Temasek Polytechnic

School of Informatics and IT

**Diploma in Information Technology (IT)**

Software Test Specifications (STS)

**Project Particulars**

|  |  |
| --- | --- |
| **Tutor** | Mel Goh |
| **Class** | P02 |
| **Project Title** | Delonix Regia Hotel Management System |

**Project Team’s Particulars**

|  |  |
| --- | --- |
| **Matric Number** | **Student Name** |
| **1502698B** | **Gideon Ler** |
| **1506489B** | **Tan Hai kang** |
| **1500632C** | **Eugene Sim** |
| **1502041E** | **Brandon Chew** |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 5/30/2017 | 1.1 | Designing of the system | Gideon  Hai Kang  Brandon  Eugene |
| 1/6/2017 | 1.2 | Building of the program | Gideon Hai Kang Brandon  Eugene |
| 2/6/2017 | 1.3 | Integrating the software | Gideon  Hai Kang  Brandon  Eugene |
| 5/6/2017 | 1.4 | Doing system testings and test log on the software | Gideon  Hai Kang  Brandon  Eugene |

Contents

[1. Distribution of Workload 4](#_Toc484504223)

[2. Module development and Unit testing 5](#_Toc484504224)

[2.1. Staff management Module 5](#_Toc484504225)

[2.1.1 Module development 5](#_Toc484504226)

[2.1.2 Unit Testing 5](#_Toc484504227)

[2.1.3 Ensure quality testing/development 6](#_Toc484504228)

[2.1.4 Other testing methods 6](#_Toc484504229)

[2.2. Booking and Room availability Module 7](#_Toc484504230)

[2.2.1. Module development 7](#_Toc484504231)

[2.2.2 Unit Testing 7](#_Toc484504232)

[2.2.3 Ensure quality testing/development 8](#_Toc484504233)

[2.3. Login, user management and room occupancy 8](#_Toc484504234)

[2.3.1 Module development 8](#_Toc484504235)

[2.3.2 Unit Testing 8](#_Toc484504236)

[2.3.3 Ensure quality testing / development 11](#_Toc484504237)

[2.3.4 Other testing method 11](#_Toc484504238)

[2.4. Guest management Module 12](#_Toc484504239)

[2.4.1 Module Development Process 12](#_Toc484504240)

[2.4.2 Unit Testing 12](#_Toc484504241)

[2.4.3 Ensure quality testing / development 12](#_Toc484504242)

[3. System Integration 13](#_Toc484504243)

[4. Test Log 14](#_Toc484504244)

[4.1. Staff management module 14](#_Toc484504245)

[Flow of events 14](#_Toc484504246)

[Scenario: 17](#_Toc484504247)

[Test case: 17](#_Toc484504248)

[Test data: 19](#_Toc484504249)

[4.3. Booking and Room availability Module 21](#_Toc484504250)

[4.3.1. Flow of events - Adding new reservation 21](#_Toc484504251)

[Identify Test case: 22](#_Toc484504252)

[Identify Test Data: 22](#_Toc484504253)

[4.3.2. Flow of events - Update reservation 23](#_Toc484504254)

[Identify Test case: 23](#_Toc484504255)

[Identify Test Data: 24](#_Toc484504256)

[4.3.3. Flow of events - Delete reservation 24](#_Toc484504257)

[Identify Test case: 24](#_Toc484504258)

[Identify Test Data: 25](#_Toc484504259)

[4.4. Login and User management Module 25](#_Toc484504260)

[4.5. Guest management Module 30](#_Toc484504261)

# 1. Distribution of Workload

|  |  |
| --- | --- |
| **Construction & Testing** | **Members** |
| * Staff Management module * Report module (Housekeeping schedule) * Module development and unit testing * System integration * Test log (Staff management) | Gideon Ler |
| * Booking and room availability module * Reporting module (Room status) * Module development and unit testing * System integration * Test log (Booking and room avaliability) | Tan Hai Kang |
| * Guest management module * Report Module (Guests per room) * Report Module (All Guests) * Module development and unit testing * Test log (Guest management) | Brandon Chew |
| * User management module * Login/Registration module * Report module (Room occupancy) * Module development and unit testing | Eugene Sim |

# 

# 

# 

# 

# 

# 2. Module development and Unit testing

## 2.1. Staff management Module

### 2.1.1 Module development

For the staff management module, the main functions are CRUD. Create, read, update and delete of staff records, viewing of existing staff and searching for existing staff through the search box by the Staff ID. Therefore, before developing the main idea is the have the ability to do CRUD functions while being able to search for the staffs by their ID. Additionally at the report section, I have done a housekeeping schedule report with its main purpose is just to display the staff names and their allocated duties to Mr Wang to see. The duties are room maintenance, security, general maintenance and estate maintenance.

