account

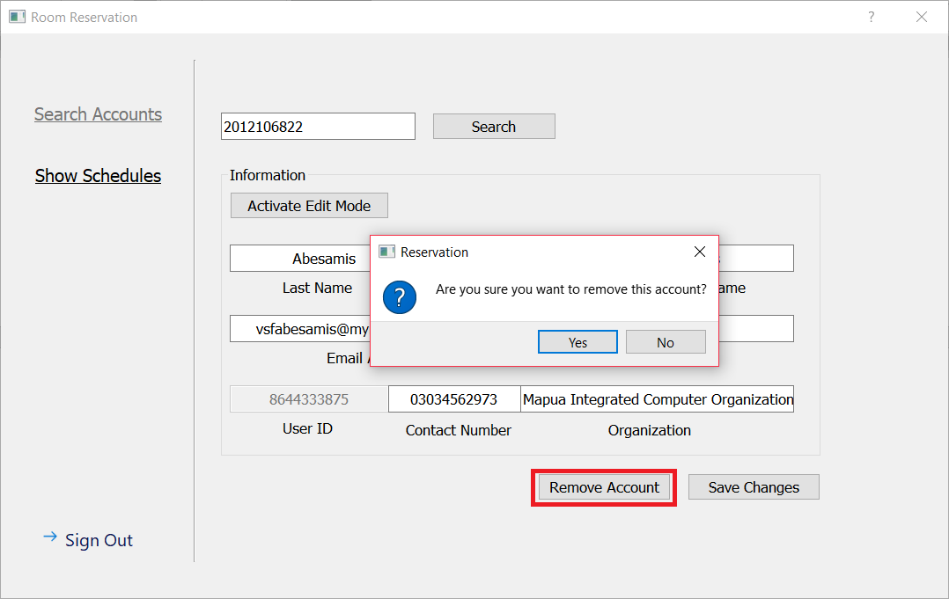


Figure G.13: Removing Account

When remove account is successfully done, the account will not exist anymore, thus it becomes unsearchable.

