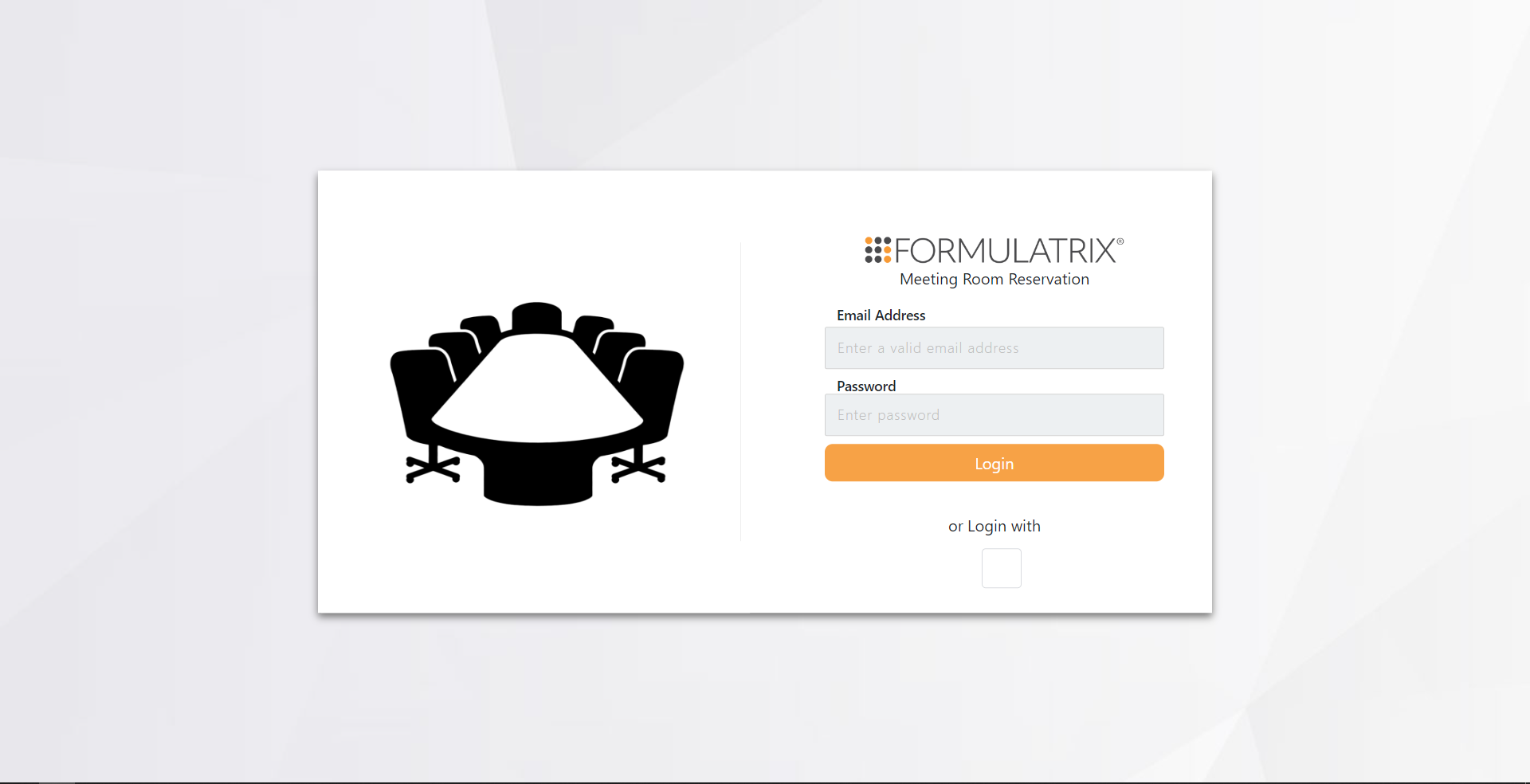
# **🏡Meeting Room Reservation V2.0🏢**

****

**Adibya Rizaldy \ 👩🏽‍💻 :** [@adibyaar](https://github.com/adibyarr)

**Bimanjaya Aji Krismawan \ 👩🏽‍💻 :** [@bimanjayaaji](https://github.com/bimanjayaaji)

**Haidar Amir Faruqi \ 👩🏽‍💻 :** [@hamirf](https://github.com/hamirf)

github repository: [Repo Link](https://github.com/adibyarr/Meeting_Room_Batch3.git)

08/31/2023

Meeting Room Organizer is a website that functions as a platform for an employee to book a room. This website could make the booking process less tedious by allowing users to directly look for available rooms at a particular day and a particular time.

## **❗Problem Description**

The need to book a room for meetings using the usual Google Calendar system can sometimes become tedious when many rooms are already used by others. This website can make the process more painless by providing users with several features to book a room for meeting.

This website could also be effective for announcements/invitations of meeting purposes because the attendees are listed using their emails, so they could get the notification of the arranged events.

Typically, in a standard Google Calendar system, users should subscribe to all rooms calendars before they can book the rooms. This website solves that problem where the users can immediately look for the available rooms at a particular day and particular time and book them, and don't have to subscribe manually to all room calendars because all calendars are managed only by Admin.

This website provides several capabilities for the users:

1. Admin Role:

Other than having user's usual capabilities, Admin has several other abilities, such as:

* Admin can organize every room in the office which is going to be displayed for the users by using “Add", "Edit", and "Delete” features in admin's "Room List" menu.
* Admin can manage every user that are registered in the website. Admin can update or downgrade a user's role. This feature is available in the admin's "User List" menu.

1. User Role:

* Users can see all of his/her events in the website "Home" menu.
* Users can reserve/book a room by using the reservation feature in the "Reservation" menu.
* Users can see and monitor all existing rooms in the office in the "Room List" menu.
* Users can modify its own username, first name, and last name in "Account" menu

## **🧰 Course of Development**

### 🗼**Batch 2 Development Process**

This website was initiated as a final project in Formulatrix SE Bootcamp Batch 2. The first development phase details could be seen in this public .docs file inside the attached [link](https://docs.google.com/document/d/1qdeQ4Wlt7XhKZQLiQb5PAzGIS5uxwj7R/edit#heading=h.gjdgxs).

### 🗼**Batch 2 Development Summary**

Batch 2 had made quite a lot of progress in developing the website. However, it has some drawback that could be improved:

1. The calendar that was used in the website was the user's own calendar. So, technically, new users/employees should add all of the room's calendars manually to their own calendar. This process exactly replicates the use of the usual Google Calendar process, just on a different platform.
2. Booking system didn't have a feature to provide and/or look for all available rooms in a particular day and particular time. So, users should book the room by looking for a vacant room manually and provide the details of the event directly.

### 🗼**Batch 3 Development Objectives**

Based on the last development from Batch 2, we decided to focus and build the core feature of this website:

1. To make the website handles all of the room's calendar by itself, so the user doesn't have to import all of rooms calendars on their own
2. To make the booking room experience less tedious for user by providing and looking for them all of the available rooms in a particular day and a particular time

There are other changes and/or improvements that we made based on Batch 2 development, you will see them as you go along reading this docs. Fasten your belt!

## **🛠️ Technical Overview**

Keywords :

1. ASP .NET Core MVC
2. Entity Framework
3. Google Calendar OAuth Credential

### 🔨ASP .NET Core MVC Framework

Initially in Batch 2, this website was built using Blazor Pages web development framework, considering its rapid development advantage. Here, we decided to completely rebuild the website's architecture using Model View Controller structure or MVC.

There are several benefits in using MVC framework rather than Blazor Pages. You can search them online, but our main concern here is to make the development of this website easier to be maintained.

MVC enforces a clear separation of concerns between the model (data and business logic), view (presentation and UI), and controller (handles user input and orchestrates the flow). This separation can lead to more maintainable and modular code.

To learn more on how to practically interact with ASP .NET Core MVC in VSCode, you can refer to these several sources:

* Beautifully written [tutorial](https://www.notion.so/ASPNET-MVC-Configuration-On-VsCode-Using-CLI-e0322b68567e4a8c8a0f0a8d48926e74) from Mbak Dhea (Batch 1 alumni)
* [Dotnet Tutorials](https://dotnettutorials.net/lesson/introduction-asp-net-core-mvc/)

### 🔨Entity Framework

EF provides a way to interact with relational databases using a higher-level, object-oriented approach. In essence, Entity Framework bridges the gap between application's object-oriented code and the relational data stored in a database.

EF is used in this website to work with a database that contains User's data and their Role and also Room's data. This context will be elaborated more along the docs. SQLite and SQLiteStudio were used to develop this website.

### 🔨Google Calendar OAuth Credential

To run this website, we need to make an Admin's Google Account that will hold all of the room's calendar. Then, we need to get its OAuth Credential so that the application will be able to access Admin's Google Calendar data -- in other words, all rooms's calendar in Google Calendar -- and perform CRUD action to the data. You can make a new OAuth credential by going to Google Console.

### 🔨General Requirements

To build the website, these dependencies/tools are needed:

1. NET SDK v7.0.305
2. ASP .NET Core MVC RazorPages
3. Sqlite and SqliteStudio
4. Google Calendar OAuth Credential
5. Visual Studio Code

## **💹 Diagrams**

### **🌲Class Diagram**



### 

### 

### **🌲Entity Relation Diagram**

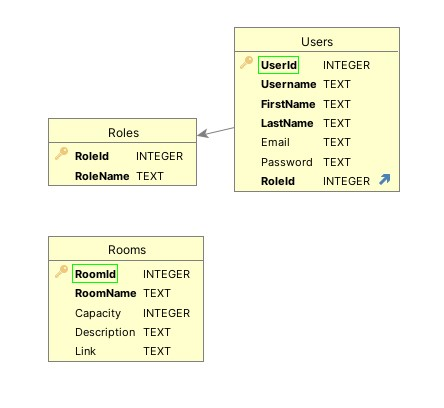
****

Image above explains the relationship of the tables inside the database (will be called DB afterward). There are 3 tables inside the DB:

1. Roles

This table represents a list of roles that exist in the website's context. It contains RoleId and RoleName. The current roles are "Admin", "Manager", and "Engineer". Generally, there are no significant differences between Manager and Engineer Role (in this website context), both are considered as ordinary User.

1. Users

This table represents each user's identity, such as "UserId", "Username", "Email", "Password", and "RoleId". It also has the user's first and last name attributes. This table has a relationship to the Roles table. One user can have only one role, but one role can be obtained by many users.

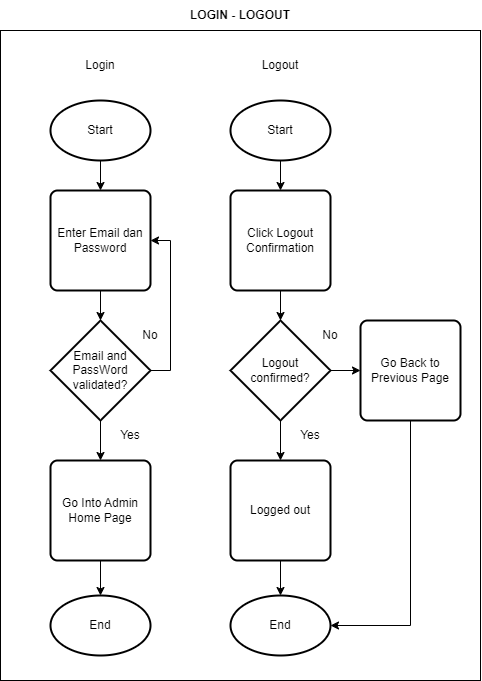
1. Rooms

This table represents all existing rooms in the office. It has attributes such as "RoomId", "RoomName", "Capacity", "Description", and "Link". This link is filled with each room's Calendar ID. You can look for this ID inside the calendar settings.

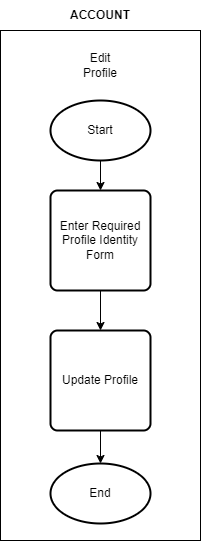
### **🌲Flow Diagram**

There are 5 different flow diagrams based on the flow context of the use-case. They are explained in these pictures.

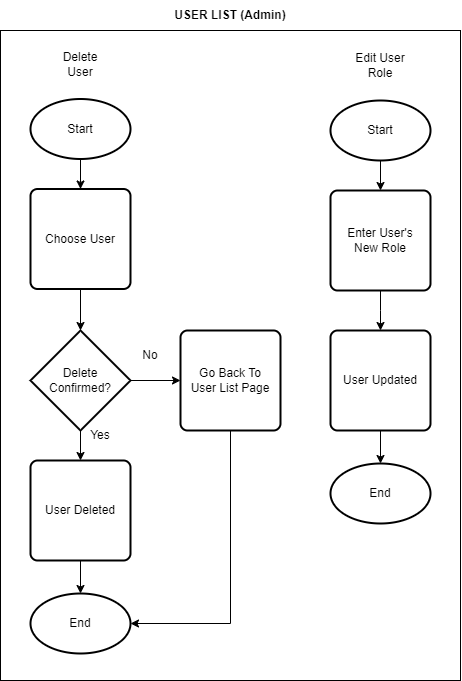
1. Login - Logout



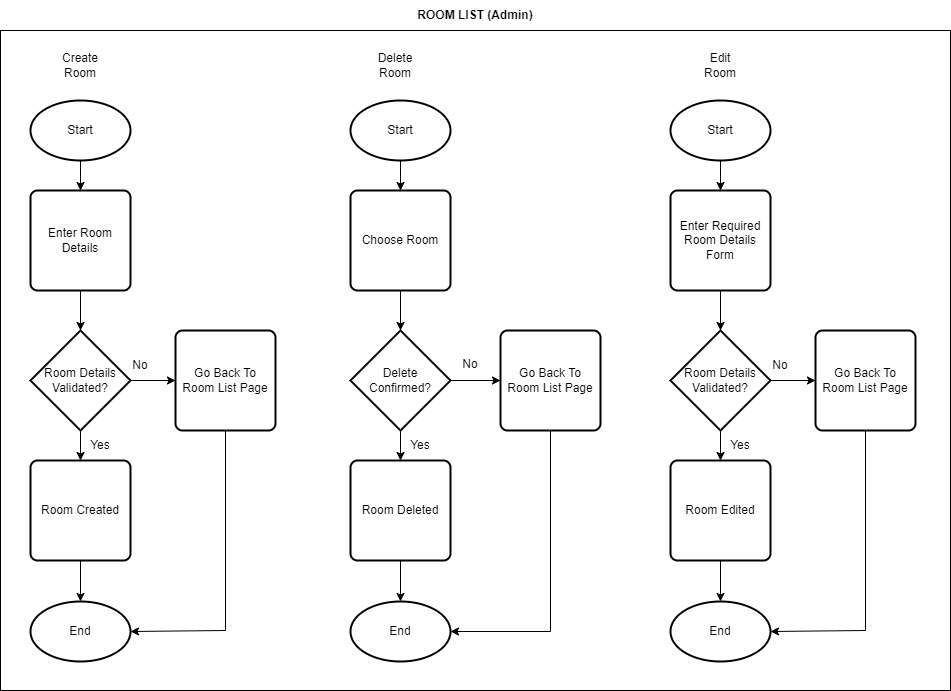
2. Account



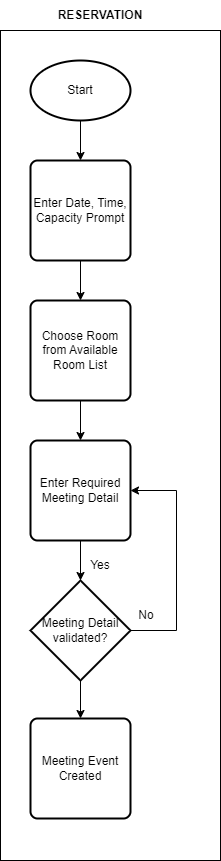
3. User List (Admin)