The design should be easy to use for the admin, and the words have to be in bold. The buttons to update and delete which are used to input information are all within 1 page, while the create button would prompt a new form to pop up for the admin to perform these functions. The database would then be displayed at the bottom in the form of a data grid view. As for the housekeeping schedule report, the report is meant to display the staff’s duties daily, based on their specific duties.

### 2.1.2 Unit Testing

Unit testing refers to the testing done on a single unit of code, and it is normally done by the developers. Its purpose is to validate that each unit of the software, with a few inputs and a single output. Some advantages of unit testing include saving development time and it also makes it easier to change and refactor codes. Below are some of the unit tests cases that I have done for the Staff management module.

1. Test if the create can submit empty values
2. Test if the update can function if user does not select the rows they wish to update
3. Test if the can function if user clicks delete without selecting any rows
4. Test if can function if user clicks search without typing anything in the search box
5. Test if it can function if able to create/update even when the contact numbers and postal code values are not in integers
6. Test if Create works
7. Test if update works
8. Test if delete works

Here are the following expected results:

1. Message appear saying “(value) is required”, process stops
2. Message appear saying “Double click on the existing rows you wish to update”, process stops
3. Message appear saying “Double click on the existing rows you wish to delete”, process stops
4. Message appear saying “Please input the Staff\_ID”, process stop.
5. Message appear saying “Contact number should be in integers”, and “Postal code should be in integers”, process stops
6. Message appear saying “Created successful”, process completed
7. Message appear saying “Update successful”, process completed
8. Message appear saying “Delete successful”, process completed

### 2.1.3 Ensure quality testing/development

To ensure quality testing and development, I have chosen to test the software using specific input data to test. This consists of 3 types of data, normal, illegal and abnormal data. For example, in this test case, “Test if able to search in the staff ID search function without entering in the staff ID”, here are my datatypes i have used.

Normal data: Staff ID (2041, 2042, 2043)

Abnormal data: Staff ID (3000, 3001, 3002)

Illegal data: Staff ID (Three thousand, Three thousand and one)

Illegal data is tested to ensure that the program is able to handle the data exceptions without crashing, abnormal data is used to ensure that the test conditions and test programs are done correctly, and normal data is the expected datatype of the system. Hence, specific input data’s help to ensure that the software is able handle such exceptions and inputs without crashing.

Other ways to ensure quality testing and development include testing individual components constantly, testing the integrated components constantly, incremental testing and making sure that the codes are able to run on the same environment and having developers constantly communicate with each other and leaving comments on their codes, to ensure everyone is on the same page.

### 2.1.4 Other testing methods

Black box testing-

This testing method is used to test the functionalities of the software; hence the source code of the software is not needed. The name is called black box as just like a black box, testers are unable to see inside of the program. The way this testing works is testers are to give the input and output of the system to see if the specifications are followed. For example, if the tester is trying to find out if it can delete the data values from the table without selecting any rows to delete. The tester would then provide the input (click delete) and see what is the outcome of the input.

This testing case is useful as it tries to find the errors of the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

## 2.2. Booking and Room availability Module

### 2.2.1. Module development

The purpose of booking and room availability module is to check the room availability in the hotel, the number of empty and occupy room in the hotel. Other functions are to create a reservation, this function involve the use of CRUD, Create, read, update and delete. The create are essentially for the software user to create a booking detail which consist of the customer detail and room detail. As for Read, it allow the software user to view the record of booking which can be done by entering the booking id at the searchbox. The Update button allow the software user to modify the booking detail, this is important as it is common for customer to request changing room. Lastly the delete feature allow the deletion of record as cancellation happens.

