VISUAL DESIGNS

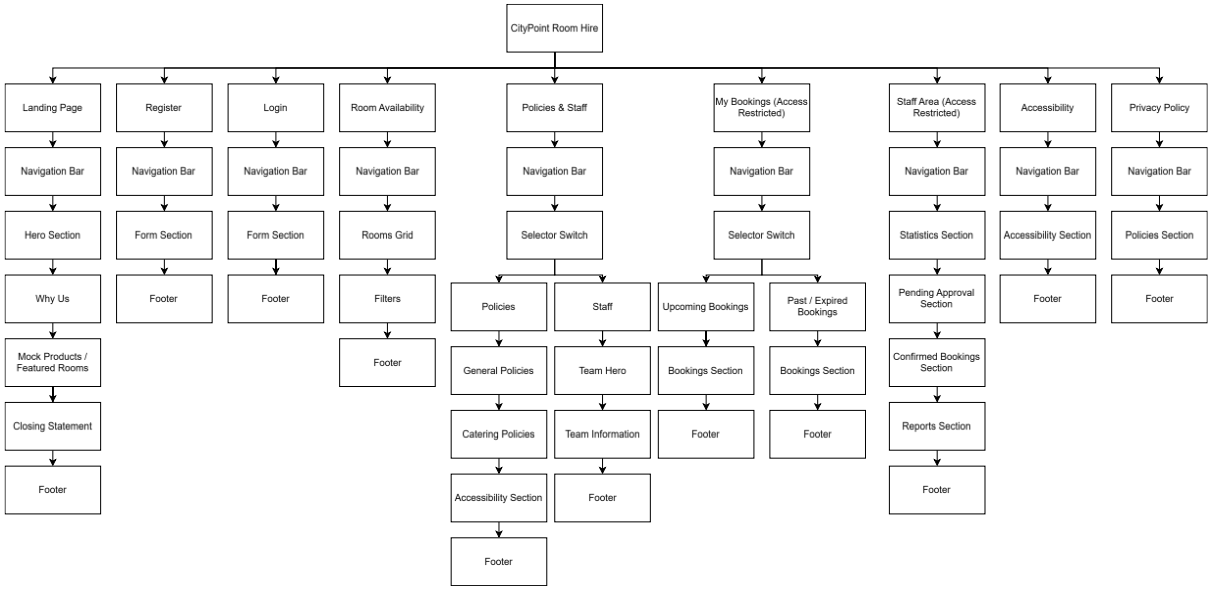
Denas Malysevskis

Table of Contents

Visual Designs 2
Site Map 2
Visual Design 3
Index Page 3
Room Availability 4
Policies & Staff 5
My Bookings 6
Login 7
Register 7
Staff Area 8
Algorithm Design 10
Register Flowchart 10
Login Flowchart 12
Uploading Rooms 13
Edit Rooms 14
Creating Bookings 15
Data Requirements Design 16
ER Diagram 16
Database Dictionary 16
AspNetUsers 16
AspNetRoles 17
AspNetUserRoles 17
Staff 17
Bookings 17
Rooms 18
Test Strategy 18