4. Room List (Admin)



5. Reservation



### 

### 

### 

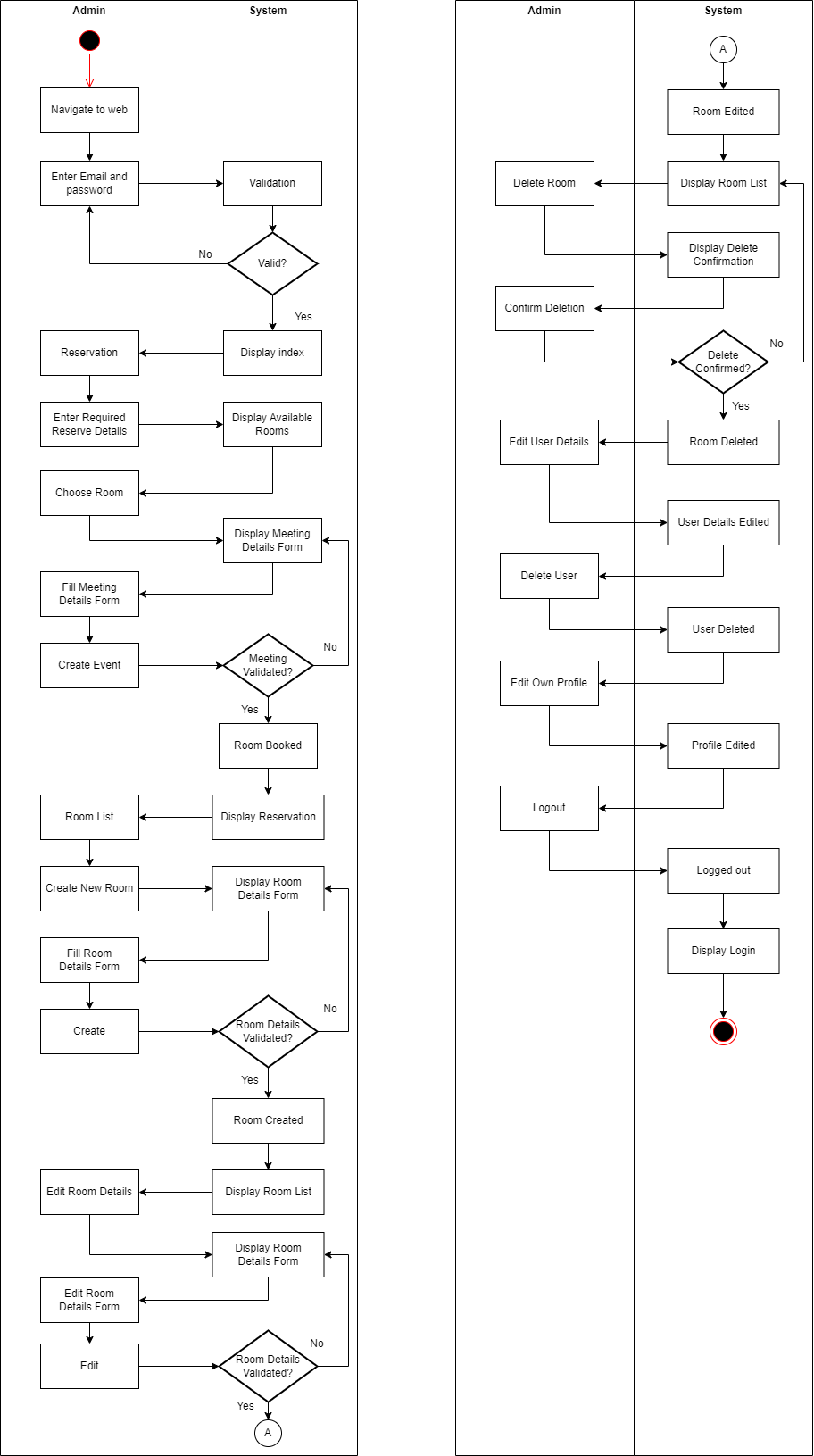
### 

### 

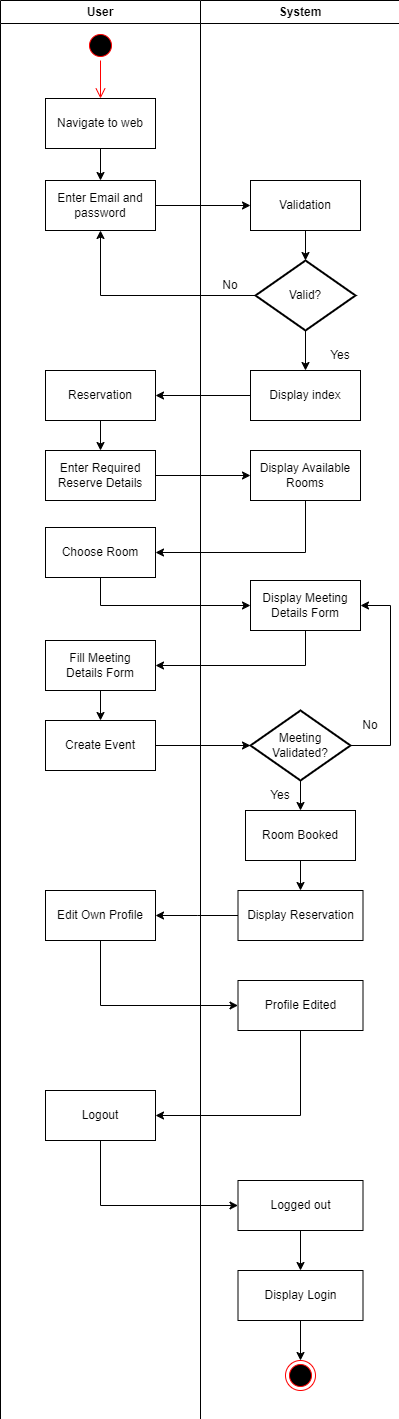
### 

### **🌲Activity Diagram**

1. Admin



2. User



## **🆕 Use Case Explanation**

*both : Ordinary user and admin have access to the feature*

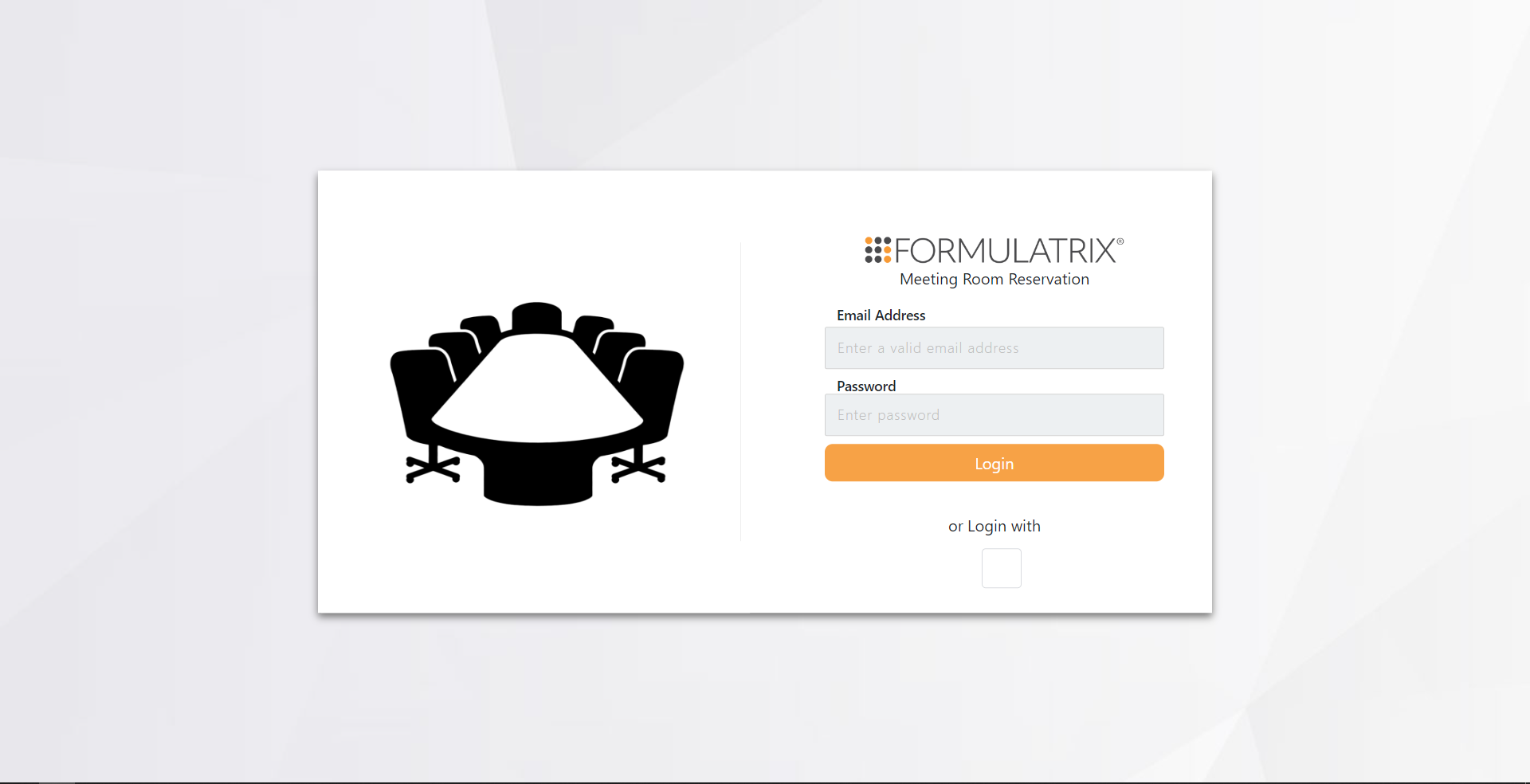
*admin : Only admin that has access to the feature*

*both-Adm : Ordinary user has read only access, whereas admin has CRUD access to the*

*feature*

### 📘 Login

#both



Login page doesn't differ that much to the one that was built in Batch 2. Here, users should enter an Email and Password that was registered in the website.

* Before :

In Batch 2, users could log into the website using their own Google account. But, the last development had its own drawbacks. When user logged into the website, the calendar that was registered was \*\*\*user's own calendar\*\*\*, not the system's calendar (as mentioned earlier in the docs). So, in a way, the login process was related to the overall calendar data, which is not what we want. We want the website to manage its own all rooms calendars, this part will be elaborated more as we go.

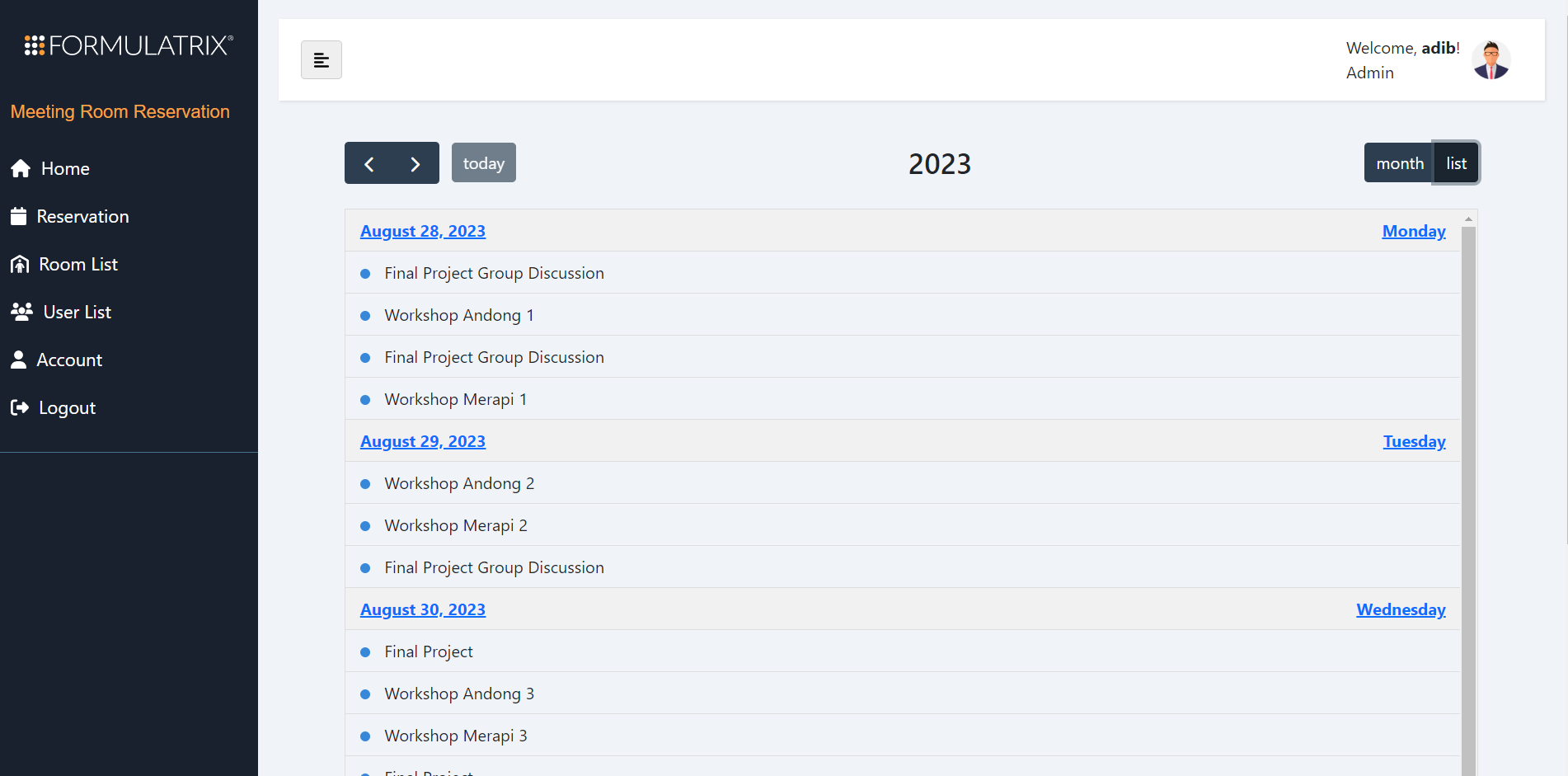
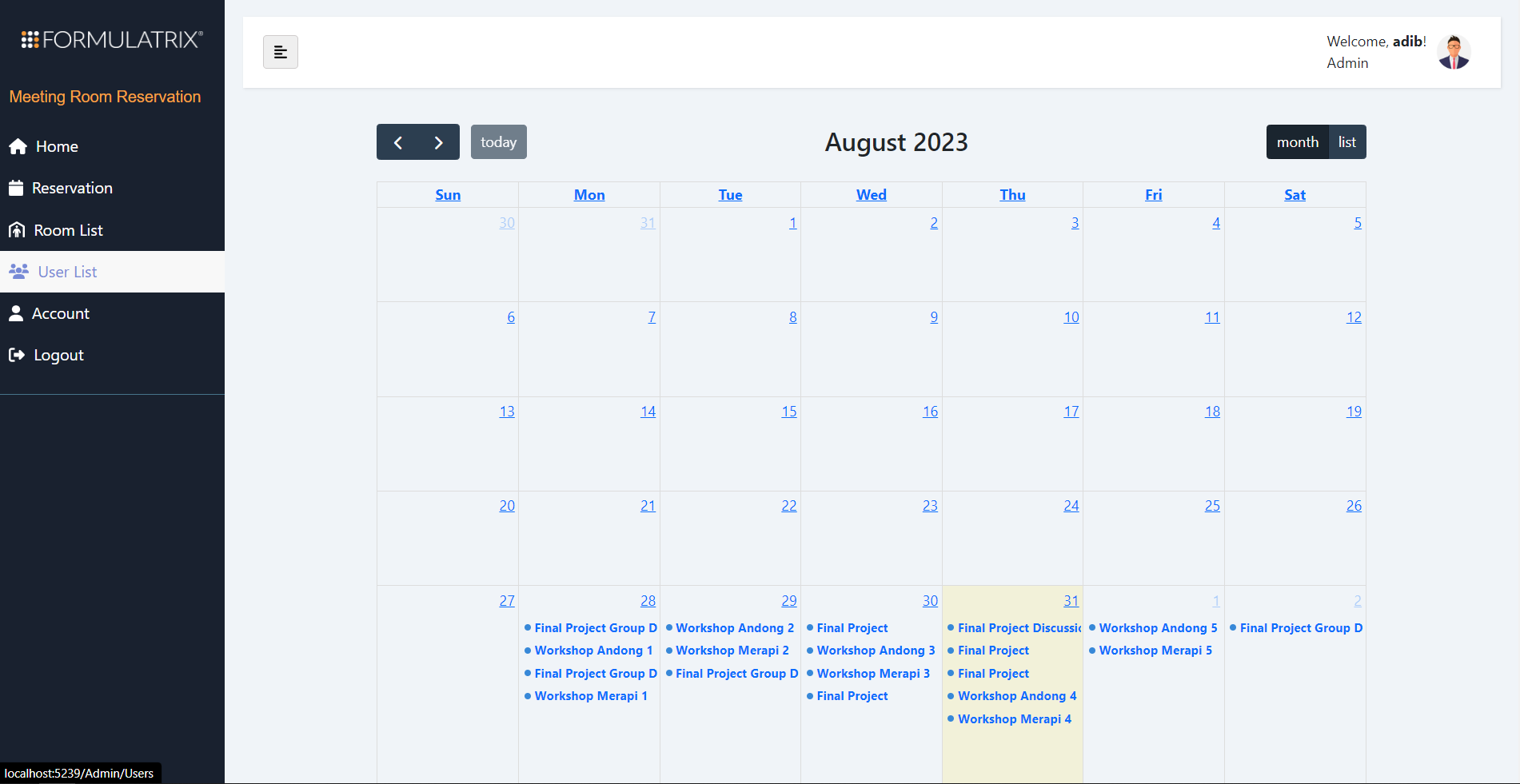
* After :

We could log into the website using email and password that was registered in the DB. Note that the registered accounts are added manually using SQLiteStudio. So, if you want to log into the website, you could check the DB using SQLiteStudio and use accounts that are already registered there, or better yet you can make your own account manually.

