APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 41-60
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Enhancing Room Reservations Efficiency: The Development of the PTTA Booking System

Han Siaw Wee[1], Rosmamalmi Mat Nawi[1]*

[1] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA*
[2] *Perpustakaan Tunku Tun Aminah, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA*

*Corresponding Author: rosmamalmi@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.003

Article Info

Received: 12 June 2024
Accepted: 8 May 2025
Available online: 30 June 2025

Keywords

PTTA, Library, Booking System, Web-based, Structured-approach, Agile model, Booking Process

Abstract

Perpustakaan Tunku Tun Aminah (PTTA) relied on manual techniques like phone reservations and Google Forms, resulting from the lack of an online booking mechanism. The objective of this system is to propose using a structured approach to design and develop a customised booking system for the library. The project aims to use an Agile approach to design and develop a customised room booking system for the library. Key library operations such as booking processing, user accommodations, account administration, and reporting are intended to be streamlined by this system. In general, it will offer an online booking platform for library spaces, greatly enhancing record-keeping, information retrieval, and booking administration responsibilities for both staff and patrons.

1. Introduction

PTTA UTHM provides various facilities for physically disabled people, such as circulation and consultation, studying and group discussion, courses, meetings and seminars, and so on, to everyone who needs them. Among these facilities, the facility for courses, meetings and seminars was managed by only two of the staff in PTTA, and they needed to manually key in all the bookings into the system, which is a lack of manpower.

The main problem was ineffective receiving a lot of phone calls to book the rooms every day by only the two staff members who managed more than 20 rooms simultaneously. Furthermore, the staff needed to check the room availability individually from the log book. Then it was a critical problem that some of the people that wanted to book the room which was not that suitable for them, therefore the staff needed to ask the number of people and the purpose for using the room again and lastly write down the booking details into log book manually. Students, staff, and other organisations were also unable to see whether periods were already reserved by others due to the opaque booking mechanism. All parties experienced difficulty and added complexity in the scheduling process as a result of the frequent disagreements and clashes in booking times caused by this lack of visibility.

The first objective is to propose a PTTA Rooms Booking System using a structured approach. Furthermore, it is to develop a PTTA Rooms Booking System that helps to enhance the PTTA room booking process. The third is to evaluate the performance of the PTTA Rooms Booking System.

In order to address the issues that PTTA employees face, a PTTA Rooms Booking System is made to simplify the booking procedure, lessen scheduling conflicts, improve communication, and guarantee effective resource utilisation by offering a centralised, easily navigable platform for making and managing reservations. The PTTA Rooms Booking System will provide a platform for people to select the number of people they want to use, and they can view all the details for the rooms. Most importantly, people can view the room availability so they can book the rooms without any clashing, and the staff are not required to manually fill in any bookings.

The expected outcome for this word is to create a booking system that's easy to use and helps make booking smoother. It aims to make things work better for both users and administrators. If the development is successful, this system will have features like an easy-to-understand interface, instant updates on availability, and reports showing helpful information.

Overall, Chapter 1 describes the introduction of the developed system, whereas Chapter 2 will elaborate on the literature review of the developed system and the comparison with other booking systems. The methodology of the developed system is discussed in Chapter 3, followed by Chapter 4, which will explain the result and discuss the developed system. Last but not least, the conclusion and recommendation for the system are described in Chapter 5.

2. Related Work

Chapter 2 addresses the study's associated work, including the relevant term, PTTA's current booking procedure, and associated current systems.

2.1 Room Booking System

Typically, the room booking system is a system that enables booking platforms to handle reservations and accept online bookings and appointments [1]. Besides, the room booking system is the system that enables guests to make secure online bookings. It is also a process or method that allows a customer to book a service or a particular place for specific periods. This system may involve coordinating several different parties or entities. Then, if the process of making bookings heavily depends on an exchange program, it is considered a significant part of this system.

2.2 Study of Existing PTTA Rooms Booking Process

PTTA UTHM employs a manual booking system, requiring customers to call by phone or utilise Google Forms to book rooms. This approach can lead to human errors in data reporting and analytics for the library, which is time-consuming and ineffective, and occasionally, staff may provide incorrect room information to users. Furthermore, maintaining user information records is a challenging task with the manual system, and user data files may be discarded when storage capacity is reached.

2.3 Study of Existing System

This section explains three existing related systems to the PTTA Rooms Booking System, which are the Booking.com Hotel Booking Website [2], Monash University Library Bookings [3], Universiti Tunku Abdul Rahman (UTAR) Library Room Booking System [4], and Taylor University Library Room Booking System [5].

Booking.com is one of the world's largest online travel agencies, offering millions of accommodation options to travellers worldwide. It was founded in 1996 and has since grown to become one of the leading players in the online travel industry [6]. Travellers can utilise the search feature to specify their preferences and make direct bookings. Typically, guests can cancel without incurring charges up to shortly before their planned arrival. In addition to hotel accommodations, the platform also facilitates reservations for vacation rental villas.

Monash University's Library bookings system is a booking system that enables its students to book and use, ensuring that they have a dedicated space for studying, group projects, research, or meetings. This reduces the frustration of searching for a room on a crowded campus. On its system, it provides a precise and concise interface for the students to select their preferred rooms. All the room details and capacity are listed completely to allow students to choose and confirm the rooms. The timetable for the bookings can also be viewed, and the students can view the slot status in the timeline.

UTAR Library Online Booking System provides two main category rooms for booking: Library Discussion Room and Refinitiv Terminals. Users can reserve the discussion room using the online booking system.

However, Refinitiv Terminals only apply to physical access at the UTAR Library. Its students can select the date, discussion room, and appointment time to book the room.

Taylor University Library's room booking system is similar to Monash University Library's. It allows the students to choose the room as listed. Students can easily access information about room capacity, available equipment, and both available and unavailable time slots at a glance. Students may select the location, category of the room, and the capacity they prefer during the booking process to reduce the searching time and make the process effective.

The developed system and the current systems are compared in Table 1.

Table 1 *Comparison between the existing system with the developed system*

Features/System	Booking.com	Monash University Library Room Booking System	UTAR Library Room Booking System	Taylor University Library Room Booking System	PTTA Rooms Booking System
Registration /Log-in	√	√	√	√	√
Room details	√	√	**X**	√	√
Room availability	√	√	√	√	√
Room bookings	√	√	√	√	√
Room Management	√	√	√	√	√
Notifications	√	√	√	√	√
Rooms bookings status	√	√	√	√	√
Report analytic	√	√	√	√	√

Legend: √ = Yes **X** = No

3. Methodology/Framework

The Agile methodology is employed as the software development life cycle (SDLC) for constructing this system. This approach is chosen due to its flexibility in accommodating changing requirements, incorporating user feedback, and enhancing user satisfaction by presenting system prototypes to the target users [7]. Therefore, involving the target users throughout the project's development cycles can help reduce the chances of the project not working out as expected. Fig. 1 displays the phases of the Agile model, including its six key phases: Requirements, Design, Development, Testing, Deployment, and Review. The activities and output for each stage are summarised in Table 2.

