PennStateSoft

Meeting Scheduling System (MSS)
Vision

Software Requirements Specifications Document

Version 1.1

TEAM MEMBERS

Name
Huy Tran
Garrett Adams
Eesaa Philips

Revision History

Date	Version	Description	Author
04/Jul//2021	1.0	Completed Initial Version	Huy Tran, Garrett Adams, Eesaa Philips
18/Aug/2021	1.1	Revised Use Cases and UI descriptions	Huy Tran

Table of Contents

Project Analysis and Vision

1. Introduction

The purpose of this Software Requirement Specification (SRS) is to collect, analyze, and define high-level needs and features of the Meeting Scheduling Software (MSS). The SRS focuses on the capabilities needed by the stakeholders and the target users, and why these needs exist.

The details of how the MSS fulfills these needs are detailed in the use-case and supplementary specifications.

The introduction of the document provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this document.

1.1 Purpose

The purpose of this document is to define the requirements and specifications, outline the interfaces of the MSS, and learn who will be involved in and gain benefits from the MSS.

1.2 Scope

The purpose of this document is to outline and detail the design of the MSS and how it will provide a web-based application to both clients and administrators of PennStateSoft. It will outline the product interfaces, specify product functions and requirements, as well as provide useful information.

1.3 Definitions, Acronyms, and Abbreviations

MSS: Meeting Scheduling Software

SRS: Software Requirements Specification

UI: User Interface

XSS: Cross-site scripting attack which is an injection of malicious script code into the web application by the attacker in the client-side within the user's browser or in the server-side within the database.

1.4 References

There are no references contained in this document.

1.5 Overview

The document is organized in the following order:

a. Document Introduction
This section will briefly introduce the purpose and scope of this document.

b. Problem Description
This section will focus on the outline of user interfaces, user information that studies user environment and potential users, identify factors that may affect MSS, and specify all requirements under specific categories.

c. Team Member Log Sheets
This section includes a log of the contributions of individual team members.

2. Problem and UI description

2.1 Project Purpose, Scope, and Objectives

This project aims to create a web-based application for PennStateSoft. The application provides distinguished services for clients and administrators. For client use, the system offers basic functionalities for booking and managing meeting schedules. For administrator use, the system provides administrative functions to manage the system and meeting room service. All detail will be discussed in the following sections.

2.1.1 Login Page

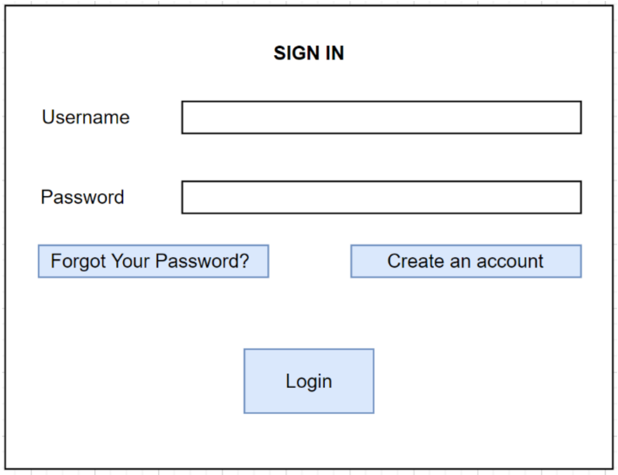
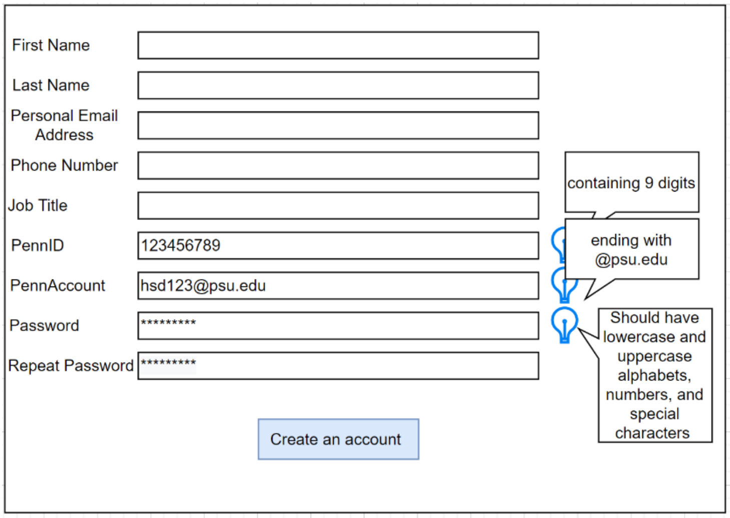


Figure 1: Sign In Page UI

On this page, the user must input the correct username and password then hit the “Login” button to gain an access to the system. If the user inputs an incorrect username or password, the system will pop up the “Your username/password is incorrect” message above the “Login” button. If the user forgets the password, the user could click on the “Forgot Your Password” button to create a new password. If the user does not have an account, the user can click the “Create an account” button. This button will direct the user to the Sign-Up Page to create a new account.

2.1.2 Account Creation Page



PennID 123456789

PennAccount hsd123@psu.edu

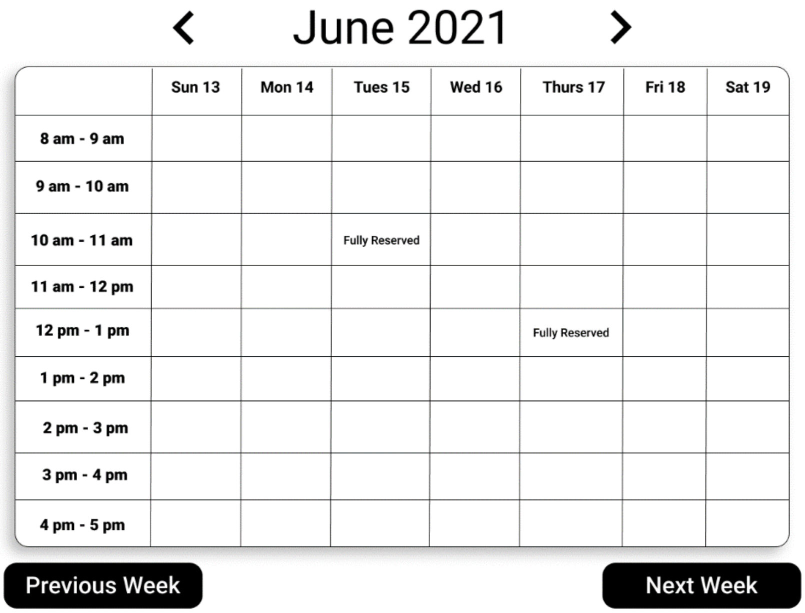
Password *********

Repeat Password *********

Figure 2: Account Creation Page UI

This page allows the user to input personal information which includes name, personal email, and job title. This page will also check whether the user is an employee of the company by checking PennID and PennAccount. The user can create their password on the password field. Because of security concerns, the password will be encoded. When everything is done, the user can click the “Create an account” button to create a new account.
To help a new employee to use this page, on the right-hand side, it has lightbulb icons that will help a new user what kind of credential information should be there.

2.1.3 Calendar View



June 2021

	Sun 13	Mon 14	Tues 15	Wed 16	Thurs 17	Fri 18	Sat 19
8 am - 9 am							
9 am - 10 am							
10 am - 11 am			Fully Reserved				
11 am - 12 pm							
12 pm - 1 pm					Fully Reserved		
1 pm - 2 pm							
2 pm - 3 pm							
3 pm - 4 pm							
4 pm - 5 pm							

Figure 3 Calendar Page UI

The figure above is the Calendar View Page, also is a home page, where a weekly calendar is loaded. Each cell represents a 1-hour window. The user can click to the cell to make a reservation. After clicking, there is a pop-up window which allows the user can choose the meeting room and make a payment (if the user chooses to a special room). The user can click the left or "Previous Week" button to see the previous week's history. The user can click right arrow or "Next Week" button to see the next week's plan.

2.1.4 Reserve a Special Room

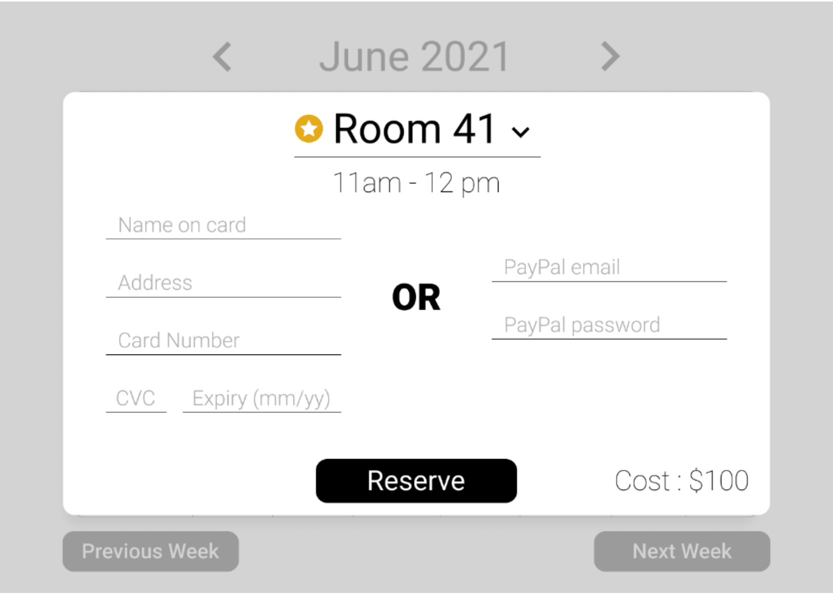


Figure 4: Reserve a Special Room UI

Above the time section, there is a dropdown box where the user can choose a meeting room. If the user chooses to reserve a special room, the above modal will appear. The user enters their billing information if the room costs money (which will be auto-filled if they have reserved before). Reserved rooms will not appear in the roomselection dropdown field. The text fields are protected by the whitelist method to prevent data injection. If the inputs are incorrect, the system will display an error message. The MSS integrates the payment system allowing the user to use either a credit card or PayPal.