Checking if Account is Removed

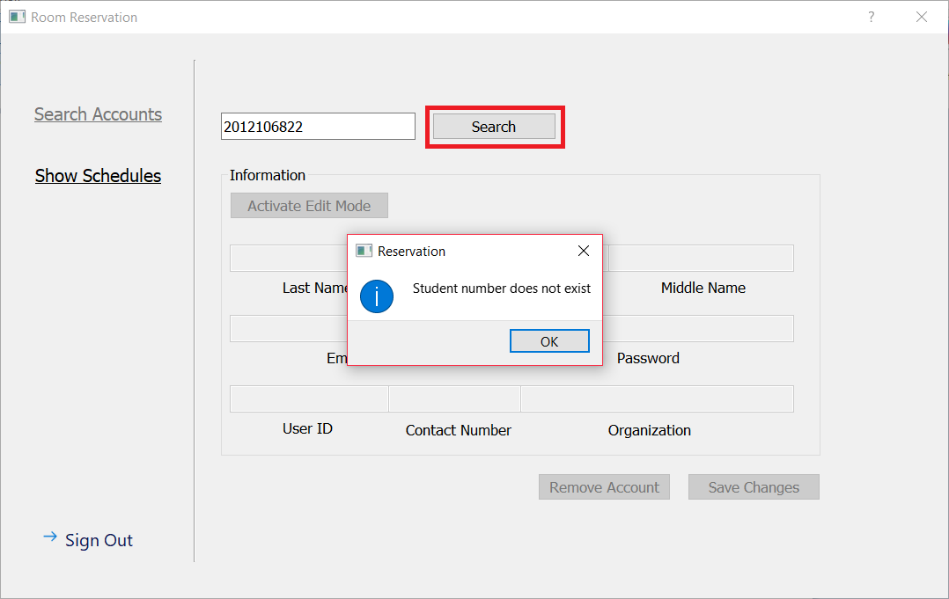


Figure G.14: Account Removed Successfully

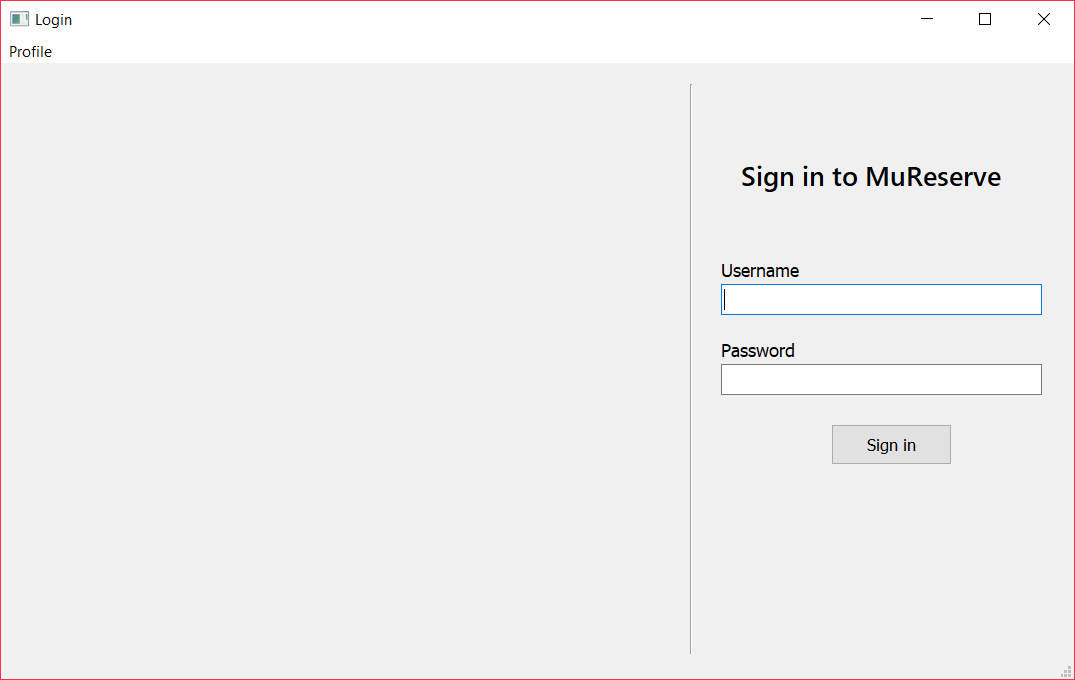