Fig. 1 *Agile Development Model [8]*

Table 2 *Software Development Activities and Their Task*

Phase	Task	Output
Requirement phase	• Gather requirement • Determine the project schedule, activities and output	• Project proposal • Develop Gantt chart
Design phase	• Design system flow, context diagram, data flow diagram • Design wireframe for user interfaces • Design entity relationship diagram • Analyse user requirement to get functional requirements • Carry out feasibility study on functional requirements	• Context diagram • Data flow diagram • Wireframe for user interfaces • Entity relationship diagram • Functional requirements
Development phase	• Develop program code using HTML, CSS, JavaScript • Launch the system prototype in local environment	• System prototype
Testing phase	• Alpha test against system prototype • Fix error if any error is found during the alpha test	• System prototype
Deployment phase	• Deploy the system prototype in live server	• System prototype is available in live server and can be accessed by target user
Review phase	• Demonstrate the system prototype features to target user • Target user explores and reviews the system prototype • Target user gives feedback and satisfactory level on system prototype	• Feedback and satisfactory level on system prototype

3.1 Requirement Phase

The goal of the requirement phase is to collect the functional requirements for the developed system. During this phase, the system's functional requirements were gathered through an interview with the staff of PTTA. The system developer elicited numerous functional requirements during this interview, which will be thoroughly examined in the subsequent phase. Moreover, the interview questions are included in Appendix A.

3.1.1 Functional Requirement & Non-functional Requirement

Functional requirements represent the system needs requested by the system's intended users [9]. Whereas an NFR (non-functional requirement) is any quality or trait that focuses on the features or limitations of the system rather than how it relates to its capabilities directly [10]. Table 3 and Table 4 detail this system's specific functional and non-functional requirements.

Table 3 *Functional Requirements of System*

No.	Functional Requirements	Descriptions
1	Room Details Module	It shows all the information about the rooms such as the maximum number of people that can be accommodated, the size and shape of the table, the devices that can be used in the room, and so on to ensure users can select the most suitable room.
2	Room Bookings Module	The system displays real-time room availability, making

Table 3 *Functional Requirements of System (cont.)*

		it easy for users to see when rooms are free or occupied. Additionally, customers have a variety of booking choices at their disposal, including the ability to reserve a room for a particular day and time or set up recurring reservations for a number of gatherings or events. Users have the option to change or remove their reservations.
3	Room Availability Module	The administrator may update the room availability with the available date and time slots for each room to allow users to make the room reservation.
4	Email Notifications Module	It provides real-time notifications to administrators about new bookings and real-time notifications to users about whether the booking status is approved or rejected.
5	Room Bookings Status Module	The administrator may be able to take immediate actions directly from the notifications, such as approving a booking request or responding to a user query.
6	Account Management Module	The users may always update their latest personal information to ensure the booking process is smooth, and the administrator can always reach them.
7	Report Analytic Module	It helps to view the total number of room bookings monthly or yearly so that the admins can quickly know whether they have achieved the target or goal that they set before. Additionally, it may export reports for additional analysis and offer insights into user activity, such as the most popular rooms, peak booking periods, and user engagement patterns.

Table 5 *Non-functional Requirements of System*

No.	Non-functional Requirements	Descriptions
1	Security	Only users who have logged in should be permitted to access both the confidential data and their personal information within the system.
2	Reliability	The system is expected to maintain a minimum uptime of 99% after it is launched.
3	Performance	The system should promptly respond to user requests and ensure minimal loading times.
4	Scalability	The system must be able to manage and store growing amounts of data and increase usage after it is launched.
5	Usability	The system's user interfaces should be user-friendly, easily navigable, and capable of displaying the system status to users.

3.2 Design Phase

This phase aims to create the system's design by examining the user requirements collected earlier. In the initial agile iteration, user requirements are analysed to establish the context and data flow diagrams, outlining how the system will function. An entity relationship diagram is also created, and the database server is set up for the upcoming phase. A feasibility study checks if the things the system needs can work with the current technology. Also, the system wireframes are created. These designs help the system developer when developing the web interface.

3.2.1 DFD CD

The context diagram of the suggested system, sometimes called DFD Level 0, is displayed in Fig. 2 and demonstrates the data flow between entities and processes in terms of input and output. The suggested system has three user categories: administrators, the UTHM community, and other organisations. The DFD Level 1 (see Appendix A) breaks down the context diagram into multiple intricate processes. The DFD Level 2 for each process can be found at Appendix B.

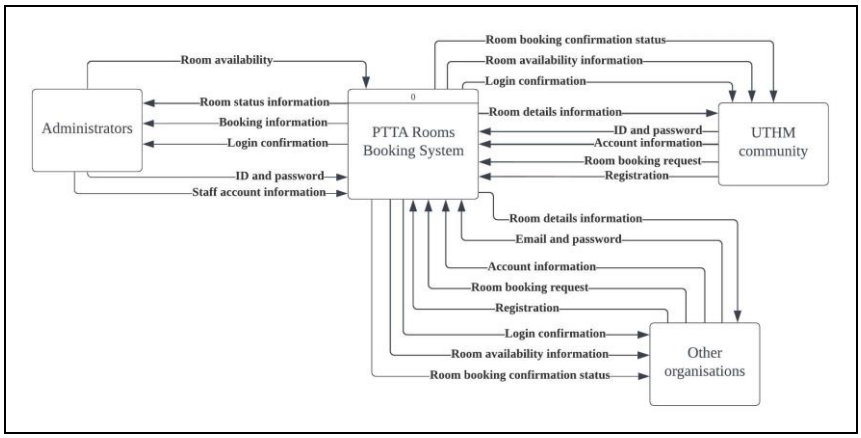


Fig.2 *DFD CD of Developed System*

Fig. 1 shows the data flow diagram context diagram (DFD CD). The context diagram has three entities: the administrators, the UTHM community, and the other organisations (user outside UTHM). For the administrator entity, the room availability and staff account information are transferred to the system while the system transferred the room status information, booking information, login confirmation, and ID and password to the administrator. For UTHM community, ID and password, account information, room booking request and registration are transferred to the system whereas the login confirmation, room availability information, and room booking confirmation status are transmitted to the UTHM community entity. The other organisations entity has the same data flow as UTHM community.

3.2.2 Flowchart

A flowchart is a helpful visual aid that provides a clear, graphical representation of the operation of a complex system or activity. Fig. 3 illustrates the flowchart of the UTHM community and other organisations that book the room. Other flowcharts are included in Appendix C.

3.2.2.1 Flowchart – UTHM Community and Other Organisations Book Room Process

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 41-60

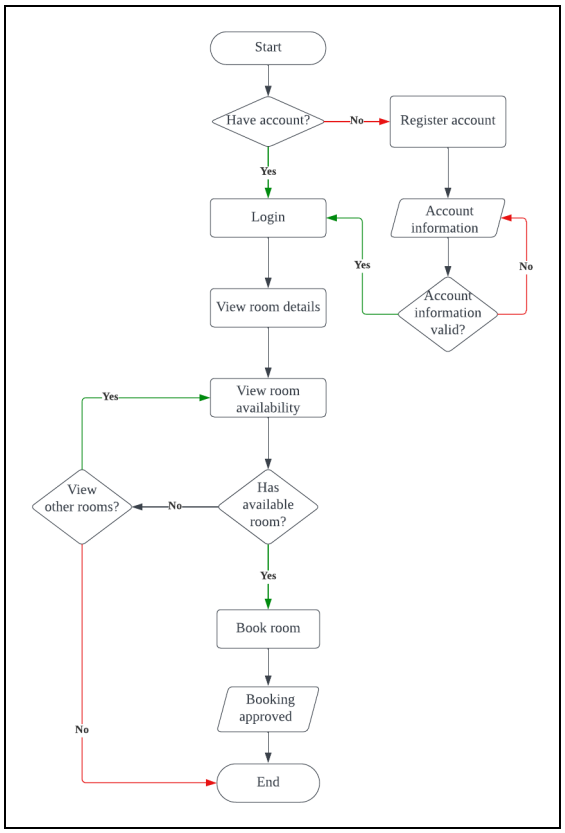


Fig. 3 *Flowchart for Users to View Room Availability and Book Room*

Fig. 3 depicts the flowchart detailing the process of users viewing room availability and booking a room. This flowchart encompasses a total of five distinct processes, four decision points, and two data input and output illustrated within the flowchart.