2.1.5 Reserve a Free Room

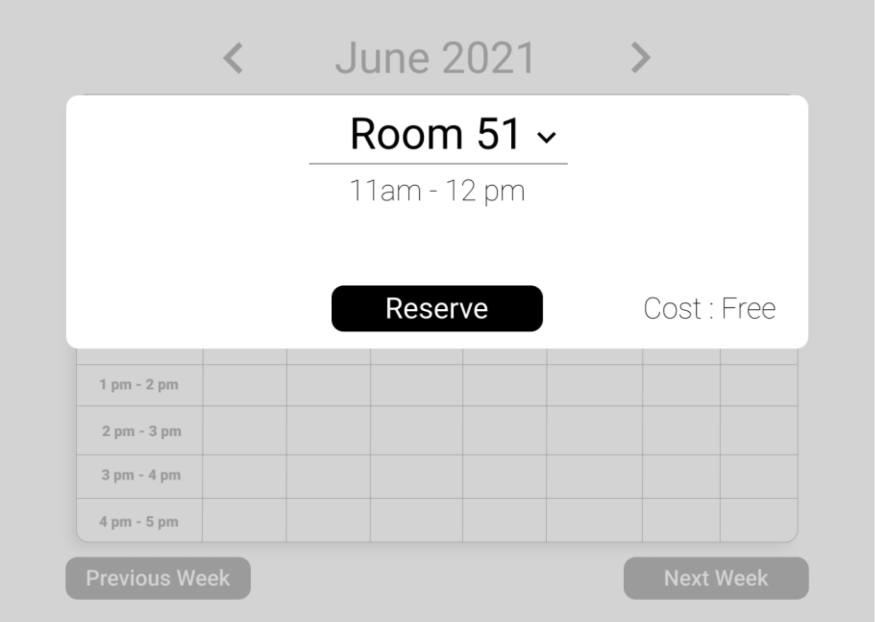
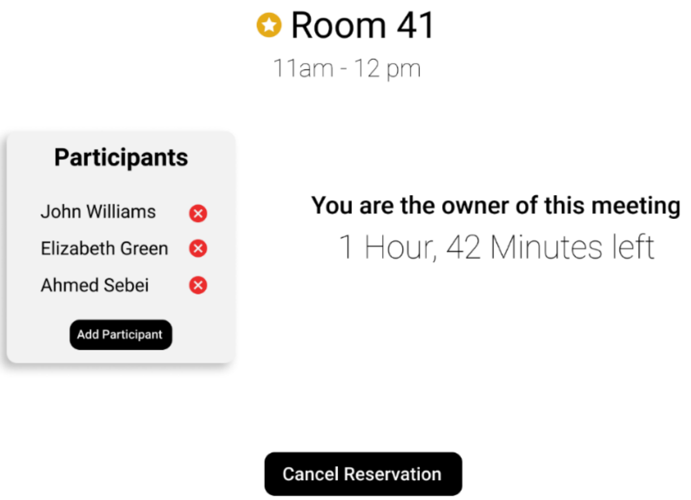


Figure 5: Reserve a Free Room UI

When the user clicks on an available time slot in a free room, the above modal will appear. They can click “reserve” or click outside or press “escape” to exit the dialog and cancel the reservation process.

2.1.6 Manage Meeting Room



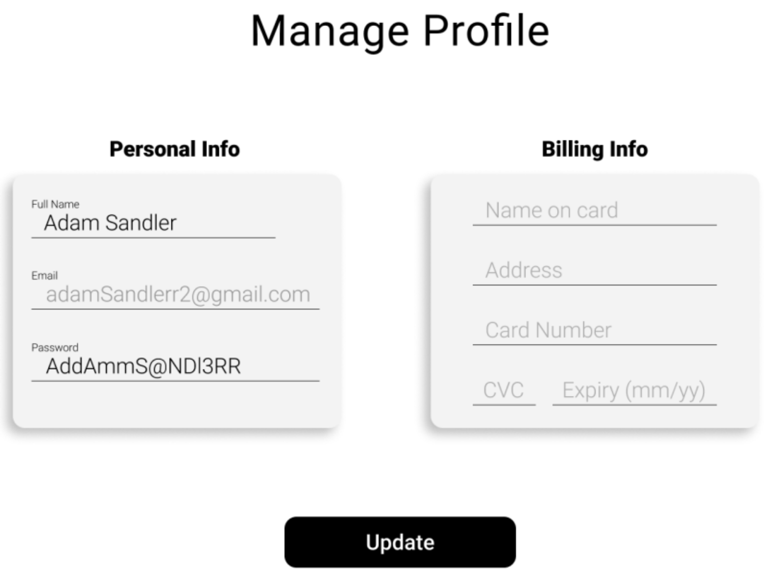
You are the owner of this meeting

1 Hour, 42 Minutes left

Figure 6: Manage Room Page UI

After the user successfully reserves a room, they are directed to the above screen where they can add/remove participants or cancel their reservation. To add a participant, the user can click the “Add Participants” button. To cancel the reservation, the user can click the “Cancel Reservation” button. If a client cancels the reservation 10 minutes before the meeting starts, they will get a refund. Otherwise, they will only receive 60% of their money back. To remove a participant, the user can click the cancel button next to the name of a participant that the user desires to remove. Besides the participant list, the system displays the time left before the meeting is coming.

2.1.7 Manage Profile



Full Name: Adam Sandler

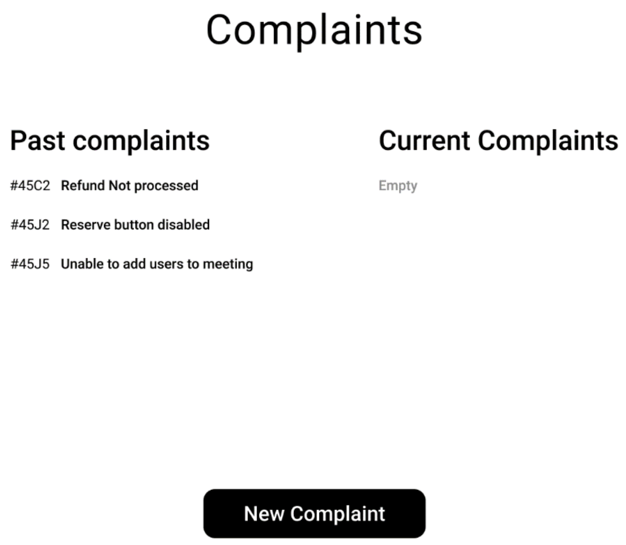
Email: adamSandlerr2@gmail.com

Password: AddAmmS@NDl3RR

Figure 7: Manage Profile UI

The user can edit their personal information (except their email) and billing information which will be auto-filled in the reservation pages. The administrator will have access to the same page, but they can only edit the billing information. All text fields are protected by the whitelist method. When the user is done, the user can click the "Update" button, then the user profile will be updated on the database.

2.1.8 Client Complaints



#45C2 Refund Not processed

#45J2 Reserve button disabled

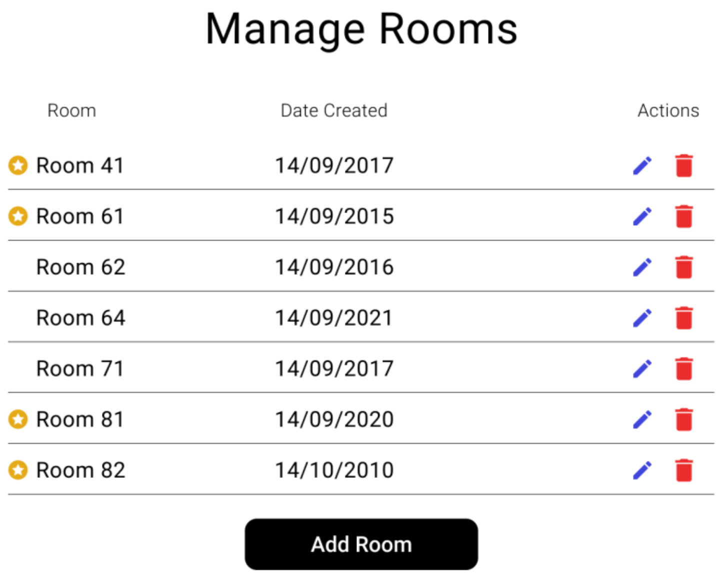
#45J5 Unable to add users to meeting

Empty

Figure 8: View/Submit Complaints UI

The client uses this page to manage their complaints. They can add a new complaint by clicking the "New Complaint" button, view pending complaints, and view past complaints. A complaint is either open, pending, or closed.

