Applied Information Technology And Computer Science Vol. 3 No. 2 (2022) 209-224
© Universiti Tun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

A Transformation of Back-Office System from MVC to Blazor and Secure Password Using PBKDF2

Mohamad Aizuddin Mustaffa Kamal[1], Chuah Chai Wen[1*]

[1]Fakulti Sains Komputer dan Teknologi Maklumat,
Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2022.03.02.014
Received 20 July 2022; Accepted 30 September 2022; Available online 30 November 2022

Abstract: DIY Tour Asia is an online booking company for package tours founded in 2012. DIY Tour Asia online platform has two different web applications. They are customer application and Back-Office System. The customer application allows customers to view and book available package tours. The Back-Office System allows administrators and staffs to manage the business in term of inventory, price, availability, and reporting. DIY Tour Asia Back-Office System is using the Model-View-Controller (MVC) framework as their backbone. MVC framework is causing a lot of code duplicates and is tightly coupled. Migrating to the Blazor framework can reduce code duplication through the Dependency Injection (DI) method. In addition, the system uses PBKDF2 to strengthen password security from brute force attacks. The PBKDF2 algorithm hashes the password before storing it in the database. The Object-Oriented Software Development is chosen for the methodology. The proposed system is developed using C# language. The developed system allowed only staff members to log in to the system. After login to the system, the user may choose to interact with any of the five modules which are administration, profile management, hotel, hotel booking, and hotel sales report. However, the administration module is only for administrators. The administration, profile management and hotel modules support CRUD process. The authorized user may view details of hotel booking and cancel them. The hotel sales report module allows user to export the results in three formats which are Microsoft Word, Microsoft Excel, and Portable Document Format.

Keywords: Blazor, PBKDF2, Web-Based, C#

1. Introduction

DIY Tour Asia is an online booking company for packaged tours founded in 2012. DIY Tour Asia offers an online platform to book package tours, mainly to destinations in Asia such as China, Indonesia, Japan, Korea, and other Southeast Asia countries. For tourists that travel in groups, DIY Tour Asia offers a discount on bulk booking.

DIY Tour Asia online platform has two different web applications, there are customer application and Back-Office System. The customer application allows customers to book available Asia package

*Corresponding author: cwchuah@uthm.edu.my
2020 UTHM Publisher. All rights reserved.

tours. The Back-Office System allows merchants, staffs, and administrators to customize Asia package tours in term of inventories, prices, and availabilities. They may access customer booking orders to update or cancel the orders.

Currently, the DIY Tour Asia Back-Office System is using the Model-View-Controller (MVC) framework as their backbone. The Model represents the database records. This layer manages data and allows database connectivity. The View is responsible for displaying the Model's data. The Controller gets the request from View and retrieve data from Model. Then, the Model responds to View for displaying output.

One of the disadvantages for MVC is most of the codes that does not fit in the View and Model are dumped into the Controller. The abundance of codes in the Controller makes it difficult to read and maintain. Then, the framework does not allow multiple database provider such as SQL Server, MySQL, and Oracle. This reduces the scalability for the system. Next, the framework depends highly on JavaScript inside the web page. This results in code duplication and consumes high storage space.

The proposed project uses Blazor framework. It utilizes Model and Pages. The Model represents database records. The Pages handles the output and the business logics. These two components ease the developer to read and maintain the algorithm. Then, Blazor framework supports Dependency Injection (DI). Dependency Injection is a method for accessing services in the program such as multiple database provider. Next, Blazor framework supports the Razer Syntax. It allows the mixing of C# into standard HTML for logic and data manipulation [1]. This may reduce the dependency on JavaScript language and avoid code duplications.

2. Related Work

2.1 Blazor Web Framework

Blazor is a framework that enables developers to create web applications using C# and HTML. Blazor are divided by three components which is the razor pages, data services, and model. The razor components allow the usage of HTML template and C# code when needed. The services are algorithm functions that may be reused multiple times. These services may also interact with the model. The model acts as database records. This framework eliminates the needs to use JavaScript. Thus, making it much simpler to code. Figure 1 illustrates the Blazor Server project structure.

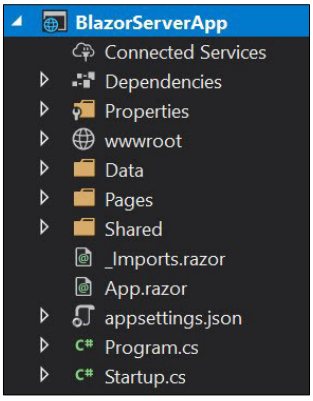

Figure 1: Blazor Server Project Structure [7]

2.2 Model-View-Controller (MVC) Web Framework

MVC is a development framework designed to isolate the data model from the presentation layer (This eases any changes to the presentation layer without the needs to interfere with the model. The model represents the database records. The view presents the data to the user. The controller processes the interactions between view and model. User input is managed here and delivered to the model in the

form of instructions defining what the model should obtain. Changes to the data in model affect the view for user as well. Figure 2 illustrates the MVC project structure.

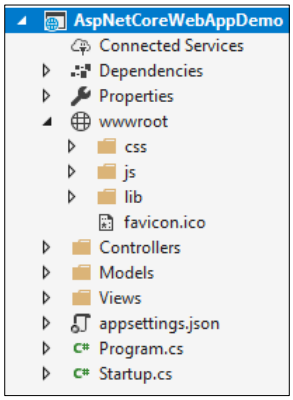

Figure 2: MVC Project Structure [8]

2.3 Secure Hash Algorithm 256 (SHA-256)

SHA256 is used to generate the MAC of HMAC. The input of SHA256 is called a message. The process of SHA256 is illustrated in Figure 3.