3.3 Development Phase

The implementation phase encompasses creating the project plan and executing its associated tasks. Typically initiated after project approval, this step involves delivering a detailed report on the PTTA Rooms Booking System. Additionally, it's important to handle risk and issue logs, as well as change logs that detail any modifications to the project plan and how they affect the project's advancement. Tools and software utilised during the implementation phase include Lucid Chart and Miro. All deliverables are conducted in alignment with the agreements established between the developer and stakeholders during the concept phase.

3.4 Testing Phase

During this phase, the developer runs an alpha test on the system to ensure that the tasks a user might do flow smoothly without mistakes. The developer does the alpha test, which is the initial testing stage, to ensure the system meets its functional requirements and operates without errors. The developer imitates potential user actions to uncover any hidden issues in the system. The developer addresses errors found during testing before moving on to the next phase.

3.5 Deployment Phase

During this phase, the system developer migrates the system prototype from the local environment to the live server, making it accessible to the intended users. Subsequently, the system developer showcases the system prototype to the target users. Following the demonstration, the users are granted access to explore the system independently to acquaint themselves with its features. A designated period, typically a few days, is allocated for the users to explore the system before transitioning to the next phase.

3.6 Review Phase

During this phase, feedback and satisfaction levels regarding the system prototype are gathered from the target users. The collected feedback is carefully analysed and reviewed to enhance and refine the system prototype.

Subsequently, a decision is made to initiate a new iteration, starting from the design phase and continuing to the review phase, to enhance the system prototype further. Alternatively, the SDLC may conclude if three iterations have been completed.

4. Result and Discussion

This section explains the system implementation process with their algorithms and the system testing using the system functional testing approach.

4.1 System Implementation

The section outlines the processes used to build, integrate, and deploy the system, ensuring it functions according to the specified requirements and design.

4.1.1 Update Room Availability Module

Fig. 4 depicts the update room availability page. The administrator will initially select the date and then update each room's availability accordingly.

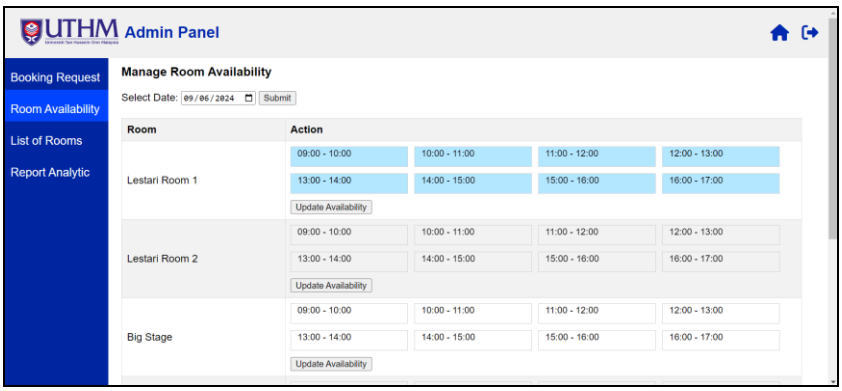


Fig. 4 *Update Room Availability Page*

Fig. 5 shows the algorithm for updating the room availability. If the administrator clicks the “Submit” button without selecting the date, an alert message will pop out to remind the administrator to select the date. Otherwise, it will show the room and its time slots. Next, the time slots will be updated successfully and shown to the user system if the administrator selects at least one time slot to update; otherwise, an alert message will pop out to remind them to select at least one time slot to update.

Algorithms 1

Start

1. Check for the booking status update.

```
if ($stmt->affected_rows > 0) {
            // Fetch user's email, name, and booking details
            $stmt = $con->prepare("SELECT email, name, room, date_time,
purpose, participants, organisation, category, userID, role, programLink
FROM bookings WHERE id = ?");
            if ($stmt === false) {
                $errorMessage = "Prepare failed: (" . $con->errno . ") " .
$con->error;
                header("Location:
admin-dashboard.php?error=prepare_failed&message=" .
urlencode($errorMessage));
                exit();
            }
            $stmt->bind_param("i", $bookingId);
            $stmt->execute();
            $stmt->bind_result($email, $name, $room, $date_time, $purpose,
$participants, $organisation, $category, $userID, $role, $programLink);
            $stmt->fetch();
            $stmt->close();
```

2. Send the email.

```
// Send email notification using PHPMailer
            $mail = new PHPMailer(true);
            try {
                // Server settings
                $mail->isSMTP();
                $mail->Host = 'smtp.gmail.com'; // Replace with your SMTP
server
                $mail->SMTPAuth = true;
                $mail->Username = 'datagenerall3@gmail.com'; // Replace
with your SMTP username
                $mail->Password = 'iyzr qjfs wetw jjzk'; // Replace with
your SMTP password
                $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
                $mail->Port = 587;

                // Recipients
                $mail->setFrom('no-reply@uthmptta.com', 'PTTA Rooms
Booking System'); // Replace with your from address and name
                $mail->addAddress($email, $name); // Use $email and $name
from the database
```

3. Receive the email.

End

Fig. 5 *Algorithm of Update Room Availability*

4.1.2 Room Booking Module

Fig. 6 shows the booking time slots page, whereas Fig.7 and Fig.8 display the booking confirmation page. On the booking time slots page, there is a date selection to allow users to select the desired date for booking, and then the available time slots and booked time slots will be displayed in a table. After the user selects their preferred time slots and click the "Book" button, they will be redirected to the booking confirmation page to fill out the booking information for the further process.

Fig. 6 *Booking Time Slots Page*

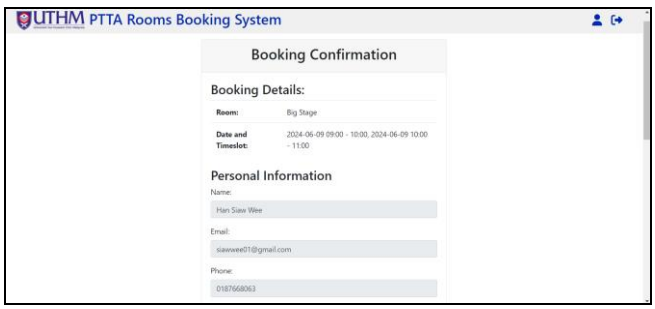


Fig. 7 *Booking Confirmation Page*

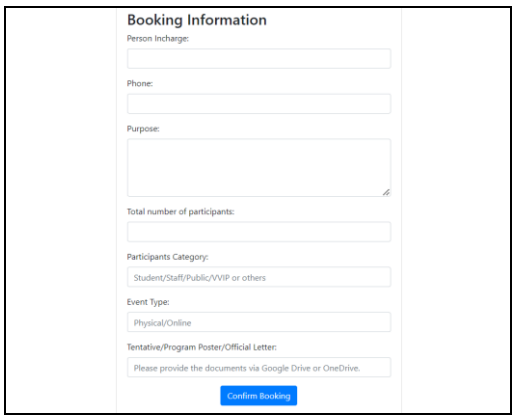


Fig. 8 *Booking Confirmation Page (cont.)*

The algorithm for making a booking request is depicted in Fig. 9. Firstly, it will check the date to view time slots. If the users do not select the date, then they cannot view the timeslots. Next, the algorithm will check to view the available time slots and booked time slots; the selected time slots are checked to avoid empty data and also the expired time slots. After all the date and selected time slot(s) are valid, the algorithm will further the process to the booking confirmation to allow users to enter the booking details.