### 2.2.2 Unit Testing

Unit testing refers to the practice of testing certain functions or area of our code. This gives us to validate that our functions work as what we expected. For example the create function, when given a set of inputs, we can determine if the function is working correctly and is handling the failures when executed. The main objective is to identify failures in our logic to help improve the quality of the code that of a function.

Below are some of the unit tests cases that I have done for the Booking and room availability module: -

1. Submit with empty value.
2. Type alphabet on field with int datatype
3. Checkin date > checkout date

### 2.2.3 Ensure quality testing/development

The following are some of the guideline we followed to ensure quality testing/development: -

* **Easy to write**. As developers have to write lots of unit test for different cases for application behavior, hence, the code should be kept simple for test routines to be easier.
* **Readable**. The unit test should be clear. Quality unit test are able to show the behavioral aspect of the application, hence it should be clear and easy to understand on which scenario is tested. If the fail test it will be easier to debug the error.
* **Unit test should be fast**. Developers write unit tests to repeatedly run and check for bug. If the process is slow, the chances of developer skip running them is higher. Slow unit tests may also indicate that either the system is environment-dependent.

## 2.3. Login, user management and room occupancy

### 2.3.1 Module development

The login module also ensures that each employee job type would login to a different homepage. This is to ensure employees account have limitation from the administrator account.

For the user management page, the create, retrieve, update, delete function has been implemented for the administrator to manage the user database through the form created. For the room occupancy function, its purpose is to display all particulars of guest living in all the rooms in the hotel.

### 2.3.2 Unit Testing

Unit testing is a test where individual unit of the software is being tested. This is to validate the software to perform as programmed. In this case of unit testing for login and user management, the white box testing is used to test the structure of the codes. For this white box testing, test will be carried out on the function capability to Create, Retrieve, Update and Delete of users in the database and to login into the program with the correct credentials in the database. These are the the test cases created:

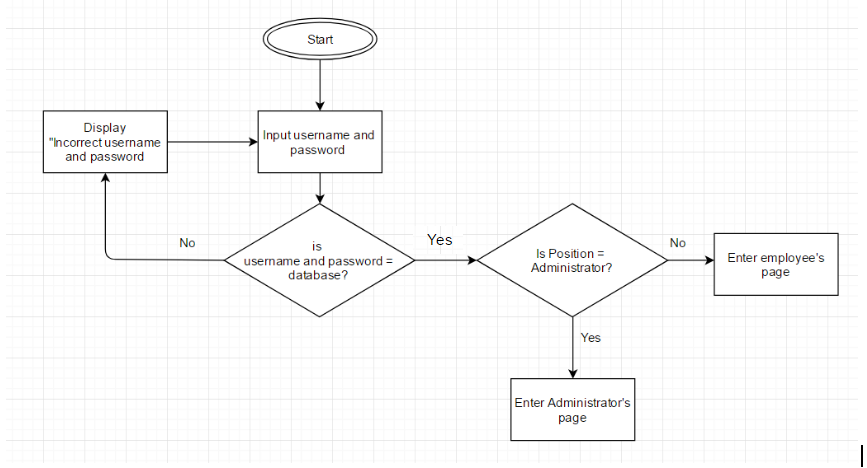
Test cases for login

1. User with specific role login into their designated homepage
2. Login return “Incorrect username and password” if credentials are incorrect.