Figure H.1: Login MainWindow

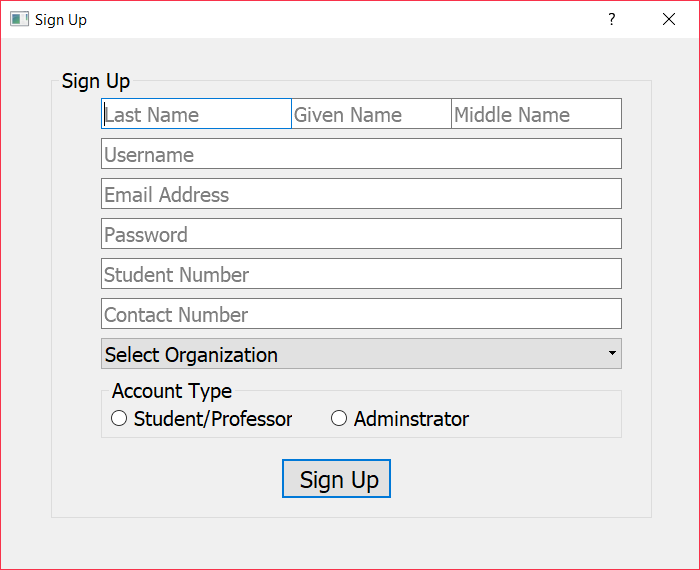


Figure .: Signup Dialog

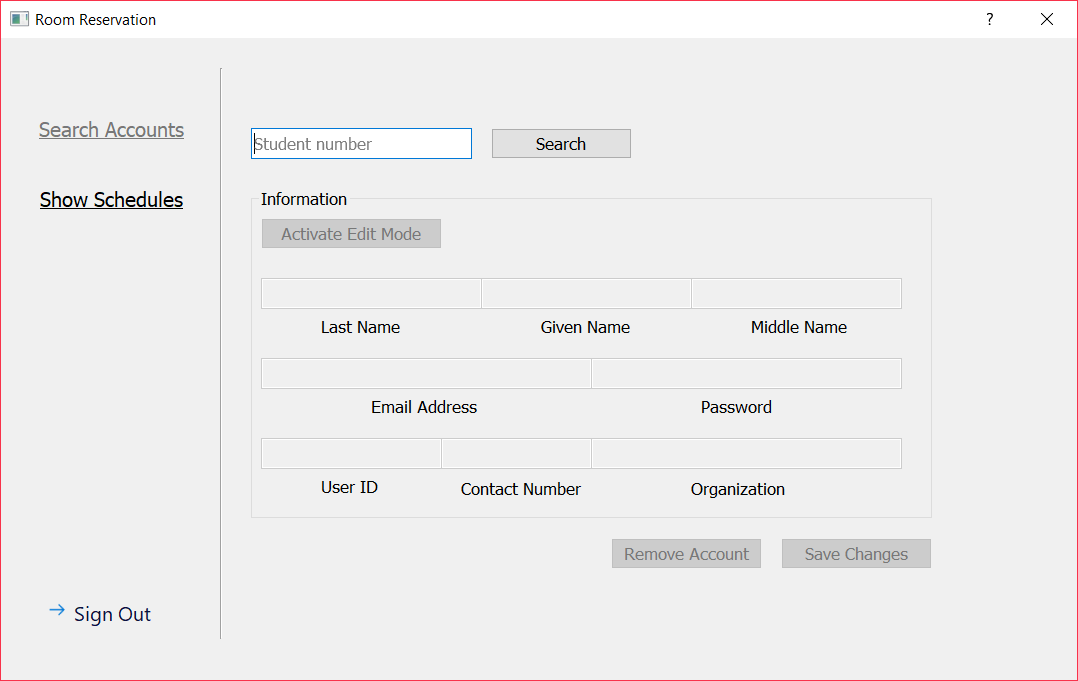


Figure .: Search Accounts (Admin)