Algorithms 2

Start

1. Check the date to view the time slots.

```
<form method="get" action="">
            <label for="selected_date">Select Date:</label>
            <input type="date" id="selected_date" name="selected_date"
<?php if (isset($_GET['selected_date']))
            echo 'value="' . htmlspecialchars($_GET['selected_date'])
. '"'; ?> min="<?php echo date('Y-m-d'); ?>">
            <input type="hidden" name="room_id"
                value="<?php echo isset($_GET['room_id']) ?
htmlspecialchars($_GET['room_id']) : ''; ?>">
            <button type="submit">Submit</button>
        </form>
```

2. Check to view the available and booked time slots.

```
if (isset($_GET['selected_date'])) {
      $selectedDate = $_GET['selected_date'];
      // Assuming $_GET['room_id'] contains the room ID to fetch
      $room_id_to_fetch = $_GET['room_id']; // Get the room ID from the
      URL parameter

      $sql = "SELECT room_id, room_name FROM room_details WHERE room_id =
      $room_id_to_fetch"; // Include 'room_id' in the SELECT query with a
      WHERE clause
      $result = $con->query($sql);
```

3. Check to select the available time slots.

```
var timeSlot = element.innerText.trim();
var currentDate = new Date();
var currentTime = currentDate.toTimeString().slice(0, 5);
var [startTime, endTime] = timeSlot.split(' - ');
var selectedDate = new
Date(document.getElementById('selected_date').value);

if (currentDate.toDateString() === selectedDate.toDateString() &&
isTimeExpired(currentTime, endTime)) {
    alert('This time slot has expired.');
    return;
}
function isTimeExpired(currentTime, endTime) {
    var [currentHour, currentMinute] = currentTime.split(':').map(Number);
    var [endHour, endMinute] = endTime.split(':').map(Number);
    return currentHour > endHour || (currentHour === endHour &&
currentMinute > endMinute);
}

if (element.classList.contains("selected")) {
    element.classList.remove("selected");
    selectedTimeSlotsByRoom[roomId] =
selectedTimeSlotsByRoom[roomId].filter(slot => slot !== timeSlot);
    if (selectedTimeSlotsByRoom[roomId].length === 0) {
        delete selectedTimeSlotsByRoom[roomId];
```

(a)

```
    } else {
        var isConsecutive = selectedTimeSlotsByRoom[roomId] &&
selectedTimeSlotsByRoom[roomId].length > 0
            ? startTime ===
selectedTimeSlotsByRoom[roomId].slice(-1)[0].split(' - ')[1]
            : true;

        if (isConsecutive) {
            element.classList.add("selected");
            if (!selectedTimeSlotsByRoom[roomId])
selectedTimeSlotsByRoom[roomId] = [];
            selectedTimeSlotsByRoom[roomId].push(timeSlot);
        } else {
            alert('Please select consecutive time slots.');
        }
    }
```

4. Book the time slots.

```
var selectedDate = document.getElementById("selected_date").value;

        // Collect all selected time slots
        var selectedTimeSlots = [];
        for (var roomId in selectedTimeSlotsByRoom) {
            selectedTimeSlotsByRoom[roomId].forEach(function (timeSlot) {
                selectedTimeSlots.push(selectedDate + ' ' + timeSlot);
            });
        }

        if (selectedTimeSlots.length > 0) {
            // Set the selected information to the hidden input field
            document.getElementById("selectedInfo").value =
selectedTimeSlots.join(', ');

            // Submit the form
            document.getElementById("bookingForm").submit();
        }
```

5. Fill out the booking details.

```
Enter the person incharge, name, organisation, purpose, number of
participants, event type, program link, category of participants, and
staff id/matric no.
```

6. Make a booking request.

End

(b)

Fig. 9 *Algorithm of Room Booking Request*

4.1.3 Room Management Modules

Fig. 10 depicts the manage room page. The administrator can add and update the room details such as room name, capacity, room images, and room description to the system. Moreover, they are also allowed to delete the room if needed.

Fig. 10 *Manage Room Page*

Fig. 11 depicts the algorithm for managing the room. Firstly, it has to check the rooms, then enter the room details into fields to add and update the room. If the room images are not inserted, then the add room and update room processes cannot be done. To delete the room, the algorithm will check the room id; if the room id exists, then the deletion can be done.

Algorithms 3

Start

1. Check for the rooms.
   If need to add a new room, click the "Add" button.
2. Validation checks for the room added.

```
foreach ($_FILES["room_img"]["tmp_name"] as $key => $tmp_name) {
                $target_file = $target_dir .
basename($_FILES["room_img"]["name"][$key]);
                if (move_uploaded_file($tmp_name, $target_file)) {
                    $uploaded_files[] = $target_file;
                    echo "The file " .
htmlspecialchars(basename($_FILES["room_img"]["name"][$key])) . " has been
uploaded.<br>";
                } else {
                    echo "Sorry, there was an error uploading your
file.<br>";
                }
            }
```

3. Add the new room.

End

(a)

Algorithms 4

Start

1. Check the rooms.
   If the rooms have to update new details, then click the "Update" button.
2. Validation checks for the room updated.

```
foreach ($_FILES["room_img"]["name"] as $key => $name) {
        $target_file = $target_dir .
basename($_FILES["room_img"]["name"][$key]);
        if (move_uploaded_file($_FILES["room_img"]["tmp_name"][$key],
$target_file)) {
            $uploaded_files[] = $target_file;
        } else {
            echo "Sorry, there was an error uploading your file.";
        }
    }
```

3. Update the room.

End

(b)

Algorithms 5

Start

1. Check the rooms.
   If the room has to be deleted, then click the "Delete" button.
2. Check the deleted rooms.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['room_id'])) {
    // Include your database connection file or initialize the database
connection here
    require('db.php'); // Replace 'db.php' with your actual database
connection file
    // Get the room_id to be deleted
    $room_id = $_POST['room_id'];