2.1.9 Administrator Manage Rooms



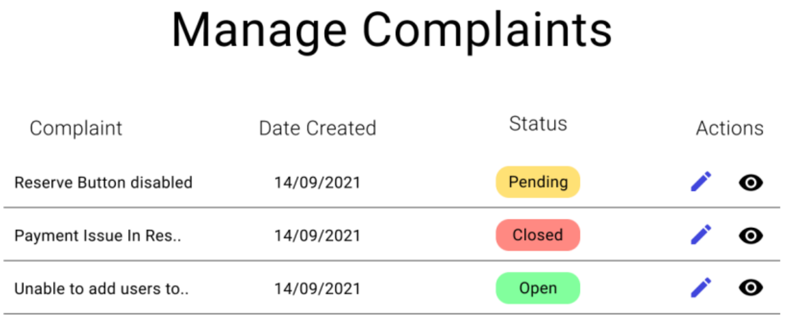
Manage Rooms

Room	Date Created	Actions
★ Room 41	14/09/2017	✎ 🗑
★ Room 61	14/09/2015	✎ 🗑
Room 62	14/09/2016	✎ 🗑
Room 64	14/09/2021	✎ 🗑
Room 71	14/09/2017	✎ 🗑
★ Room 81	14/09/2020	✎ 🗑
★ Room 82	14/10/2010	✎ 🗑

Figure 9: Admin Manage Rooms Page UI

The administrator uses this page to manage the rooms. They can delete a room by clicking the “Trash” icon or edit it by clicking the “Pen” icon. A room that is in use cannot be modified until it has no pending reservations. The administrator can add a new room by choosing the “Add Room” button.

2.1.10 Administrator Manage Complaints



Manage Complaints

Complaint	Date Created	Status	Actions
Reserve Button disabled	14/09/2021	Pending	✎ 👁
Payment Issue In Res..	14/09/2021	Closed	✎ 👁
Unable to add users to..	14/09/2021	Open	✎ 👁

Figure 10: Manage Complaints UI

The administrator uses this page to manage complaints. They can close a preview of a complaint by clicking the “Eye” icon or edit it by clicking the “Pen” icon. Editing a complaint allows the administrator to change its status and add comments to it. When a complaint is closed, the client is notified via email.

2.2 Product Functions

2.2.1 All User Functions

i. Creating a new account
The system shall allow the user to create an account using their unique company email address. This will be done by filling through the Sign-Up page. This page will allow a new user to create a new account and validate whether the user is a company employee by checking PennID and PennAccount. The credentials will not be shared outside.

ii. Displaying traditional working hour calendar
The meeting time shall befall in traditional 8-5 business hours. The calendar will be displayed on the main page. The calendar has 9 one-hour time slots, and they are clickable. The client can click on any available time slot to book a meeting room. When the reservation is done, the system will display a new meeting at that time slot, and the user can click on the slot to get the meeting detail or edit it (if the client is a meeting creator).

iii. Having hint and help features
The system shall have hints and help features to support clients and administrators. The help features will be shown on the top right of the system. The hint feature will be shown on the specific page. The hint and help features are clickable, and the text must be readable and understandable.

iv. Serving more than 100 users
The system shall run well if more than 100 users are using the system. If there are more than 100 users, the system may run slowly but shall not crash or terminated.

2.2.2 Client Functions

i. Updating profile
The system shall allow the user to check and update their profile information including payment information. This will be done after logging in to the system and clicking the profile icon. After clicking the profile icon, the profile page will show up and allow the user to fix his/her personal information including payment method. If the user identity is inappropriate, the system would direct the user safely to the "Page Not Found" page and immediately log out. If the user identity is appropriate, the system would update the database.

ii. Reserving a meeting room
The system shall allow the client to reserve a room for a meeting. This function is for client use only. Inappropriate users will be directed to the "Page Not Found" page and logged out. An appropriate client can reserve a normal room or special room. If the client cannot reserve the room, the system will display the picked room is not available during that time and allow the client to pick another room.

iii. Paying for special rooms
The system shall allow the client to pay for special rooms. If the client reserves a special room, the billing form will show up. This form allows the client to input his/her payment method, including using PayPal, and the fee must be $100.

iv. Displaying or editing meeting room detail
The system shall allow the client to display and edit all the meetings. This function will be done by clicking on the meeting timeslot. When the client clicks on the meeting, it will display the meeting name, time, room number, and room type. If the client is the meeting creator, it will show who will participate and allow the creator to add/remove participants.

v. File a complaint
The system shall allow the user to file a complaint. This function will be done on the main page. When the client clicks on the "File a complaint" button, it will display the complaint form. In this form, the client needs to fill in the meeting room and time, client ID which should be PennID, and complaint section. The complaint form will be saved on the database for later review, and only administrators can read and respond to it.

vi. Restricting client to view only meeting (s)he will participate
The system shall restrict the client from viewing meetings that the client will not participate in. If the client is removed from a meeting, the system immediately updates the change and reports to the client that (s)he has been removed from the meeting.

2.2.3 Administrator Functions

i. Viewing, editing, and creating rooms
The system shall allow the administrator to view, edit, and create meeting rooms. This function is for administrators only. The administrator can view all meetings' detail (including who will join the meeting) in a week or a day and all available as well as unavailable rooms. The administrator can edit the room detail and remove or add available rooms. The administrator can also create a new room. A created room shall contain a unique number. If the user having no authority to access this function, the system will direct the

user to the “Page Not Found” and log out.

ii. Updating clients’ billing information
The system shall allow the administrator to update the clients’ billing information. This function is for administrators only. If the user having no authority to access this function, the system will direct the user to the “Page Not Found” and log out.

iii. Viewing and responding to client complaints
The system shall allow the administrator to view complaints and respond to them. This function is for administrators only. If the user having no authority to access this function, the system will direct the user to the “Page Not Found” and log out.

2.3 User Description

2.3.1 User Environment

The website will mainly run on the desktop. The website can run on mobile, but it is not intended or optimized for mobile use. A mobile version will be created later using flutter with the same APIs as the web application.

2.3.2 User Profiles

Clients and administrators are expected to have the basic computer knowledge to navigate the site efficiently.

2.4 Assumptions and Dependencies

It is assumed that any user has basic computer skills and knowledge.

2.5 Constraints

The system must be web-based, accessible at any time, and able to handle up to 5000 concurrent users. The system must be supported by Windows and macOS.

2.6 Specific Requirements

2.6.1 Applicable Standards

The payment will operate under card standard regulations: it will only accept Visa/Mastercard or PayPal. The requests will rely on the TCP through RESTFUL APIs.

2.6.2 System Requirements

The website should run on any browser, including Internet Explorer version 11 and up, and should be supported on Windows and macOS. Plus, the website should handle up to 5000 users with no noticeable performance dips.

2.6.3 Performance Requirements

The backend will run on a test server (personal device). In the future, it will be deployed to multiple Linux servers with NGINX to distribute the load evenly among the servers.

2.6.4 Quality Requirements

2.6.4.1 Correctness

The system should meet all the requirements without any software faults or failures as indicated in the use cases. The system should pass penetration testing, unit testing, system testing, and integration testing.

2.6.4.2 Reliability

The system should not fail more than once every 50 days.

2.6.4.3 Efficiency

The website should load and perform within the standards of modern website applications. The website should load in under 4 seconds.

2.6.4.4 Integrity

Only administrators can view client personal details. Clients cannot view other clients' data.

2.6.4.5 Usability

The system should be intuitive and not require more than 2 hours of training for administrators.

2.6.4.6 Testability

The system should incorporate unit testing in the development phase as well as manual testing after deployment. Any detected issues will be added to the GitHub "issues" ticket system so the developers can fix it and link the associated code.

2.6.4.7 Interoperability

The system should generate an excel sheet for the week's schedule which will be in the standard .xlsx format.

2.7 Analysis model

2.7.1 Summary Use Case Diagram

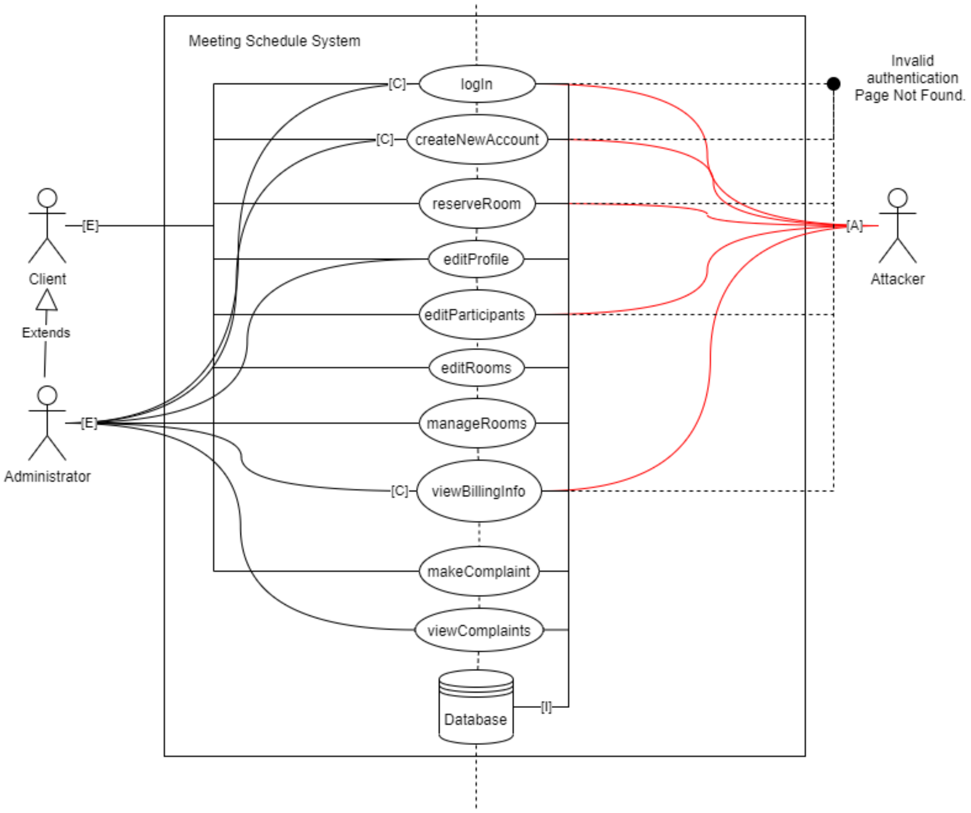


Figure 11 MSS Use Case Diagram

The figure above describes how the clients and administrators would use the system as well as the interactions between the system and the database. The MSS Use Case Diagram also points out the potential threats an attacker may penetrate, and how the system would respond if it is attacked.

