Techniques used

- Functional programming
    - Abstraction
- PHP session
- MySQL connection and query
    - While loops
- Dynamic database operation: adding, updating and deleting
- Data filtering
    - Decomposition
    - Logical operators
    - Conditional statements
- Graphical User Interface (GUI)

Functional programming

admin > modular > booking.php

```php
<?php
include('../config/constants.php');
include('login-check.php'); ?> <!-- Since booking.php is used in all pages,
I can input the constant.php here to avoid more repetition of code -->

<html>
    <head>
        <title>Cat Hotel Admin</title>

        <link rel="stylesheet" href="../css/admin.css">
    </head>

    <body>

        <div class="booking text-center">
            <div class="wrapper">
                <ul>
                    <li><a href="index.php">Home</a></li>
                    <li><a href="manage-admin.php">Admin</a></li>
                    <li><a href="manage-booking.php">Bookings</a></li>
                    <li><a href="manage-room.php">Room</a></li>
                    <li><a href="manage-service.php">Services</a></li>
                    <li><a href="logout.php">Logout</a></li>
                </ul>
            </div>
        </div>

```

Figure 1: Navigation bar for web application

To make my program more efficient, I have used functional programming. Since my website contains lots of pages of similar characteristics, functional programming can be used to prevent rewriting identical code for different files. For example, the code above is written for the navigation bar of each web page and is written in an independent file. Instead of having to repeatedly write this code for different page files, I can simply just use the PHP code: *<?php include('modular/booking.php') ?>* to call this one file. By breaking down the files of my website into functions, it increases efficiency because it avoids having to repeatedly load a larger body of code which does the same process; this is likely to happen if procedural

programming was used instead. Functional programming, as opposed to procedural, takes up less memory space due to less code written and makes it much faster for users to access different pages (this suits the client's wishes as mentioned in criterion A). Additionally, through functional programming, if an error occurs, it will only need to be fixed once which further emphasises the efficiency. Functional programming also increases the abstraction of my code; large lines of identical code are hidden under a singular function call but its functionality remains intact. This simplifies the code for future development by the client.

PHP sessions

```php
<?php
    //start session
    session_start();
```

Figure 2: Starting a PHP session

```php
$_SESSION['login'] = "<div class='success'>Login successful</div>";
$_SESSION['user'] = $username; //checks if user is logged in or not

header("location:".SITEURL.'admin/');
```

Figure 3: Using the global session variable to acknowledge that an admin has logged in[1]

admin > logout.php

```php
<?php

    include('../config/constants.php'); //include the mysql connection constant
    session_destroy(); //unsets user session and logout from the system

    header("location:".SITEURL.'admin/login.php');

?>
```

Figure 4: Terminating the session once admin has logged out and redirecting back to login page

Unlike a regular application, websites have no way of acknowledging who the current user of it is because the HTTP address doesn't stay the same[2]. Since my program needs a section for admins and customers, it is crucial that unauthorised customers should be prevented from accessing the admin page. When an admin logs out, a customer shouldn't be able to simply change the URL to gain admin privilege. For this reason, I've implemented a PHP session system to temporarily store the information of the admin. I have chosen sessions instead of cookies because sessions store information temporarily within a server directory

[1]"PHP Sessions." W3Schools, Refsnes Data, www.w3schools.com/php/php_sessions.asp.
[2]"PHP Sessions." W3Schools, Refsnes Data, www.w3schools.com/php/php_sessions.asp.

whereas cookies store information permanently on the user's computer. To maintain security, the login information in my website should be stored temporarily because it isn't guaranteed the same user will use the same computer each time hence why sessions have been chosen over cookies. This session system essentially stores information about the admin by generating a unique session ID and ensures that the whole web application passes the admin's information.[3] I used the code above: *session_destroy()* to terminate the session once the admin logs out. This way, if a customer attempts to access the admin page by modifying the URL, they will be denied access because there is no existing session ID to permit them to do that. This system ensures that the admin and customer section remains separate.