    // Prepare and execute the DELETE query
    $query = "DELETE FROM room_details WHERE room_id = ?";
    $stmt = mysqli_prepare($con, $query);
    if ($stmt) {
        mysqli_stmt_bind_param($stmt, "i", $room_id);
        $result = mysqli_stmt_execute($stmt);
        if ($result) {
            // Room successfully deleted
            header('Location: list-room.php'); // Redirect to a page after
deletion
            exit();
```

3. Delete the room.

End

(c)

Fig. 11 *Algorithms of Room Management*

4.1.4 Update Account Details Module

Fig. 12 shows the update account details page. The user can update their names, phones, and organisations in the system if there are any new updates according to their personal information. Furthermore, the system also has a change password function to allow users to update their passwords frequently to enhance security (See Appendix E).

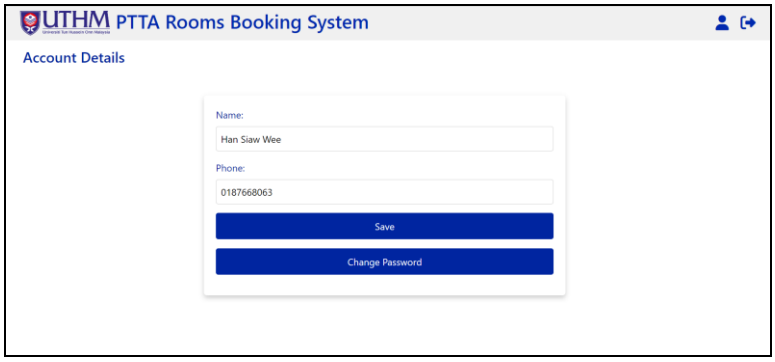


Fig. 12 *Update Account Details Page*

The algorithm for updating user account details is shown in Fig. 13. If users want to update their name, phone, or organisation details, they should enter the new information and click "Save". To choose which database table and fields to update, a switch statement looks up the userType. First name, last name, and phone number are updated for staff and student users. For other users, their organisation is updated in addition to the abovementioned fields. If they want to change their password, they must click the “Change Password” button. It will check all the required fields, such as current password, new password, and confirm password. Then, it will check whether the new password is the same as the confirmed one and whether the current one is the same as the one in the database. If yes, the password changed successfully; else it will show the error message.

Algorithms 6

Start

1. Check to update account details .
   If the user has to update the latest name, phone, or organisation, enter the latest info then click “Save”.

2. Update account details.

```
switch ($userType) {
    case 'student':
        $updateQuery = "UPDATE student_acc SET student_acc_firstname = ?, student_acc_lastname = ?, student_acc_contactNo = ? WHERE student_acc_email = ?";
        $stmt = $con->prepare($updateQuery);
        $stmt->bind_param("ssss", $firstName, $lastName, $phone, $user);
        break;
    case 'staff':
        $updateQuery = "UPDATE staff_acc SET staff_acc_firstname = ?, staff_acc_lastname = ?, staff_acc_contactNo = ? WHERE staff_acc_email = ?";
        $stmt = $con->prepare($updateQuery);
        $stmt->bind_param("ssss", $firstName, $lastName, $phone, $user);
        break;
    case 'other':
        $updateQuery = "UPDATE other_acc SET other_acc_firstname = ?, other_acc_lastname = ?, other_acc_contactNo = ?, other_acc_organisation = ? WHERE other_acc_email = ?";
        $stmt = $con->prepare($updateQuery);
        $stmt->bind_param("sssss", $firstName, $lastName, $phone, $organisation, $user);
        break;
    default:
        // Handle incorrect user type selection
        echo "<script>alert('Invalid user type');</script>";
        exit;
}
```

3. Check to change password.
   If the user has to change password, click the “Change Password” button.

4. Change password.

```
if (isset($_POST['change_password'])) {
    $id = mysqli_real_escape_string($con, $_POST['id']);
    $role = mysqli_real_escape_string($con, $_POST['role']);
    $oldPassword = mysqli_real_escape_string($con, $_POST['currentPassword']);
    $newPassword = mysqli_real_escape_string($con, $_POST['newPassword']);
    $confirmPassword = mysqli_real_escape_string($con, $_POST['confirmPassword']);

    if (!$oldPassword || !$newPassword || !$confirmPassword) {
        echo json_encode(['status' => 422, 'message' => 'All fields are mandatory!']);
        return false;
    }

    if ($newPassword !== $confirmPassword) {
        echo json_encode(['status' => 422, 'message' => 'New Password and Confirm Password do not match!']);
        return false;
    }

    $roleColumn = "{$role}_acc_password";
    $roleTable = "{$role}_acc";
    $roleIdColumn = "{$role}_acc_id";

    $verifyQuery = "SELECT $roleColumn FROM $roleTable WHERE $roleIdColumn = ?";
    $updateQuery = "UPDATE $roleTable SET $roleColumn = ? WHERE $roleIdColumn = ?";

    $stmt = mysqli_prepare($con, $verifyQuery);
    mysqli_stmt_bind_param($stmt, 'i', $id);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);

    if ($result) {
        $row = mysqli_fetch_assoc($result);
        if (!$row || !password_verify($oldPassword, $row[$roleColumn])) {
            echo json_encode(['status' => 422, 'message' => 'Incorrect Old Password']);
            return false;
        }
    }

    $hashedPassword = password_hash($newPassword, PASSWORD_DEFAULT);

    $stmt = mysqli_prepare($con, $updateQuery);
    mysqli_stmt_bind_param($stmt, 'si', $hashedPassword, $id);
    $update_run = mysqli_stmt_execute($stmt);

    if ($update_run && mysqli_stmt_affected_rows($stmt) > 0) {
        echo json_encode(['status' => 200, 'message' => 'Password Changed Successfully']);
    } else {
        echo json_encode(['status' => 500, 'message' => 'Password Change Failed']);
    }
}
```

End

(a) (b)

Fig. 13 *Algorithm of Update Account Details*

4.1.5 Room Bookings Status Module

Fig. 14 shows the booking request page. The administrator can take the action whether to approve or reject the booking request in the system.

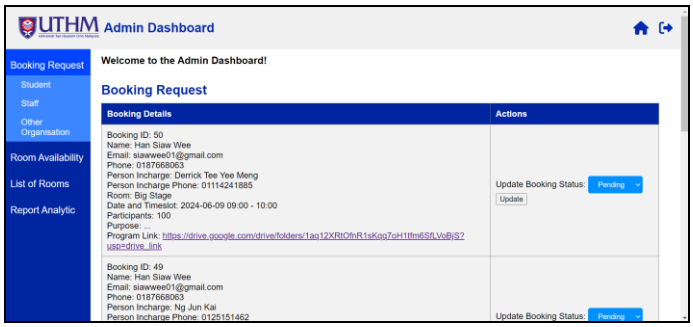


Fig. 14 *Booking Request Page*

The algorithm for the administrator to update the booking status is shown in Fig. 15. Firstly, the algorithm will verify the booking ID and the status of the booking request. Then, the booking request is updated successfully if the booking id and the status can be extracted from the POST data. Once the booking request has been updated, the email for notify the user about the booking status will be sent.

Algorithms 7

Start

1. Check for the booking request.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Check if the booking ID and new status are set in the POST data
    if (isset($_POST['booking_id']) && isset($_POST['new_status'])) {
        // Extract booking ID and new status from the POST data
        $bookingId = $_POST['booking_id'];
        $newStatus = $_POST['new_status'];
```

2. Update the booking request.

```
$stmt = $con->prepare("UPDATE bookings SET status = ? WHERE id = ?");
        if ($stmt === false) {
            $errorMessage = "Prepare failed: (" . $con->errno . ") " .
$con->error;
            header("Location:
admin-dashboard.php?error=prepare_failed&message=" .
urlencode($errorMessage));
            exit();
        }
$stmt->bind_param("si", $newStatus, $bookingId);
$stmt->execute();
```

3. Send the email to the user.

End

Fig. 15 *Algorithm of Booking Request*

4.2 Testing

The purpose of testing is to ascertain whether the system's effectiveness satisfies the user-specified requirements for the system. This testing aims to ensure that all of the system's components operate accurately and without error. The room booking, notification, room booking status, and updating room availability modules are all included in this test case. Table 6 displays the test case of the update room availability module. There are a total of two pass test cases for this module. This test case is to ensure the administrator updates the room availability without missing essential data.