2.7.2 Summary Use Case

Project Name:		Meeting Schedule System
Use Case Name:		Meeting Schedule System
User Goal:		Clients can keep track, manage, and create meeting schedules. Administrators can keep track and manage the system
Level:		Summary
Relevant User Reqs:		User password and user ID and/or unique company email
Relevant System Reqs:		None
Primary Actor:		The Users
Precondition:		None
Minimal Guarantee:		The system terminated
Success Guarantee:		The system updates users' activities
Fail Case:		The system is terminated without an error
Consequence of Failure:		Users stop using the product
Associated Risks:		Product purpose violation
Trigger:		The users sign in to the system
Success Scenario:	**Step**	**Actions**
	1	A user logs in the system <<Login>>.
	2	A user creates a new account <<Create an account>>.
	3	The system initializes the environment.
	4	Client chooses and edits the meeting room and time then makes reservations.
	5	Client files a complaint.
	6	Client edits his/her profile.
	7	Administrator adds/deletes rooms.
	8	Administrator views and response to complaints.
Extensions:		**Branching Scenarios**
1		**Condition: Attacker logs in the system**
		Step Actions
		1 The system records user IP address and time being attacked then uploads the record to GitHub
		2 The system directs the attacker safely to Error Page.
2		**Condition: The system cannot identify a new user's identification number or company email**
		Step Actions
		The system pops up a message telling the user to wait a few seconds for updating the database and provide the user the contact information of the HR office
3		**Condition: Environment performance is too slow**
		Step Actions
		1 If the loading time excesses the limited time by 1 minute, the system will record the issue and send it to GitHub "issues" ticket.
		2 The system asks the user to reload the page.
4		**Condition: Room is not available or unable to make a reservation.**
		Step Actions
		The system pops up a message telling the user to pick another room or another time
5		**Condition: Complaint form is not loaded**
		Step Actions
		The system pops up an error message telling the user to reload the page
6		**Condition: Administrator cannot add/delete a room**
		Step Actions
		1 System reloads the page
		2 System terminated

The Summary Use Case states how the system works in the Success Scenario and how the system behaves when something is gone wrong in the Extensions. The role of this use case is to clarify how the system works and reacts in general. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

2.7.3 Task Use Case: Login

Project Name:	Meeting Schedule System	
Use Case Name:	Login	
User Goal:	Login to system	
Level:	Task	
Relevant User Reqs:	User must be client/admin	
Relevant System Reqs:	None	
Primary Actor:	The Client/Admin	
Precondition:	Client must have an account	
Minimal Guarantee:	The system reloaded	
Success Guarantee:	The system logs in	
Fail Case:	Client has an account and cannot log in	
Consequence of Failure:	Client cannot use the product	
Associated Risks:	Client forgoes attempts to use the system	
Trigger:	The client pulls up the site in the browser and clicks login	
Success Scenario:	**Step Actions**	
	1	Use case begins when the user opens the site
	2	The system prompts the user to enter credentials
	3	The client enters them and clicks login
	4	The system verifies credentials and logs user in
Extensions:	**Branching Scenarios**	
4a	**Condition: Client enters the wrong credentials**	
	Step Action: The system tells the user that the credentials were wrong and asks them to try again	

The figure above is the Login Use Case which states in detail how the system works in the Login Page in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

2.7.4 Task Use Case: Reserve a Room

Project Name:	Meeting Schedule System
Use Case Name:	Reserve a normal room
User Goal:	Client can reserve a normal room without paying any fee
Level:	Task
Relevant User Reqs:	User must be a client
Relevant System Reqs:	None
Primary Actor:	The client
Precondition:	Client must have the authorization to use the system
Minimal Guarantee:	The system reloaded
Success Guarantee:	The system updates the client’s room reservations.
Fail Case:	Client pays a fee for reserving a regular room or cannot make a reservation.
Consequence of Failure:	Meeting cannot be made so clients stop using the product
Associated Risks:	Room policy violation
Trigger:	The client clicks on any available time slot of the calendar.
	Step Actions

Success Scenario:	**1**	A client clicks on any available time slot of the calendar
	2	The system pops up the reservation form showing the meeting time
	3	The client chooses an available room from the dropdown. If the client reserves a special room, the client needs to pay $100.
	4a	The client clicks <<Reserve>> for the confirmation.
	4b	The client clicks outside or presses “escape” for exiting and canceling the reservation process
	5	If the user confirms the reservation, the system will update the database.
Extensions:		**Branching Scenarios**
4a		**Condition: Client cannot make the reservation**
		Step Actions
		1 The system shows an error message that the room has already been reserved at the selected time slot.
		2 The page is updated to show the latest changes in the room reservations.
4a		**Condition: The client has been charged for reserving a regular room**
		Step Action: The system immediately refunds 100% of the money.

The figure above is the Login Use Case which states in detail how the system works in the Login Page in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

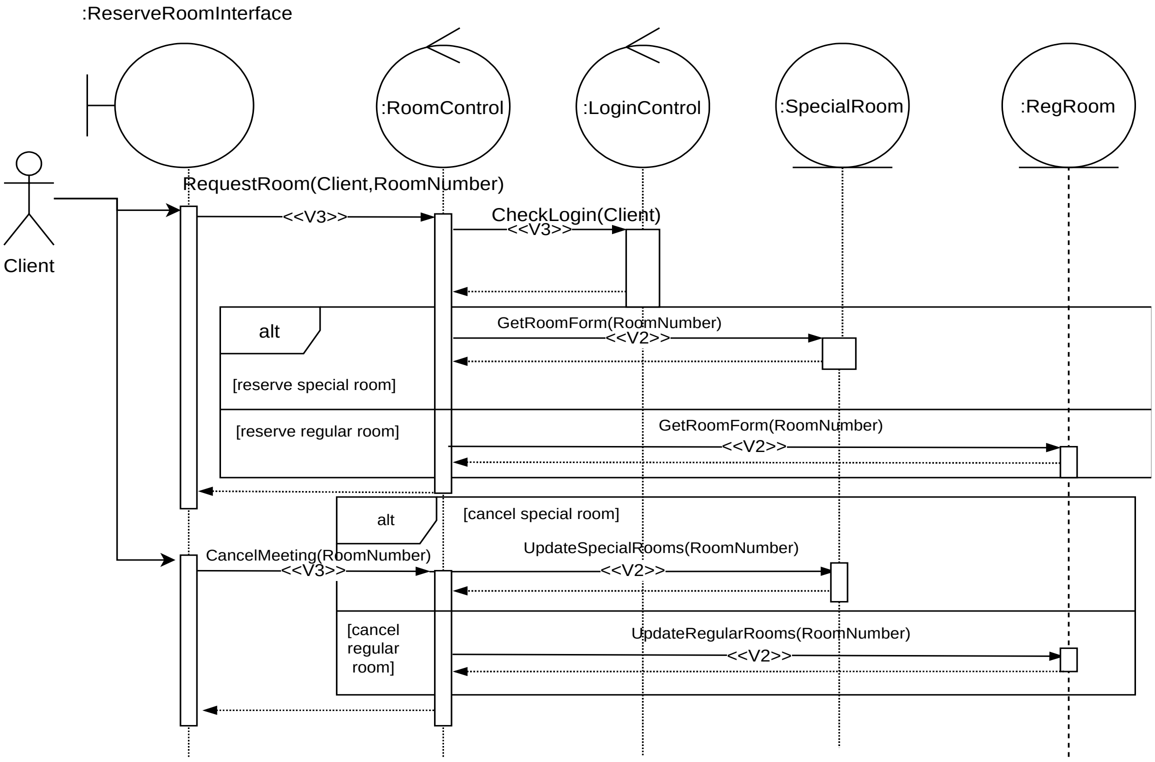


Figure 12 Room Reservation Sequence Diagram

The diagram above shows the logic behind the room reservation working. The RoomControl and LoginControl will check the user authentication. If the user has the authority, the user will provide service. Otherwise, the system will provide no service.