We tried to integrate the Google Login feature, but unfortunately we didn't have enough time to complete it. This will be added in the Future Development Suggestions part.

### 📘 Home

#both



The Home menu displays a calendar that contains all registered events in all rooms. This Home menu differs a little bit than the last development from Batch 2.

* Before :

Home appearances in Batch 2's development were built using Google Calendar's IFrame feature. It had its own advantages and disadvantages. First of all, it allows developers to easily make a calendar display, just by copying the provided IFrame link from Google. Batch 2 used this IFrame to display all rooms calendars by giving users options to choose between all room calendars. But the drawbacks of using IFrame is that, -- as far as we know -- we cannot interact with it in any way. IFrame only displays calendar periods.

* After :

In the future, we want to develop this Home display so that users can interact with it, maybe even doing some CRUD actions from it.

So, we have been developing the Home menu's calendar display using the fullcalendar package (link attached below). This package allows developers to build custom calendars for many purposes. The last version (v6.1.8) of this package also allows developers to integrate Google Calendar with this customized fullcalendar.

As you can see in the image above, users could switch between the calendar based on the month. Users could also change the view from month-grid display to list view.

For now, this Home calendar displays all calendars and all events inside one calendar. We realize that this will probably make the Home appearances more and more messy and violate the private meeting principle. The Home calendar should only display the currently logged in user's events. This part will be added into Future Development Suggestions.

[FullCalendar](https://fullcalendar.io/) Info: To integrate Google Calendar data to FullCalendar, you might need to generate a new API Key, different from earlier OAuth credentials.

### 📘 Reservation

#both

We made lots of changes in the Reservation section from Batch 2's last development.

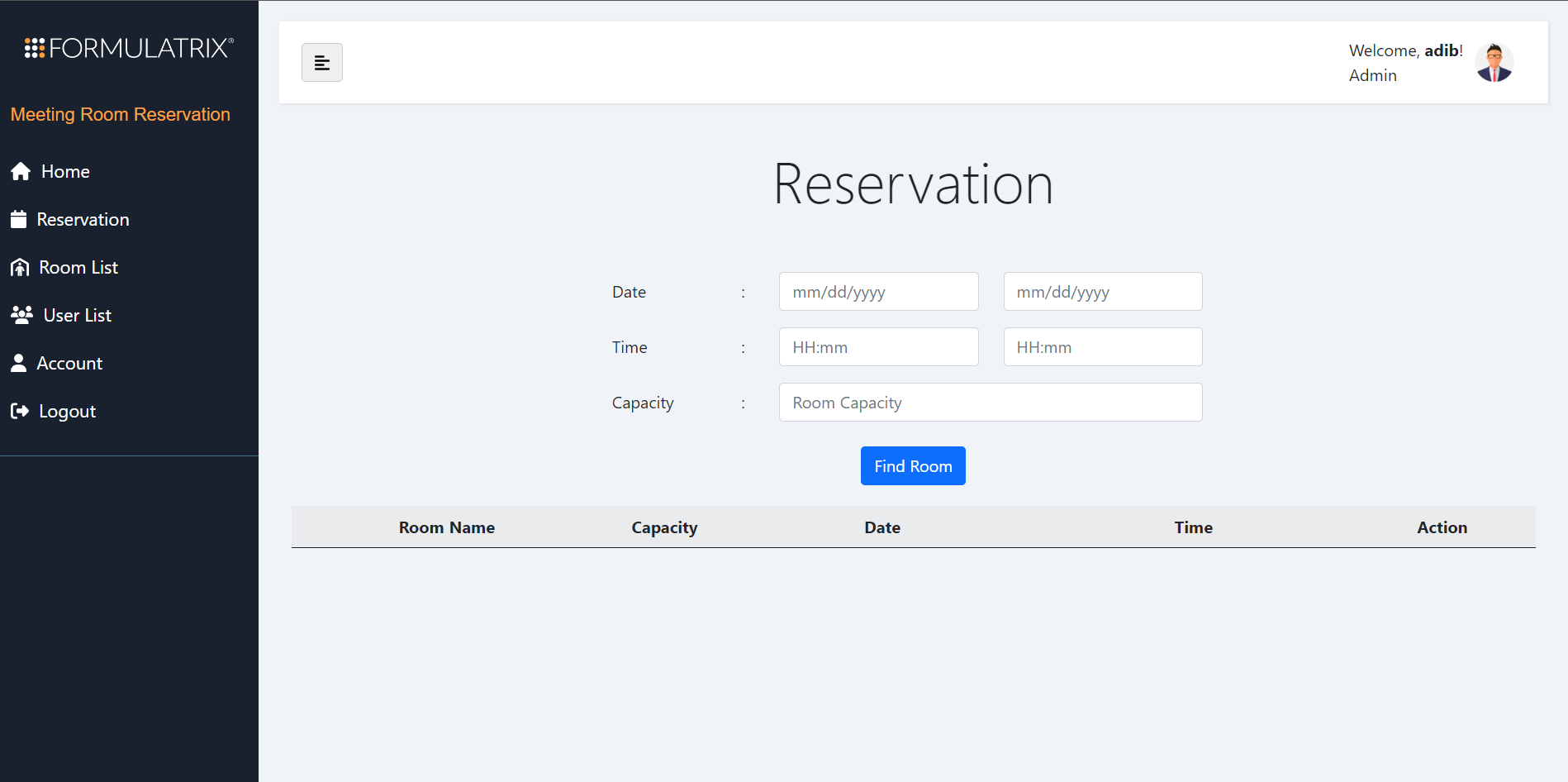
* Before :

Last development's reservation system was still limited in several ways. First of all, users were still using their own Google Calendar. Second, users had no way to know which rooms that are available to be booked/reserved on some particular day and/or time, so users could only guess and straight off the bat book some particular room for some particular timespan. This doesn't eliminate the tedious part of using the original Google Calendar, if anything this system only makes it harder, no offense.

* After :

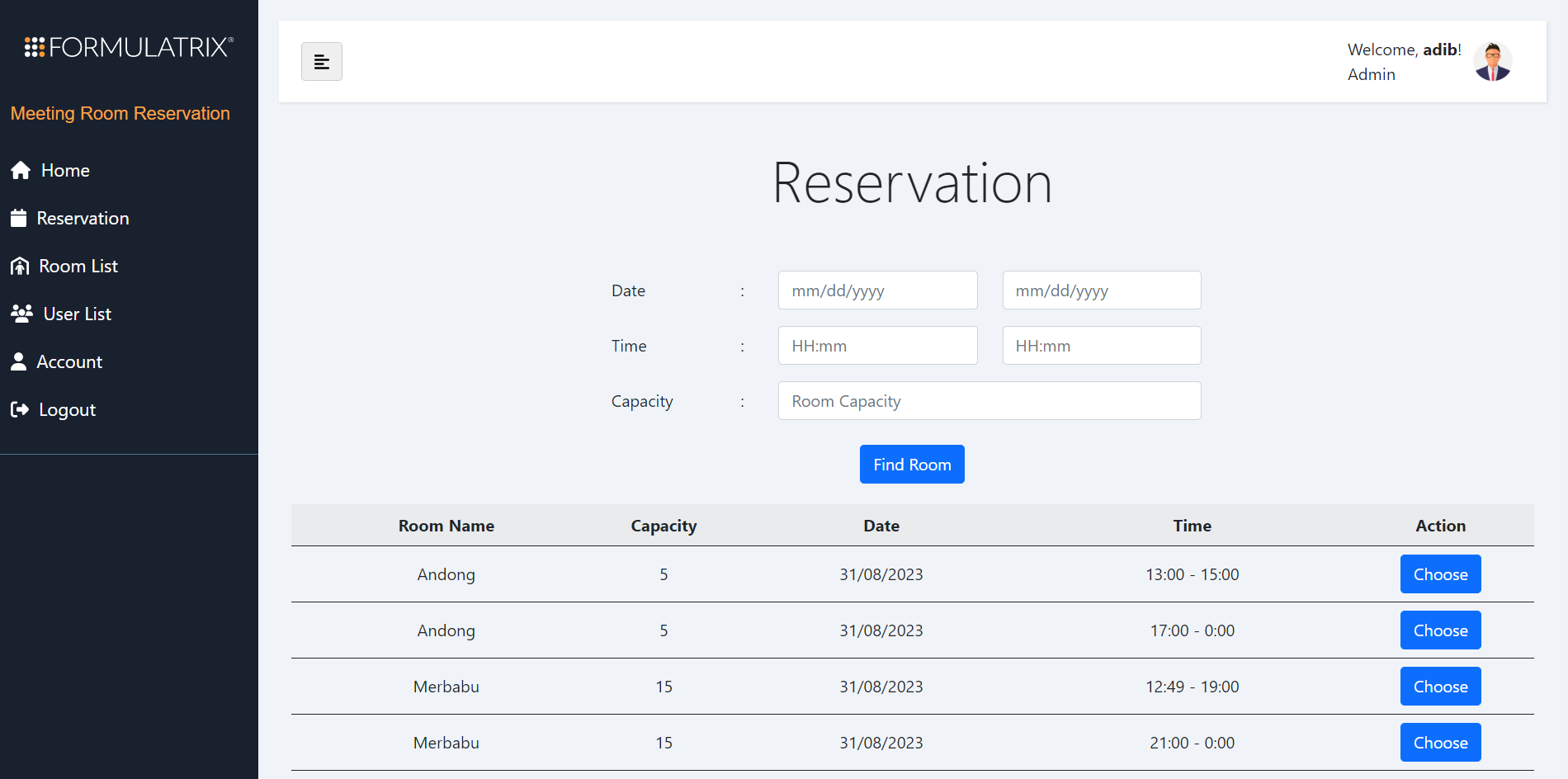
We successfully made changes in this system. We will guide you through the steps on how users could book/reserve a room for some particular day and/or time.

1. Reservation Page



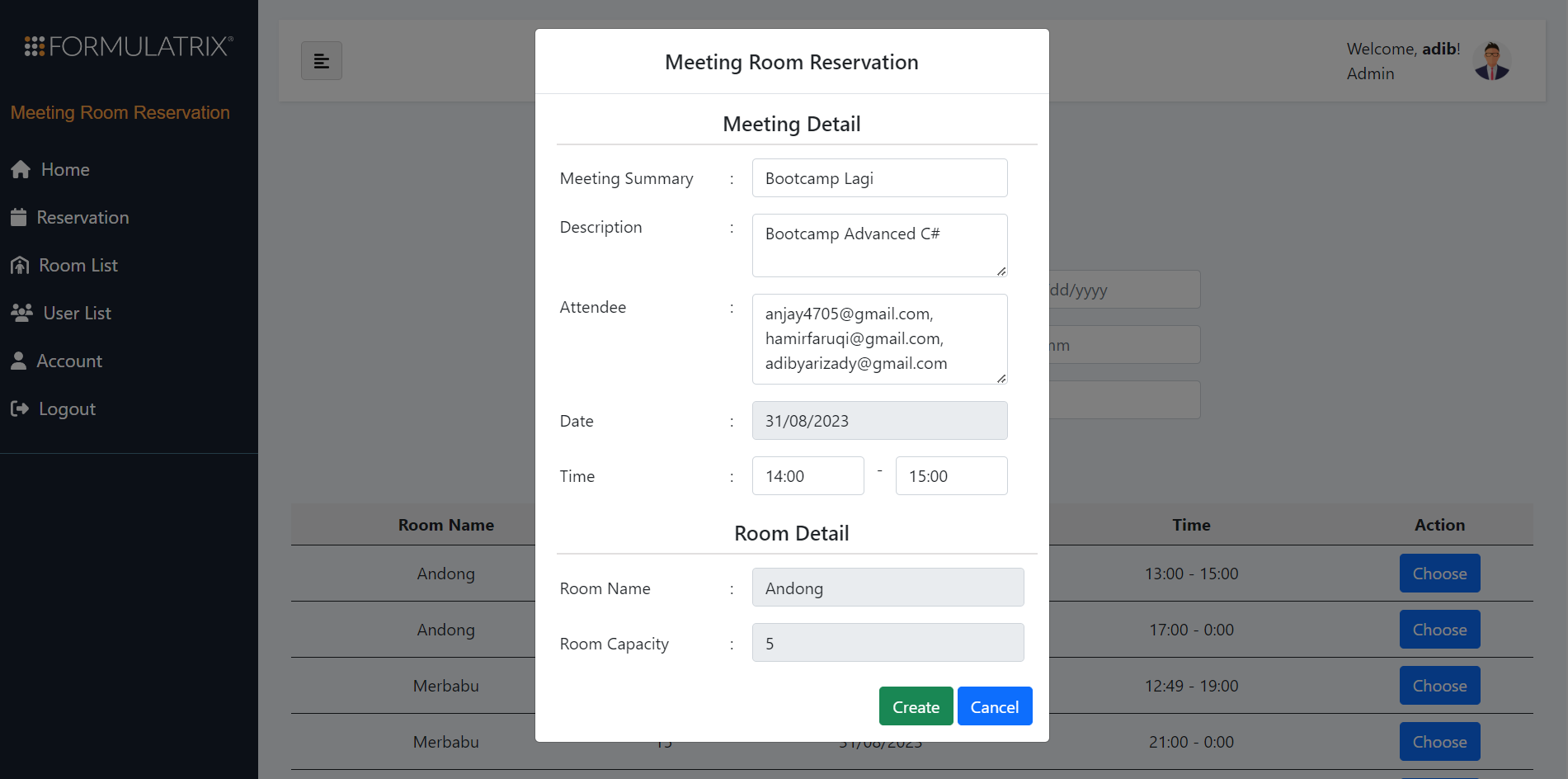
The Reservation landing page was shown in the picture above. It has some prompts to be filled optionally by the user. The prompts are start date - end date, start time - end time, and room's required capacity. Because these prompts are not mandatory to be filled, different combinations of the filled prompt will result in different available rooms list context. You can see this by trying all of its combinations.