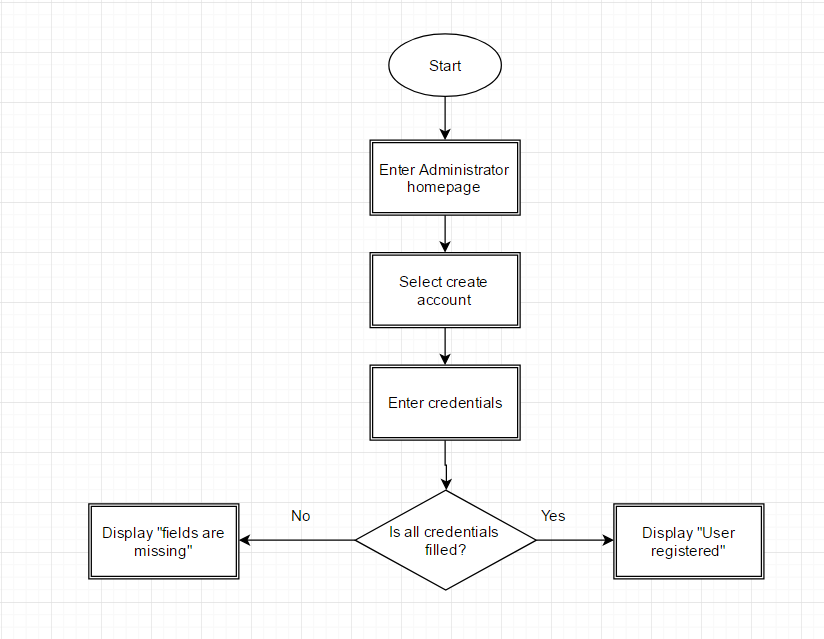
Test cases for user management

1. Search function able to return values based on combobox given.
2. Update able to display "Update date is successful” and update database.
3. Delete function able to remove data selected on the data grid view.
4. Back button able to return user to the previous page.

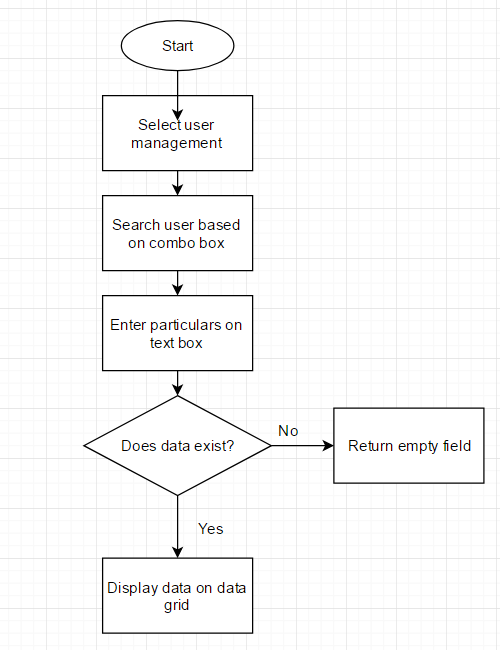
Flow diagram for login



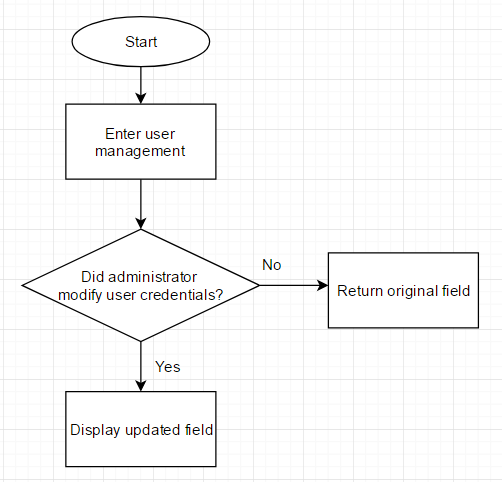
Flow diagram for account creation



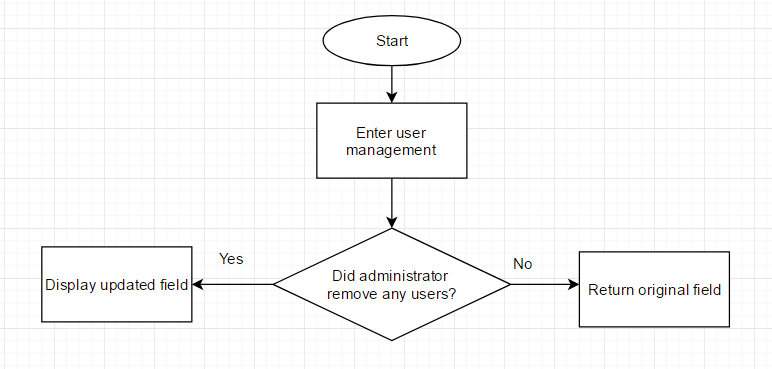
Flow diagram for search



Flow diagram for update



Flow diagram for delete



### 2.3.3 Ensure quality testing / development

To ensure the quality of the module, i have managed to implement the search update and delete into a single form to prevent user from wasting time on exiting and clicking. With bigger font size for buttons and labels, users would not feel irritated when using the program. The search function also allows the administrator to search by the user’s id, first name, last name, username or position. This is to ensure efficiency when looking for a certain user.