2.7.5 Task Use Case: Account Creation

Project Name:	Meeting Schedule System
Use Case Name:	Account Creation
User Goal:	Clients can create client accounts and admins can create client or admin accounts
Level:	Task
Relevant User Reqs:	None
Relevant System Reqs:	None
Primary Actor:	The client or administrator
Precondition:	Client/admin must be a company employee
Minimal Guarantee:	The system reloaded
Success Guarantee:	The client/admin creates an account
Fail Case:	The client/admin is unable to make an account with the necessary information
Consequence of Failure:	The website will have no users and be rendered inoperable
Associated Risks:	User is unable to access the system, so the company may stop using the product
Trigger:	The client/admin clicks the button for account creation
Success Scenario:	**Step Actions**
	1 User clicks on <<Create an account>> button of the Sign In page
	2 The system pops up the form to create an account
	3 The client/admin enters all the information needed
	4 The user clicks <<Create an account>>
	5 The system shows a success message and will update the database.
Extensions:	**Branching Scenarios**
4	**Condition: Client/admin provides an email that is not a company email**
	Step Actions
	1 The system shows an error message that the email is not a company email and prompts the user to try again
	2 The page is updated to show the latest changes in the room reservations.
4	**Condition: The client/admin enters a password that does not fit the criteria**
	Step Actions: The system rejects the account creation and prompts the user to try again and informs them of the criteria

The figure above is the Account Creation Use Case which states in detail how the system works in the Account Creation Page in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

2.7.6 Task Use Case: Edit Profile

Project Name:	Meeting Schedule System
Use Case Name:	Edit Profile
User Goal:	Client can edit their profile after their account has been made
Level:	Task
Relevant User Reqs:	None
Relevant System Reqs:	None
Primary Actor:	The client
Precondition:	Client must have an account
Minimal Guarantee:	The system reloaded
Success Guarantee:	The client edits the profile as intended
Fail Case:	The client is unable to make changes to their account
Consequence of Failure:	The client will be dissatisfied with the software
Associated Risks:	N/A
Trigger:	The client clicks the button for editing the account

		Step Actions
Success Scenario:	**1**	Use case begins when the user clicks on the edit profile button
	2	The system pops up the user's account information. The personal information on one side, and the billing information on the other.
Success Scenario:	**3**	The client makes the desired amendments
	4	The user clicks <<Save>>
	5	The system shows a success message and will update the database.
Extensions:		**Branching Scenarios**
4		**Condition: Client attempts to change pages before saving the form**
		Step Actions
		1 The client is reminded that they must press save
		2 The page waits for the client to confirm whether they want to save or not
3		**Condition: The client attempts to edit their profile in a way that is outside the parameters (such as an invalid password or credit card number)**
		Step Action: The system rejects the edit and informs the client must remain within the parameters.

The figure above is the Account Edition Use Case which states in detail how the system works in the Update Profile Page in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

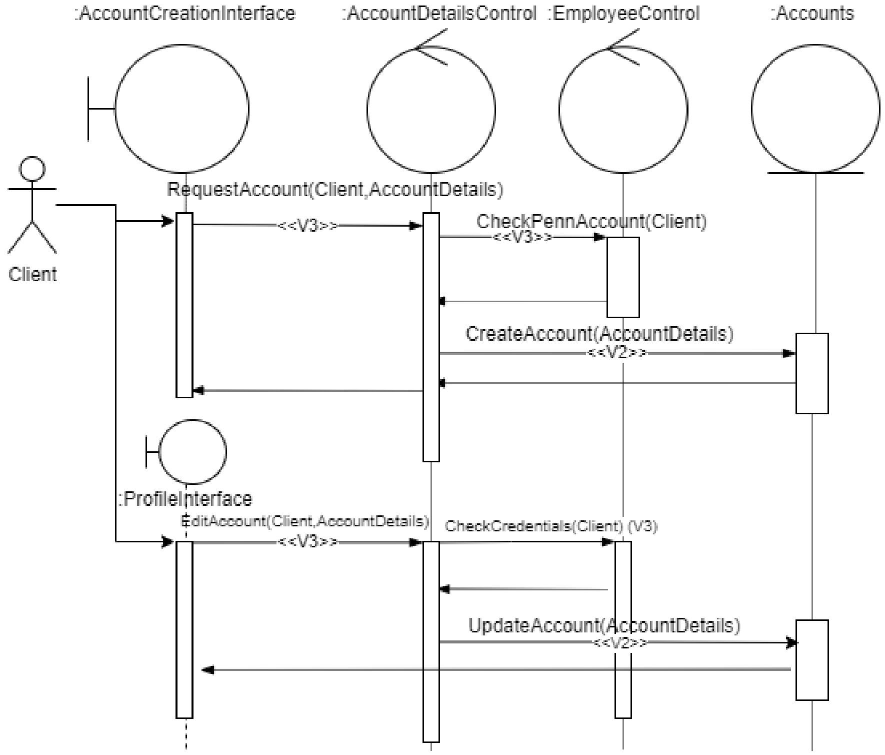

:AccountCreationInterface
:AccountDetailsControl
:EmployeeControl
:Accounts
Client
RequestAccount(Client,AccountDetails)
<<V3>>
CheckPennAccount(Client)
<<V3>>
CreateAccount(AccountDetails)
<<V2>>
:ProfileInterface
EditAccount(Client,AccountDetails)
<<V3>>
CheckCredentials(Client) (V3)
UpdateAccount(AccountDetails)
<<V2>>

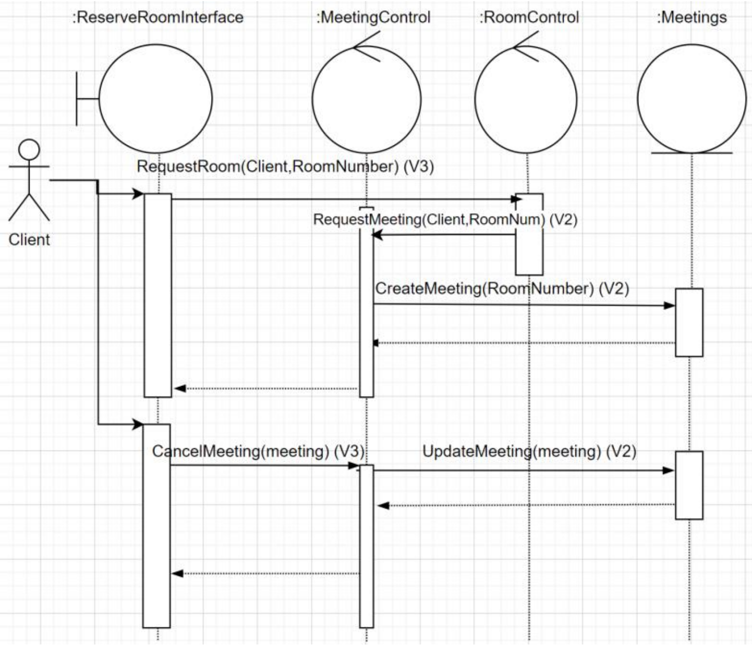
Figure 13 Account Creation and Profile Edition

The diagram above shows the logic behind the account creation and profile update. The AccountDetailControl and EmployeeControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.7 Sub-Function Use Case: Meeting Creation

Project Name:	Meeting Schedule System	
Use Case Name:	Meeting Creation	
User Goal:	Upon reserving a room, a meeting is created	
Level:	Sub-Function	
Relevant User Reqs:	None	
Relevant System Reqs:	None	
Primary Actor:	The client	
Precondition:	Client must have an account	
Minimal Guarantee:	The system reloaded	
Success Guarantee:	The meeting is created once the room is reserved	
Fail Case:	The meeting is not created	
Consequence of Failure:	The client will not be able to change the important properties of the meeting such as “participants”	
Associated Risks:	Product purpose violation	
Trigger:	The client clicks <<Reserve>>	
Success Scenario:	**Step**	**Actions**
	1	Use case begins when the user clicks the <<Reserve>> button
	2	The system automatically assigns a meeting to that reservation
	3	The client adds participants or cancels the meeting
	4	The user clicks <<Save>>
	5	The system shows a success message and will update the database.

The figure above is the Meeting Creation Use Case which states in detail how the system creates a meeting in the Success Scenario. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.



:ReserveRoomInterface
:MeetingControl
:RoomControl
:Meetings
Client
RequestRoom(Client,RoomNumber) (V3)
RequestMeeting(Client,RoomNum) (V2)
CreateMeeting(RoomNumber) (V2)
CancelMeeting(meeting) (V3)
UpdateMeeting(meeting) (V2)

Figure 14 Meeting Creation

The diagram above shows the logic behind the account creation and profile updation. The MeetingControl and RoomControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.8 Task Use Case: Edit Meeting Participants

Project Name:	Meeting Schedule System
Use Case Name:	Edit Meeting Participants
User Goal:	Client can add/remove people from meeting
Level:	Task
Relevant User Reqs:	None
Relevant System Reqs:	None
Primary Actor:	The client
Precondition:	Client must have an account and there must exist a meeting
Minimal Guarantee:	The system reloaded
Success Guarantee:	The client edits meeting as intended
Fail Case:	The client is unable to make changes to their meeting
Consequence of Failure:	The client will be dissatisfied with the software
Associated Risks:	Product purpose violation
Trigger:	The client clicks the room with the reservation
Success Scenario:	**Step Actions**
	1 Use case begins when the user clicks on the room with the reservation
	2 The system pops up a list of participants
	3 The client makes the desired amendments. Red button to remove participant and “add participant button” to add them
	4 The user clicks <<Save>>
	5 The system shows a success message and will update the database.
Extensions:	**Branching Scenarios**
4	**Condition: Participants has a meeting time conflict**
	Step Actions
	1 The system will send the notification to the participants having time conflict
	2 The system will notify the meeting creator that there are participants who may not join the meeting due to time conflict

The figure above is the Meeting Management Use Case which states in detail how the system manages a meeting in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