1. Finding Available Rooms



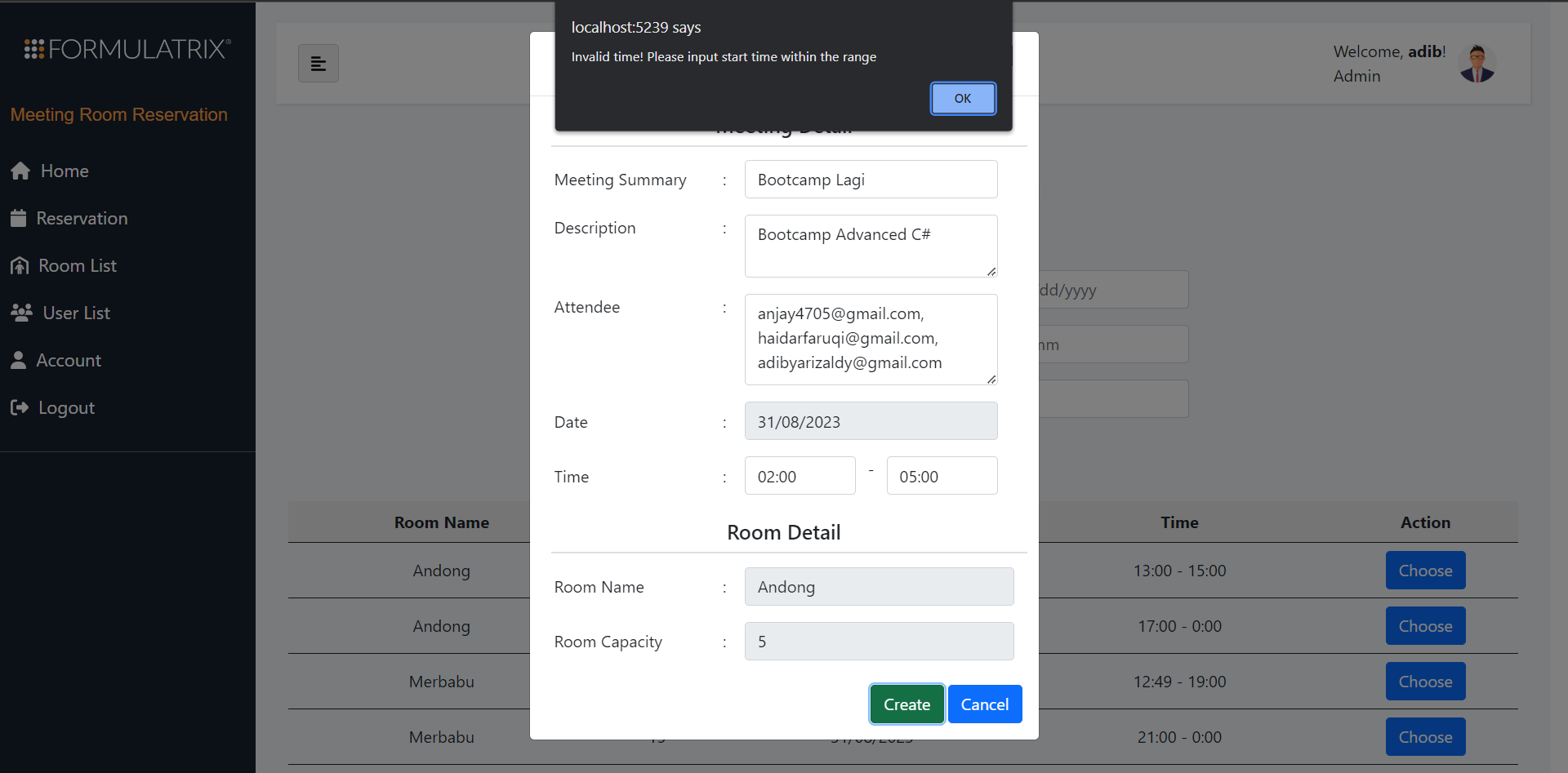
When all prompts are empty, and we click the "Find Room" button, we will be provided with a list of rooms that are available to be booked/reserved in the present day. You can see the detailed information of each room inside the tables, the columns display Room Name, Capacity, Date, Time when the room would be vacant, and a button that reads Choose. When, user has decided which room to book/reserve, the user can click the "Choose" button.

1. Meeting Detail Pop-Up



After the "Choose" button is clicked, we will be faced with a pop-up with Heading titled "Meeting Detail". Here, users could fill all of the prompts to make an event in Google Calendar. These prompts are "Meeting Summary" which is the event's title, "Description" of the event, "Attendee" or users that will attend the event, and the "Time" that user will specifically book for the event. Date, Room Name, and Capacity are automatically filled by the system and can't be filled by the user.

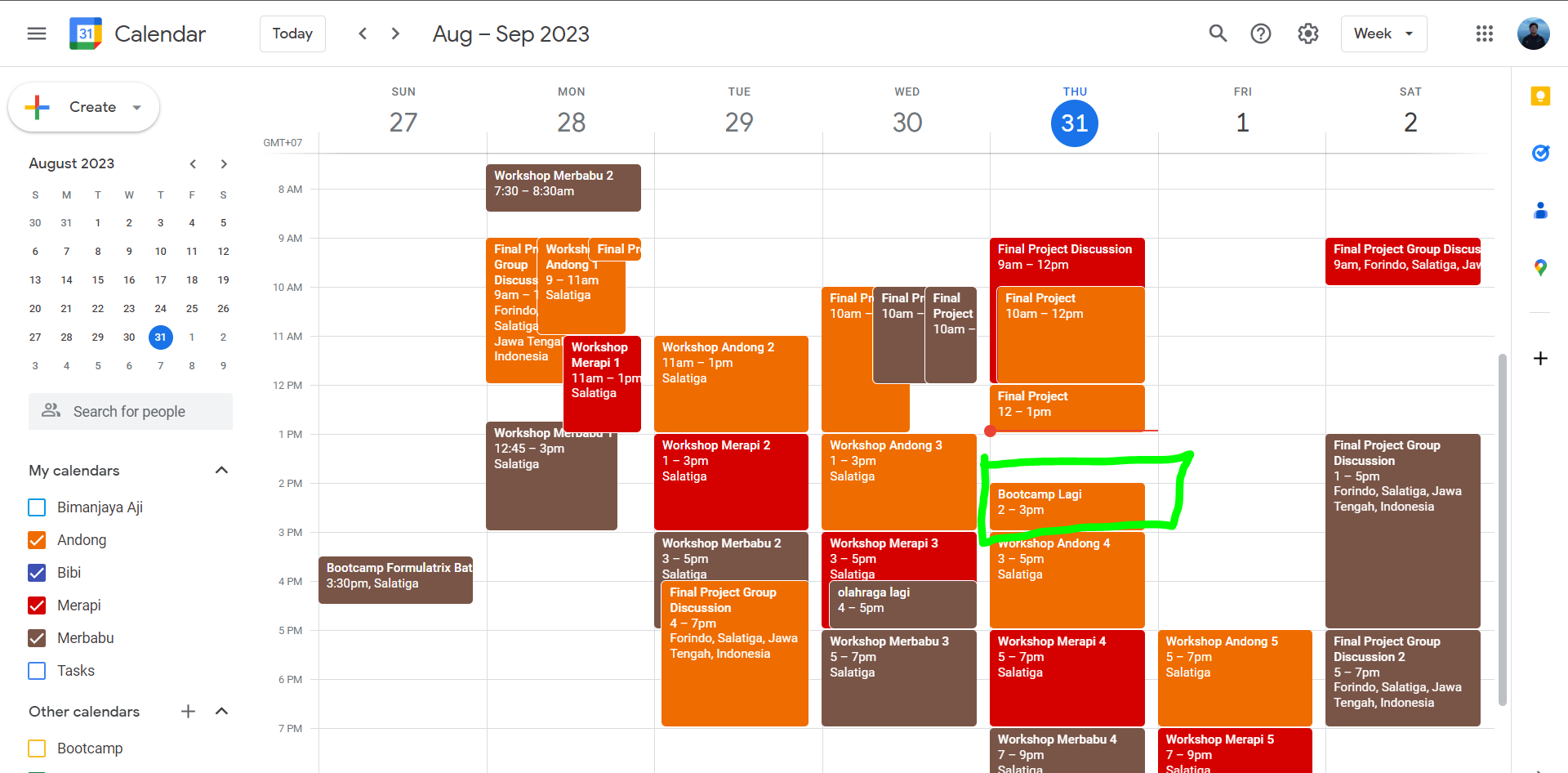
Note that, to add an attendee, the user should type out all of the attendee's email and separate them using a comma.



When the user has already chosen a room to be booked and inserted some invalid range of time that is printed in the earlier list, the system will display a warning "Invalid time!". So, users should not insert time ranges outside the time boundary that is printed in the Time column of the table.

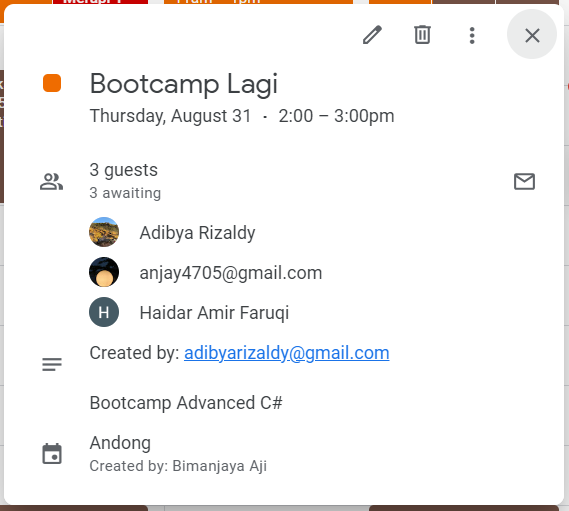
In the example above, we chose an option Andong that is available from 13:00 to 15:00, but instead we insert a 3 hour timespan from 02:00 until 05:00. By the way, in practice, users won't be able to choose invalid time because the dropdown provides valid time only. In other words, you can only insert invalid time by typing it manually.

1. Admin's Google Calendar



If the event is successfully created, it should be present in admin's Google Calendar display as shown in the picture above. As you can see in the green box, our event was created.

1. Event's Details



The details of the event is shown in the picture above. It is filled exactly with our earlier prompt inputs.

Because we are using admin's Google Calendar (in this case Bimanjaya Aji's calendar), the event is noted to be created by Bimanjaya Aji. As far as we know, Google Calendar doesn't provide a way to change this value to another Google account. So, we came to make a solution to automatically include the creator of this event in the "Description" section. As you can see, the creator of the event is the user with email adibyarizaldy@gmail.com, the current user.

### 📘 Room List

#both-adm

In the Room List menu, we could create a new room, edit existing room details, and delete rooms.

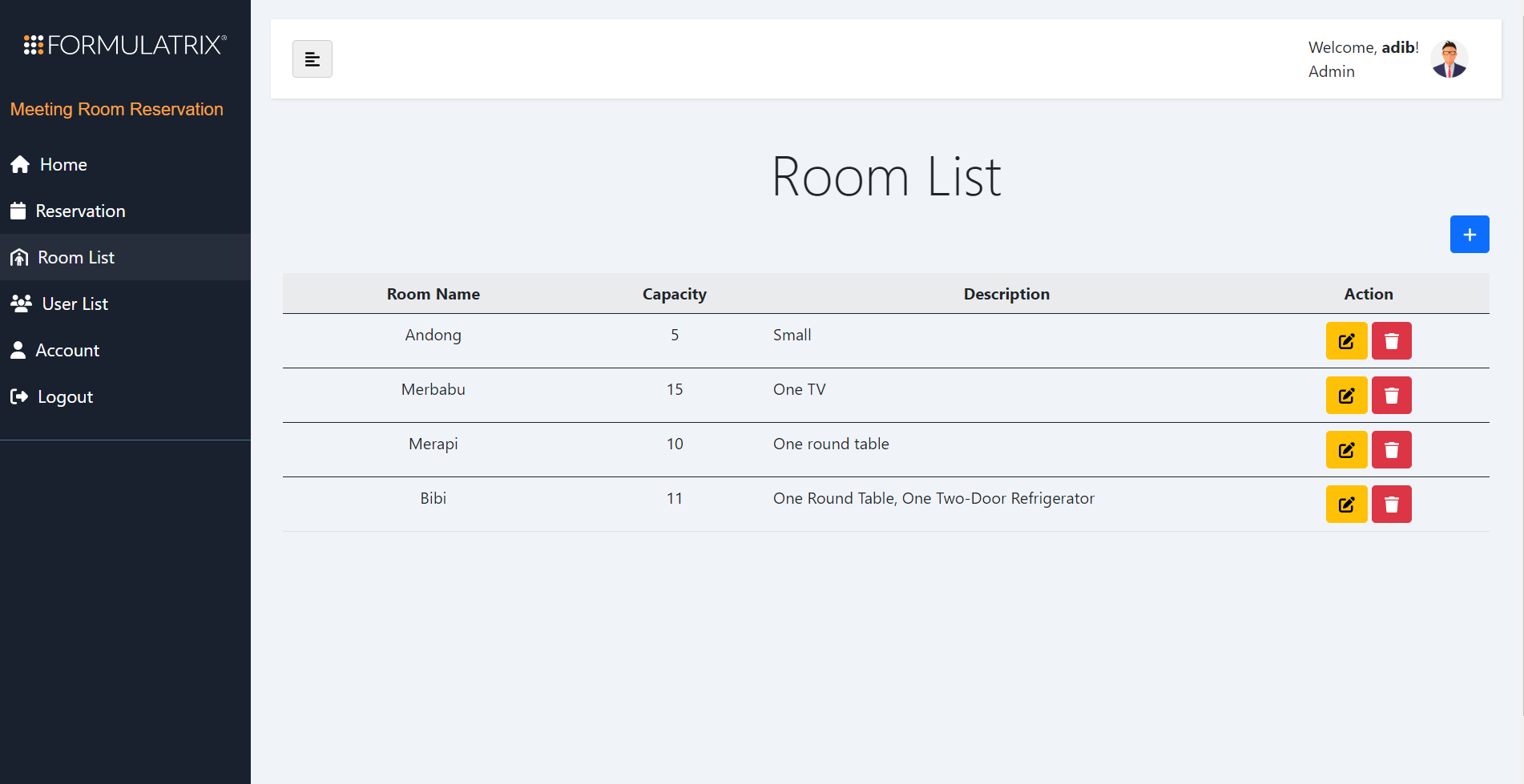
* Before :

Batch 2's room management was separated in different menus. The separation was between "Add Room", and "Select Room". The Add Room menu allows admin to add/create new rooms, whereas "Select Room" allows admin to edit and delete rooms, and also delete events inside each room.