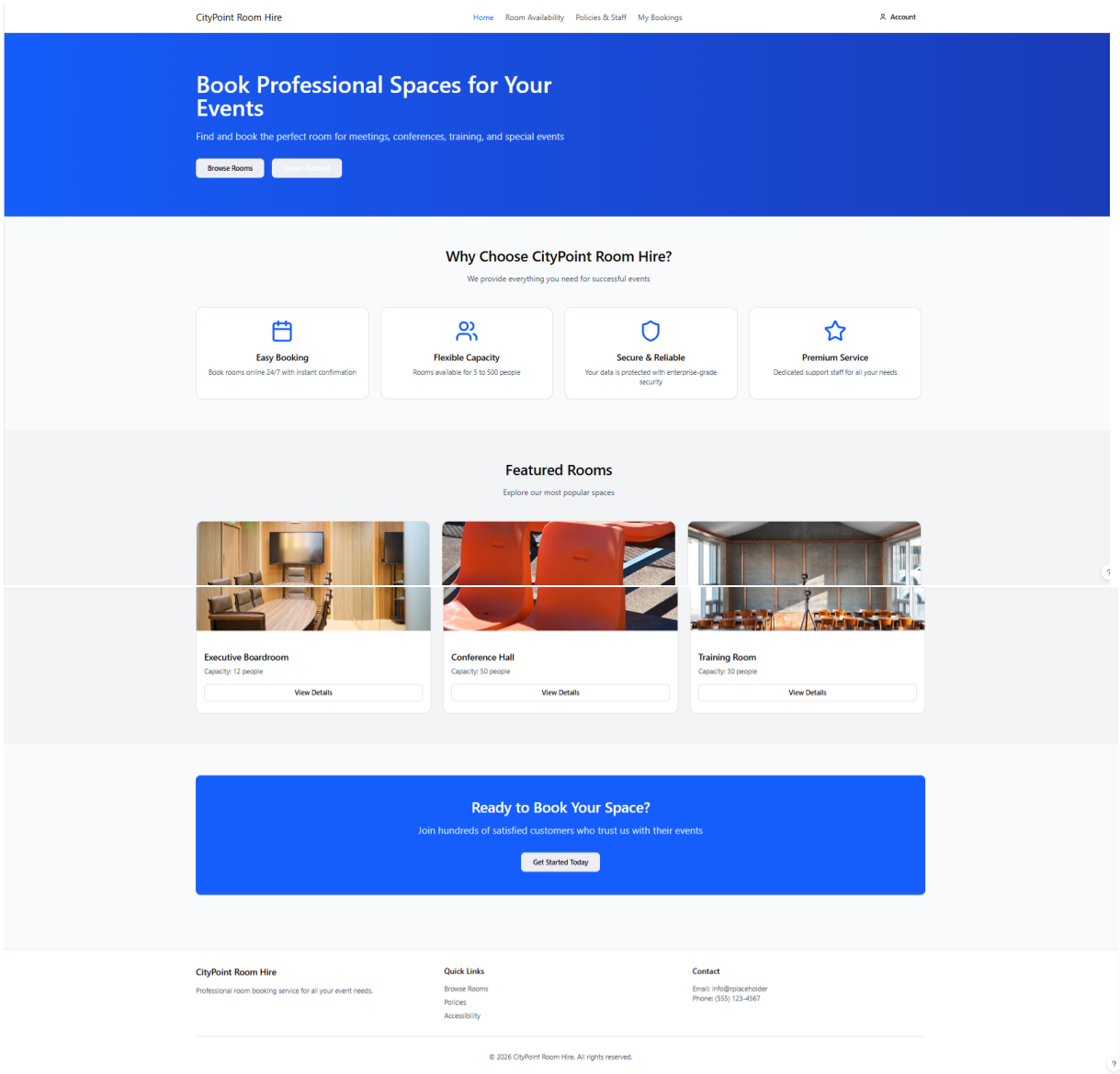
Visual Designs

Site Map



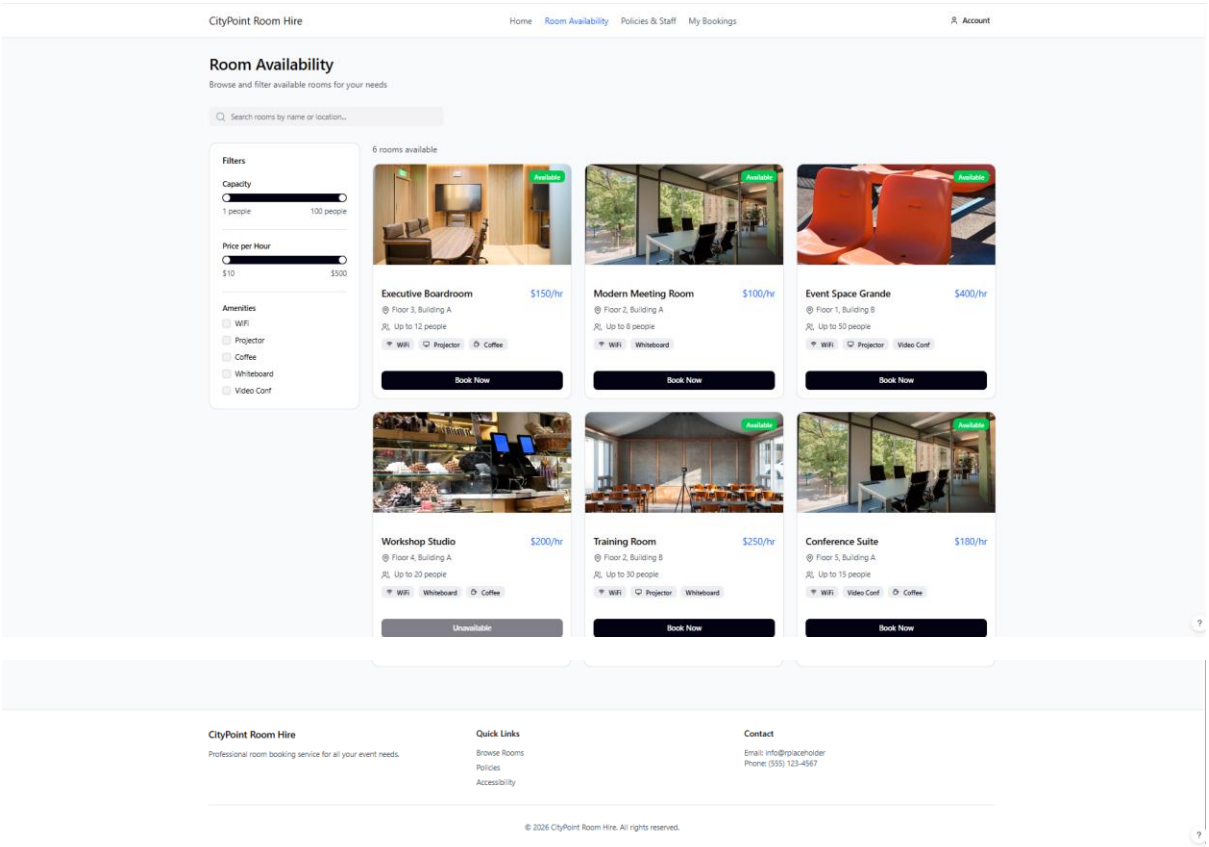
Visual Design

Index Page



On the Index Page, I have gone for a simplistic, office style look that has strictly content and minimal assets to reduce the chances of distraction while maintaining enough content to keep the user hooked to the website. I have used free assets without copyright using Tailwind's extensive library of icons and images. I have then used Unsplash to provide images for the featured rooms. I have then implemented another hook to entice the user to create an account with the client. For the page to link, I have added the 'home' directory in the navigation bar to ensure users can access this page at any time.

Room Availability

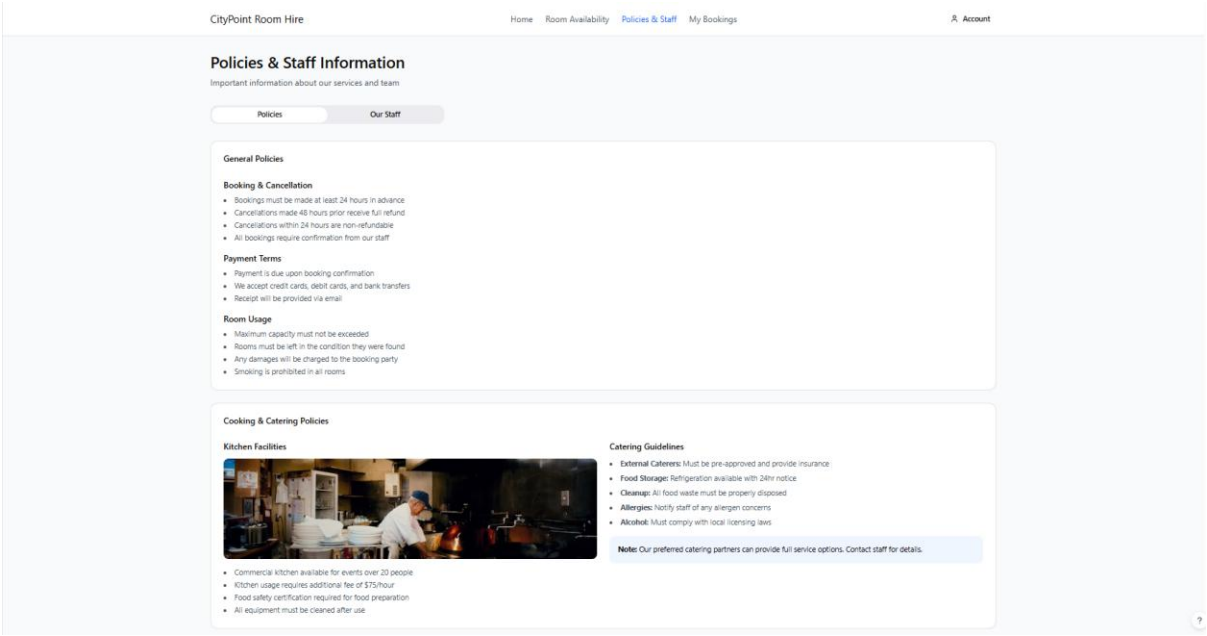


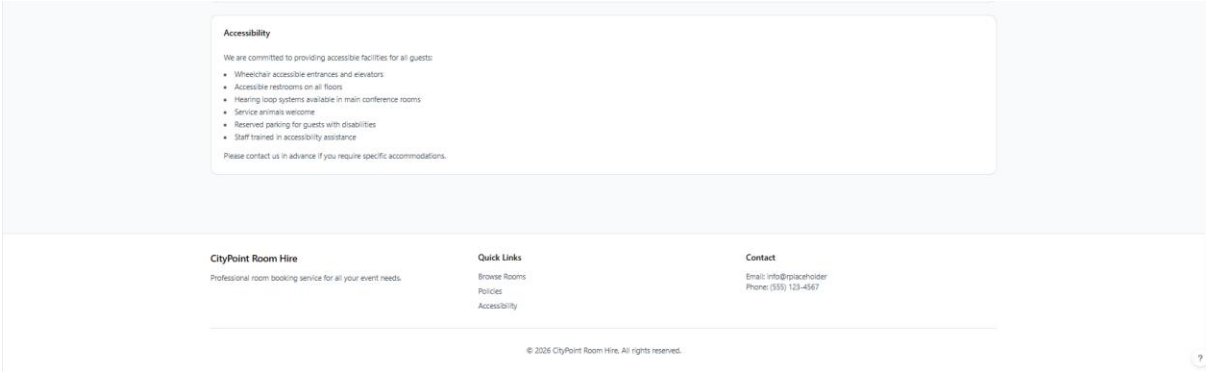
On the Room Availability page, I have continued to go for the plain, office style looks for the website. On the left, I have implemented filters to help the user look for the room that fits best. This is done by prompting the user to choose capacity of the room they need, the price / budget they have to spend per hour and any amenities that they would like. The system then automatically filters the possible options.

To the right of that, I have used a grid system to allow placement of new rooms in a 3 by (whatever amount is required to fit all rooms, e.g., 3x3 or 3x5) to ensure consistent positioning. In all other pages, there is a directory to the available rooms on the navbar.