MySQL connection and query

```php
$conn = mysqli_connect('localhost', 'root', '', 'cat-booking') or die(mysqli_error());
```

Figure 5: Line of code to connect database to web application[4]

```php
<?php
    $sql = "SELECT * FROM tbl_admin"; //sql query to fetch all admin data
    $res = mysqli_query($conn, $sql); //executing the query

    if($res == TRUE){

        $row_count = mysqli_num_rows($res); //function to check the number of rows in the database
        $idchange=1; //avoids the id number not showing in order

        if ($row_count>0){

            while ($rows=mysqli_fetch_assoc($res)){ //while loop runs as long as there is data in the database

                $id=$rows['id'];
                $username=$rows['username'];
                $password=$rows['password'];

                ?>
```

Figure 6: MySQL query used to fetch admin data from database

The website uses MySQL database to store hotel's data in a unified location, as preferred by the client. To access the MySQL database, I used the MySQL method instead of PDO as the MySQL method has a 6.5% better performance than PDO so this delivers data faster which is ideal for delivering user requests quickly on the website.[5] Above, I've used a while loop instead of a for loop to carry out the data retrieval process as this avoids creating extra variables that may take up unnecessary memory space; the while loop here simultaneously

---

[3] Bradley, Angela. "Understanding How PHP Sessions Work." ThoughtCo., 5 Feb. 2019, www.thoughtco.com/basic-php-sessions-2693797.

[4] Ali, Inshal. "How to Connect MySQL Database with PHP Websites." Cloudways, 7 Dec. 2021, www.cloudways.com/blog/connect-mysql-with-php/#connection.

[5] Marjanovic, Dejan. "PDO vs. MySQLi: Which Should You Use?" EnvatoTuts, 2012, code.tutsplus.com/tutorials/pdo-vs-mysqli-which-should-you-use--net-24059.

creates a variable *$rows=mysqli_fetch_assoc($res)* (used to fetch data) and the boolean condition to run the loop. Hence, the while loop is more efficient than for loop as it doesn't need to create separate variables for the loop's condition and the data retrieval. By being more efficient, I allow the website to operate faster which is ideal for the client's aim for improving productivity.

Dynamic MySQL database operation

```
<form action="" method="POST">
    <table class="tbl-30">
        <tr>
            <td> Username: </td>
            <td><input type="text" name="username" placeholder="Enter username"></td>

        </tr>

        <tr>
            <td> Password: </td>
            <td><input type="password" name="password" placeholder="Enter password"></td>

        </tr>

        <tr>
            <td colspan="2">
            <input type="submit" name="submit" value="Add Admin" class="btn-secondary">
        </td>

        </tr>

    </table>

</form>
```

Figure 7: HTML form for adding data[6]

```
$username = $_POST['username']; //getting date from the form using the name
$password = $_POST['password'];

//sql query of data, name on left is name of column; name of right is name of value of data from form. Insert query.
$sql = "INSERT INTO tbl_admin SET
    username='$username',
    password='$password'
";
```

Figure 8: MySQL query adding the form data to the database[7]

As the information I store in the web application doesn't remain static, I need to be able to use operations to modify the contents of my database; these include adding, updating, and deleting data. For adding data into the database, I used a HTML form which the users can

[6] "2. Food Order Website with PHP and MySQL (Add and Display Admins)." YouTube, uploaded by Vijay Thapa, www.youtube.com/watch?v=o-6cL61-uDw&list=PLBLPjjQlnVXXBheMQrkv3UROskC0K1ctW&index=2.

[7] "How to Connect HTML Form with MySQL Database using PHP." YouTube, uploaded by Technical Babaji, 31 Mar. 2019, www.youtube.com/watch?v=2HVKizgcfjo&t=135s.

input information into, as shown by the example in figure 7. I used an HTML form, instead of any other forms, as it allowed the users to input data and also allowed the use of predefined methods like *"POST"* which made the data processing part with MySQL less complicated to do; when the HTML form posts the data, the PHP code can easily retrieve this data by using '$_POST'. These predefined methods increase the abstraction of my program as the code uses predefined global variables and methods.[8] Increasing the abstraction within my code will allow the client to further develop the website easily as they do not have to worry about extra complexity. After this, I used the MySQL query method to parse the form data into the database. In figure 8, I initialised the variables *$username* and *$password* according to the input by the user (which is retrieved by the code *$_POST*) and used these variables to be inputted within the SQL query.

Data filtering

```
if ($status!='Removed'){
    ?>
    <tr>
        <td><?php echo $idchange++ ?></td>
        <td><?php echo $fullname ?></td>
        <td><?php echo $email ?></td>
        <td><?php echo $room ?></td>
        <td><?php echo $length_stay ?></td>
        <td><?php echo $additional_services ?></td>
        <td>$<?php echo $total ?></td>
        <td><?php echo $status ?></td>

        <?php
            if ($status=='Pending confirmation'){ //admin needs to confirm booking first before being added to system i.e. customer needs to pay first befor
                ?>
                <td><a href="<?php echo SITEURL; ?>admin/add-customer.php?id=<?php echo $id; ?>"class="btn-secondary">Confirm Booking </a></td>
                <td><a href="<?php echo SITEURL; ?>admin/decline-customer.php?id=<?php echo $id; ?>"class="btn-tertiary">Decline Booking </a></td>
                <?php
            }
            else if ($status=='Booked'){ //admin can checkout customer if they are currently booked
                ?>
                <td><a href="<?php echo SITEURL; ?>admin/checkout-customer.php?id=<?php echo $id; ?>"class="btn-danger">Checkout Customer </a> </td>
                <?php
            }
            else{ //admin can decide whether to keep or remove customer data once they have checked out
                ?>
                <td><a href="<?php echo SITEURL; ?>admin/delete-customer.php?id=<?php echo $id; ?>"class="btn-tertiary">Remove Customer </a></td>
                <?php
            }
        ?>
    </tr>
    <?php
```