Table 6 *Test Case of Update Room Availability Module*

No.	Test Case	Expected Output	Actual Output	Result
1	Select nothing and click the "Submit" button.	"Please select a date before submitting." will be displayed as an alert message.	As expected output	Pass
2	Select nothing for each room and click the "Update" button.	"Please select at least one timeslot before updating." will be displayed as an alert message.	As expected output	Pass
3	Select the date required to update	The booking slots for updates will be	As expected	Pass

Table 7 *Test Case of Room Booking Module (cont.)*

	and click the "Submit" button.	displayed.	output		
4	Select the timeslot required to update for each room and click the "Update" button.	"Selected times for 'Room' saved successfully" will be displayed as an alert message.	As output	expected	Pass

Table 7 illustrates the total five pass test cases for the room booking module. This test case ensures the user selects a valid date and available time slots to complete the booking process.

Table 7 *Test Case of Room Booking Module*

No.	Test Case	Expected Output	Actual Output		Result
1	Select the date required to have a book and click the "Submit" button to view the room availability.	The available booking slots and booked slots will be displayed.	As output	expected	Pass
2	Select the timeslot required to book and click the "Book" button.	Redirecting user to booking confirmation page.	As output	expected	Pass
3	Select the expired time slot.	"This time slot has expired." will be displayed as an alert message.	As output	expected	Pass
4	Select non-consecutive time slots.	"Please select consecutive time slots." will be displayed as an alert message.	As output	expected	Pass
5	Let the input fields blank.	"Please fill out this field" will be displayed as an alert message.	As output	expected	Pass

Table 8 depicts the total number of five pass test cases for add, update and delete room modules. Its purpose is to prevent form submissions with missing essential data, ensuring that the fetching available rooms and time slots are based on a valid date.

Table 8 *Test Case of Add, Update and Delete Room Modules*

No.	Test Case	Expected Output	Actual Output		Result
1	Let the input fields blank for the add room page.	"Please fill out this field" will be displayed as an alert message.	As output	expected	Pass
2	Fill out all the input fields for the add room page.	"New room added successfully" will be displayed as an alert message and redirected to the room details page.	As output	expected	Pass
3	Let the input fields blank for the update room page.	"Please fill out this field" will be displayed as an alert message.	As output	expected	Pass
4	Fill out all the input fields for the update room page.	"Room details updated successfully" will be displayed as an alert message and redirected to the room details page.	As output	expected	Pass
5	Click the "Delete" button.	"Are you sure you want to delete this room?" will be displayed as an alert message.	As output	expected	Pass

Table 9 shows the test cases of the update account details module with two pass results. It ensures that users provide all necessary information in the correct format before submitting the data to the system.

Table 9 *Test Case of Update Account Details Module*

No.	Test Case	Expected Output	Actual Output	Result
1	Let the input fields blank.	"Please fill out this field" will be displayed.	As expected output	Pass
2	Fill up all the input fields.	"Data updated successfully" will be displayed as an alert message.	As expected output	Pass

Table 10 displays the room booking status module with two total numbers of pass test cases. It prevents the submission of invalid or incomplete actions that could cause errors in the database or system.

Table 10 *Test Case of Room Booking Status Module*

No.	Test Case	Expected Output	Actual Output	Result
1	Click the "Update" button without selecting any actions.	An error message will be displayed in the URL.	As expected output	Pass
2	Click the "Update" button with selecting the actions.	An email will be sent to the user and redirected to the admin dashboard page.	As expected output	Pass

5. Conclusion and Recommendation

In summary, PTTA's operational issues might be resolved. Other libraries looking to update their booking procedures could benefit from adopting an intuitive and well-thought-out online booking system for library spaces. Maintaining this study through real-world applications and iterative enhancements will help create a workable system that will benefit patrons and library employees equally. A thorough event calendar that shows all scheduled and open timeslots will be included in the future to improve user experience by facilitating the visualisation and selection of users' favourite times. The system should also enable users to choose and reserve numerous timeslots at the same time. This would minimise the requirement for multiple individual requests and allow the admin panel to process group booking requests more quickly. Moreover, by putting in place an email notification system that manages several timeslots in a single booking, administrators and users will both receive consolidated emails that summarise every time slot that has been reserved, improving clarity and communication.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Han Siaw Wee, Rosmamalmi Mat Nawi, Irwan Jubri, Suhaimi Mohd Sukor;* ***data collection:*** *Han Siaw Wee;* ***analysis and interpretation of results:*** *Han Siaw Wee, Rosmamalmi Mat Nawi;* ***draft manuscript preparation:*** *Han Siaw Wee, Rosmamalmi Mat Nawi. All authors reviewed the results and approved the final version of the manuscript.*

An author name can appear multiple times, and each author name must appear at least once. For single authors, use the following wording:

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] Derek. (2023, February 28). *Reservation system | Outsourcing Glossary | Outsource Accelerator*. https://www.outsourceaccelerator.com/glossary/reservation-system/

[2] Alderighi et al. (2022). Consumer perception of price fairness and dynamic pricing: Evidence from Booking.com. *Journal of Business Research*, *145*, 769–783. https://doi.org/10.1016/j.jbusres.2022.03.017

[3] Space Availability - Caulfield Library - Library Bookings - Monash University. [Accessed: 25-Dec-2023].

[4] Utarlibdr. [Accessed: 25-Dec-2023].

[5] Space Availability - Taylor Library - Calendar - Western Libraries. [Accessed: 25-Dec-2023].

[6] Booking.com: The largest selection of hotels, homes, and vacation rentals. (n.d.). Booking.com. https://www.booking.com/content/about.html

[7] Singh et al. (2019). A Framework For Transitioning Of Traditional Software Development Method To Distributed Agile Software Development. IEEE. https://doi.org/10.1109/icict46931.2019.8977654