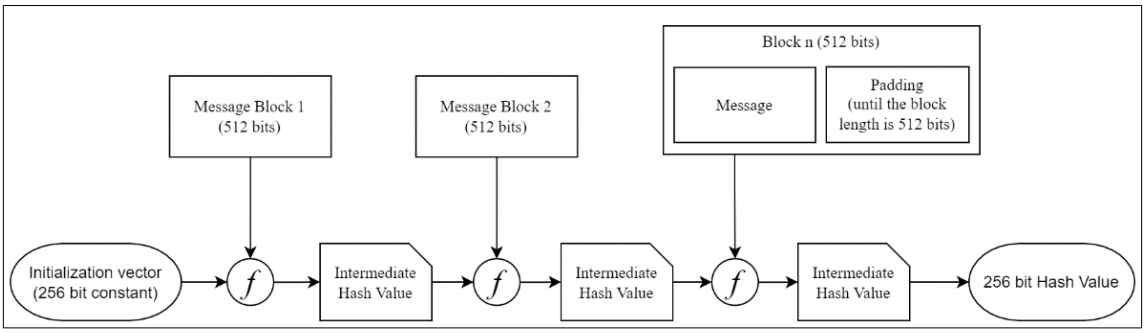

Figure 3: SHA256 Algorithm Process [2]

The SHA256 algorithm follows these steps [2]:

1. Obtain the input message and confirm that its length is a multiple of 512 bits. This is accomplished by adding padding.
2. Parse the pass message into N 512-bit pieces.
3. Repeat through all the steps in step 2.
   - 3.1 Initialize the message schedule which is a sequence of 64 32-bit words.
   - 3.2. Initialize the eight key variables a, b, c, d, e, f, g, h using the previous iteration's hash values H0, H1, H2, H3, H4, H5, H6, H7 (H0 to H7 are initialized with constants in the first iteration).
   - 3.3. Execute 64 iterations in which the key variables a through h are rotated in a certain order. The message is placed into the hash in this step, which involves a lot of bitwise mixing.
   - 3.4. Calculate the new intermediate hash values $H0$ to $H7$ as $H0 = H0 + a$, $H1 = H1 + b$ and so forth.
4. To get the message digest, concatenate $H0$ to $H7$ and return it.

2.4 Key Derivation Function (KDF): Password-Based Key Derivation Function 2 (PBKDF2)

PBKDF2 is commonly used to obtain cryptographic keys from passwords and can also be used for key storage. It is resistant to GPU-based brute-force attacks that can reduce hundreds and thousands of the guess rates per second. The PBKDF2 key derivation formula has five input parameter as follows [3]:

Derived Key = PBKDF2(PRF, Password, Salt, c, dkLen)

1. PRF is a pseudorandom function of two parameters such as a keyed HMAC_SHA256.
2. Password is the user input password.
3. Salt is a can be a random word converted into bits.
4. c is the number of iterations to repeat the computation.
5. dkLen is the needed bit-length of the derived key.

3. Methodology/Framework

The methodology used to develop the back-office system is Object-Oriented System Development (OOSD) is a design, analysis, and development technique for systems and applications. In Figure 4, there are four phases in the Object-Oriented System Development Life Cycle, which include object-oriented requirement analysis, object-oriented analysis, object-oriented design, and object-oriented implementation and testing [4].

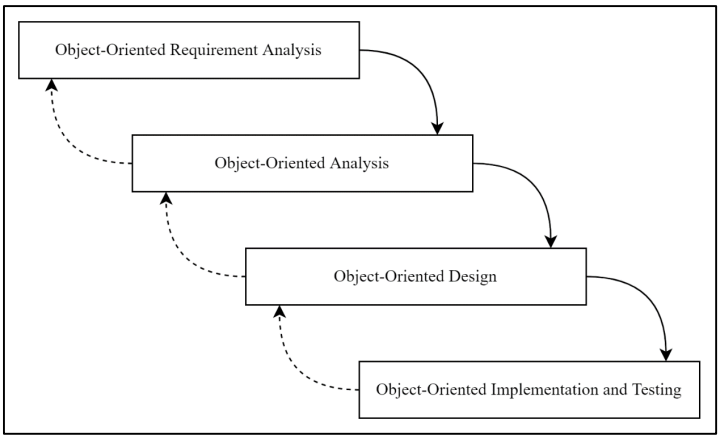

Figure 4: Object-Oriented System Development Life Cycle [4]

3.1 Object-Oriented Requirement Analysis

In the object-oriented requirement analysis phase, the requirements for the proposed system are analyzed from an interview with the AlphaReds Solution company representative and observation of the existing systems. Based on the finding, the basic requirement properties of the Back-Office System for DIY Tours and Travel are customer's booking, hotel management, activity listing, maintenance setup, and promotions. Then, the security requirements are confidentiality, authorized access, and non-repudiation.

The security aspect includes confidentiality, authorized access, and non-repudiation. For confidentiality, the registered users' passwords are stored in the hashed form using PBKDF2. Using PBKDF2 ensures that the hash is resilient to brute force attacks. Next, authorized access means the application only allows users with permission to access the pages on the application. Each role has different permissions that allow them to access a specific web page. Lastly, the non-repudiation makes it that everyone is accountable for their action. Every record has a modified-by attribute. Every time a

piece of information is updated, the modified-by attribute update too according to the current account name. This provides strong proof against repudiation.

There are two existing systems reviewed. Extranet by Emerging Travel Group and Vrbo by Expedia Group. Emerging Travel Group is a company that advertises property listings for tours and travels. Property owners may sign up with Extranet to manage their property listings. Expedia Group is one of the world's largest full-service travel brands. It seeks to offer the most diverse range of holiday locations, airfares, car rentals, cruises, and in-destination activities. Vrbo, which is under Expedia Group, includes a back-office system for owners and managers. The difference between both existing systems and the proposed system is the lack of customizable access control and lack of reports generator.

Specifications and properties of materials, equipment, and other resources used in the current study should be described in this section. Should a bulleted list be required, it may be included and should look like this:

3.2 Object-Oriented Analysis

In the object-oriented analysis phase, the modules for the proposed system are planned out. There are a total of six modules for the system. The modules are login, administration, profile management, hotel booking, hotel, and hotel sales report. Table 1 explains the six modules of the proposed system.

Table 1: Modules for the Proposed System

Modules	Description
Login	• Authorized users may login to the system using email and password. • Passwords must consist of alphabets (lowercase and uppercase), numbers, special characters, and minimum length of 10 characters. • Passwords in the database are hashed with PBKDF2 algorithm. • All users may request for password reset by clicking on the Forgot Password link.
Profile Management	• All users may change their password by entering their old password and new password. • All users may update their own user profile. • All users may update their own organization profile
Administration	• Only administrator may create, read, update, and delete user accounts. • Only administrator may create, read, update, and delete the associated organization. • Only administrator may create, read, update, and delete user group and assign them page access.
Hotel	• All users may create, read, update, and delete hotel information. • Hotel information includes hotel name, country, city name, room name and more. • All users may create, read, update, and delete room rate plan. • Room rate plan includes pricing, room rate, and inclusion details for a room
Hotel Booking	• All users may read and update customers' hotel booking.
Hotel Sales Report	• All users may read and filter the results of hotel sales report. • All users may export the filtered hotel sales report results into Microsoft Excel and PDF format.