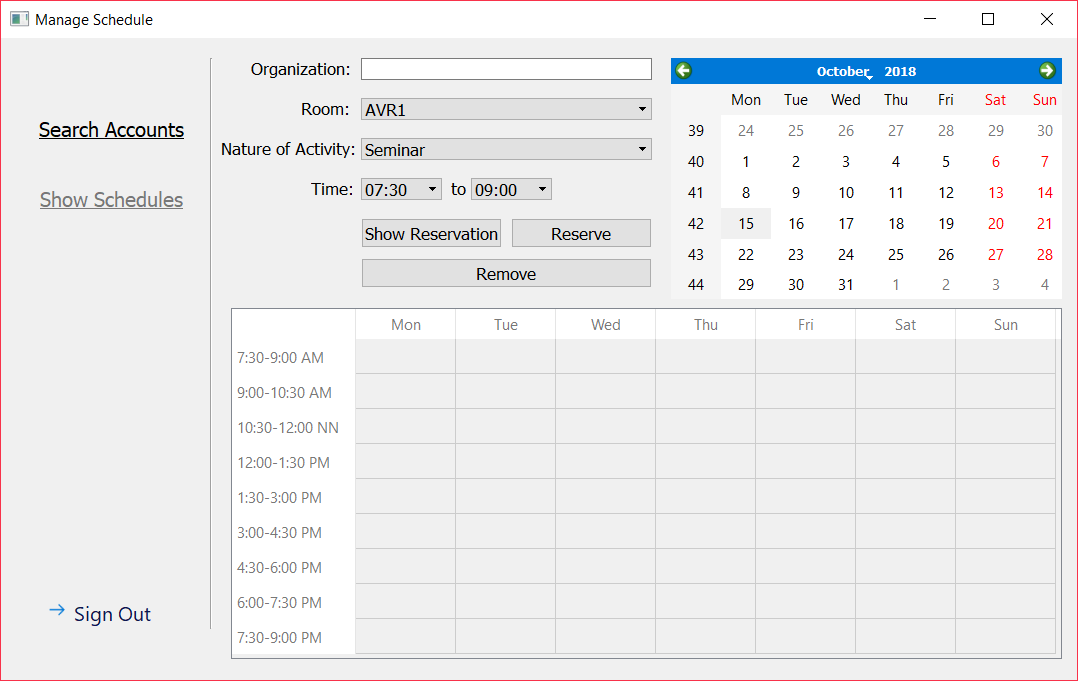


Figure .: Show Schedules (Admin)

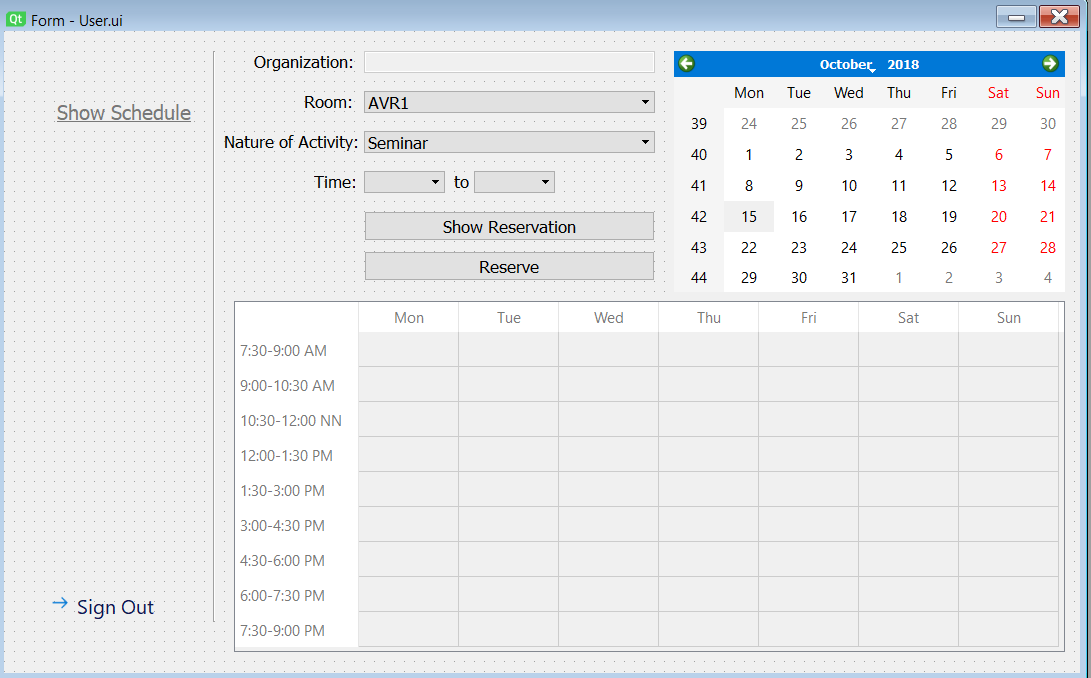


Figure .: Show Schedule (User)

# Analysis

In results obtained, the automation of room reservations becomes much easier because our algorithm immediately avoids conflicts when reserving for a schedule. Our program automatically shows all available schedule to from the table, also the data on the box for the time only shows the available schedule so that it becomes much easier to reserve a room without looking for a vacant with mere observation.

# Conclusion

In conclusion, we created a program that automates the process of reserving rooms. Also, we can conclude that our program makes reserving room for events much easier because of the algorithm implemented on our program.