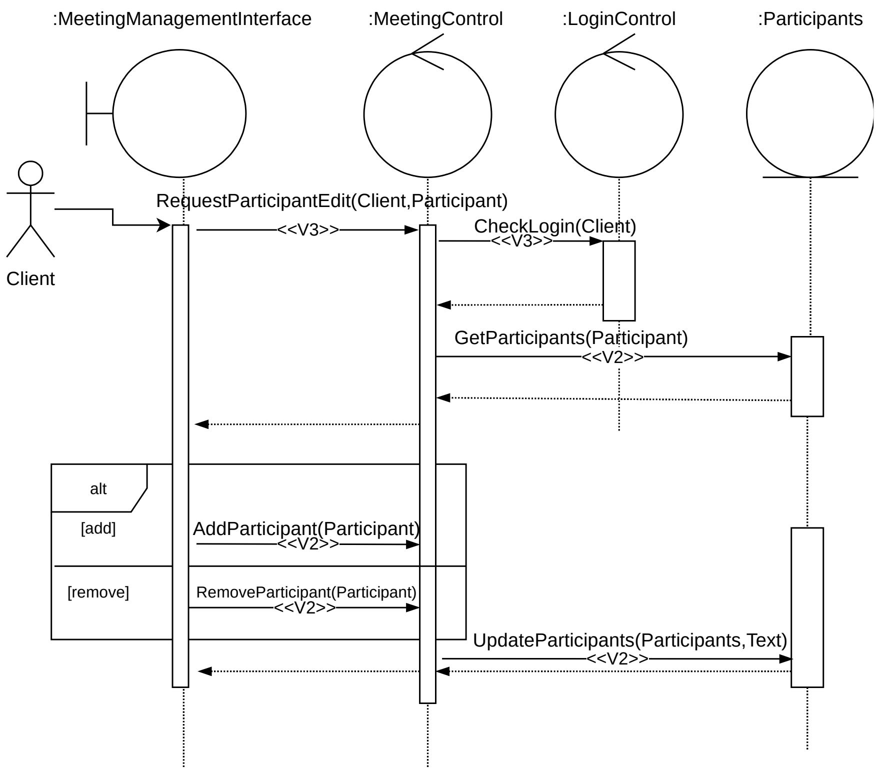

:MeetingManagementInterface
:MeetingControl
:LoginControl
:Participants
Client
RequestParticipantEdit(Client,Participant)
<<V3>>
CheckLogin(Client)
<<V3>>
GetParticipants(Participant)
<<V2>>
alt
[add]
AddParticipant(Participant)
<<V2>>
[remove]
RemoveParticipant(Participant)
<<V2>>
UpdateParticipants(Participants,Text)
<<V2>>

Figure 15 Meeting Management Sequence Diagram

The diagram above shows the logic behind the meeting management. The MeetingControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.9 Task Use Case: Display/Edit Created Meetings

Project Name:	Meeting Schedule System	
Use Case Name:	Edit Meetings	
User Goal:	Client can cancel or view the meetings they made reservations for	
Level:	Task	
Relevant User Reqs:	None	
Relevant System Reqs:	None	
Primary Actor:	The client	
Precondition:	Client must have an account and there must exist a meeting	
Minimal Guarantee:	The system reloaded	
Success Guarantee:	The client can view their meetings and change them accordingly	
Fail Case:	The client is unable to make changes to their meetings	
Consequence of Failure:	The client will be dissatisfied with the software	
Associated Risks:	Meeting time conflict may arise	
Trigger:	The client clicks the "my meetings" button on their profile	
Success Scenario:	**Step Actions**	
	1	Use case begins when user clicks on the button to view their meetings
	2	The system pops up a list of their meetings
	3	The client makes the desired amendments.
	4	The user clicks “save”
	5	The system shows a success message and will update the database.

Extensions:	Branching Scenarios
2	**Condition: There is no meeting**
	Step Action: The system displays a message saying that the profile has no meeting
4	**Condition: Participants has a meeting time conflict**
	Step Actions
	1 The system will send the notification to the participants having time conflict
	2 The system will notify the meeting creator that there are participants who may not join the meeting due to time conflict

Note: This use case is going to serve as the use case for the functional requirement of the system displaying the meetings that the user is assigned to participate in. This display will be shown after the client presses the "my meetings" button.

The figure above is the Display/Edit Created Meeting Use Case which states in detail how the system displays and edits a meeting in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

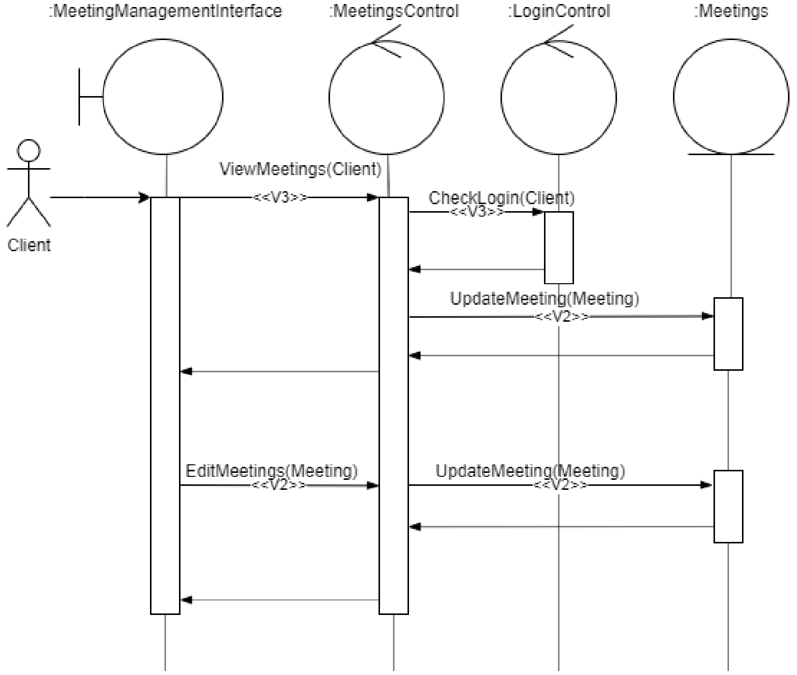


Figure 16 Display/Edit Created Meetings

The diagram above shows the logic behind how the system displays and allows a user to edit a created meeting. The MeetingControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.10 Task Use Case: File Complaint

Project Name:	Meeting Schedule System	
Use Case Name:	File Complaint	
User Goal:	Client can file a complaint to the system administrators	
Level:	Task	
Relevant User Reqs:	None	
Relevant System Reqs:	None	
Primary Actor:	The client	
Precondition:	Client must have an account	
Minimal Guarantee:	The system reloaded	
Success Guarantee:	The client can send a complaint that is received by an administrator	
Fail Case:	The client is unable to file a complaint	
Consequence of Failure:	The client will be dissatisfied with the software	
Associated Risks:	User experience cannot be improved	
Trigger:	The client clicks the “file a complaint” button on the main page	
Success Scenario:	**Step**	**Actions**
	1	Use case begins when the user clicks on the “complaints” button on the menu
	2	The system shows the previous complaints and current complaints
	3	The client clicks on the “new complaint” button.
	4	The client fills out their complaint issue and can add 2 images up to 5 Mb each
	5	The client clicks “save” to create the complaint
	6	The system shows a success message and updates the database
Extensions:	**Branching Scenarios**	
	Condition: the user edits an open or pending complaint	
	1 The client clicks on “edit” to open an existing complaint	
	2 The client can delete the complaint or edit the complaint message and images	
	3 The client then clicks “save” to update their changes	
	4 The system shows a success message and updates the database	
	Condition: the user clicks on a closed complaint	
	1 The system shows the complaint in preview mode. No edits can be made	

The figure above is the File Complaint Use Case which states in detail how the system files a complaint in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

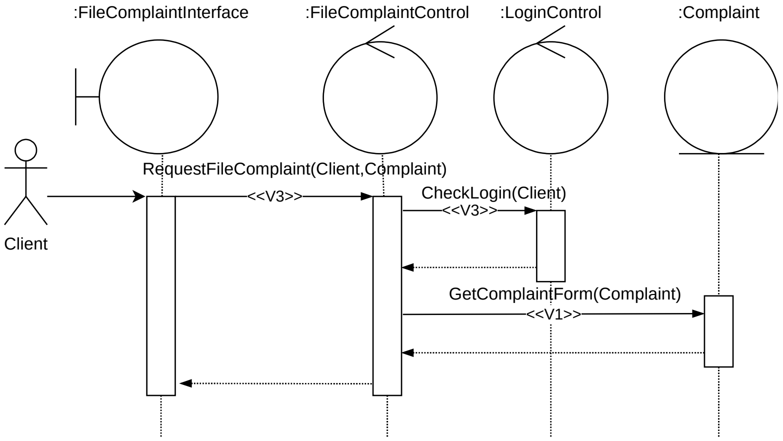


Figure 17 File Complaint Sequence Diagram

The diagram above shows the logic behind how the system files a complaint. The FileComplaintControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.11 Task Use Case: Administrator Add/Delete Rooms

Project Name:	Meeting Schedule System
Use Case Name:	Admin Add/Delete Rooms
User Goal:	The administrator can change the rooms that are available for reservations
Level:	Task
Relevant User Reqs:	None
Relevant System Reqs:	None
Primary Actor:	The administrator
Precondition:	Administrator must have an account
Minimal Guarantee:	The system reloaded
Success Guarantee:	The admin adds and deletes rooms as desired
Fail Case:	The admin is unable to edit the list of rooms
Consequence of Failure:	The software will not accurately reflect the rooms that can be reserved
Associated Risks:	Meeting room conflict may arise, so the company may stop using the product
Trigger:	The admin selects the “edit rooms” button
Success Scenario:	**Step Actions**
	1 Use case begins when the user clicks on the “edit rooms” button
	2 The system pops up the list of rooms available to clients and the option to add another one or delete an existing one
	3 The admin makes the desired changes and hits “save”
	4 The system shows a success message and saves the changes
Extensions:	**Branching Scenarios**
3	**Condition: A room that the administrator wants to delete will be used**
	Step Action: The system pops up the error message telling: “The room has been reserved! Please come back when the meeting ends.”