3.3 Object-Oriented Design

In the object-oriented design phase, the architecture of the proposed system is designed. The database table, classes, user interface (UI), and test plans for the proposed system are all designed.

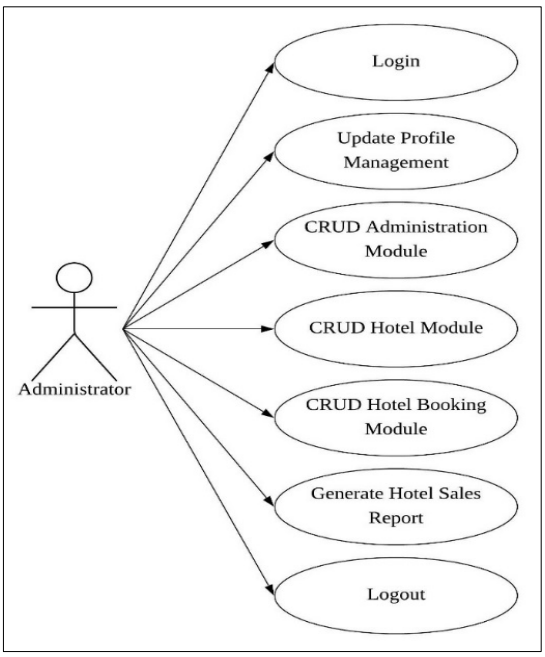

Figure 5: Use Case for Administrator

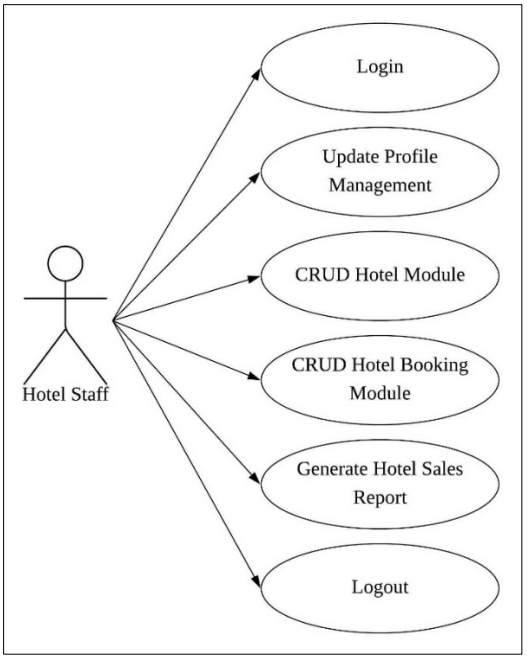

Figure 6: Use Case for Hotel Staff

As shown in the use case diagram in Figure 5, the administrator may perform seven actions. This includes login, update profile management, CRUD administration module, CRUD hotel module, CRUD hotel booking module, generate hotel sales report and logout.

Figure 6 illustrate the use case diagram for hotel staff. They may perform six actions which are login, update profile management, CRUD hotel module, CRUD hotel booking module, generate hotel sales report and logout.

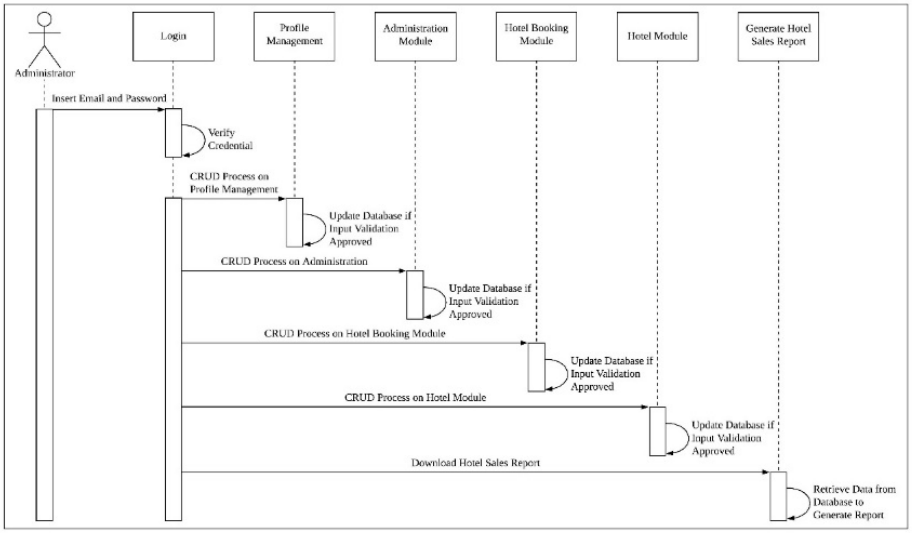

Figure 7: Sequence Diagram for Administrator

As shown in Figure 7, administrator have six actions that they can perform. Administrator needs to login to their account to use the system. After login, administrator can choose to do CRUD process on

four module which are profile management, administration module, hotel booking module, and hotel module. Lastly, they can download hotel sales report from the system.

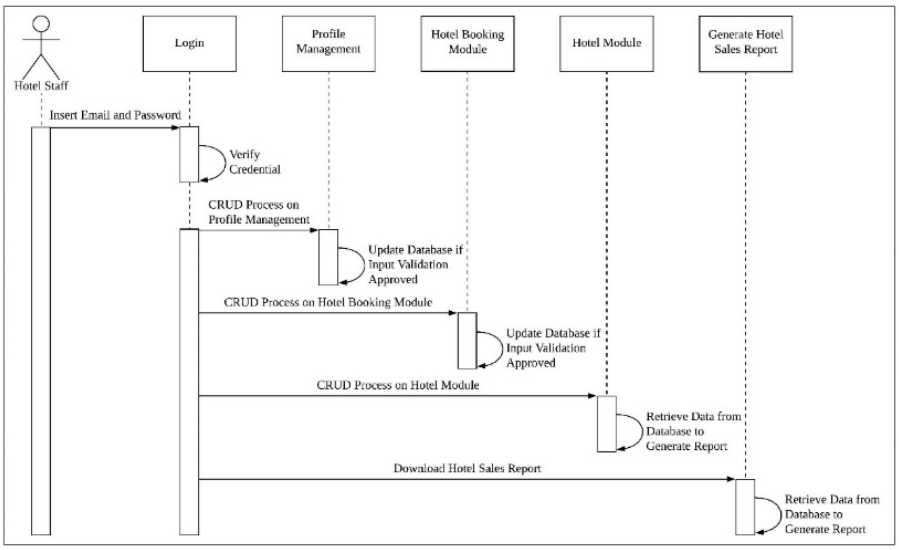

Figure 8: Sequence Diagram for Hotel Staff

As shown in Figure 8, hotel staff have five actions that they can perform. Hotel staff needs to login to their account to use the system. After login, hotel staff can choose to do CRUD process on three modules which are profile management, hotel booking module, and hotel module. Lastly, they can download hotel sales report from the system.