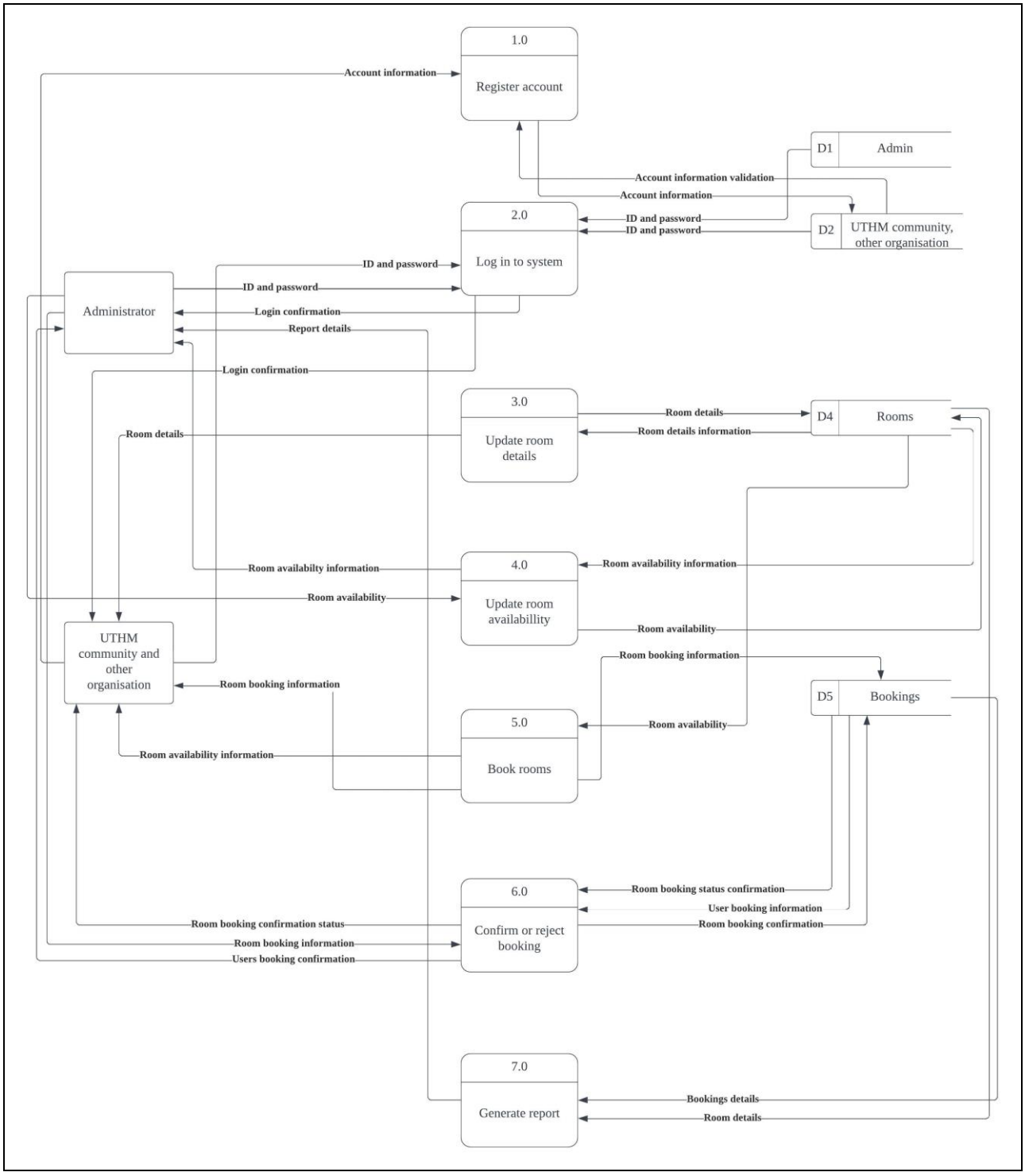
[8] Jayathilaka, C. (2021, December 15). Agile Methodology - Chathmini Jayathilaka - Medium. Medium. https://medium.com/@chathmini96/agile-methodology-30ec4cdf3fc

[9] Martin. (2023, October 28). What is a Functional Requirement in Software Engineering? Guru99. https://www.guru99.com/functional-requirement-specification-example.html

[10] Horkoff. (2019). Non-Functional Requirements for Machine Learning: Challenges and New Directions, ResearchGate.? https://doi.org/10.1109/re.2019.00050

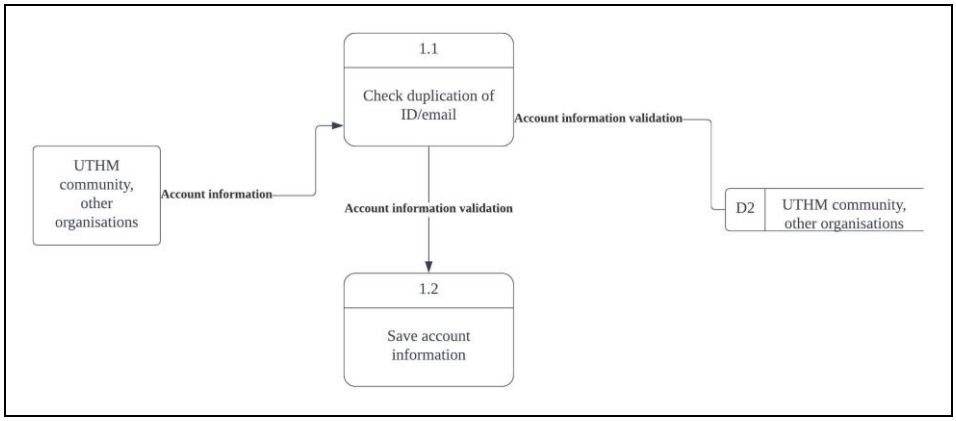
Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 41-60