* After :

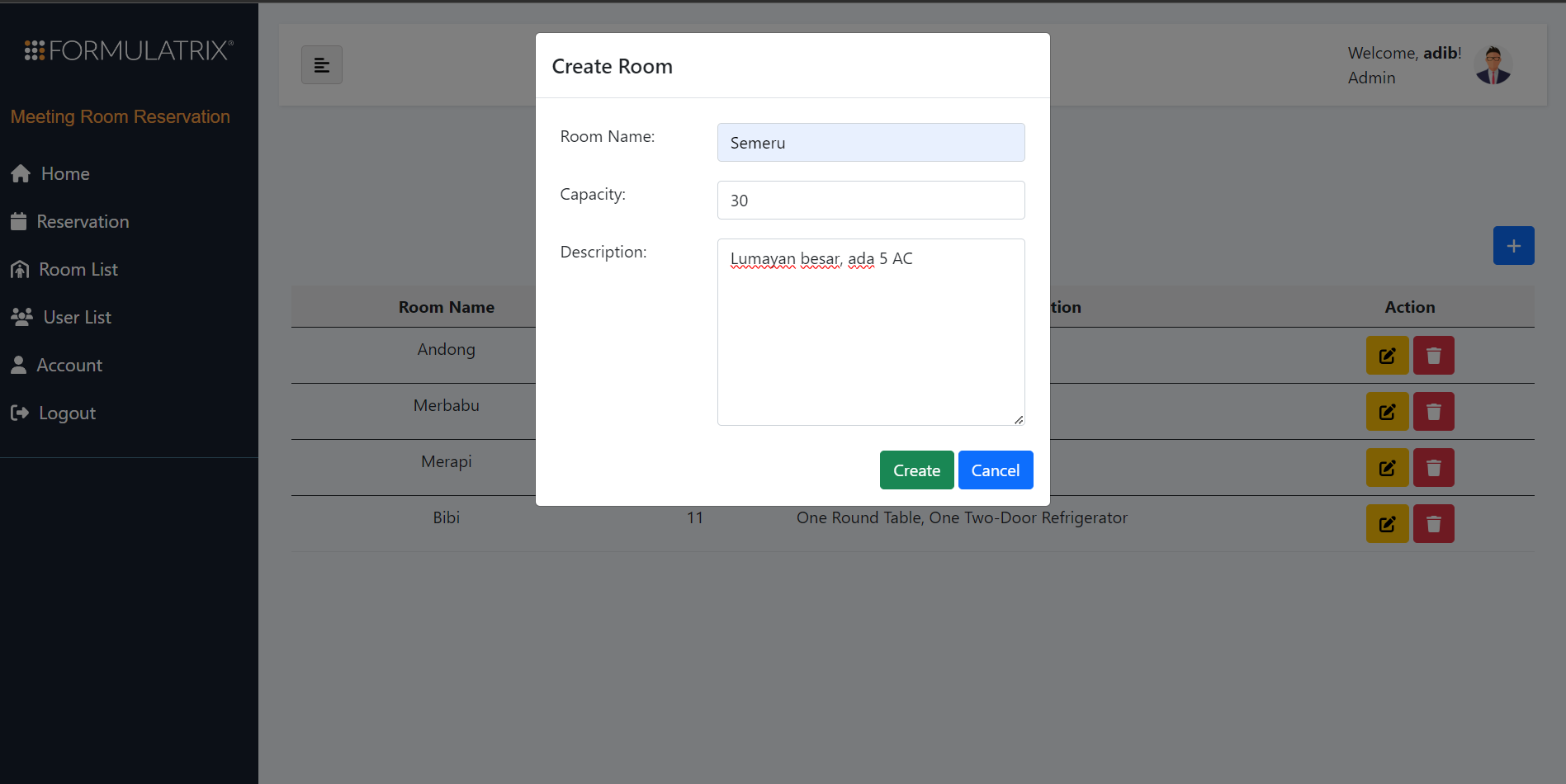
We decided to merge all of this feature into one page. So inside the Room List page, we could create a new room, modify existing room details, and delete rooms. The following passages will explain more on how to use each of these features.

1. Room List Page



In the Room List page, we could see a table of rooms. The columns are Room Name, Capacity, Description, and Action that contains Edit and Delete buttons. This one is quite straightforward.

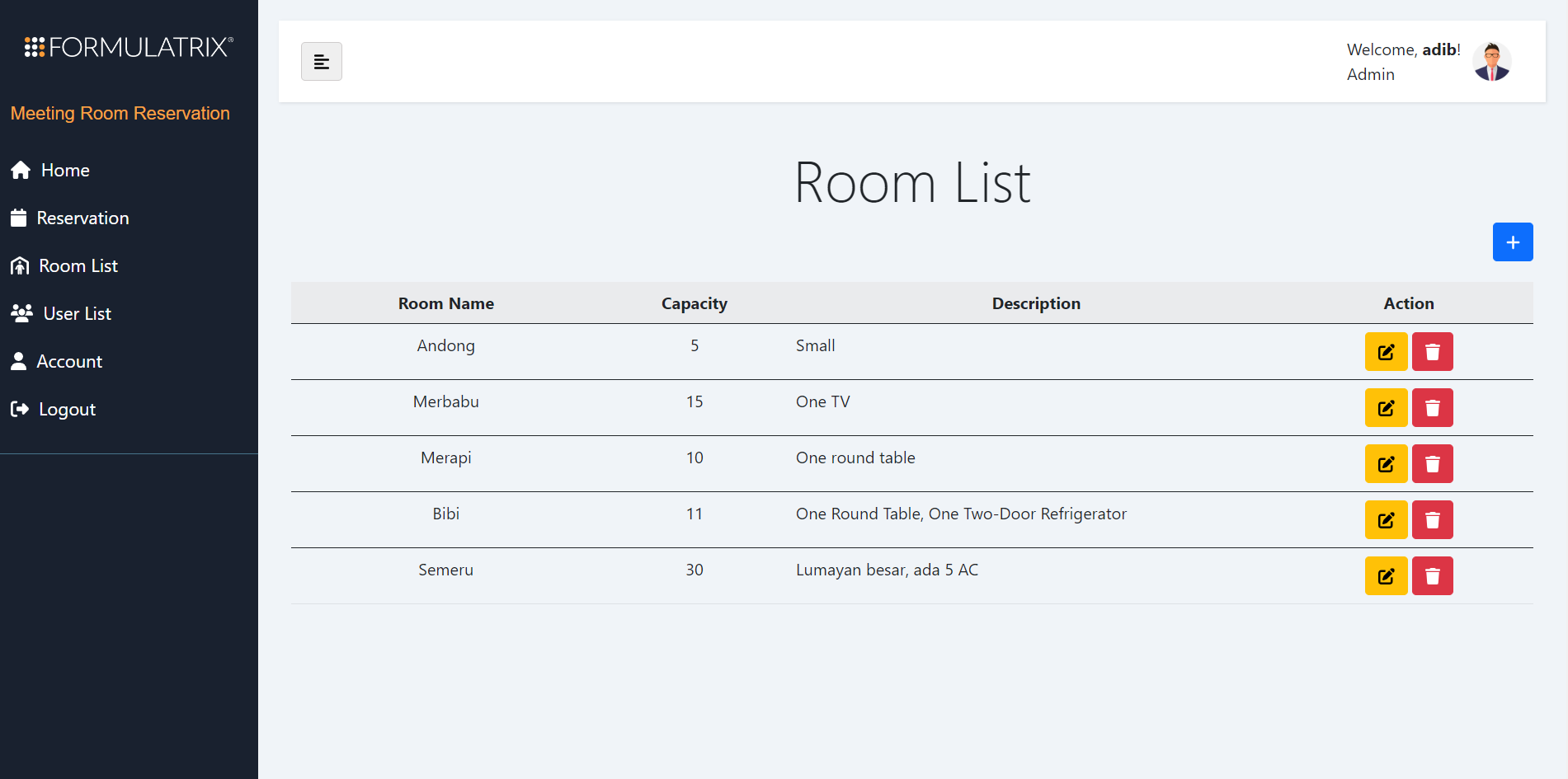
1. Create Room



Admin has the ability to create room by clicking the blue with plus icon at the top right of the screen.

After clicking the blue-plus icon, a pop-up will appear and the admin can fill in the prompts provided. There are Room Name, Capacity, and Description. Note that all of them are required.

After filling the form, simply click the green-create button, and the room will be created.



Above image shows that a new room with the name Semeru has been successfully created.

When creating a new room, that new room data will be added to the Room DB.

Aside from inserting new room data to the DB, the system will also automatically create a new room calendar inside Admin's Google Calendar.

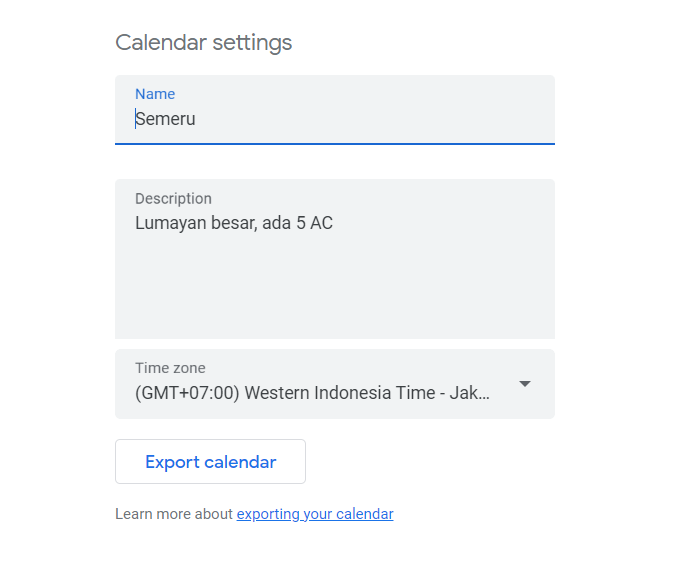
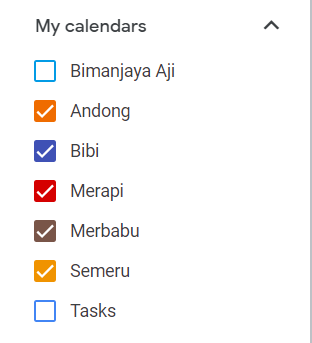
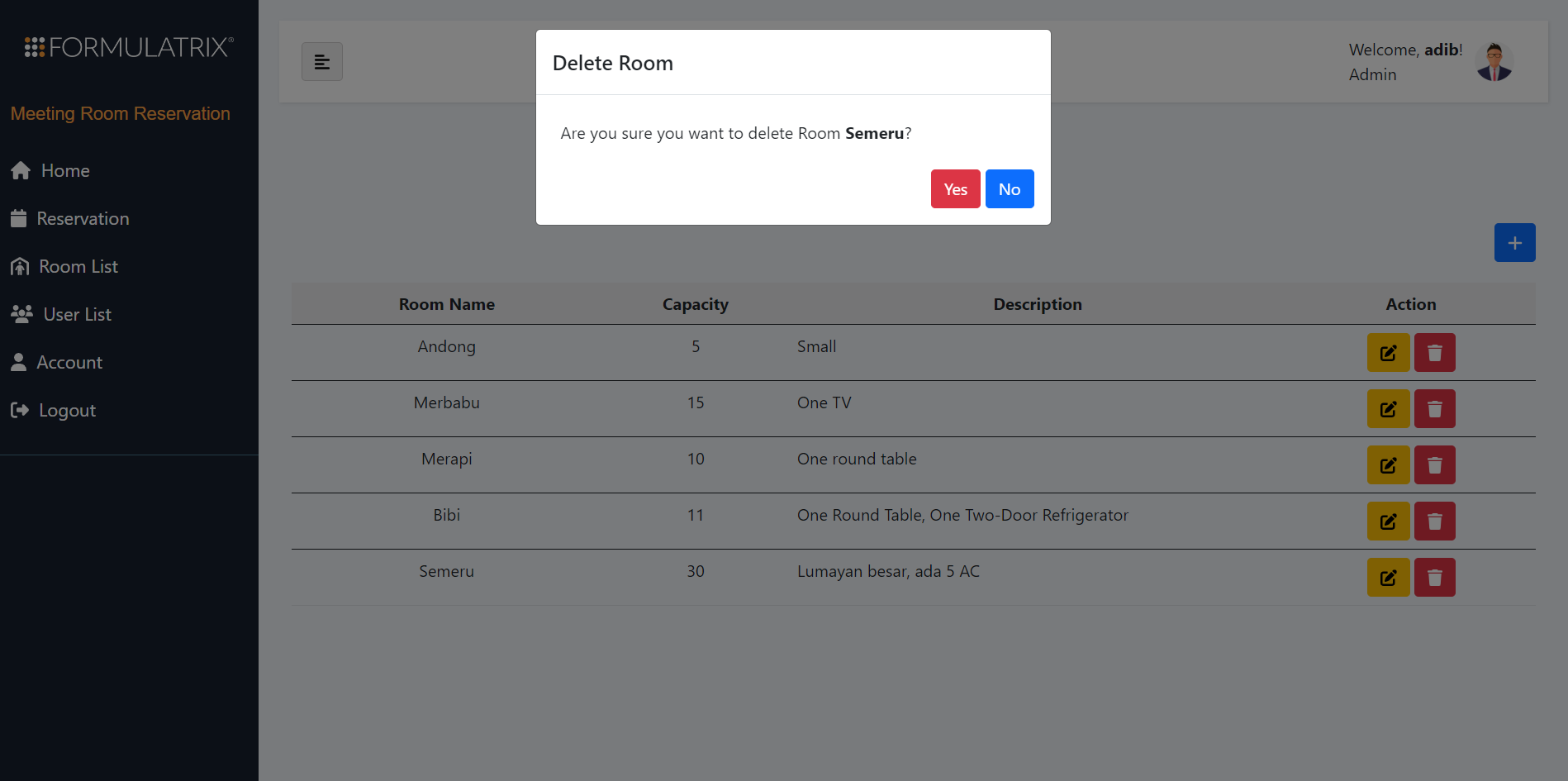


Image above shows that a new room with name Semeru has been successfully created inside Admin's Google Calendar.

1. Delete Room



Admin also has the ability to delete a room from the Room List menu. This action will delete that particular room data from the system's local DB and also automatically remove that room calendar from Admin's Google Calendar.

To delete a room, Admin can simply click the red-trash button at the right side of the table.

After clicking the red-trash button, a confirmation pop-up will appear, validating the delete action. To proceed deleting, just click Yes.