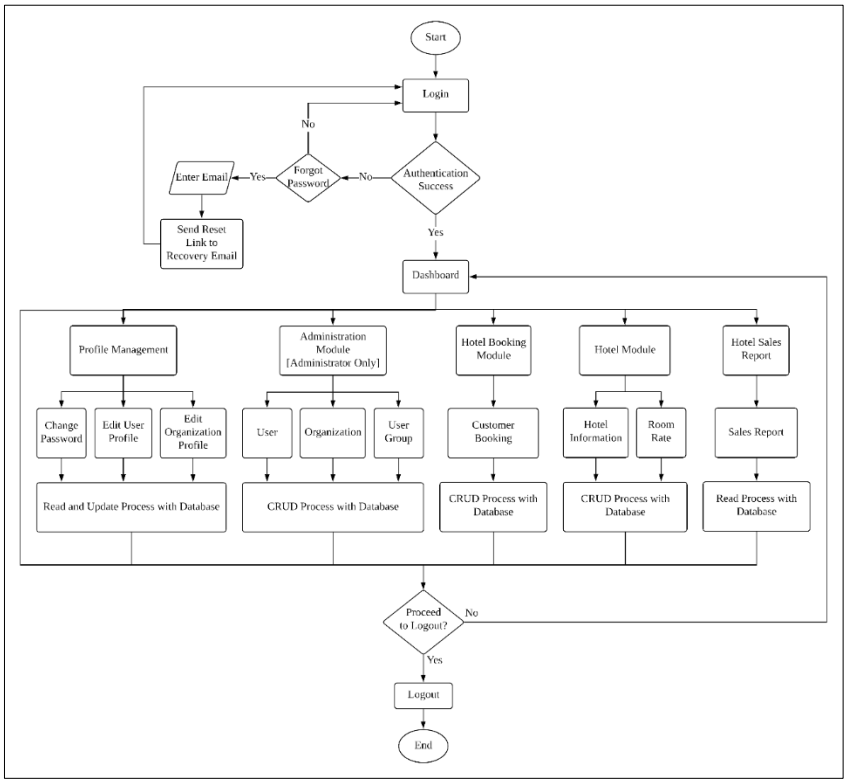

Figure 9: Activity Diagram for Proposed System

Based on Figure 9, after login, user may choose between five modules. These modules are profile management, hotel booking module, hotel module, hotel sales report and administration module. However, only administrators are allowed to access the administration module. If user chooses the

profile management, they may change password, edit user profile, and edit organization profile. If user chooses the administration module, they may CRUD user, organization, and user group information. If user chooses hotel booking module, they can view and update customers booking. If user chooses hotel module, they may CRUD hotel information and room rate. Finally, if user chooses hotel sales report, they may export and download the sales report from the database. After user is finished interacting with the system, they may choose to proceed to log out or choose another module from the main menu.