Policies & Staff

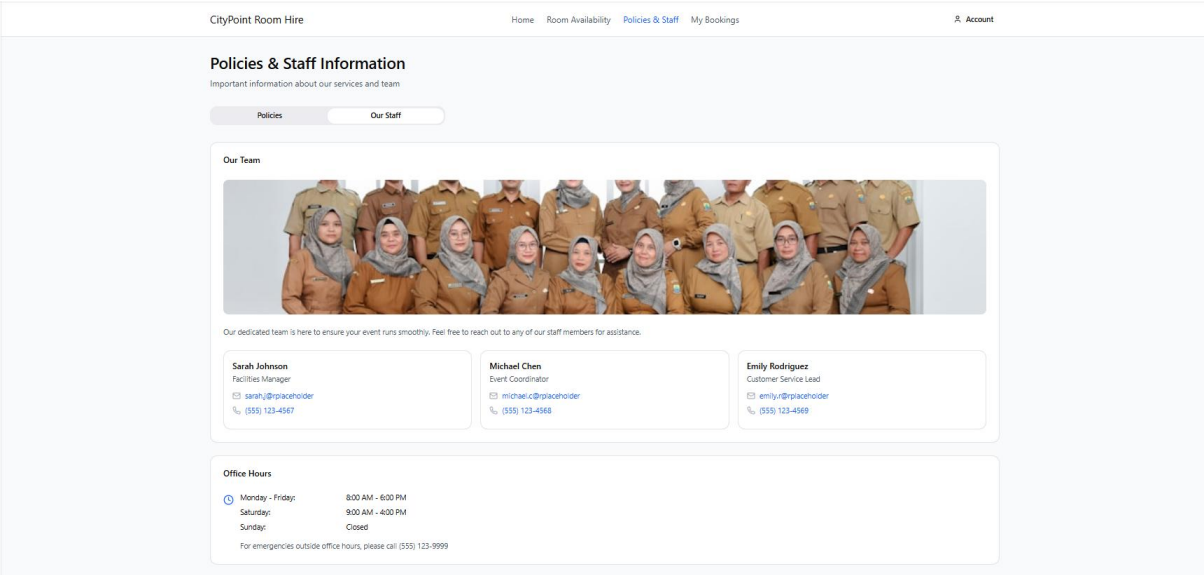
Policies Page





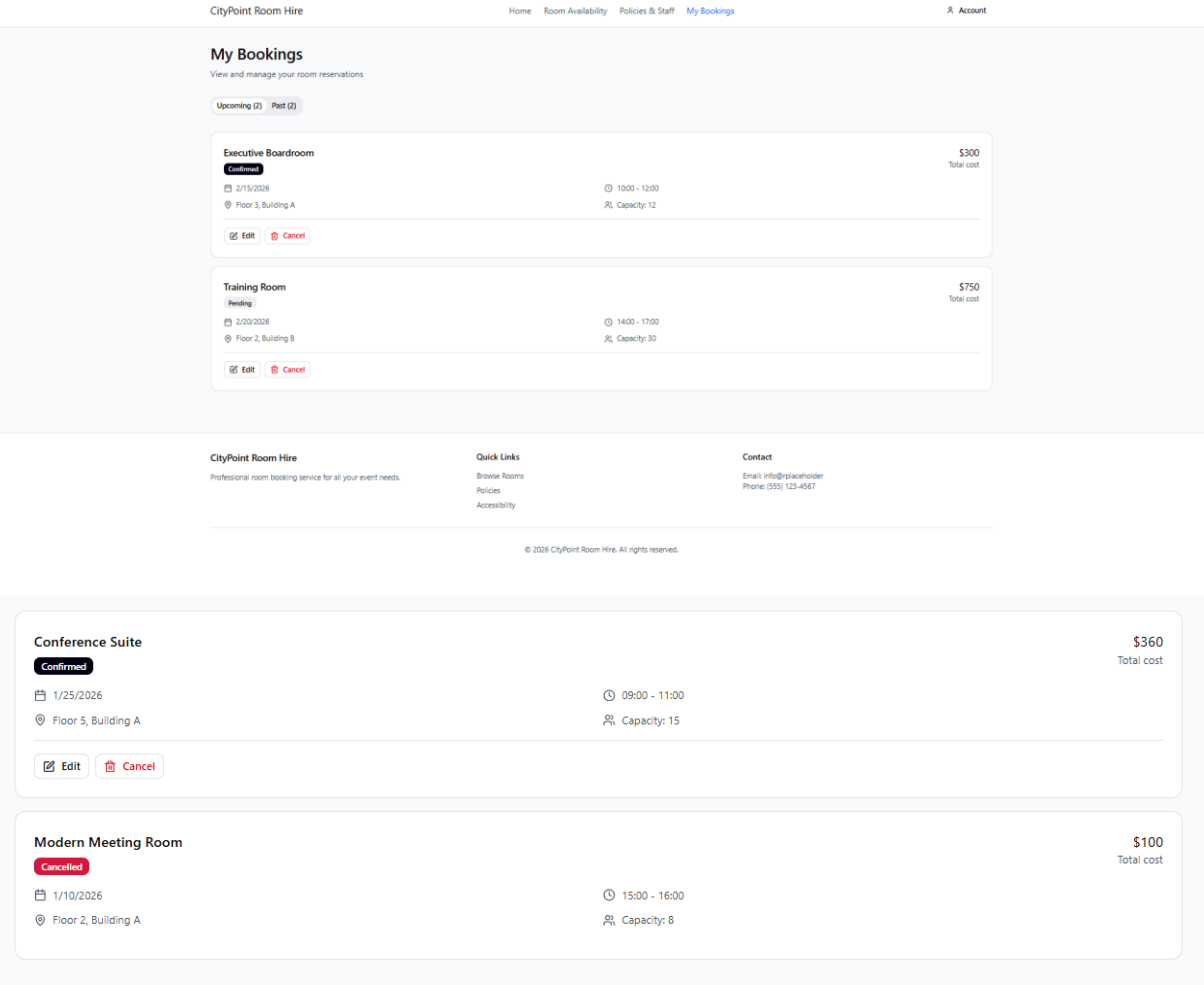
On the policies page, I have added a section selector to reduce clutter on the page. This allows the user to choose which information they would like to see instead of giving both at the same time. Users are able to click the adjacent button to swap content. For the policies content, I have provided general policies and catering policies combined with accessibility, whereas I have placed staff information in the staff section.

Staff Page



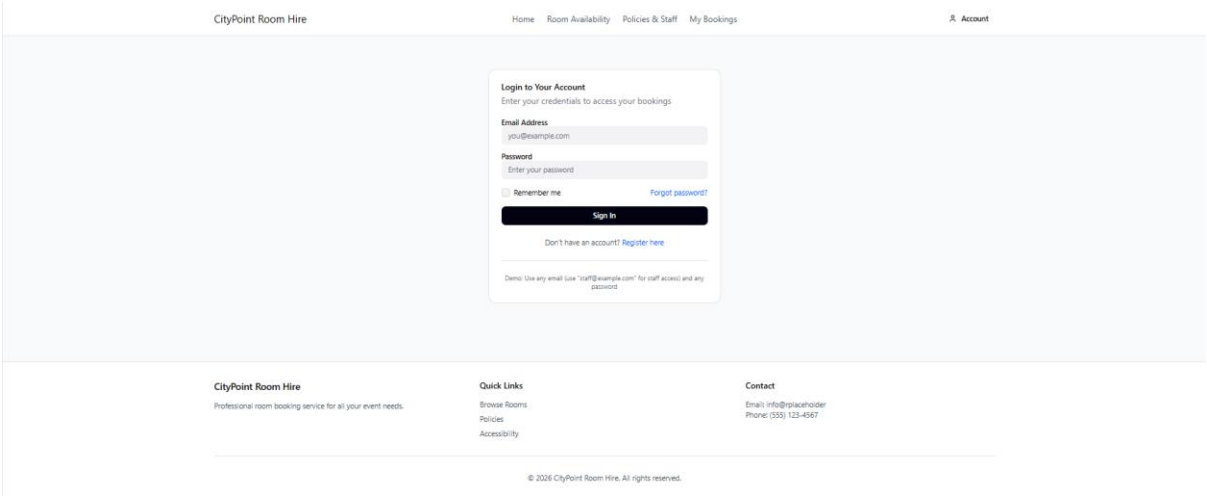
The staff section provides information to the user for contact details to the managers/leads of each division of staff.

My Bookings



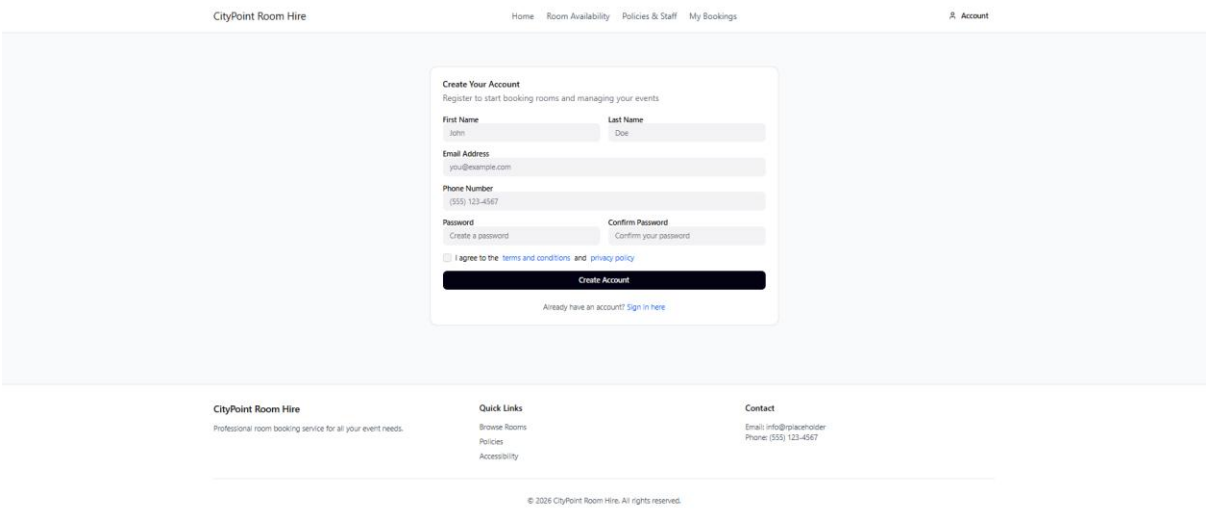
On the My Bookings Page, I have provided users with a Graphical User Interface to manage their bookings that are soon to happen / upcoming and past bookings for the history of their bookings all in one place. Again, this is done with a selection switch. Users can cancel and edit their bookings per to the policies. To note, this is access restricted and is not accessible without being logged in.