Appendix A: DFD Level 1

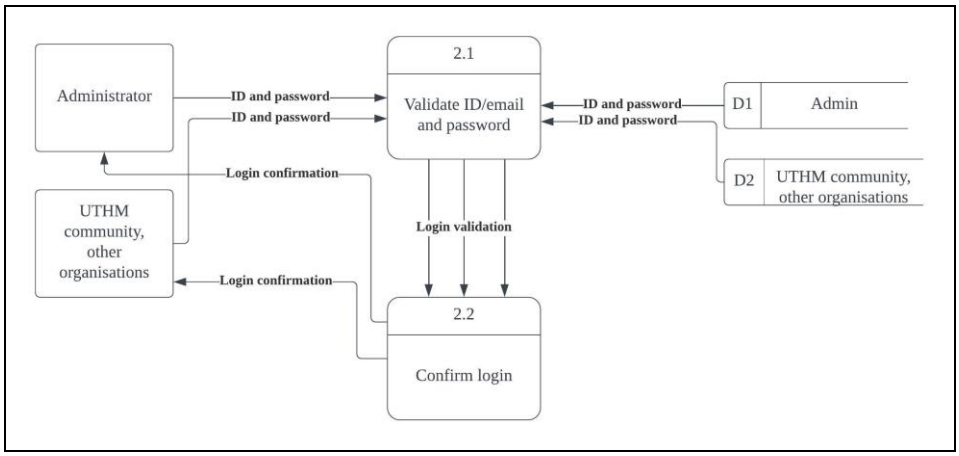


Appendix B: DFD Level 2

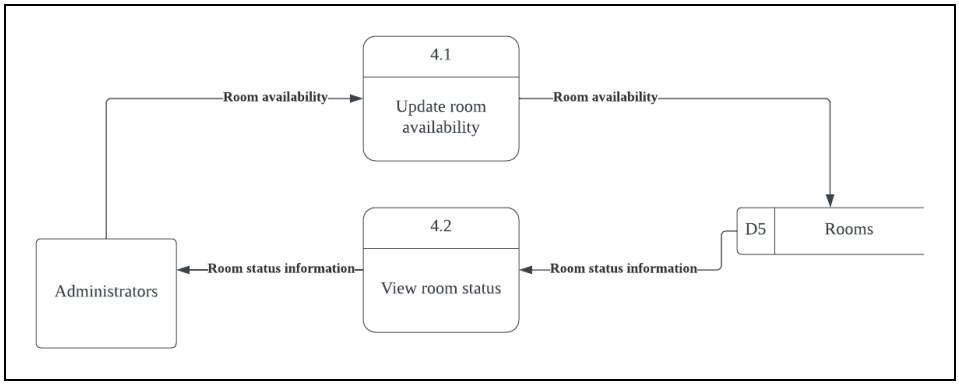
Register Account



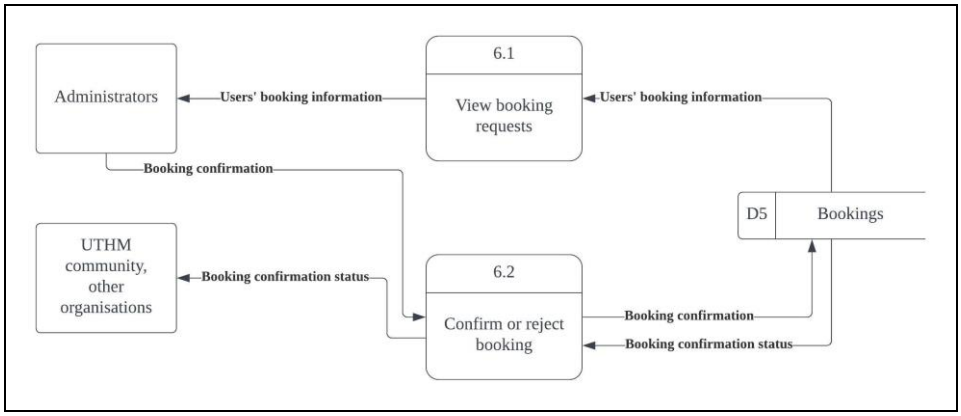
Login



Update Room Availability



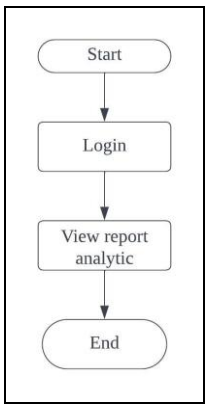
Update Booking Status



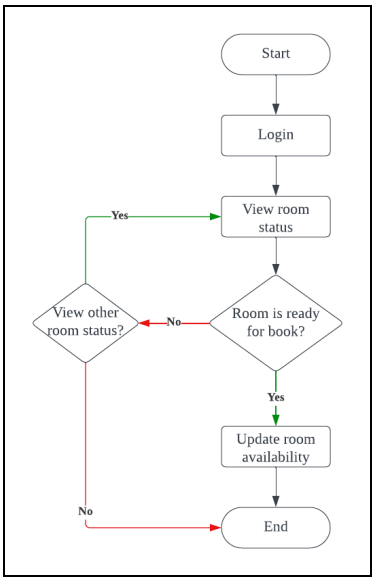
Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 41-60

Appendix C: Flowchart

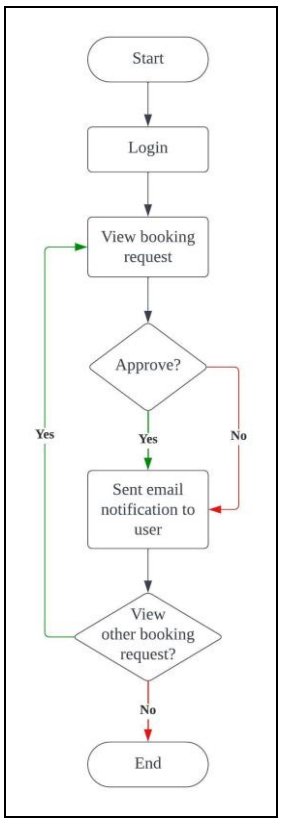
a) Administrators View Report Analytic Process



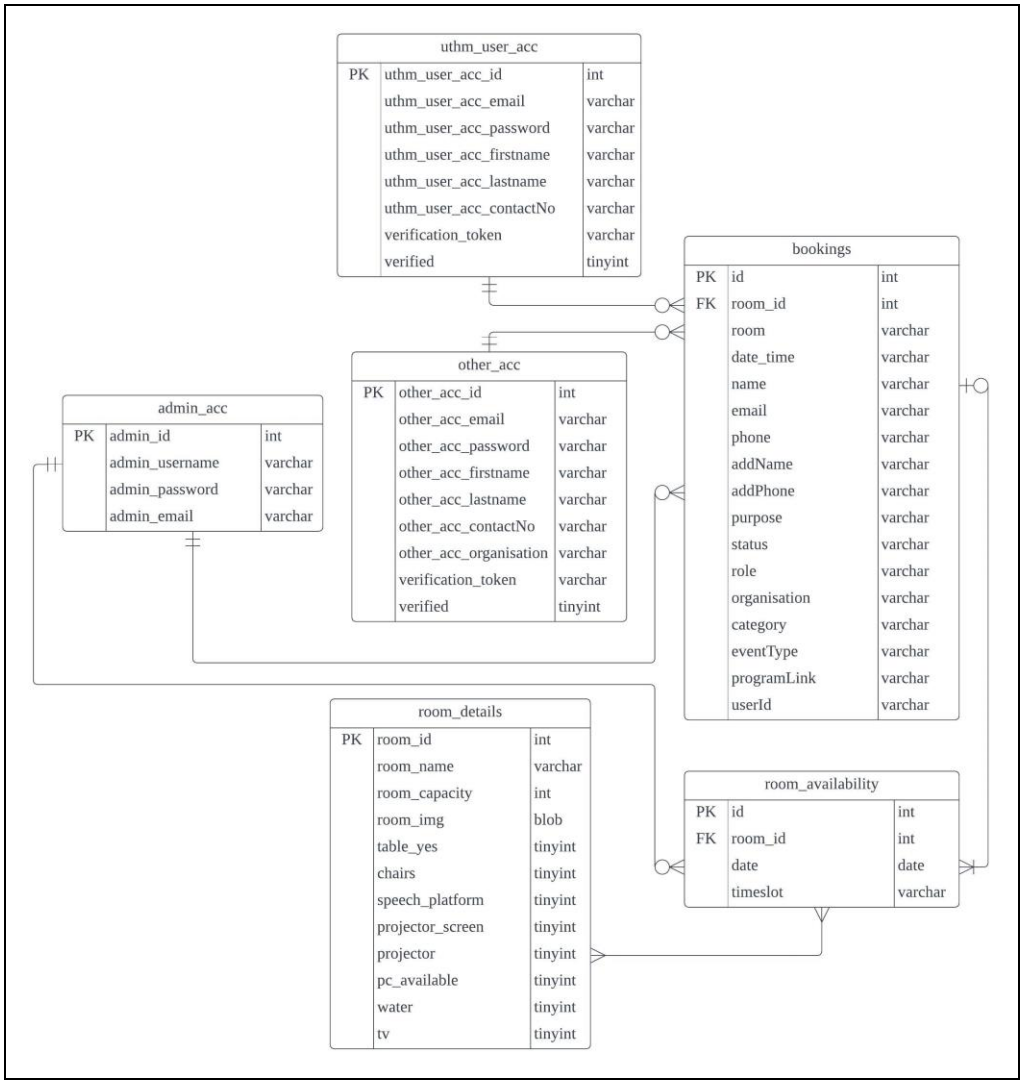
b) Administrators Update Room Availability



c) Administrators Confirm Booking Process



Appendix D: ERD



Appendix E: Change Password Interface