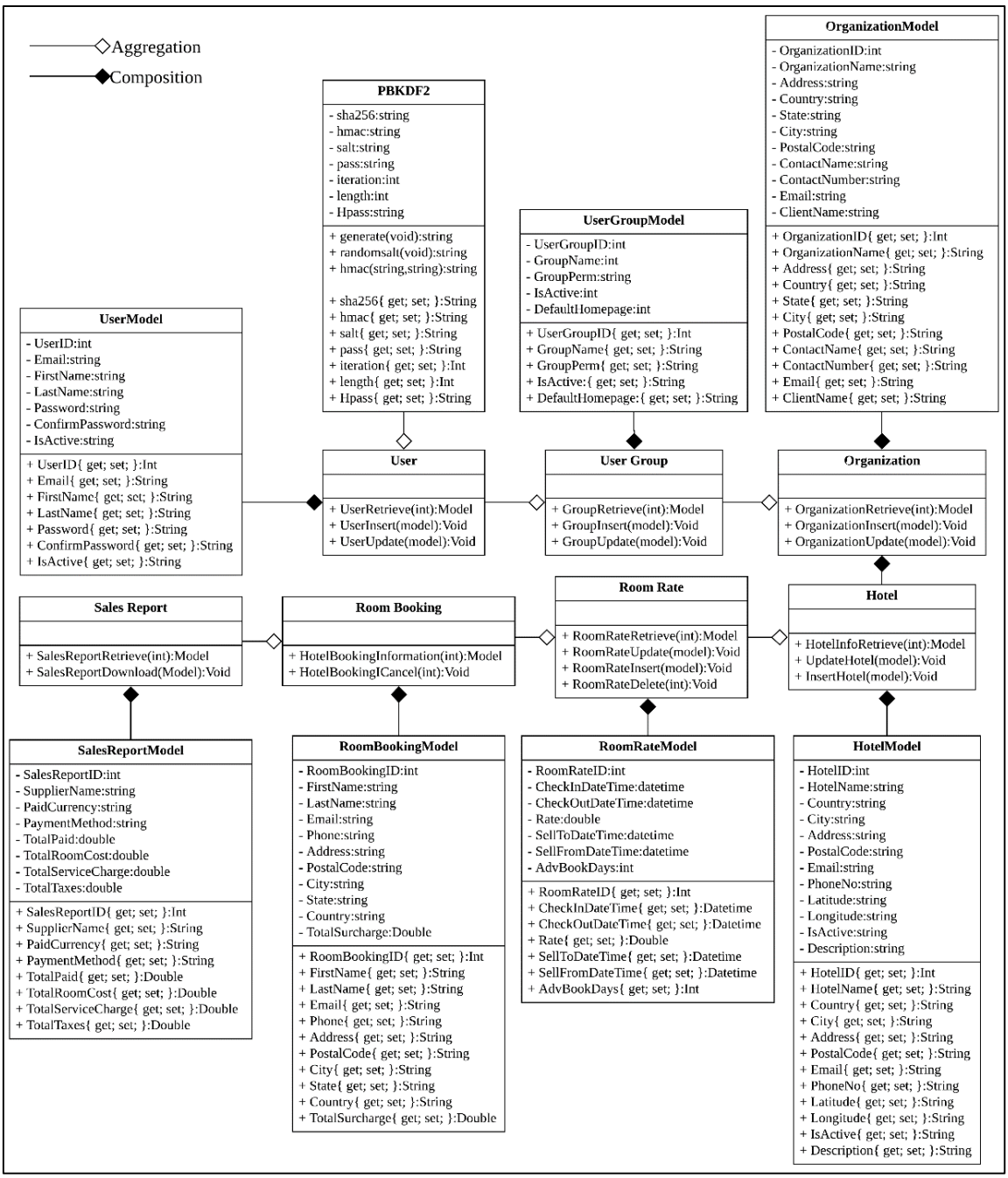

Figure 10: Class Diagram for Proposed System

As shown in Figure 10, there are a total of 7 operational classes and 8 model classes exist in the proposed system. The operational classes are Hotel, Organization, User, UserGroup, PBKDF2, RoomRate, RoomBooking, and Sales Report. The operational classes have operational methods. Then, the model classes are HotelModel, OrganizationModel, UserModel, UserGroupModel, RoomRateModel, RoomBookingModel, and SalesReportModel. Model classes have their own attributes and encapsulation methods because they represent the database.

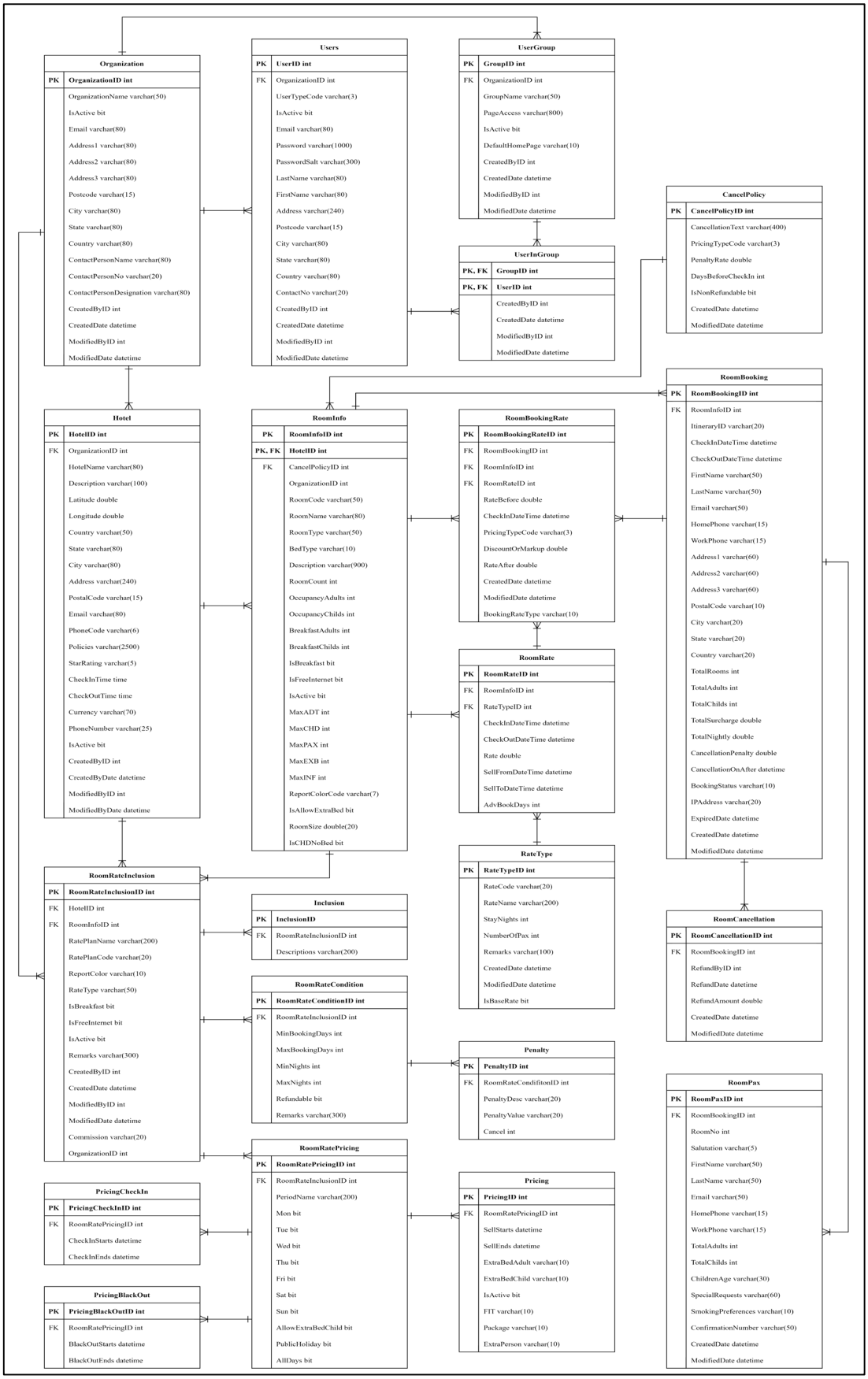

Figure 11: Entity-Relationship-Diagram

The proposed system ERD has 21 entities which are User, Organization, UserGroup, UserInGroup, Hotel, RoomInfo, RoomPax, Penalty, CancelPolicy, RoomCancellation, RoomRate, RoomBooking, RoomRateBooking, RoomRateCondition, Pricing, RoomRatePricing, Inclusion, RomRateInclusion, RateType, PricingCheckIn, and PricingBlackOut. Figure 11 illustrates the ERD of the proposed system.

Table 2: Security Test Plan

No	Check List	Result
1	Passwords must consist of alphabets (lowercase and uppercase), numbers, special characters, and minimum length of 10 characters.	Pass / Fail
2	Passwords are obscured in the input box.	Pass / Fail
3	Ensure users can only access the page based on their role.	Pass / Fail
4	Passwords in the database are hashed with PBKDF2 algorithm.	Pass / Fail
5	Random salt is added to the password before hashed with PBKDF2 algorithm	Pass / Fail

Lastly, the test plan is created to determine if the proposed system is performing as planned. There are two test plans that was designed. One is a functionality test plan and another one is a security test plan as shown in Table 2. Both test plans are critical for the testing phase.