Login



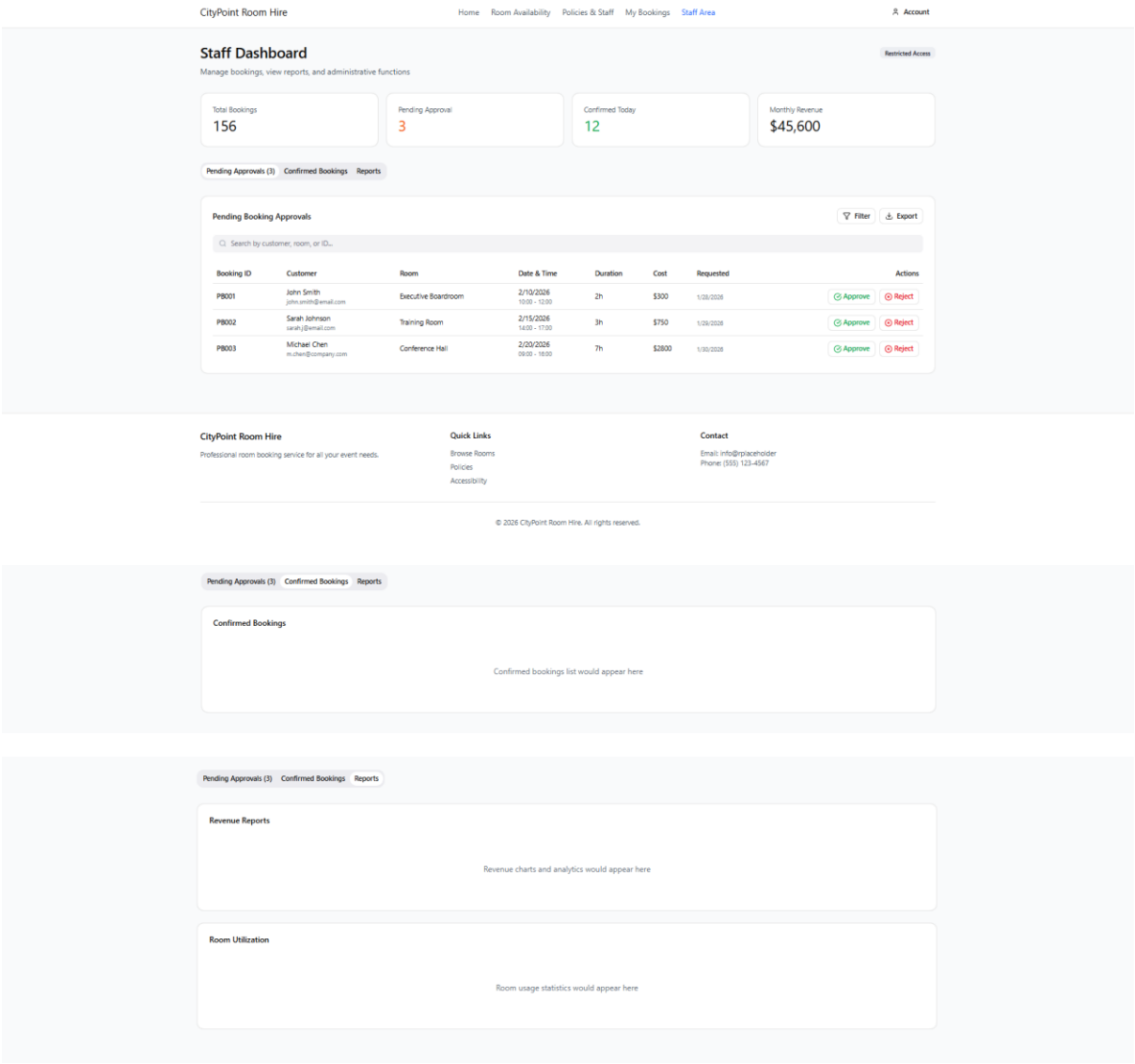
Speaking about logged in, the login page has the same simplistic style as all other pages, providing the user with a form to input their credentials to login. If the user forgets their password, there is a 'forgot password' option to reset a user's password through their email. If a user does not have an account, there is a redirect link that takes the user to go register. If a user logs into the solution, they will be taken to 'My Bookings'.

Register



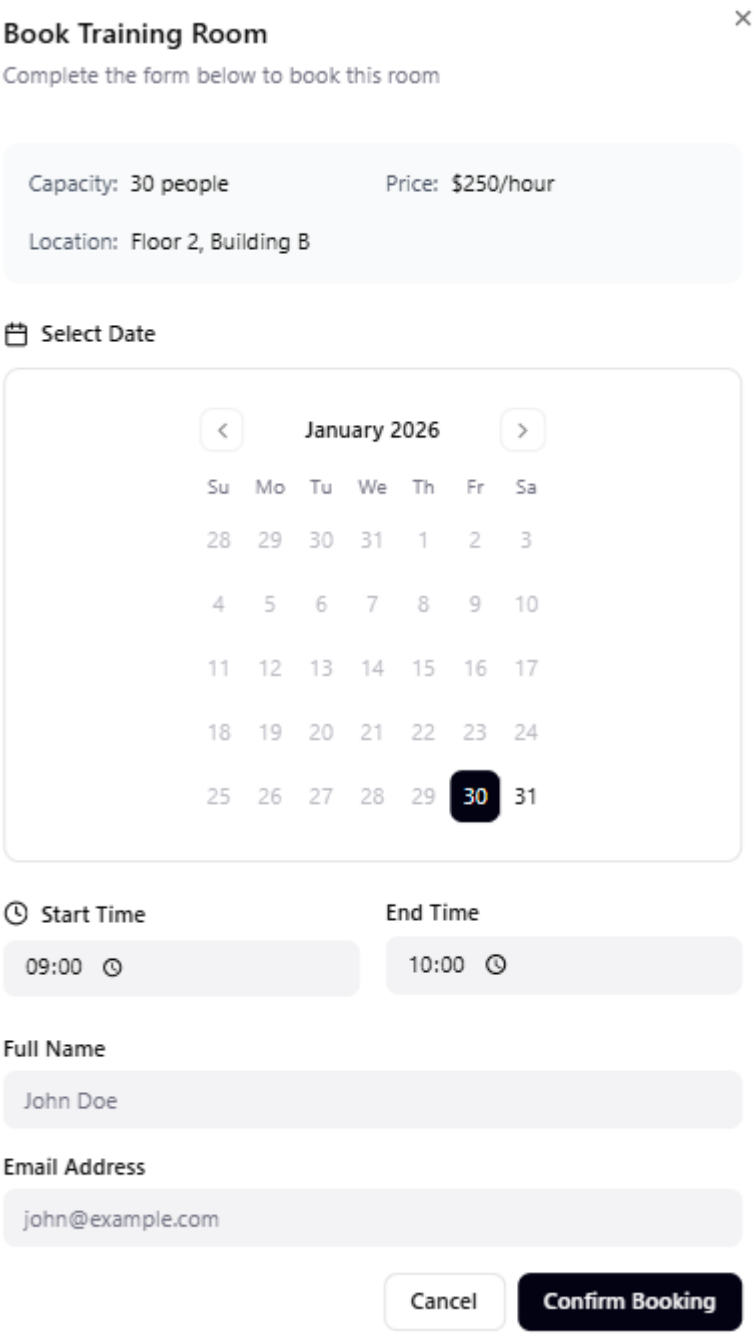
Similar to the Login Page, the Register Page has a form that allows the user to place their information into the information boxes to create an account. If the user has an account, they can choose to click the 'Sign in here' button that will redirect them to the Login Page. Once their request is submitted, the user will be redirected to the Login Page to resume the process.

Staff Area



This is the Staff area page. It is a restricted page only meant for the staff at CityPoint Room Hire. The page allows the staff of CityPoint Room Hire to approve or deny bookings as well as see confirmed bookings and reports. Also, staff can view previous bookings for administrative use.