Figure 9: Example of when data is filtered according to what its database attributes are

Automation of the website is required to satisfy my client. When fetching data, I ensured the code selects and processes the appropriate data from the user requests to provide automation. To do this, I needed to filter the data from the database and this can be done through selection statements and decomposition as this allows the code to form logical decisions on its own. In the code above, I broke down the problem into individual conditions. By using decomposition instead of tackling the entire problem at once, I've helped develop a

[8] "PHP $_POST." TutorialsPoint, www.tutorialspoint.com/php-post.

clearer process for the code for future developers. Selection statements ensure that only the necessary code should be run hence increasing the efficiency of my code.

Graphical user interface (GUI)

```
*{
    margin: 0;
    padding: 0;
    font-family: Arial, Helvetica, sans-serif;
}

.wrapper{
    padding: 1%;
    width: 80%;
    margin: 0 auto;
}
```

Figure 10: CSS classes for changing the appearance of the web application[9]

```
<div class="booking text-center">
    <div class="wrapper">
        <ul>
            <li><a href="index.php">Home</a></li>
            <li><a href="manage-admin.php">Admin</a></li>
```

Figure 11: Implementation of CSS classes into HTML code

The user-centred nature of my product makes it important to have high accessibility and usability. Since everyone doesn't possess the skills to read HTML or PHP code, it would be hard for them to navigate through my web application without graphical assistance. Through GUI, users can easily navigate through the web application as there are graphical components to assist them in doing so. Graphical components are much simpler to understand than lines of code thus making my web application more accessible. I used CSS to modify the graphical appearance of my web application like seen in figure 10 (I have used a template CSS from an online source to quicken my process). Then, in my HTML code, I included the classes created in the CSS file with the use of codes like *class="rooms"* to modify its appearance. GUI helps develop a user-friendly environment and thus encourages the high usability of my product.

Word count: 1094

[9] Thapa, Vijay. "Restaurant Food Order Theme." GitHub, github.com/vijaythapa333/web-design-course-restaurant.

Works Cited

Ali, Inshal. "How to Connect MySQL Database with PHP Websites." Cloudways, 7 Dec. 2021, www.cloudways.com/blog/connect-mysql-with-php/#connection.

Bradley, Angela. "Understanding How PHP Sessions Work." ThoughtCo., 5 Feb. 2019, www.thoughtco.com/basic-php-sessions-2693797.

"How to Connect HTML Form with MySQL Database using PHP." YouTube, uploaded by Technical Babaji, 31 Mar. 2019, www.youtube.com/watch?v=2HVKizgcfjo&t=135s.

Marjanovic, Dejan. "PDO vs. MySQLi: Which Should You Use?" EnvatoTuts, 2012, code.tutsplus.com/tutorials/pdo-vs-mysqli-which-should-you-use--net-24059.

"PHP MySQL Update Data." W3Schools, www.w3schools.com/php/php_mysql_update.asp.

"PHP $_POST." TutorialsPoint, www.tutorialspoint.com/php-post.

"PHP Sessions." W3Schools, Refsnes Data, www.w3schools.com/php/php_sessions.asp.

"2. Food Order Website with PHP and MySQL (Add and Display Admins)." YouTube, uploaded by Vijay Thapa, www.youtube.com/watch?v=o-6cL61-uDw&list=PLBLPjjQlnVXXBheMQrkv3UROskC0K1ctW&index=2.

Thapa, Vijay. "Restaurant Food Order Theme." GitHub, github.com/vijaythapa333/web-design-course-restaurant.

Images used:

Clips, Caters. Cat Lover Opens Luxury Cat Hotel. YouTube, i.ytimg.com/vi/r8INYeWaCOk/maxresdefault.jpg.

Luxurious Cat Hotels. Pinterest, www.pinterest.com/pin/273523377349125014/.

Luxury boutique cat hotel. 8 Aug. 2017. IBPSA, www.ibpsa.com/blog/2017/08/08/purrfectly-meowvelous-following-trends-luxury-boutique-cat-hotel/.