3.4 Object-Oriented Implementation and Testing

In the object-oriented implementation phase, all the database tables, classes, and user interfaces are implemented. All the tables and classes are linked as their functions require it to work properly. For example, the RoomBooking class is connected to the RoomBookingRate table. The getter and setter methods in the RoomBooking class are able to retrieve the RoomBooking data from the RoomBookingRate table. Also, the classes are linked with the UI. For example, RoomBooking class is linked with the Hotel Booking UI module. The RoomBooking class retrieved data from RoomBookingRate table and then displayed it on the Hotel Booking UI module.

In the object-oriented testing phase, the test plan is used to validate the usability of the proposed system functions. The test plan is run twice, once for each of the test plan categories. The first is a system functionality test, followed by a security test. If any errors occur, the debugging technique is used to resolve the issues.

4. Results and Discussion

The expected outcome for this project is the authorized users may login using email and password. Then, the administrator user group may access all five modules in the system. The hotel staff user group may only access four modules which are profile management module, hotel module, hotel booking module, and hotel sales report module. Authorized user may initiate the CRUD process on the profile management module, administration module , hotel module, and hotel booking module. Then, authorized user may view and filter the results of hotel sales report. They can export the filtered results into Microsoft Word, Microsoft Excel, and PDF format.

4.1 Implementation of Safe Login Error Message

This module indicates that the error login message does not reveal which credentials are correct. Figure 12 depicts an error message from a failed login attempt in the proposed system. The failed login attempt error message is "You have entered an invalid email or password". The error message does not specify which credentials are incorrect or whether the account exists.

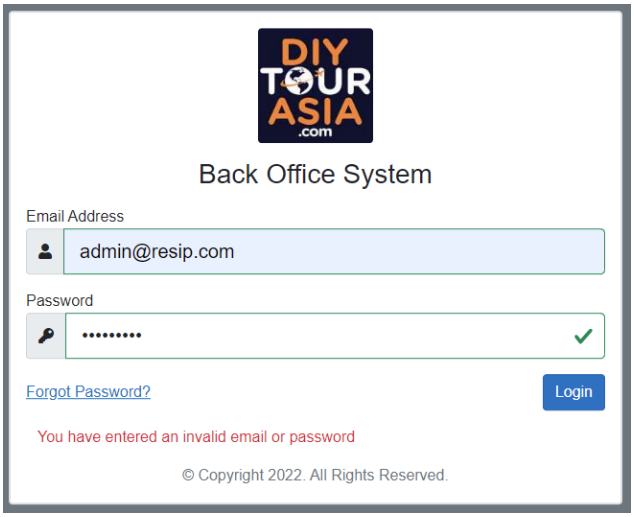

Figure 12: Error message for failed login attempt

4.2 Implementation of Strong Password Policy

The proposed system has a strong password requirement for every user account. The strong password requirement includes:

- Minimum 8 characters and maximum 30 characters long
- Include both lowercase and uppercase alphabets
- At least one number
- At least one special character

The password checking mechanism is implemented using Data Annotations inside the model class that contains all the data declarations. These functions specify a few attributes to the model data such as the required attribute, regular expression attribute, and data type attribute. Figure 13 shows the attributes assigned to the new password data.

```
[Display(Name = "New Password"), Placeholder("")]
[Required(ErrorMessage = "New Password is required")]
[RegularExpression("(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[\\W]).{8,30}",
ErrorMessage = "Password must between 8 to 30 characters, that include at least 1 uppercase character, 1 lowercase character, 1 number and 1 special character")]
[DataType(DataType.Password)]
[AllowEdit]
2 references
public string NewPassword { get; set; }
```

Figure 13: Data annotations attributes assigned to NewPassword data.

Based on Figure 13, the required attribute means that the field is required. If it is left empty, the error message “New Password is required” is displayed. Next, the regular expression attribute means the input needs to match with the pattern specified. If the pattern is not matched, the error message “Password must be between 8 to 30 characters, that include at least 1 uppercase character, 1 lowercase character, 1 number and 1 special character” is displayed. Lastly, the data type attribute decides what type of input that are going to be use. In this case, data type password will mask the password when users are typing their passwords into the input box.

4.3 Implementation of Parameterized Query

Every Structured Query Language (SQL) statement in this proposed system that involves taking user inputs such as SELECT, INSERT, UPDATE and DELETE are protected with parameterized query. The user inputs are supplied to the parameters before it is inserted into the statement to be executed. This sanitize user inputs that may contain SQL injection code from being directly appended to the query. Figure 14 shows a parameterized query for creating a new hotel room.

```
public void InsertHotelRoom(RoomList model)
{
    var connection = "StgDIYHotelDB";

    var sqlParams = new DynamicParameters();
    sqlParams.Add("@HotelID", model.ContentID);
    sqlParams.Add("@IsActive", model.IsActive);
    sqlParams.Add("@RoomName", model.RoomName);
    sqlParams.Add("@RoomCode", model.RoomCode);
    sqlParams.Add("@RoomType", model.RoomType ?? "NA" );
    sqlParams.Add("@BedType", model.BedType ?? "");
    sqlParams.Add("@RoomSize", model.RoomSize);
    sqlParams.Add("@Description", model.Description);
    sqlParams.Add("@MaxPAX", model.MaxPAX);
    sqlParams.Add("@MaxADT", model.MaxADT);
    sqlParams.Add("@MaxCHD", model.MaxCHD);
    sqlParams.Add("@MaxINF", model.MaxINF);
    sqlParams.Add("@MaxEXB", model.MaxEXB);
    sqlParams.Add("@OrganizationID", model.OrganizationID);

    sqlParams.Add("@MinADT", 0);
    sqlParams.Add("@MaxCHN", 0);
    sqlParams.Add("@MaxCHX", 0);
    sqlParams.Add("@MaxCHDINF", 0);

    string insertHotelRoom = "INSERT INTO TA2.RoomInfo (ContentID,RoomName,RoomCode,RoomType,BedType,Description," +
        "RoomSize,OrganizationID,IsActive,MaxADT,MaxCHD,MaxPAX,MaxEXB,MaxINF,MinADT,MaxCHN,MaxCHX,MaxCHDINF) " +
        "VALUES (@HotelID,@RoomName,@RoomCode,@RoomType,@BedType,@Description,@RoomSize,@OrganizationID,@IsActive," +
        "@MaxADT,@MaxCHD,@MaxPAX,@MaxEXB,@MaxINF,@MinADT,@MaxCHN,@MaxCHX,@MaxCHDINF);";

    _dapperService.Execute(insertHotelRoom, connection, sqlParams, System.Data.CommandType.Text);
}
```