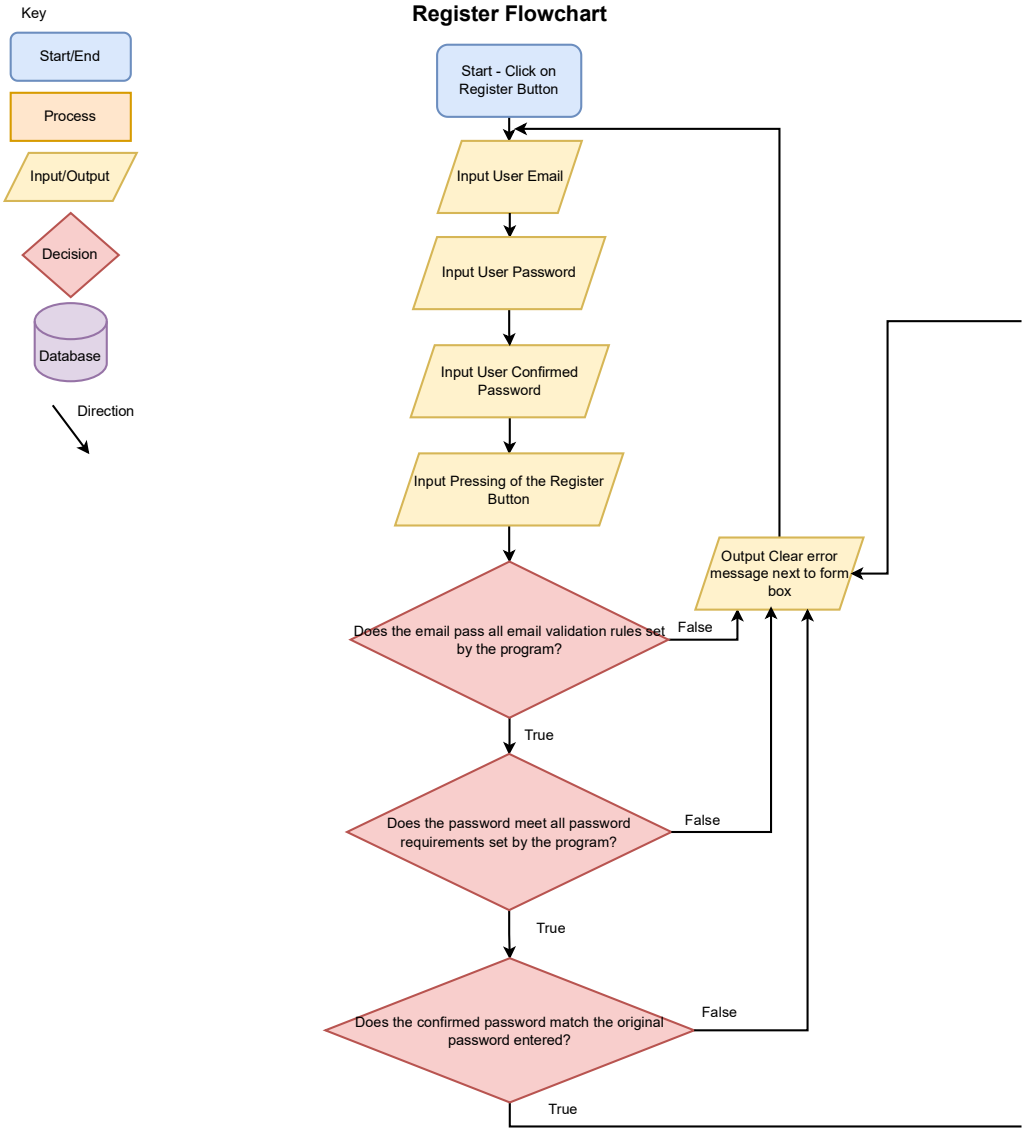
Booking Room GUI

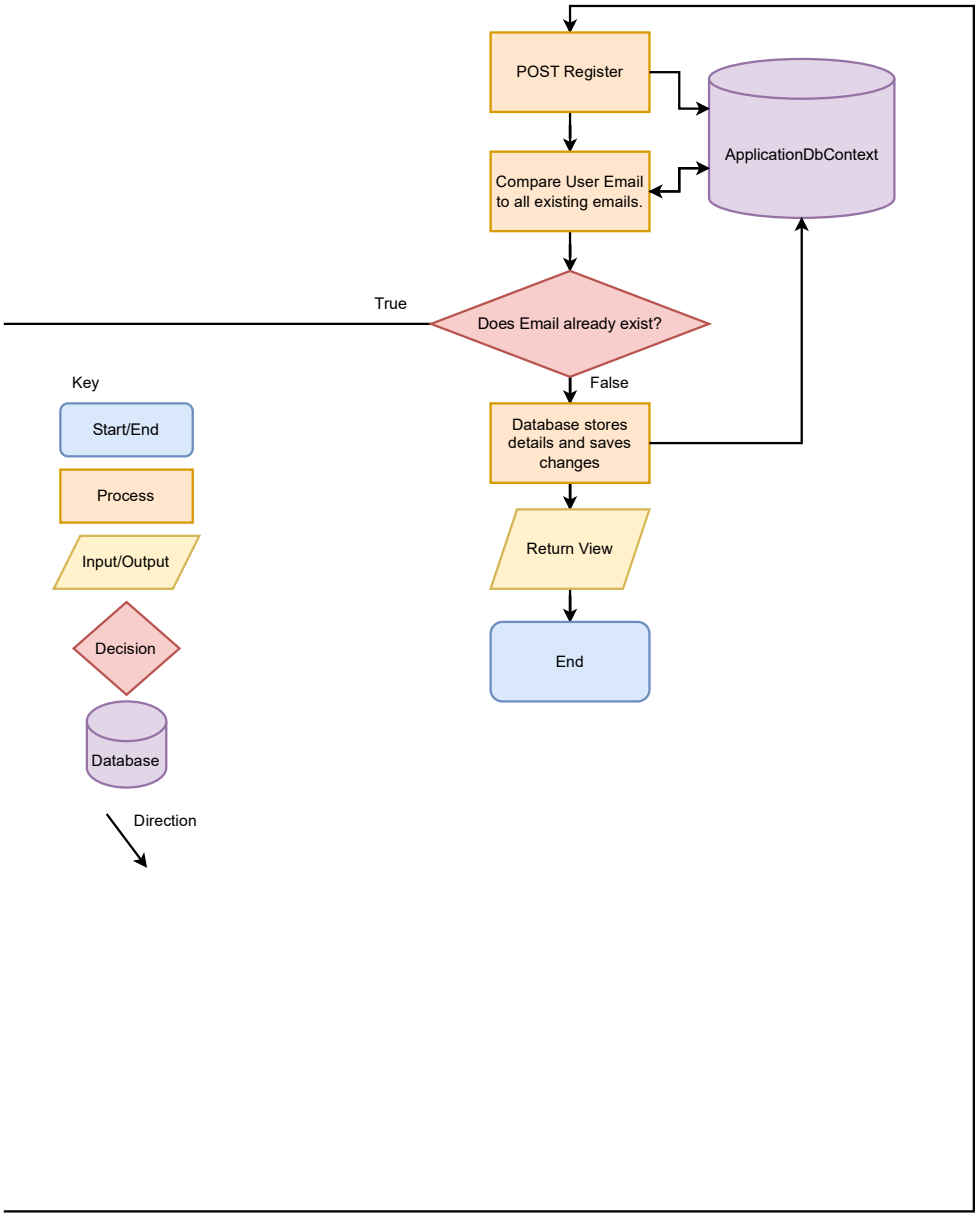


For users to book a room, They must use this Graphical User Interface to input the required information.