The figure above is the Add/Remove a Room Use Case which states in detail how the system allows an administrator to add/remove a room in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

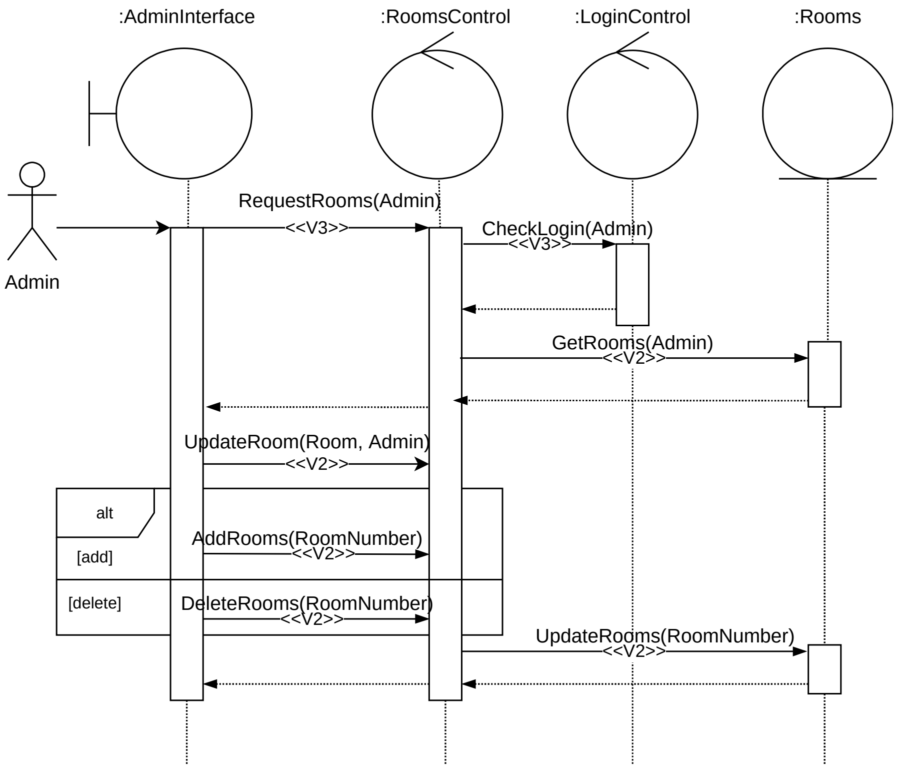


Figure 18 Admin Add/Delete Rooms Sequence Diagram

The diagram above shows the logic behind how the system adds/removes a room. The RoomControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.12 Task Use Case: Administrator Update Clients Billing Information

Project Name:	Meeting Schedule System	
Use Case Name:	Admin Update Clients Billing Information	
User Goal:	The administrator can change the billing information for a client	
Level:	Task	
Relevant User Reqs:	None	
Relevant System Reqs:	None	
Primary Actor:	The administrator	
Precondition:	Administrator must have an account	
Minimal Guarantee:	The system reloaded	
Success Guarantee:	The admin can update a client’s billing information	
Fail Case:	The client is stuck with old billing information	
Consequence of Failure:	This could interfere with payments	
Associated Risks:	The client is unable to reserve the special room.	
Trigger:	The admin selects the “Edit clients billing information” button	
Success Scenario:	**Step**	**Actions**
	1	Use case begins when the user clicks on the “Edit clients billing information” button
	2	The system pops up the list of clients
	3	The admin selects the client whose billing information he would like to change
	4	The admin makes the changes
	5	The admin presses the <<Save>> button
	6	The client is notified that their information has been changed via email

Extensions:	Branching Scenarios
6	**Condition: Client is unable to receive the change**
	Step Actions
	1 After 5 minutes, if the client is unable to receive the change, the system reloaded
	2 The system will notify the administrator to check whether the change has been saved then resend an email to the client to check the billing.

The figure above is the Update Client's Billing Use Case which states in detail how the system allows an administrator to update the client's billing in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

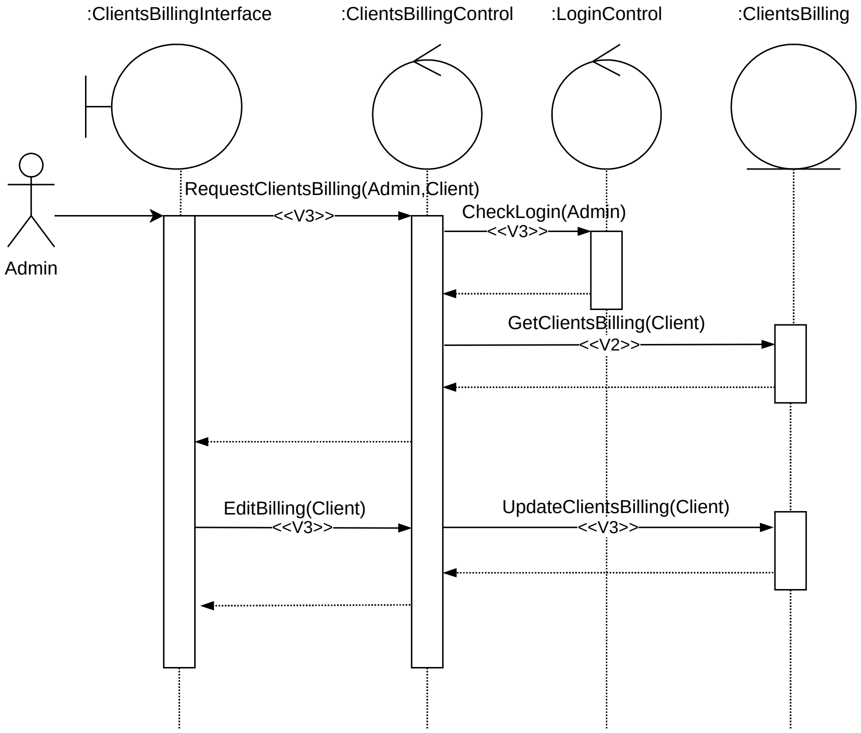


Figure 19 Edit Clients Billing Sequence Diagram

The diagram above shows the logic behind how the system edits client's billing. The ClientsBillingControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.13 Task Use Case: Administrator View/Respond to Complaints

Project Name:	Meeting Schedule System
Use Case Name:	Admin view/respond to complaints
User Goal:	The administrator can view complaints submitted by clients and respond to them
Level:	Task
Relevant User Reqs:	None
Relevant System Reqs:	None
Primary Actor:	The administrator
Precondition:	Administrator must have an account
Minimal Guarantee:	The system reloaded
Success Guarantee:	The admin can view the complaints and respond to them
Fail Case:	The admin does not receive complaints

Consequence of Failure:		The admin will not know the issues the clients are having so they will not be able to help them
Associated Risks:		The administrator cannot improve the user experience.
Trigger:		The admin selects the “inbox” button
Success Scenario:		**Step Actions**
	1	Use case begins when the user clicks on the “complaints” button on the menu
	2	The system pops up the list of complaints that have been sent in
	3	The admin selects the edit button next to the complaint
	4	The admin adds comments to the complaint. Then closes it or leaves it as pending (not fixed yet)
	5	The admin saves their changes.
	6	The client gets notified via email that the complaint has been updated
Extensions:		**Branching Scenarios**
3		**Condition: The complaint is already closed**
		Step Action: The administrator must change the status to pending to update it

The figure above is the Complaints Management Use Case which states in detail how the system allows an administrator to view/respond to complaints in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

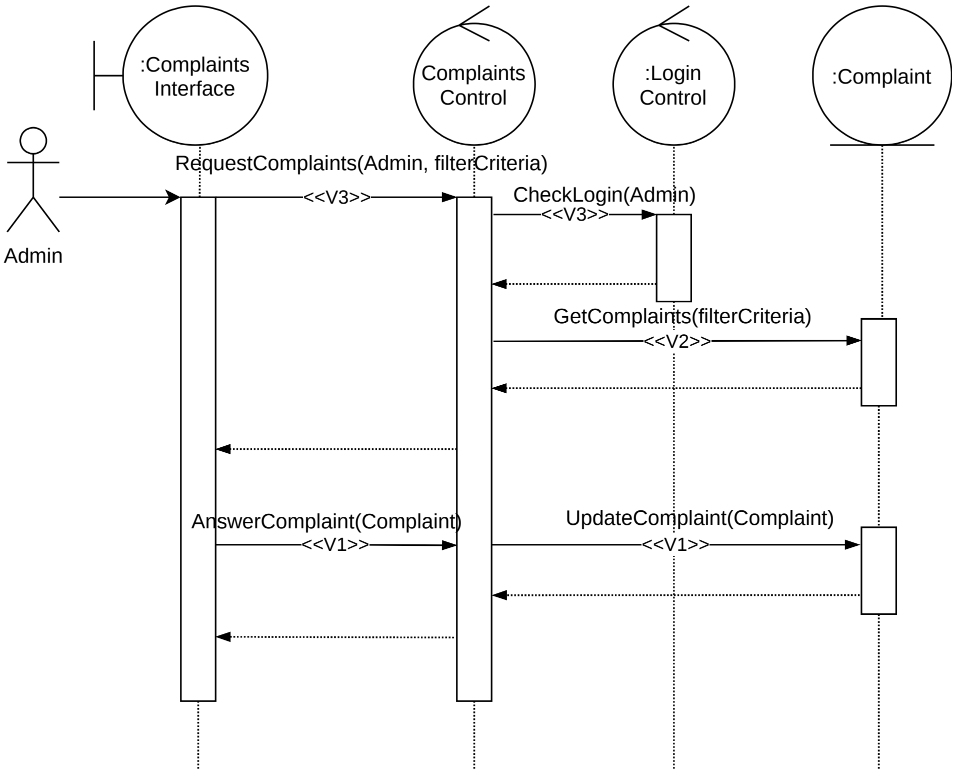

:Complaints Interface
Complaints Control
:Login Control
:Complaint
Admin
RequestComplaints(Admin, filterCriteria)
<<V3>>
CheckLogin(Admin)
<<V3>>
GetComplaints(filterCriteria)
<<V2>>
AnswerComplaint(Complaint)
<<V1>>
UpdateComplaint(Complaint)
<<V1>>