Figure 14: Parameterized query for inserting a new row in table TA2.RoomInfo

4.4 Implementation of Input Validation

The proposed system has input validation in every user input. There are various types of input validation such as input length validation, input type validation and null or whitespace validation. Input length validation ensures that user input is within the space allocated in the database. Input type validation ensures that user input is within the same type that is specified in the database. Null or whitespace validation ensures that no null value or whitespace input from user which may contradict with the database column rules. All these validations are important for web applications. Otherwise, the program may break. Figure 15 shows input validations for data Address 1.

```
[Required(ErrorMessage = "Address Line 1 is required")]
[StringLength(80, ErrorMessage = "Input exceeded maximum allowed characters that is 80")]
[DataType(DataType.Text)]
4 references
public string Address1 { get; set; }
```

Figure 15: Input validations for data Address1

4.5 Implementation of Role-Based Authorization

Role based authorization are implemented on the menu, sidebar, and page load section in this proposed system. After user has logged in, the system retrieves the group role permission from the database and populate the data in the model. Then, when pages are loaded, the system will perform permission check and only permit user role that have specific permission to access the specific pages. If a user does not pass the permission check, the components in the page will not load. Figure 16 shows the role-based authorization in menu section.

```
<AuthorizeView>
    @if(perms.AC || perms.OS || perms.UG)
    {
        <li class="nav-item" id="iconnav">
            <Blazorise.Tooltip Text="Admin" Placement="Blazorise.TooltipPlacement.Right">

                <NavLink class="nav-link" @onclick="()=>ToggleSubmenu(Models.NavSubmenu.Admin)">
                    <span class="fas fa-user px-2" style='font-size:18px;' aria-hidden="true"></span>
                </NavLink>
            </Blazorise.Tooltip>

            @if(navSubmenu == Models.NavSubmenu.Admin)
            {
            <ul class="nav flex-column ps-4">
                @if(perms.AC)
                {
                    <Blazorise.Tooltip Text="Users" Placement="Blazorise.TooltipPlacement.Right">
                    <li class="nav-item" id="iconnav">
                        <NavLink class="" href="/Users">
                            <span class="fas fa-angle-double-right" aria-hidden="true"></span>
                        </NavLink>
                    </li>
                    </Blazorise.Tooltip>
                }
                @if(perms.OS)
                {
                    <Blazorise.Tooltip Text="Organizations" Placement="Blazorise.TooltipPlacement.Right">
                    <li class="nav-item" id="iconnav">
                        <NavLink class="" href="/Organizations">
                            <span class="fas fa-angle-double-right" aria-hidden="true"></span>
                        </NavLink>
                    </li>
                    </Blazorise.Tooltip>
                }
```

Figure 16: Role-based authorization in menu section

4.6 Implementation of Role-Based Authorization

In the proposed system, PBKDF2 algorithm is used to hash the users' password before storing it in database. Figure 17 shows the C# code used for PBKDF2 hashing process. The first method is GetPasswordHash(). It is used to hash user password using the PBKDF2 algorithm. GetPasswordHash() takes user password in string and return a hash in string. The second method is GetSalt(). It is used to create a 32 bytes size of random characters. The method GetSalt() does not take any parameter and returns a string.

```
1 reference
private static string GetSalt()
{
    var cryptoProvider = RandomNumberGenerator.Create();
    byte[] b_salt = new byte[SaltByteSize];
    cryptoProvider.GetBytes(b_salt);
    return Convert.ToBase64String(b_salt);
}

3 references
public static string GetPasswordHash(string password)
{
    string salt = GetSalt();

    byte[] saltBytes = Convert.FromBase64String(salt);
    byte[] derived;

    using (var pbkdf2 = new Rfc2898DeriveBytes(
        password,
        saltBytes,
        Iterations,
        HashAlgorithmName.SHA256))
    {
        derived = pbkdf2.GetBytes(HashByteSize);
    }

    return string.Format("{0}:{1}:{2}", Iterations, Convert.ToBase64String(derived), Convert.ToBase64String(saltBytes));
}
```

Figure 17: C# source code for PBKDF2 hashing process

The PBKDF2 algorithm begins with assigning GetSalt() output to a salt variable. Then, the salt is converted from Base 64 string to 8-bit unsigned integers. Then, the Rfc2892DeriveBytes object is instantiated and takes the user plaintext password, salt, 310000 number of iterations, and the name of chosen KDF function. In this case, SHA256 is chosen. Then, the algorithm finishes by returning a string that is a concatenate of iterations, PBKDF2 hash, and salt in hex string. Figure 18 shows example of user hashed password in the database.

Hash
310000:8sFCww1v9NVLzBZKil/CabUu3wRDNqORXSpvej2IA4M=:Q8vKDezOV3KbhDN2e+lwd1f8xT5HCYQOvUawmVjciFo=
310000:OGpHJnp2KKk42/wximyX9CZsBK974HVME+QyIT0d9iQ=:4ibUy/oRMXv7LplydiDCfT3eztFMWBzwPSH9IJCwvp0=

Figure 18: User hashed password in the database

4.7 Security Functional Testing

The security test plan was used to ensure that all security features are implemented correctly. Table 3 shows the security test plan result.

Table 3: Security Test Plan Result

No	Check List	Result
1	Passwords must consist of alphabets (lowercase and uppercase), numbers, special characters, and minimum length of 10 characters.	Pass / ~~Fail~~
2	Passwords are obscured in the input box.	Pass / ~~Fail~~
3	Ensure users can only access the page based on their role.	Pass / ~~Fail~~
4	Passwords in the database are hashed with PBKDF2 algorithm.	Pass / ~~Fail~~
5	Random salt is added to the password before hashed with PBKDF2 algorithm	Pass / ~~Fail~~

4.8 User Acceptance Testing

The user acceptance form collects data from users by using Google Form. There are three sections in the Google Form which are Section A, Section B and Section C. Section A collects the respondents' information such as name, email, and job scope. Section B collects the testing result for the respondents. Lastly, Section C collects comments from the respondents regarding the system. Table 4 shows Section B questions in in the Google Form.