Algorithm Design

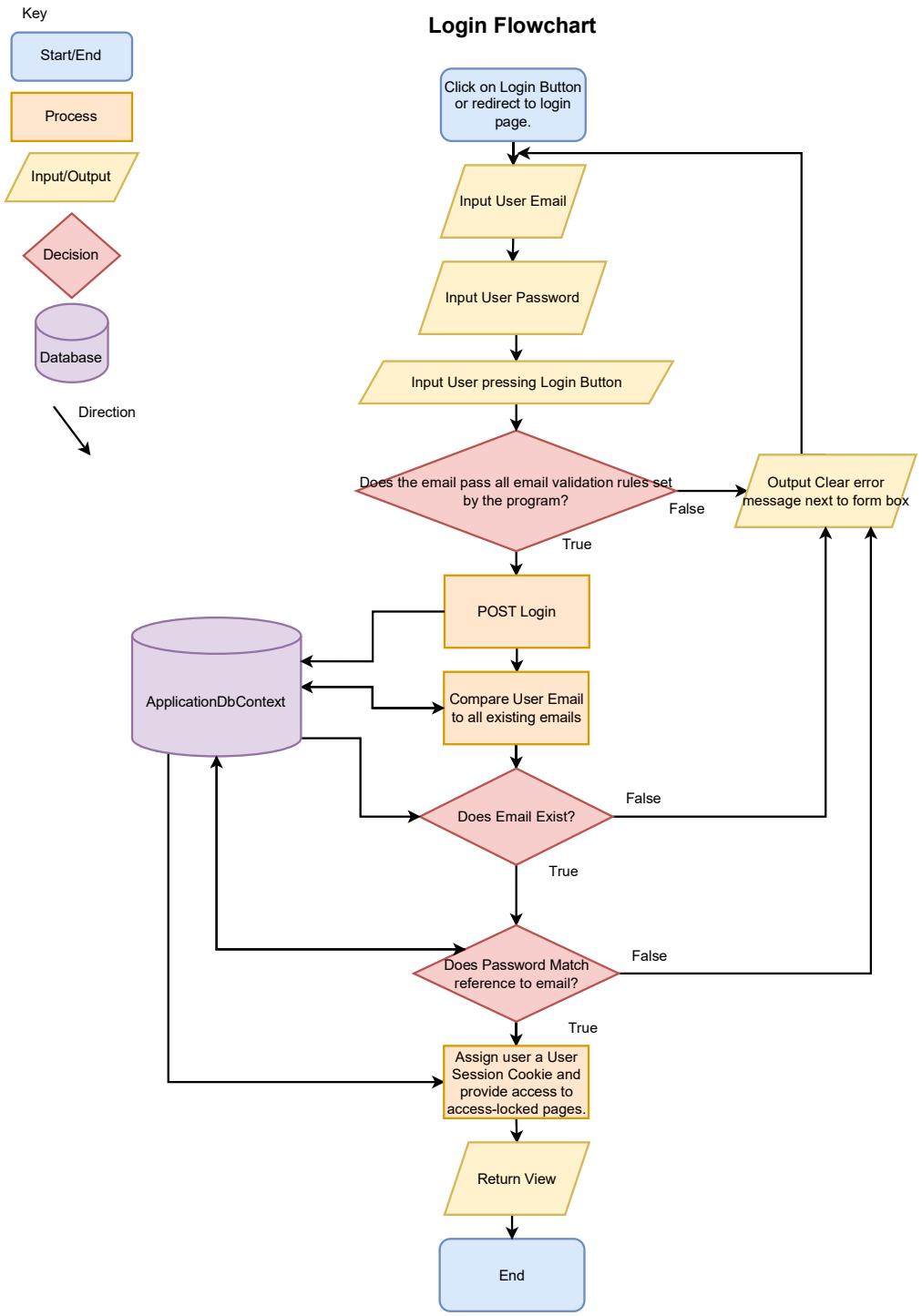
Register Flowchart





The user will be given a Form Box to input an email, a password and a confirmation password box. The details will then be validated, if any of the validation checks come back as false the user will be asked to retry. If the validation checks pass, the data will be sent through POST to the database. Then, the solution will compare to see if the email already exists. If the email already exists, the user will be prompted to use a different email or to sign in. If the email does not already exist, the details are saved in the SQL server database and the user is then returned to the view.

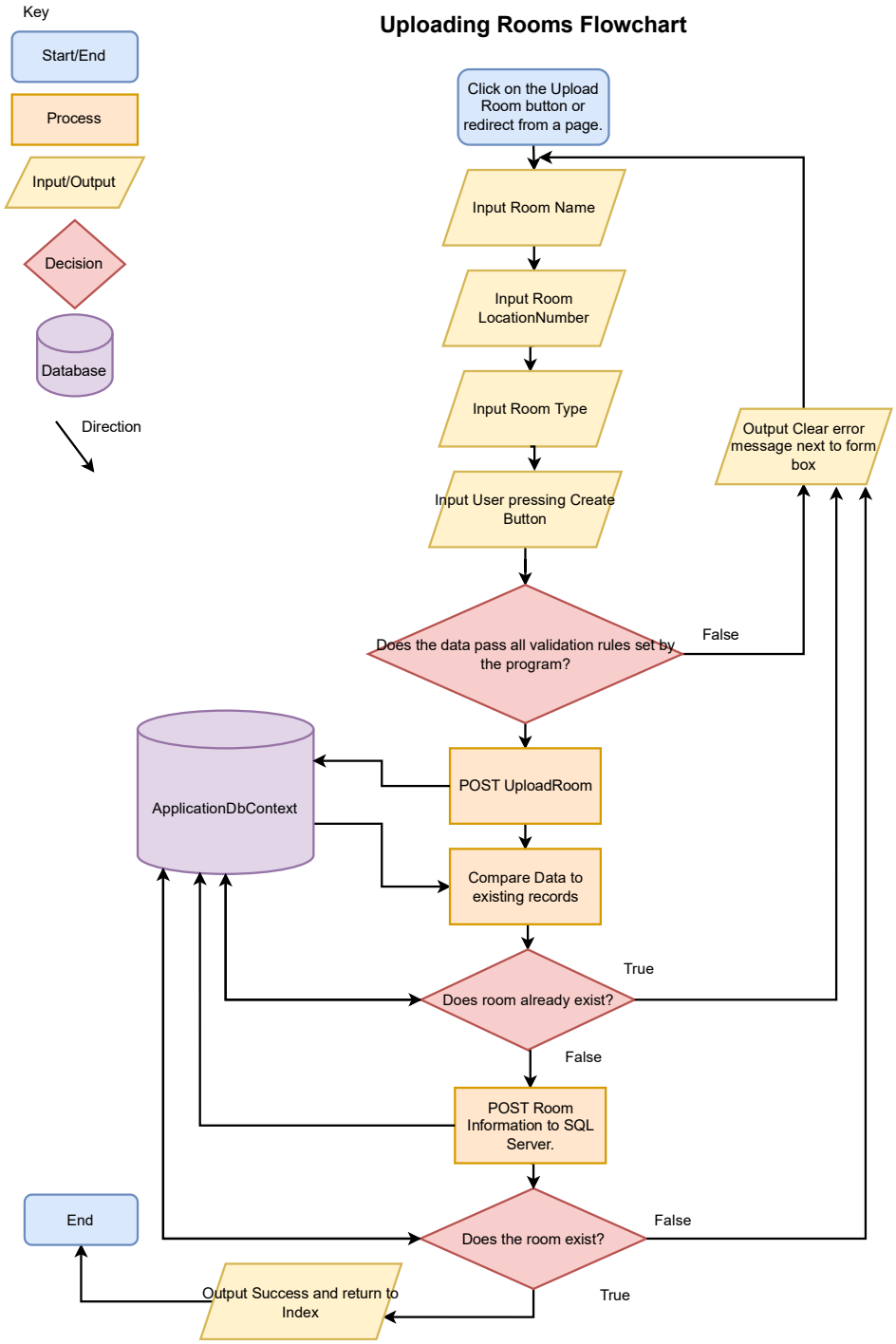
Login Flowchart



The user will be prompted to enter their email and password. The system will then validate the user's information. Then, the solution will compare the email and the password and checks if the account information is correct. If the email doesn't exist or doesn't match, or the password is incorrect, then the solution will display an error message on the form telling the user to try again. If

the data inputted is all correct, the user is assigned a session cookie and is taken to an authorised access locked page.

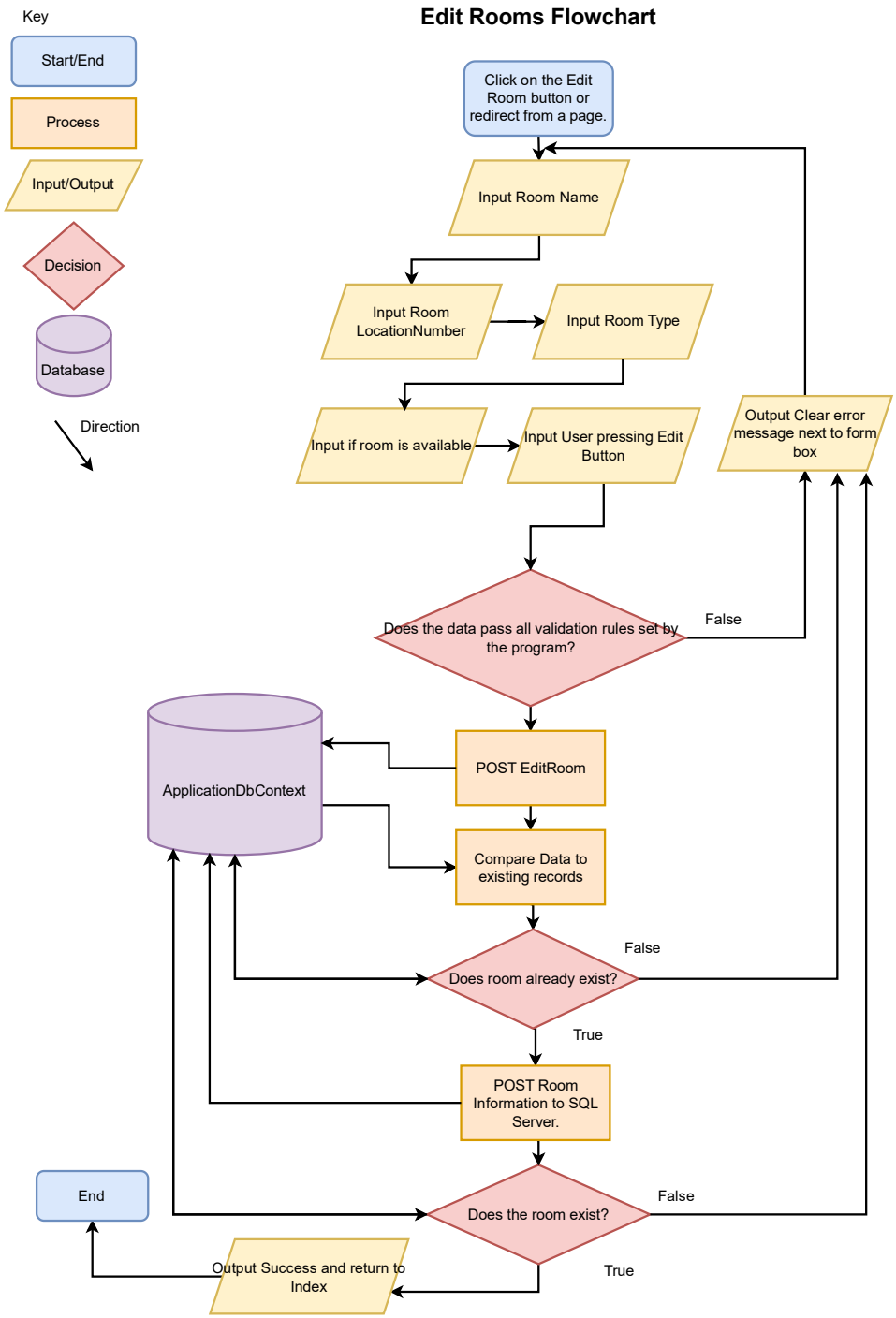
Uploading Rooms



The user will be prompted to provide information about the room, such as the Location Number, name and room type. The information then gets validated. If the validation passes, the information is POST 'ed to the database. The solution checks if the data already exists, and if it does not exist, it is