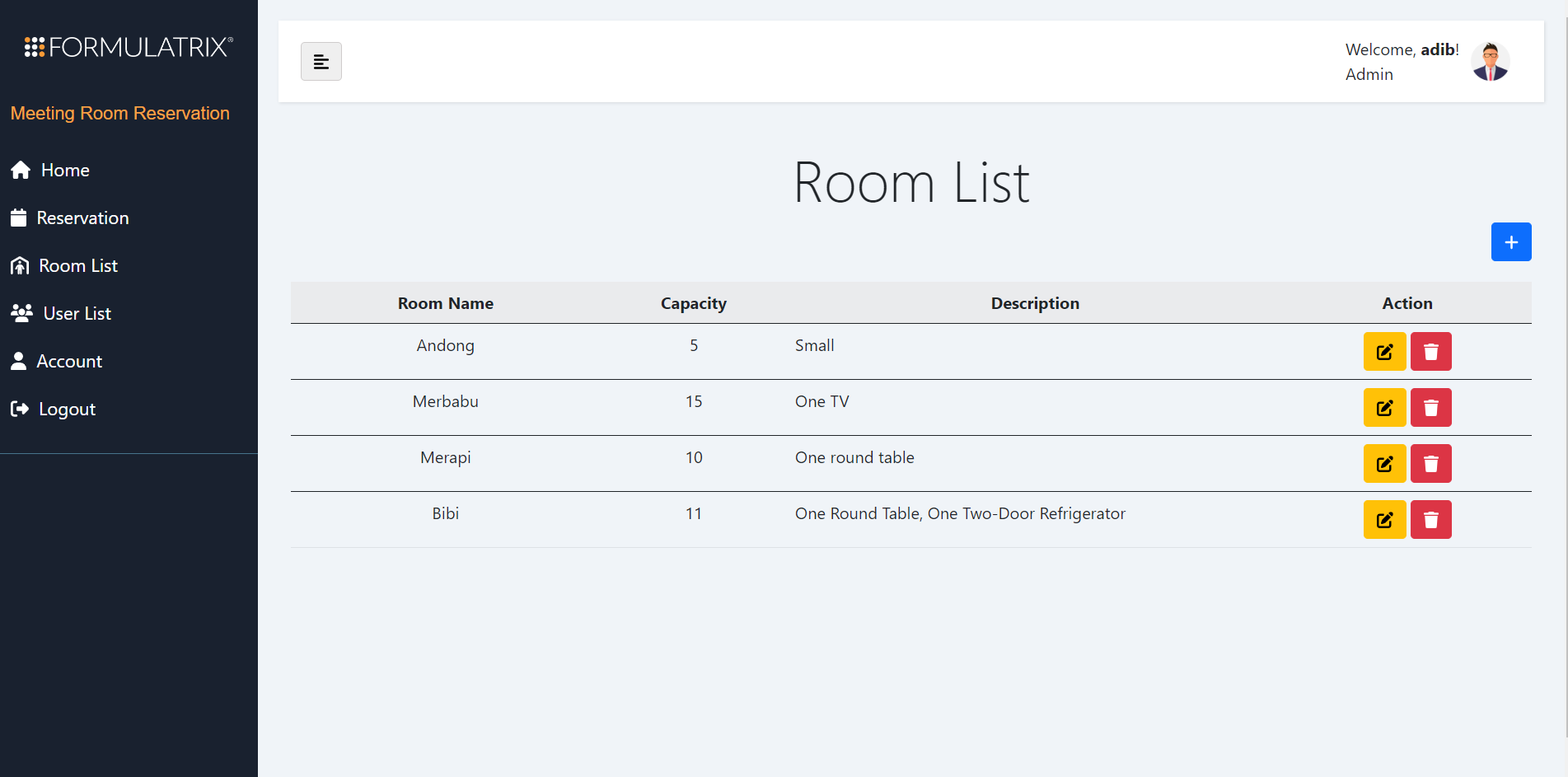
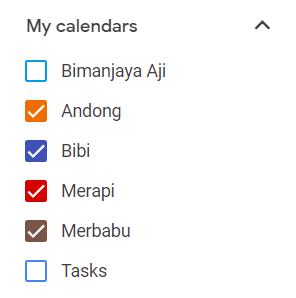
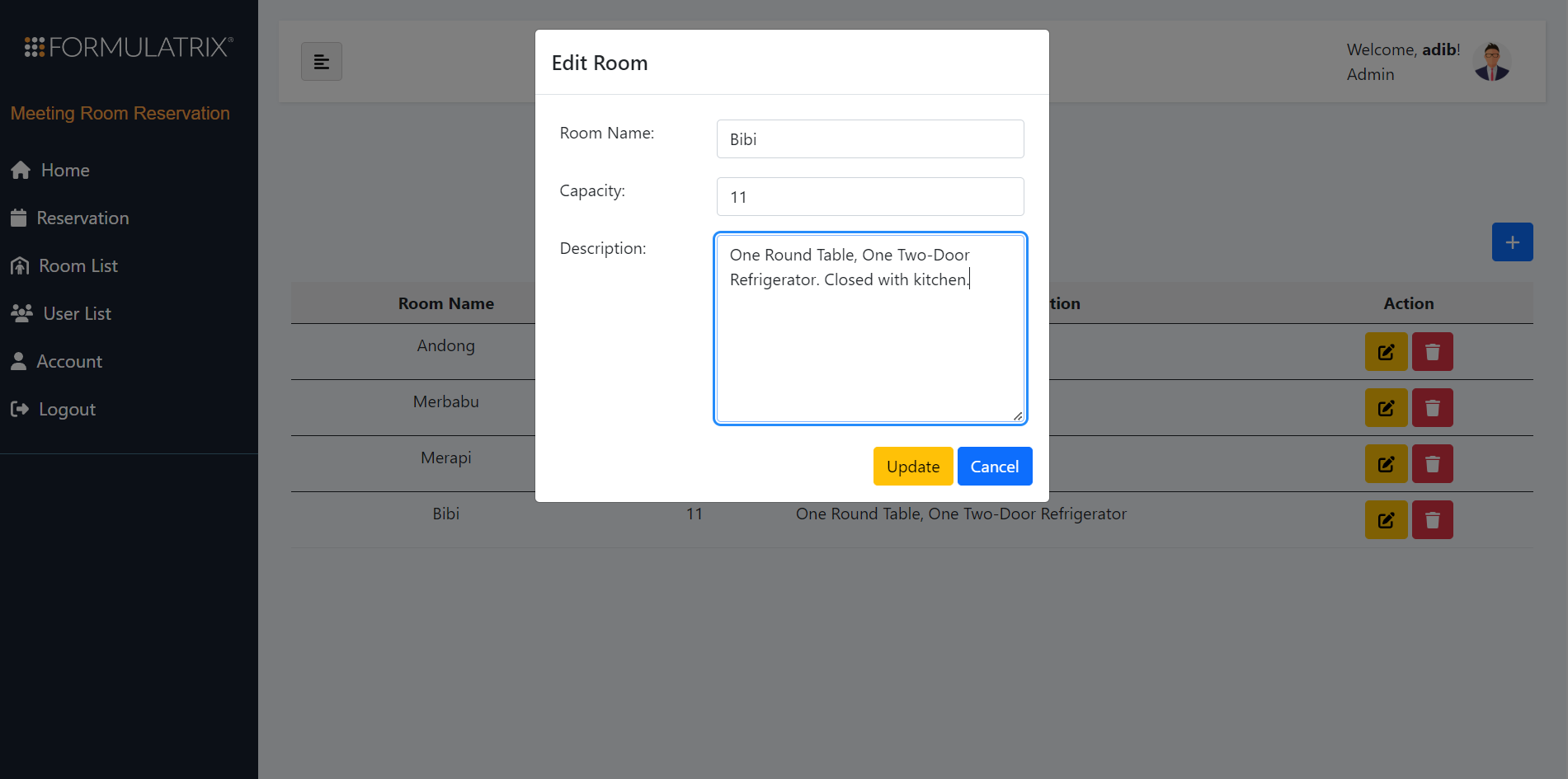


Image above shows that the previously created Semeru room has been successfully deleted from the Room List. Now, we can validate that from Admin's Google Calendar.



Inside Admin's Google Calendar, there was no Semeru. This means that the deletion process of the Semeru room was successful.

1. Edit Room



Admin also has the ability to edit room's details. By details, we mean the room's name, capacity, and description.

This action can be achieved by clicking the yellow button beside the delete button.

After clicking the yellow button, a pop-up will appear that shows the 3 properties that can be edited.

After editing the properties, to save the changes, just click the yellow Update button.

Just like Delete action, Edit action also handles -- in this case, modify -- the data inside the system's local DB and in Admin's Google Calendar.

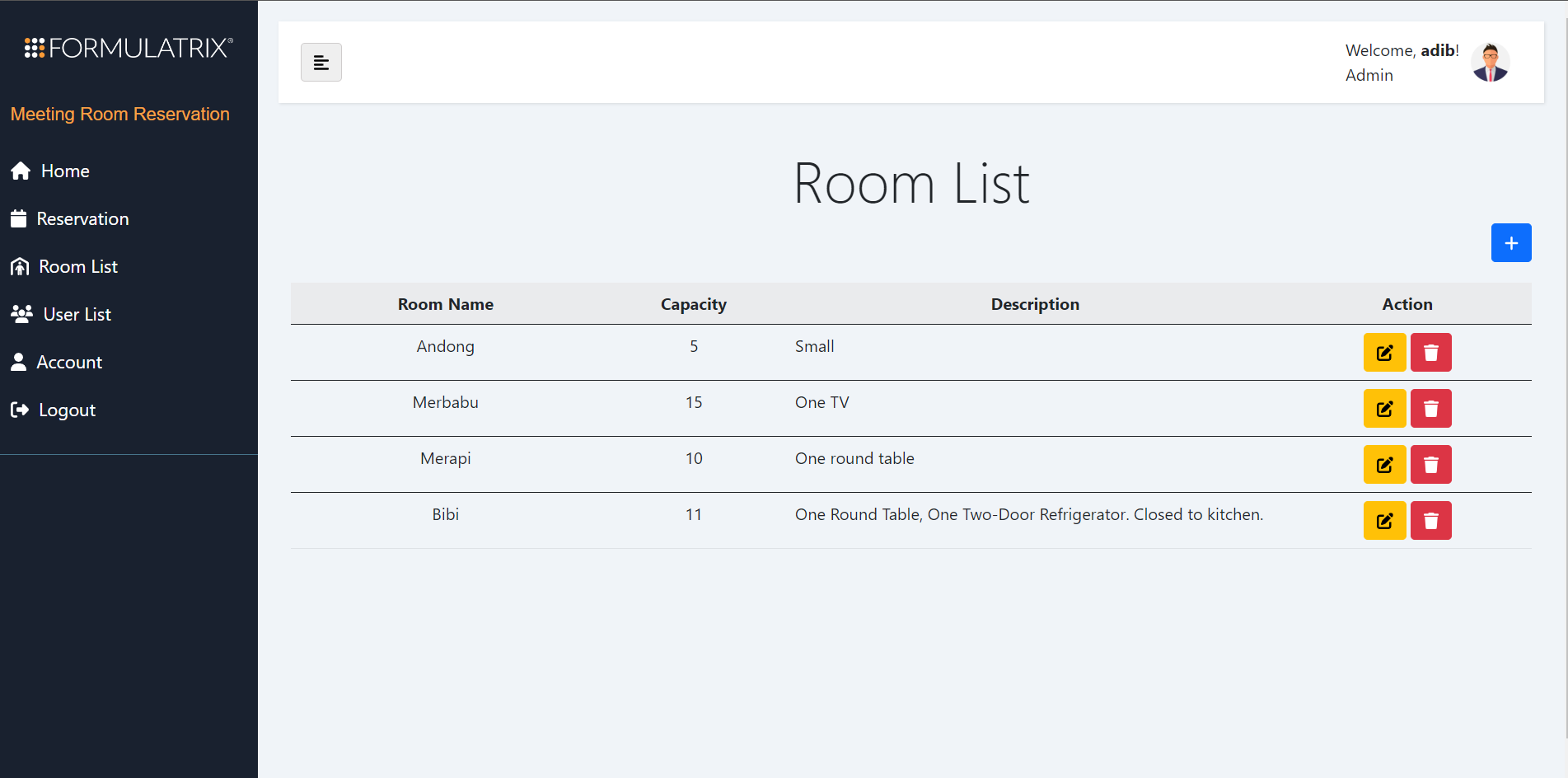
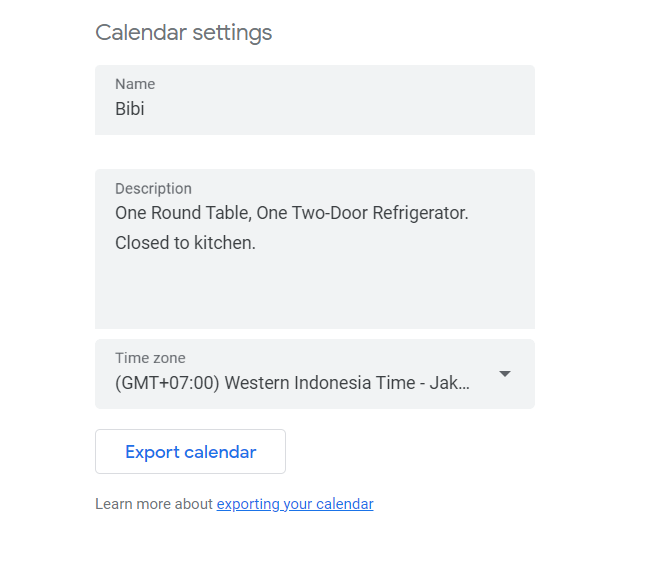


Image above shows that the editing action has been successful. We can see that Bibi's description has been added with "Closed to kitchen".



The Bibi calendar description inside Admin's Google Calendar also has been changed. This means that the editing action was successful, locally and cloudly.

### 📘 User List

#admin

In the User List menu, Admin could do some actions. These are editing user's detail, and deleting user accounts.

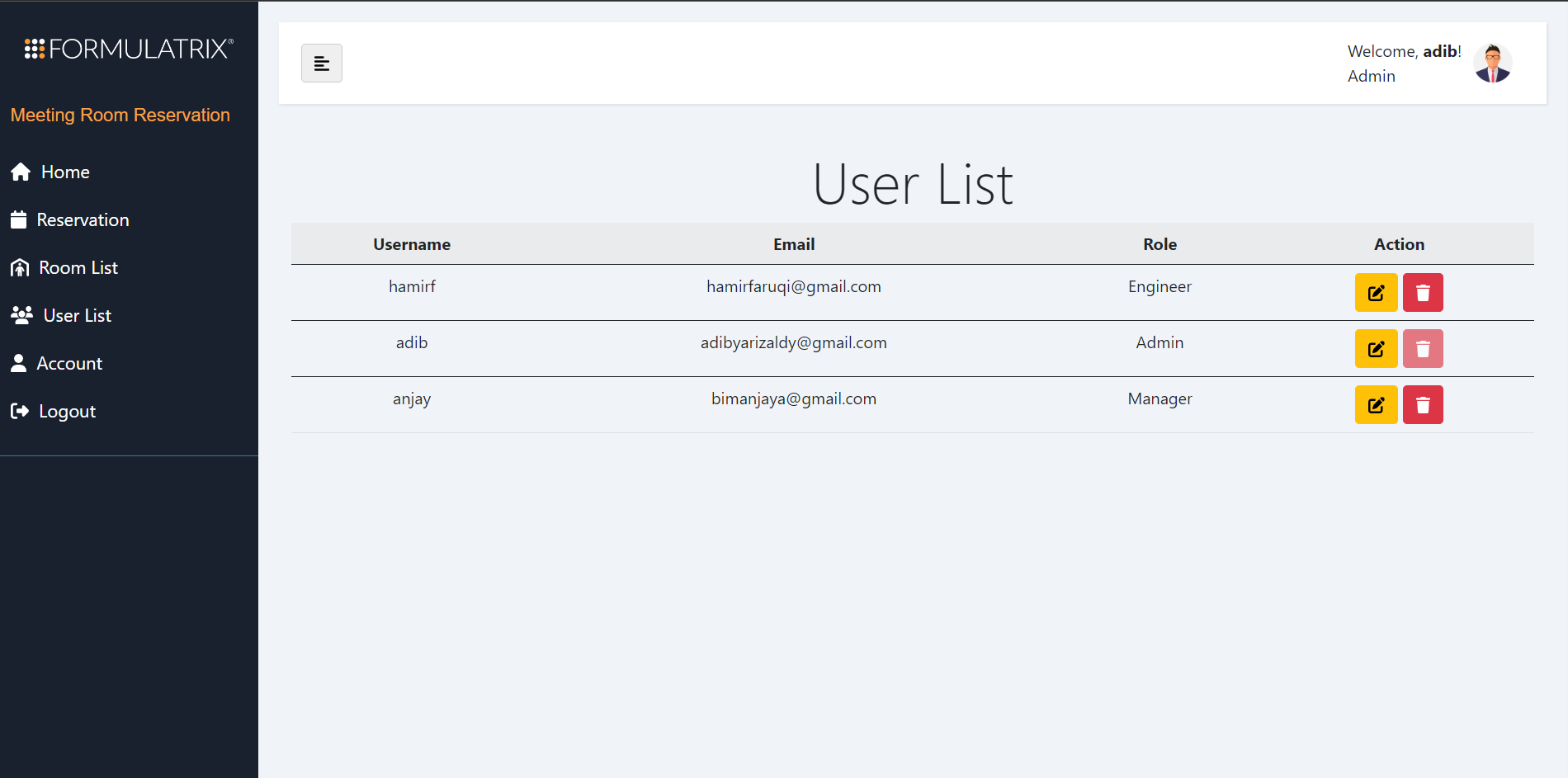
* Before :

Batch 2's development gave all Admins the ability to edit user's detail, delete user's accounts, and create new user accounts.