Figure 20 Manage Complaints Sequence Diagram

The diagram above shows the logic behind how the system manages complaints. The ComplaintsControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.7.14 Task Use Case: Administrator View/Filter Meetings

Project Name:	Meeting Schedule System	
Use Case Name:	Admin view/filter meetings	
User Goal:	The administrator can view all the meetings and filter by time, room, date, week, and participants	
Level:	Task	
Relevant User Reqs:	None	
Relevant System Reqs:	None	
Primary Actor:	The administrator	
Precondition:	Administrator must have an account	
Minimal Guarantee:	The system reloaded	
Success Guarantee:	The admin can view all the meetings and filter through them	
Fail Case:	The admin does not have access to all the meetings or cannot filter them	
Consequence of Failure:	The admin will be less capable of overseeing everything without access to this information	
Associated Risks:	User experience cannot be improved	
Trigger:	The admin selects the “meetings” button	
Success Scenario:	**Step**	**Actions**
	1	Use case begins when the user clicks on the “meetings” button
	2	The system pops up the list of all the meetings in chronological order and numerical order by room number
	3	The admin selects the “filter by” button
	4	The admin fills out which criteria they would like to filter by
	5	The system displays all the meetings that meet these criteria
Extensions:	**Branching Scenarios**	
5	**Condition: the loading time is longer than 1 minute**	
	Step Actions	
	1 The system breaks the result into many pages	
	2 The system shows the loaded result first to reduce the waiting time.	

The figure above is the View/Filter Meetings Use Case which states in detail how the system allows an administrator to view/filter meetings in the Success Scenario, and how the system behaves in the Extension if the system encounters a problem. This use case indicates who is the main actor, or the main user, pre and post conditions, minimal and success guarantee, fail case, consequence of failure as well as associated risks.

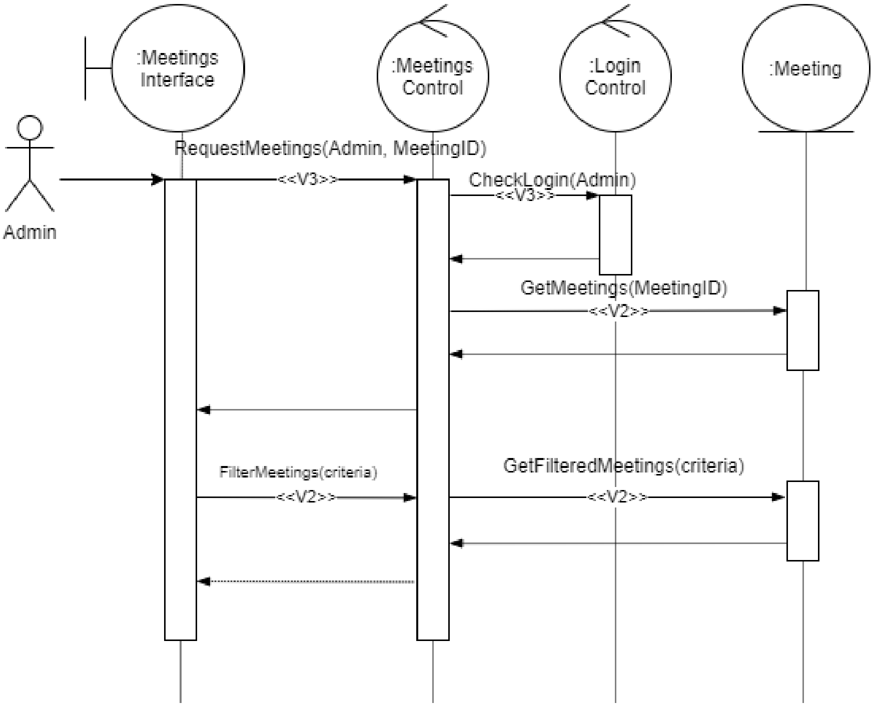


Figure 21 Administrator Views/Filters Meetings

The diagram above shows the logic behind how the system allows an administrator to view and filter meetings. The MeetingControl and LoginControl will check the user authentication. If the user has the authority, the system will provide service. Otherwise, the system will provide no service.

2.8 Vulnerability Mapping

2.8.1 High Priority (V3)

- To prevent SQL injection in any part of the system, the backend will not use concatenation to add SQL parameters.
- **RequestRoom(Client, RoomNumber), RequestAccount(Client, AccountDetails), RequestPartcipantEdit(Client, Participant), ViewMeetings(Client), CancelMeetings(RoomNumber/Meeting), and RequestFileComplaint(Client, Complaint)**:
  - The AccountDetail is sensitive. Therefore, before this data transmission, the data must be encrypted.
  - Since the data is from user input, all requests must go through the CheckLogin, CheckPennAccount, or CheckCredentials. If one of these functions returns false, the system will direct the user to the Not Authenticated page.
  - If the user did not log in, the system will direct the user safely to the Not Authenticated page.
- **CheckLogin(Client/Admin) and CheckCredentials(Client)**:
  - If the return value is false, the system will direct the user to the Not Authenticated page.
- **CheckPennAccount(Client)**:
  - This function will check PennID and PennAccount to ensure the user is a company employee. The PennID and PennAccount must follow the company ID and Account formats.
- **RequestRooms(Admin, filterCriteria), RequestComplaints(Admin, filterCriteria), RequestMeetings(Admin, MeetingID):**
  - The admin retrieves and filters the data through these interactions. If the admin does not have the privilege or is not logged in, they will be directed to a Not Authorized page or redirected to the sign-in page (respectively). This data is not sensitive, so it does not need to be encrypted.

- **UpdateMeeting(Admin, Meeting), UpdateRoom(Admin, Room), UpdateComplaint(Admin, Complain), UpdateMeetingParticipants(Admin or Client, Meeting):**
    - When updating these entities, the user might add scripting in a text field. This will cause XSS attacks. All text editors will remove any scripting before updating the data.
    - If the user does not have the privilege to update the room, they will be redirected to the Not Authorized page
    - If the user tries to update an entity in use (such as a room that is used) they will not be allowed to do so.
    - Any updates to Meetings or Rooms will be logged in a log table because this information is sensitive.
- **EditBilling(Client or Admin):**
    - If the billing information entered is not valid or is detected to be spam the user will be met with an error that the data is not valid. If an admin edits a client's data, the client will be notified and the change will be logged in the log table.

2.8.2 Moderate Priority (V2)

- It is impossible to prevent packet sniffing even when using the RESTful API standard. Therefore, any sensitive data being transmitted such as billing information or passwords will be encrypted.
- Data in the local storage will also be encrypted.
- The system will use JWT for authentication. This ensures that no data can be retrieved or updated without the user being logged in.

2.8.3 Low Priority (V1)

- If a process fails in the API or the web client, it will be handled using the appropriate exception-handling mechanism and an error will be shown to the user. Details of the errors will not be sent to the client app as not to expose the code to any potential hackers.

3. Team Members Log Sheets

3.1 Eesaa Philips

date	task	duration
June 3, 2021	UI diagrams	3 hr
June 4, 2021	2.8	1 hr
June 4, 2021	2.3, 2.4, 2.5	1 hr
July 3, 2021	Review use case diagrams	1.5 hr
July 3, 2021	Review Sequence diagrams	1.5 hr
	Total :	8 hr

3.2 Garrett Adams

date	task	duration
June 3, 2021	1.1-1.5	1hr
June 16,2021	2.1-2.3	3hr
June 23, 2021	2.6-2.7	4hr
June 29, 2021	Review/Format	2hr
	Total :	10hr

3.3 Huy Tran

date	task	duration
June 19, 2021	1.1-1.5, 2.1	2 hr
June 27, 2021	2.1.1.1, 2.1.1.2, review 2.3 and 2.6	1 hr
June 28, 2021	2.7.1	0.5 hr
June 29, 2021	2.7.4, Review 2.1.1.3-2.1.1.5	1 hr
July 1, 2021	2.2, 2.7.9 extension, fix 2.7.3	2 hr
July 2, 2021	2.7.8, 2.7.9, Review sequence diagram	2.5 hr
July 4, 2021	Final review and format	1 hr
Aug 8, 2021	Revised UI descriptions and use cases	1 hr
	Total :	11