Table 4: User Acceptance Testing

No	Question	Result
1	Users may read, update, and delete their own user profile and organization profile information.	Yes /~~No~~
2	Users may update their own password in the profile section.	Yes /~~No~~
3	Only users with administrator roles may create, read, update, and delete new user, user group and organization.	Yes /~~No~~
4	Users may create, read, update, and delete hotel information which includes hotel name, country, city name, room name and more.	Yes /~~No~~
5	Users may create, read, update, and delete room rate plan which includes pricing, room rate, and inclusion details for a room.	Yes /~~No~~
6	Users may read and update customers' hotel booking.	Yes /~~No~~
7	Users may export the filtered hotel sales report results into Microsoft Excel and PDF format.	Yes /~~No~~

There are 10 respondents for the user acceptance form. All respondents are the final year students from the faculty of computer science and technology in Universiti Tun Hussein Onn Malaysia. Figure 19 shows their response for the seven questions in Section B. Based on the chart, all the respondents choose "Yes" for all questions. The results proved that the system passes the test.

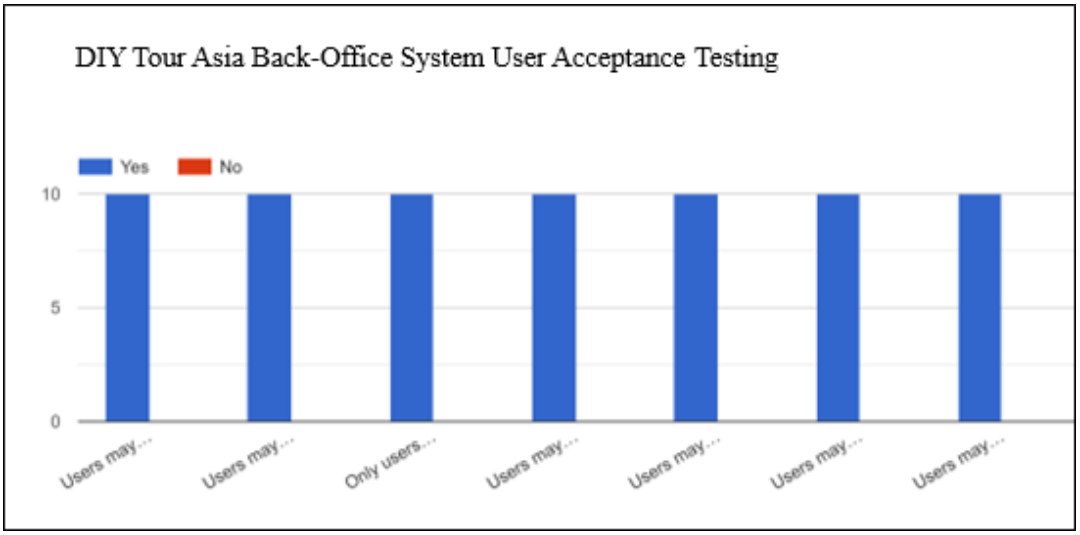

Figure 19: DIY Tour Asia Back-Office System User Acceptance Testing Chart.

For the suggestions and improvements in Section C, four of the respondents give the same suggestions. Two respondents suggest implementing a session timeout that logs user out when the session is idle for a long time. Two respondents suggest enforcing two factor authentication to strengthen the security of the user accounts.

5. Conclusion

The transformation of the back-office system from MVC to Blazor using secure password PBKDF2 is a viable project. With thorough planning, the project can complete within the allocated time. The Blazor framework makes it easier to improve or implement new modules in the future. The proposed system, DIY Tour Asia Back-Office System has several advantages. The system hashed users' password with PBKDF2 algorithm and enforces strong password requirement for user account. Moreover, the system implements parameterized query for every interaction with the database and validates all user input which includes length validation, pattern validation and null or whitespace validation. Lastly, it implements loosely coupled components for easy code maintenance. However, the proposed system also has several disadvantages. The system does not notify the user for CRUD every action taken and does not destroy the token if user is idle for a long time. During the deployment of the system on the company's server. The two-factor authentication is disabled due to having to set up a mail server. Moreover, the system does not notify or warn the user before a CRUD interaction with the database is taken. This is intuitive and may leads to human error because user might accidentally delete data from the database. For future improvement, the two-factor authentication may be enabled after setting up a mail server for the company itself. Lastly, the notification may be added using some notifications components so that the system can prevent more human errors.

Acknowledgement

This research project was supported by Universiti Tun Hussien Onn and Alpha Red Solutions Sdn. Bhd. through sepadan Re-Sip M078.

References

[1] E. Sandberg, "Evaluating Blazor : A comparative examination of a web framework." p. 37, 2021.

[2] N. A. Azeez and O. J. Chinazo, "Achieving data authentication with hmac-sha256 algorithm.," vol. 2, no. April, pp. 34–43, 2019.

[3] B. Kaliski, “PKCS \#5: Password-Based Cryptography Specification Version 2.0,” no. 2898. RFC Editor, Sep. 2000, doi: 10.17487/RFC2898.

[4] A. R. Hevner, “Object-Oriented System Development Methods,” *Adv. Comput.*, vol. 35, no. C, pp. 135–198, 1992, doi: 10.1016/S0065-2458(08)60595-1.

[5] M. Ramadhan, "Blazor Project Structure: Blazor Server vs. Blazor WebAssembly", Medium, 2021. [Online]. Available: https://medium.com/informatics/blazor-project-structure-e91a7c48ce1b. [Accessed: 27- Dec- 2021].

[6] S. Kılıçarslan, "ASP.NET Core MVC Web Application (Project Structure)", Medium, 2020. [Online]. Available: https://medium.com/net-core/asp-net-core-mvc-web-application-project-structure-3ccaa244fa66. [Accessed: 27- Dec- 2021].

[7] Kılıçarslan, S. (2020). ASP.NET Core MVC Web Application (Project Structure). Medium. Retrieved 27 December 2021, from https://medium.com/net-core/asp-net-core-mvc-web-application-project-structure-3ccaa244fa66.

[8] Ramadhan, M. (2021). Blazor Project Structure: Blazor Server vs. Blazor WebAssembly. Medium. Retrieved 27 December 2021, from https://medium.com/informatics/blazor-project-structure-e91a7c48ce1b.