* After :

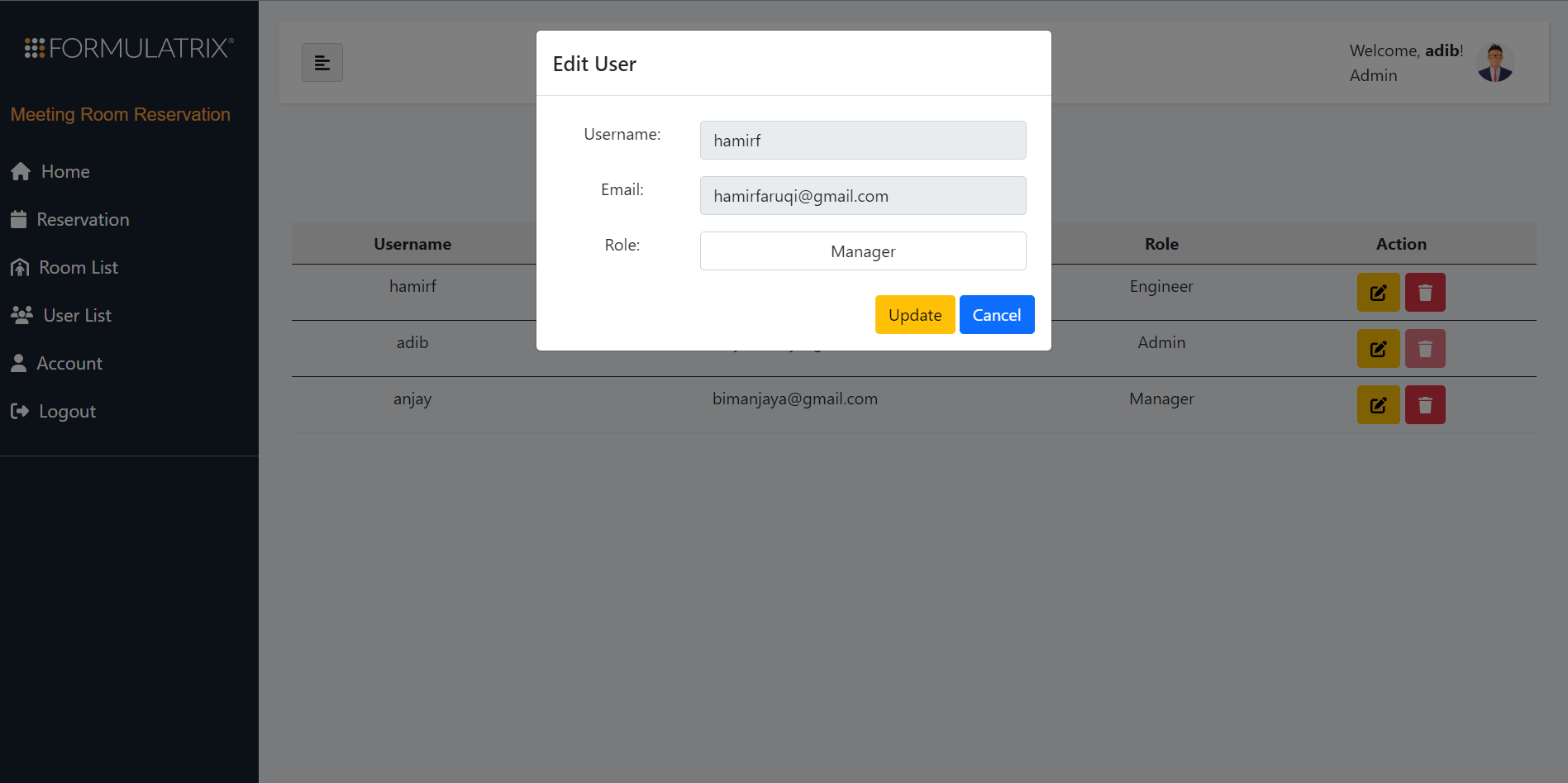
This Room List menu is quite similar to Batch 2's development. The only difference is that here, we removed Admin's ability to create new user accounts. We think that, if the application has been published and there is a new user or in this case a new employee, the account should be made by the new user/employee themselves.

1. User List Page



Inside the User List page, there are some user's properties displayed. They are Username, Email, and Role. There are also action buttons to edit the user's details, and delete the user's account.

1. Edit User Detail



To edit the user's detail, Admin can click on the yellow button.

After clicking the yellow button, a pop-up that displays the user's detail prompt will appear. In above's picture example, Admin can change the user's role. Admin can upgrade and downgrade the user's role.

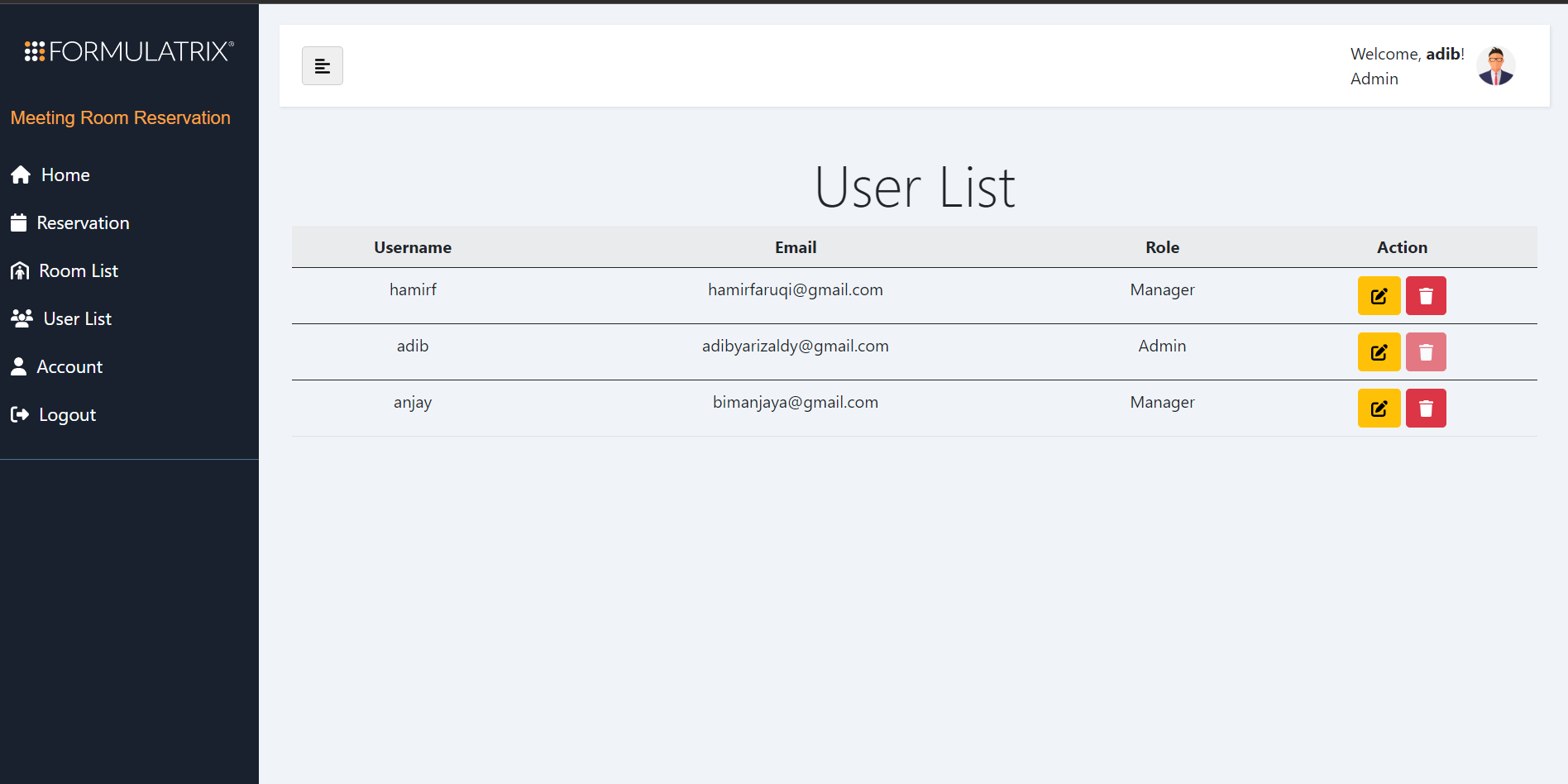
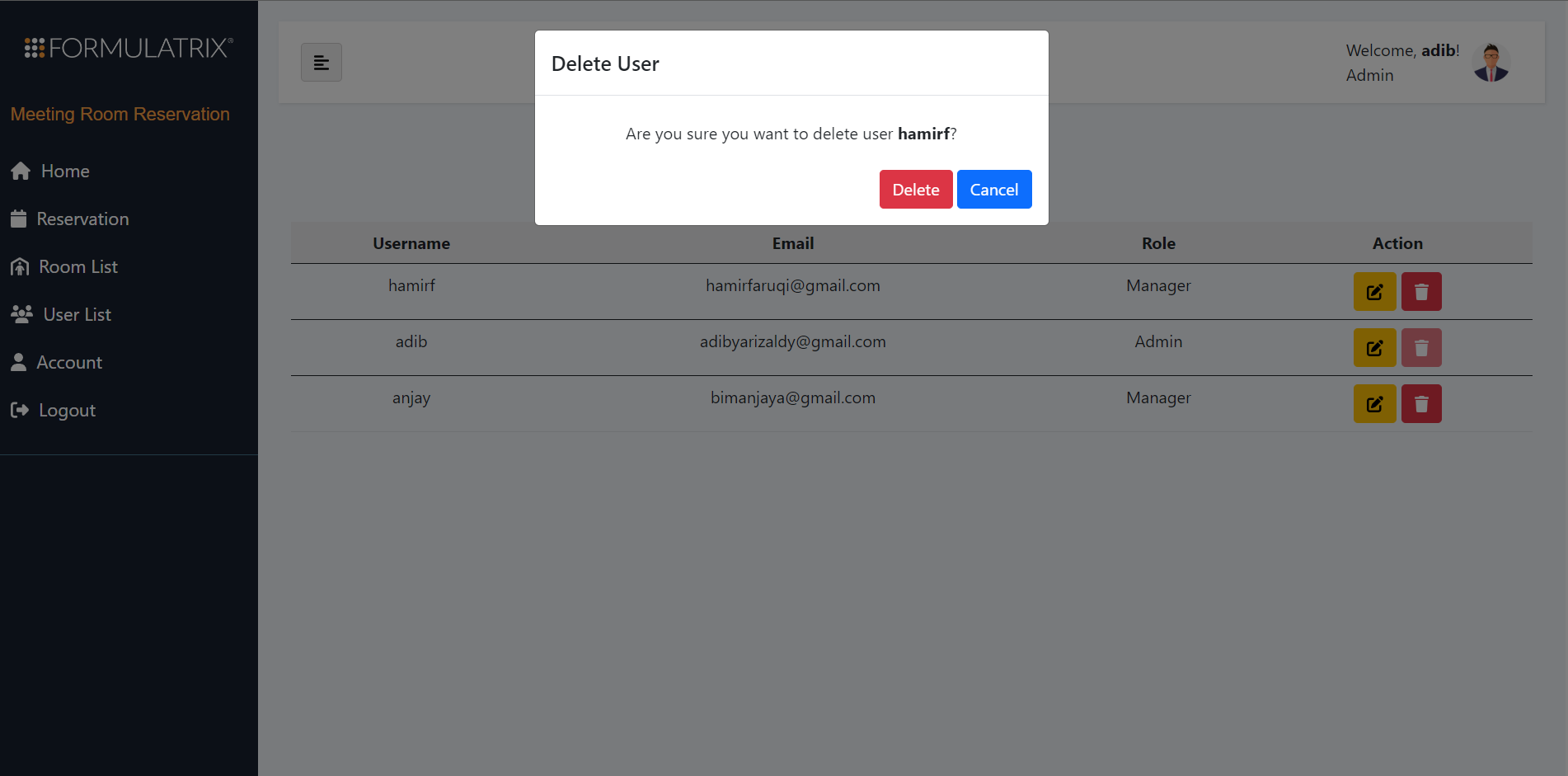


Image above shows that Admin has been succesfully changed hamirf's role from Engineer to Manager.

Note that this action will change the data inside the DB.

1. Delete User Account



Admin could also delete user's account by clicking the red-trash button.

After clicking it, a confirmation pop-up will appear, validating the deletion action.

To proceed deleting, just click Delete.

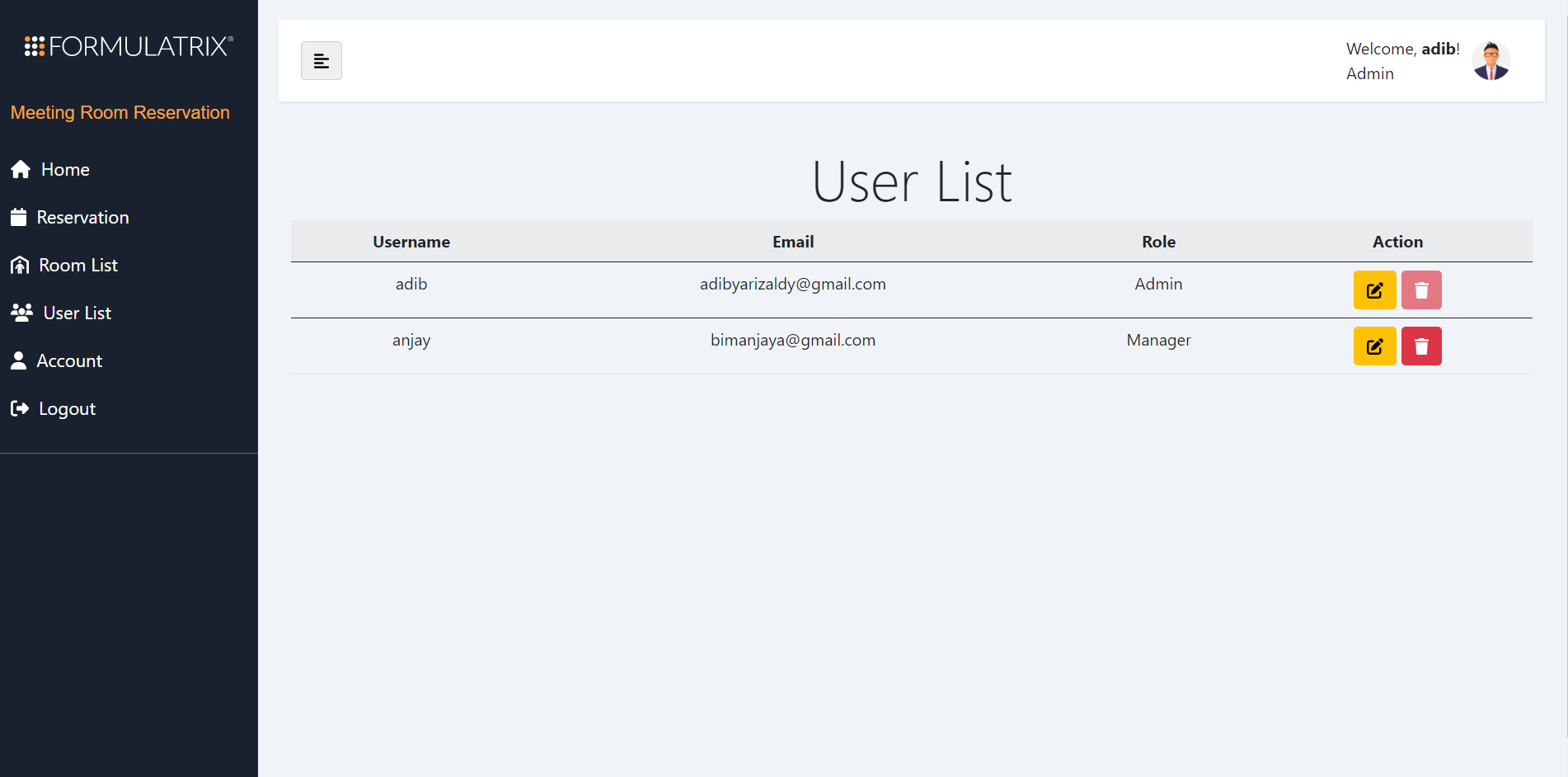


Image above shows that the deletion of hamirf account has been succesfully done.

Note that the deletion action of this process will delete user's data from DB (not soft delete).

### 📘 Account

#both

Users can look at their own profile by going into the Account menu. This menu displays the user's overall identity, such as Username, First Name, Last Name, Email, and Role.