then written to the database. A final validation check to ensure that the information was saved and the user receives a success message. If any of the validation checks fail, the user is given an error message relevant to the issue and is asked to submit again.

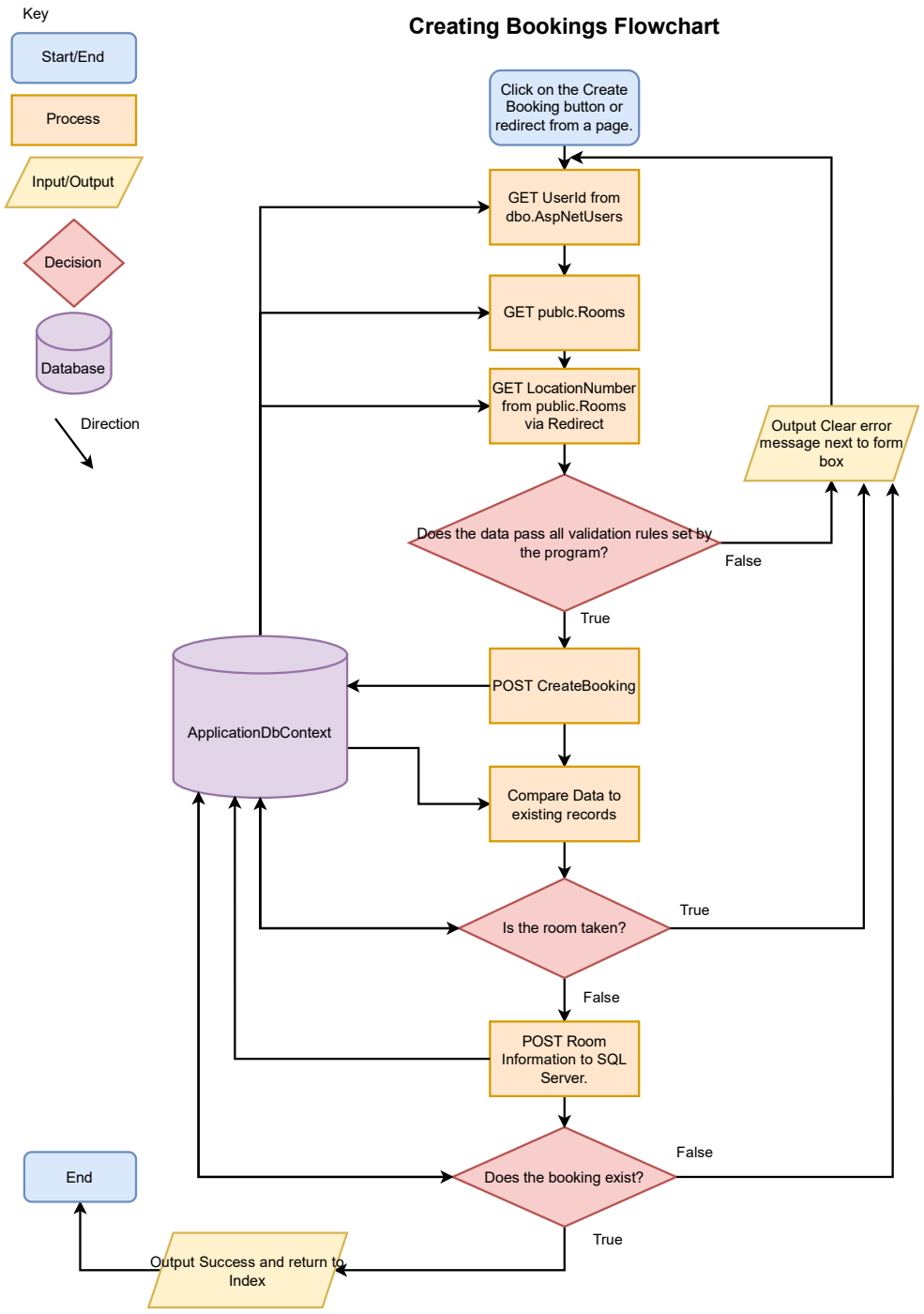
Edit Rooms



The user is prompted to edit any information about the room that is necessary. Then, the solution validates all data input. If the information is validated, the information gets posted to the database

to then look for any existing records, if there is an existing record then the solution updates the information to the SQL database. After uploading the changes, a final validation check is used to ensure the data got written. If the validation check passes, the user is given a success message and is returned to the index. If any of the validation checks fail, the user is given an error message relevant to the issue and is asked to submit again.

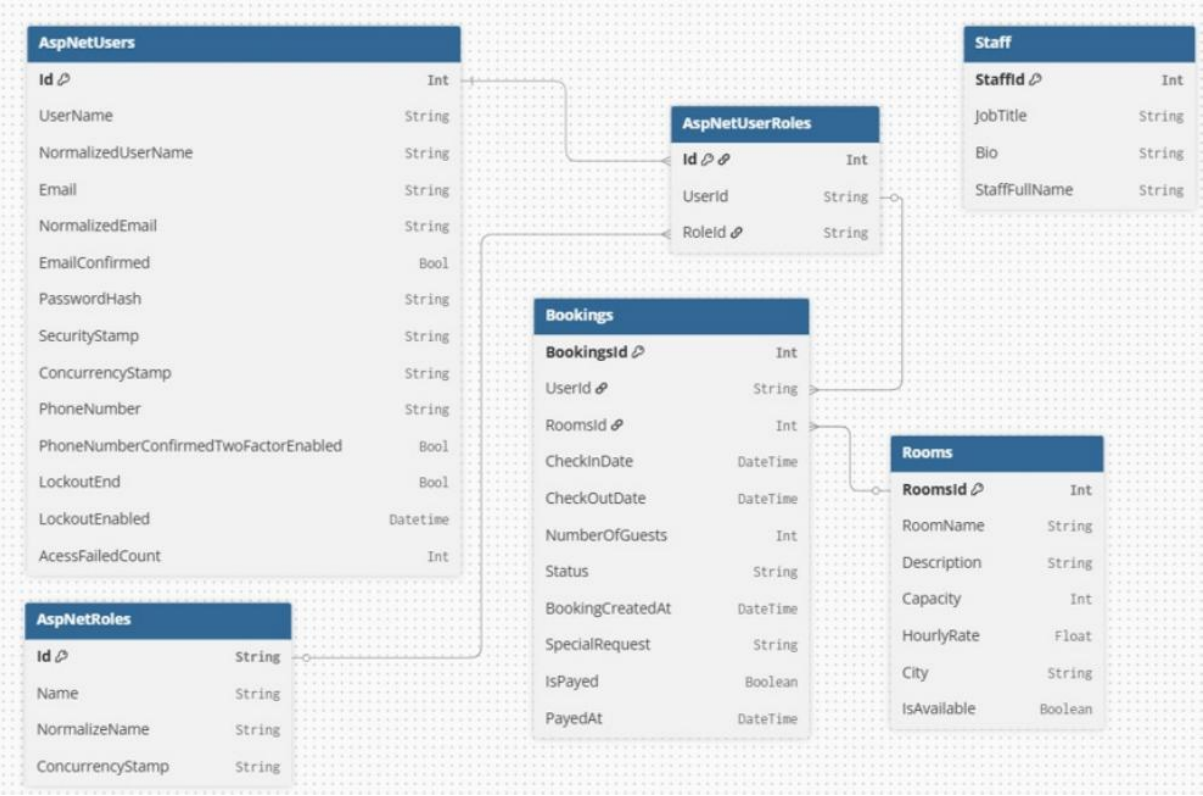
Creating Bookings



The user has clicked the “book room’’ on the front end. The solution GET ‘s the data necessary to create a booking. This includes the UserId and the public.Rooms table. Then, the data is validated to ensure no data corruption has occurred. Once the validation check is successful, the information is then checked against the database. If the room is not taken, the information is saved to the database. Then, if the validation checks shows that the room has been booked, the user is given a success message and is returned to index. If any of the validation checks fail, the user is given an error message relevant to the issue and is asked to try again or to select another room.

Data Requirements Design

ER Diagram



Database Dictionary

AspNetUsers

Column	Data Type	Required?	Default	Validation	Key Types
Id	Int	Yes	N/A	N/A	Primary
UserName	String	Yes	N/A	N/A	N/A
NorimalizedUserName	String	Yes	N/A	N/A	N/A
Email	String	Yes	N/A	Syntax Validation	N/A
NormalizedEmail	String	Yes	N/A	Syntax Validation	N/A
EmailConfirmed	Bool	No	False	N/A	N/A
PasswordHash	String	Yes	N/A	N/A	N/A
SecurityStamp	String	Yes	N/A	N/A	N/A
ConcurrencyStamp	String	Yes	N/A	N/A	N/A

PhoneNumber	String/BIGINT	No	N/A	Syntax Validation, Porting Status	N/A
PhoneNumberConfirmed -TwoFactorEnabled	Bool	No	False	N/A	N/A
LockoutEnd	Bool	No	N/A	N/A	N/A
LockoutEnabled	DateTime	No	N/A	N/A	N/A
AccessFailedCount	Int	No	0	N/A	N/A

AspNetRoles

Column	Data Type	Required?	Default	Validation	Key Types
Id	String	Yes	N/A	N/A	Primary
Name	String	Yes	N/A	N/A	N/A
NormalizedName	String	Yes	N/A	N/A	N/A
ConcurrencyStamp	String	Yes	N/A	N/A	N/A

AspNetUserRoles

Column	Data Type	Required?	Default	Validation	Key Types
Id	Int	Yes	N/A	N/A	Primary, Foreign
UserId	String	Yes	N/A	N/A	
RoleId	String	yes	N/A	N/A	Foreign

Staff

Column	Data Type	Required?	Default	Validation	Key Types
StaffId	Int	Yes	N/A	N/A	Primary
JobTitle	String	No	N/A	N/A	N/A
Bio	String	No	N/A	N/A	N/A
StaffFullName	String	No	N/A	N/A	N/A

Bookings

Column	Data Type	Required?	Default	Validation	Key Types
BookingsId	Int	Yes	N/A	N/A	Primary
UserId	Int	Yes	N/A	N/A	Foreign
RoomsId	Int	Yes	N/A	N/A	Foreign
CheckInDate	DateTime	Yes	N/A	N/A	N/A
CheckOutDate	DateTime	Yes	N/A	N/A	N/A
NumberOfGuests	Int	Yes	N/A	N/A	N/A
Status	String	No	N/A	N/A	N/A
BookingCreatedAt	DateTime	Yes	N/A	N/A	N/A

SpecialRequest	String	No	N/A	N/A	N/A
IsPayed	bool	Yes	False	N/A	N/A
PayedAt	DateTime	No	N/A	N/A	N/A

Rooms

Column	Data Type	Required?	Default	Validation	Key Types
RoomsId	Int	Yes	N/A	N/A	Primary
RoomName	String	Yes	N/A	N/A	N/A
Description	String	Yes	N/A	N/A	N/A
Capacity	Int	Yes	N/A	N/A	N/A
HourlyRate	Float/Double	Yes	N/A	N/A	N/A
City	String	Yes	N/A	N/A	N/A
IsAvailable	Bool	Yes	True	N/A	N/A

Test Strategy

Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies
N/A	**Navigation Bar**	**Blackbox Testing, Functional Testing, Integration Testing**	No Test Data Required. Must have completed the navigation bar and must be functional. Linked pages must exist and must be redirected upon when clicked. Tester will navigate through navigation bar and ensure all pages linked are redirected to and load successfully.
N/A	**Home Page**	**Whitebox testing, Blackbox testing, Unit Testing, Integration Testing**	No Test Data Required. Must have completed the layout for the page and the CSS. Tester will test that all buttons, redirects and filters/links work as expected. All homepage components should

			interact properly with other elements.
N/A	**Register**	**Whitebox Testing, Blackbox Testing, Security Testing, Functionality Testing, Integration Testing**	Test Data Required. Must have the completed layout for the page and the CSS. The register function should be fully functional and developed with implementation in the solution. Database Schema must be applied. The controller must be completed to handle registration and error handling for incorrect input of data. Login Page must be created and linked to the Register Page. There must be working logic for the function. Tester must ensure that when testing, the data inputted should have invalid and valid data, as well as erroneous data. Testers must test vulnerabilities with SQL injection and buffer overflows. **Examples of Test data for the Tester to use:** **Normal Data;** Email: "Normal@NormalData.org"

			Password: “Password123?” Confirmed Password: “Password123?” **Erroneous;** Email: “NotAnNormal&ErroneousData£org”(Should contain a @ symbol instead of &, should use a period instead of £) Password: “69” (Should be longer, Should have Letters + a capital letter, should have a symbol and is at 8 characters or more.) Confirmed Password: “420” (Should match the first inputted password, should have Letters + a capital letter, should have a symbol and is at 8 characters or more.) **Extreme Valid;** Email: “ExtremeValid@NormalData.org” Password: “Pasd123?” Confirmed Password: “Pasd123?” (Hits the minimum requirements)

			Extreme Invalid; Email: "ExtremeValid@NormalData.org" Password: "Pas123?" Confirmed Password: "Pas123?" (Does Not hit the minimum requirements)
N/A	**Login**	**Whitebox Testing, Blackbox Testing, Security Testing, Functionality Testing, Integration Testing**	Test Data Required. Must have the completed layout for the page and the CSS. The register function should be fully functional and developed with implementation in the solution. Database Schema must be applied. The controller must be completed to handle login and error handling for incorrect input of data. Register Page must be created and linked to the Register Page. There must be working logic for the function. Tester must ensure that when testing, the data inputted should have invalid and

			valid data, as well as erroneous data. Testers must test vulnerabilities with SQL injection and buffer overflows. **Examples of Test data for the Tester to use:** **Normal Data;** Email: "Normal@NormalData.org" Password: "Password123?" **Erroneous;** Email: "NotAnNormal&ErroneousData£org"(Should contain a @ symbol instead of &, should use a period instead of £) Password: "69" (Should be longer, Should have Letters + a capital letter, should have a symbol and is at 8 characters or more.) **Extreme Valid;** Email: "ExtremeValid@NormalData.org" Password: "Pasd123?" (Hits the minimum requirements) **Extreme Invalid;**

			Email: "ExtremeValid@NormalData.org" Password: "Pas123?" (Does Not hit the minimum requirements)
N/A	**Log Out**	**Blackbox Testing, Security Testing, Session Management Testing, Functionality Testing, Integration Testing**	No Test Data Required. System must allow users to logout through account settings/navigation bar. System must allow a logout and must clear session cookies when logging out. Tester must ensure that when logged out, not user session cookies are stored and that tester cannot get access to restricted material without permissions.
	Creating a booking for a room.	**Whitebox Testing, Blackbox Testing, Functionality Testing, Integration Testing**	Test Data Required. System must allow logged in users to book an available room. System must provide the user a form to input data. System must ensure that data input is validated. Database Schema must be applied. The controller must be able to manage

			bookings and ensure that the request is successful. There must be working logic in the function. Tester must ensure that when testing, the data inputted should have invalid and valid data, as well as erroneous data. Testers must test vulnerabilities with buffer overflows. Tester should ensure that they are unable to boom a room in the past, and that the booking must be at least a day in advance. Example Test Data: **Valid;** Date: 31/1/2026 Start Time: 09:00 End Time: 10:00 Full Name: John Doe Email Address: JohnnyManBoyDogCatMouseCheese@NormalData.com **Invalid;** Date: 1/1/2001 Start Time: 00:01 End Time: 23:59 Full Name: (*) Email Address: UvU@*.com

			Erroneous; Date: The Thirteenth of December Ninety Ninety Seven Start Time: 5 o'clock in the morning End Time: 7 o'clock at night Full Name: 5/4/2023 Email Address: Gaben Newell
	Viewing Available Rooms	**Blackbox Testing, Functionality Testing, Integration Testing**	Test Data Not Required. The system must allow users that are logged in to view the available rooms. Database Schema must be applied. The controller must be able to manage viewing requests and ensure that the request is successful. There must be working logic in the function.
	Editing Rooms	**Whitebox Testing, Blackbox Testing, Functionality Testing, Integration Testing**	
	Editing Bookings		
	Set Payment Method		

Column	Data Type	Required?	Default	Validation	Key Types