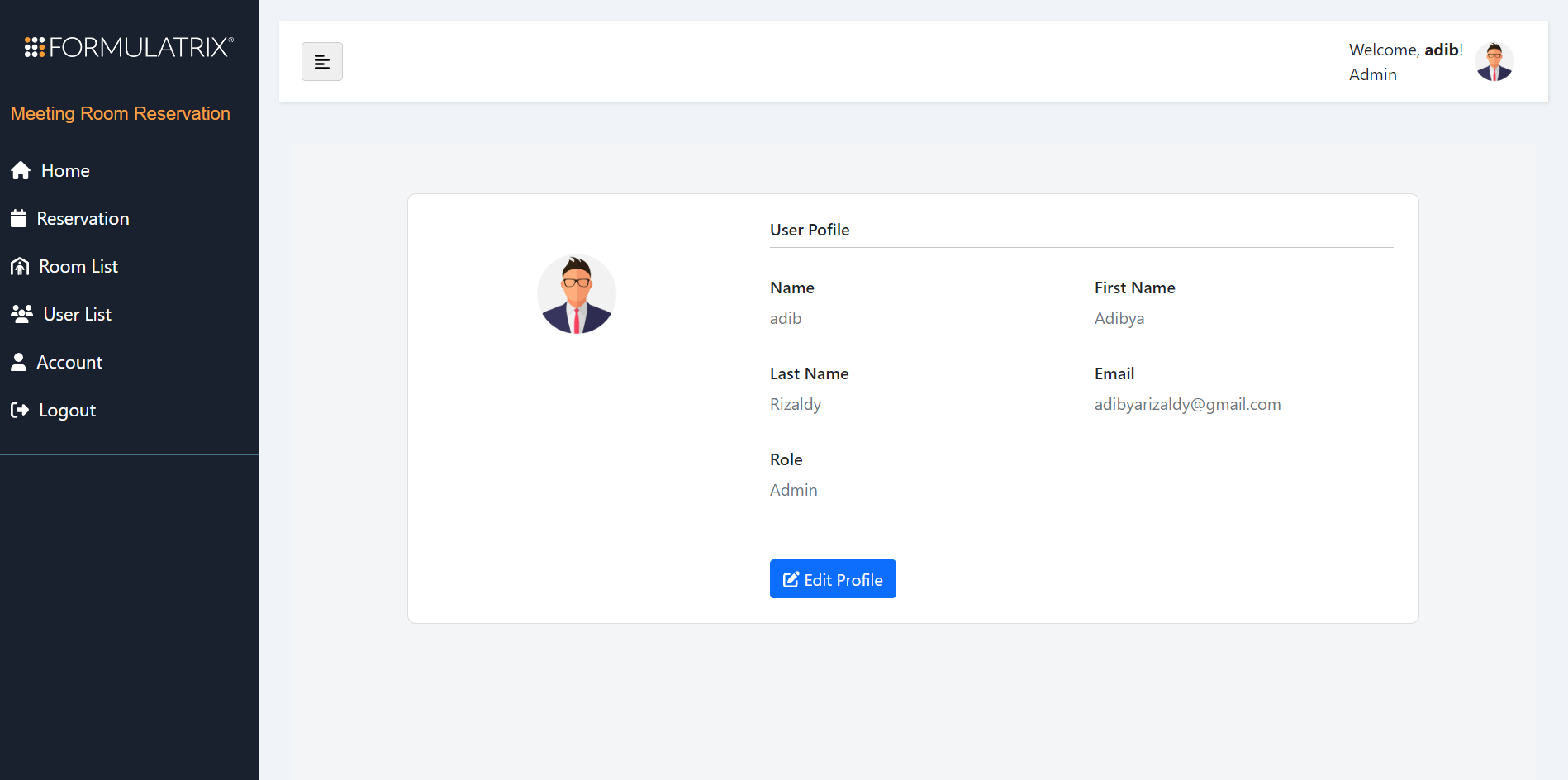
* Before :

Batch 2's development gives users an ability to modify Username, Email, and Password directly inside the Account menu.

* After :

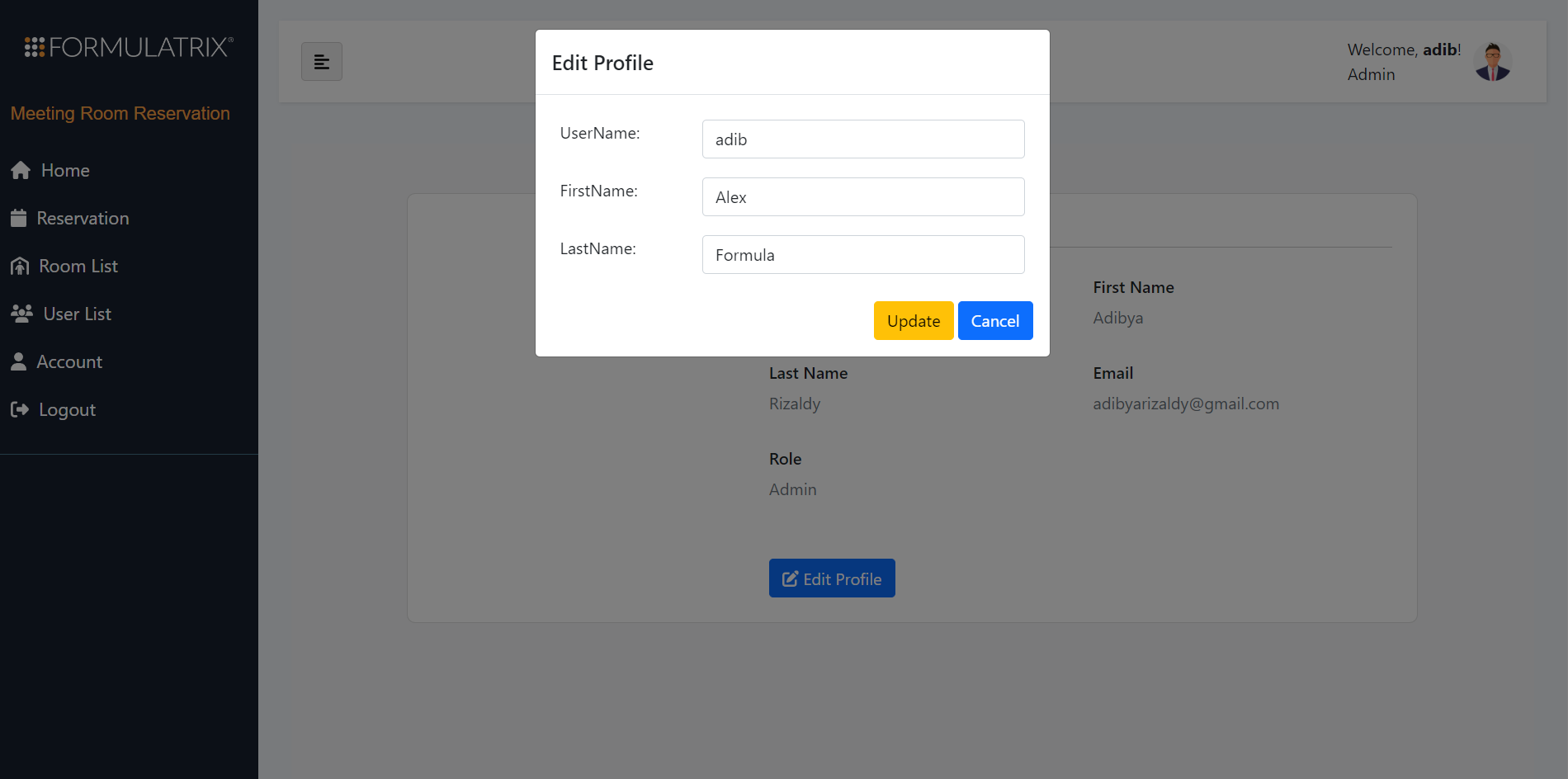
This menu is also quite similar to the last development. The difference here is that we made and edited the pop-up feature. So that, when users want to edit their identity/profile, they should edit them from the pop-up, not directly from Account's main page.

1. Account Page



The account page displays the user's identity.

1. Edit Profile Detail



Users can edit their own profile by clicking the blue-edit-profile button.

After clicking it, a pop-up will appear and user could edit the details of their profile

After editing, the user can save the changes by clicking Update.

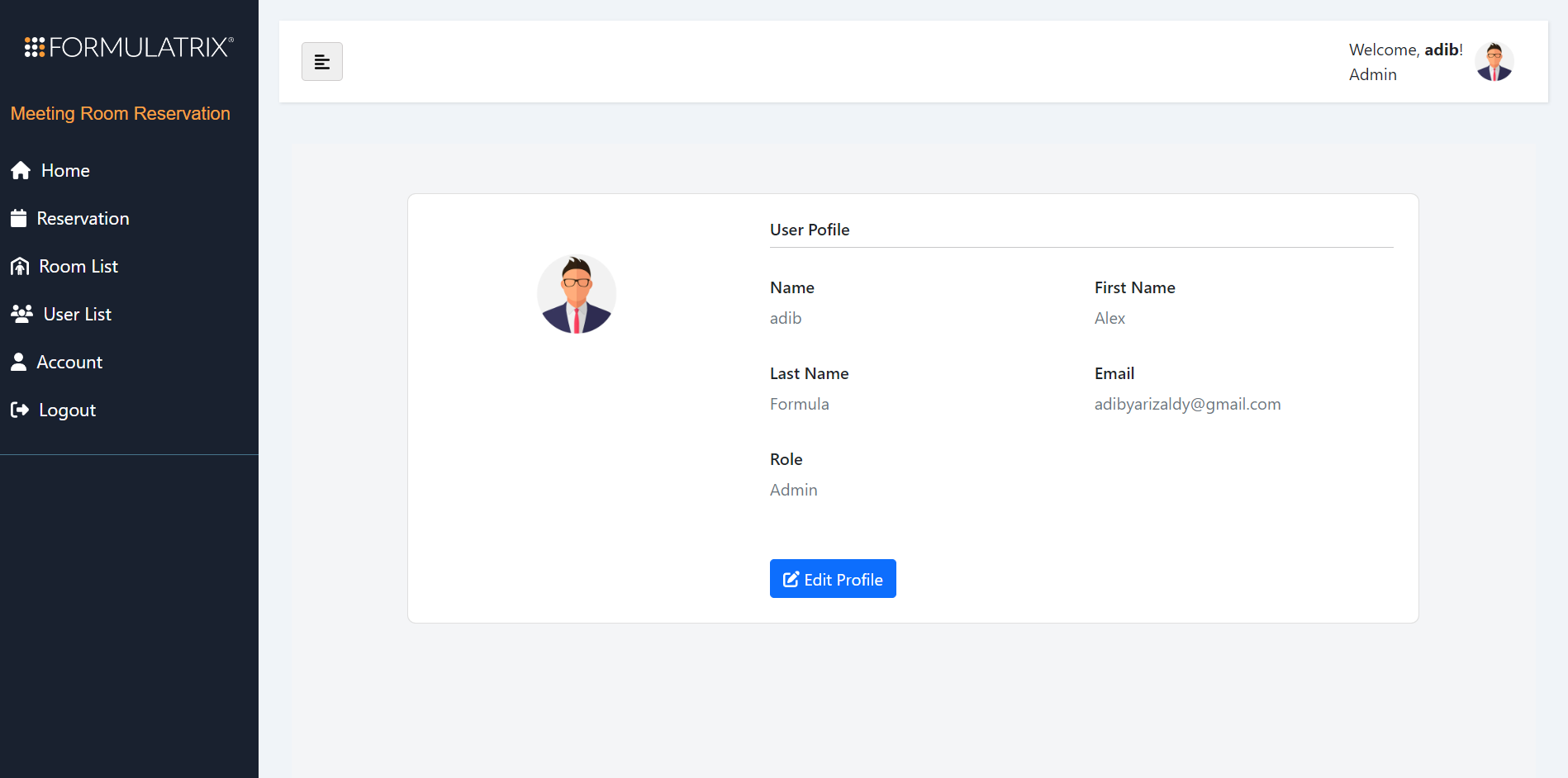


Image above shows that user successfully changed his first and last name from Adibya Rizaldy to Alex Formula.

### 📘 Logout

#both

Practically the Logout feature is the same with previous development. After clicking it, user will be redirected to the Login page.

## ⚓ Unit Test

* For testing the Meeting Room Reservation Web Application, we use XUnit and FakeItEasy packages.
* We tested every method and available output of controllers that are Login Controller, Logout Controller, and User Controller.
* We make a new class, FakeHttpSession, that implements ISession, for mocking the Session we use in the web application. Database used in the test is In Memory Database from the InMemory package of Entity FrameworkCore, and FluentAssertions for asserting the Actual and Expected condition for the readability.
* In every test of the controller the must mock / fake component is session, cause we use session for the web application to handle the app accessibility.
* There are several tests done on each controller. In Login Controller, we test the login View, whether its type of return is View, and either return a correct login view or not.
* Second, we test the login attempt. There are 2 tests here, if login attempts succeed, it has to return a Redirect To SaveLoginData Action. Or if login attempts fail, it has to Redirect To Index Action of the Login Controller.
* Third test is the save login data test. There are several cases here. If saved user data contains an Admin role, it has to Redirect To Index Action of Admin Controller. If saved user data contains another registered role, it has to Redirect To Index Action of User Controller. If there's no saved user, it has to Redirect To Index Action of Login Controller.
* For the Logout Controller, there is just one test inside. The tested method has to return Redirect To Index Action of Login Controller.
* User Controller is the last one tested controller. The first test is, Index Action test. If any registered user logged in, it has to return a home page view of a web application. The method has to return a single user data, It can't be null. And the log in user data has to be Equal to user data in the database.
* Afterwards, is to test the get method of the room list. The method expects to return a list of rooms from the database, and display it to the room list view. The name of the View has to be RoomList. And the room list in the view has to be not null, and greater than 0.
* Third test is to test the Account method. The method has to return a View named Account, should return not null of data user, and returned user data have to be the same as user data in the database.
* The last test is to test the edit profile method. If the edit data attempt is successful, the method has to Redirect To Account Action of the User Controller that will display the View of Account.

## **🗺️ Future Developments Suggestions**

1. Login

Login or Signup using Google account.

1. Account

* Warning pop-up when editing profile. If the assigned username already exists, action fails.
* Changing email and confirming it through email confirmation.
* Changing first name and last name.

1. User List (Admin)

Feature to sort users alphabetically or by role.

1. Room List

* (Admin) Directly store Calendar ID to DB after generating a new calendar (Optional. This will only apply, if you still want to keep this system/schema. Will be elaborated later in the docs. Read all of this carefully)
* (Admin) Control over all events inside each room. Admin could edit or even delete events from all rooms.
* Feature to sort rooms alphabetically or by capacity.

1. Reservation

* Feature to sort the listed available rooms based on room name, capacity, or date.
* Feature to search the listed available rooms using certain keywords.
* Drop-down of list of email in Attendee prompt when adding attendee.

1. Home

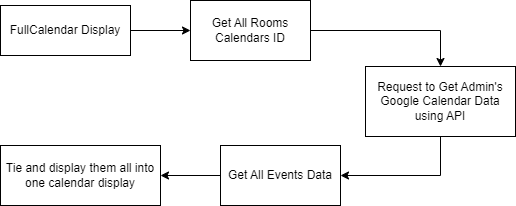
* Only shows events that the current user attended/will attend.
* Admin and user that created a particular event has control (CUD) to edit that event from the Home calendar.

1. Logout

Confirmation pop-up.

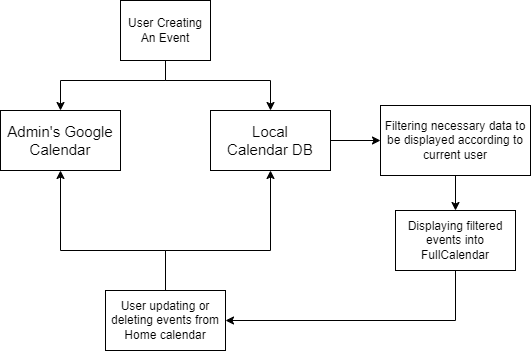
1. Specific System Development

Ideally, Home calendar only displays all events that logged in user attended or will attend. For now, the system's scheme is roughly depicted in this diagram:

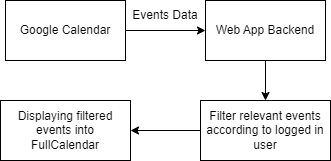


Currently we have 2 development suggestions to achieve the ideal Home feature.

1. Creating a separate local DB that contains all events from all calendars. So, roughly here's how the process will looked like:



1. Directly processing all events from Admin's Google Calendar and display them to the Home calendar. So, roughly here's how the process will looked like:



*Feel free to reach out our github profile when further elaboration required*