### 2.3.4 Other testing method

## 

## 

## 2.4. Guest management Module

### 2.4.1 Module Development Process

The guest management module is mainly on CRUD and finding out which guest is staying in which room number. Therefore, before developing the main idea is the have the ability to Create, Read, Update and Delete while being able to search for guest base on the room number that they are staying in.

With that said, the design is to make it easy to use for the staff members. Therefore, the buttons to create, update and delete and fields that are used to input information are all within 1 page. This is such that it is compact and simple to use.

### 2.4.2 Unit Testing

For my unit tests, they are done after each function is created. Different variables are inputted in order to find out bugs within the system. Some bugs are able to be removed while a few are left behind as I am unsure on how to remove them. Unit testing is done after each function to ensure that each functions are working properly and no collision of errors is done in between functions.

### 2.4.3 Ensure quality testing / development

In order to ensure quality testing and developments, fonts are made bigger and buttons are made bigger for simple to use purposes. Not only that, an agreed naming convention is set to ensure development are standardised.

This allows groupmates to understand each other code to aid in debugging if necessary.

# 

# 

# 3. System Integration

System integration(SI) refer to the process of combining components of sub-system into one system. It ensures that each integrated subsystem functions as required.

To ensure sub-system developed from different team member to be integrated during the integration phase, each of us had did an unit testing individually. The purpose is to validate that each function of the software performs as designed before we integrate the system.

We use the integration method known as Common Data Format. This approach allow us to integrate our system without the need of having the adapter to convert every application format. Systems using this method set a common or application-independent format, or they provide a service that does the transformation to or from one application into the common application.

After we combined our sub-component into one system, we did a integration testing. The purpose is to test the interfaces amongst the different units to ensure that the design/implementation of the interfaces amongst the units was done properly. It is also done to test if the integration has resulted in any error. For example, run time error, logical error, etc..

For our integration testing, we decided to use incremental testing. Incremental testing is a way of integration testing in which first you test each module of the software individually then continue testing by adding another module to it then another.

The following are benefits of using incremental testing: -

* Another extreme is that all programmers are integrated one by one, and a test is carried out after each step.
* The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.

Here is how we conducted incremental testing :

1. The staff management, booking, user management and guest management are tested individually for any errors.
2. Next, since booking and guest management are somewhat linked together, both modules are tested to see if it works. If the guest management is unable to integrate with the booking, than it will be debugged for errors and tested until both have no errors.
3. The User management is then tested together with the guest and booking to see if it is able to be integrated together. Likewise for step 2, if it is unable to integrate , than it will be debugged for errors and tested until both have no errors.
4. The staff management is tested together with step 3, and if it is unable to integrated it would be debugged for errors and tested until both have no errors.
5. Finally, the report modules are added and tested for errors, until they are error free.

# 4. Test Log

## 4.1. Staff management module

## **Flow of events**

For the staff management module, I would be testing the entire module as a whole, which includes the create, update, delete, view and search part of the module. I would also be testing on a few exception flows which I feel is important such as if the user tries to create a new staff without filling in the entire staff details, and ect.

**(Basic flow)**

1.0: View list of staff

2.0: Add new staff

3.0: Update staff information

4.0: Delete staff

5.0: View updated information

6.0: Search for staff

**(Alternate flow)**

7.0: Help

8.0: Quit

**(Exception flow)**

2.1: Uncompleted create

2.2: Postal code and Contact number input non-integer for create

3.1: Uncompleted update

3.2: Postal code and Contact number input non-integer for update

3.3: Click update without selecting any rows/clicking once on the row

4.1: Click delete without selecting any rows/clicking once on the row

6.1: Search input non-staff ID in search bar

6.2: Click search button while search is empty

* **Basic flow:**

Step 1.0: View existing staff information

The use case starts when the administrator wishes to view the list of staff and their personal information, and views the data grid view of the staff management page.

The system automatically displays the list of staff information in the data grid view, showing the staff’s first name, last name, NRIC, DOB, Email, contact number, country, address, postal code, duty, salary, bank account number and staff ID.

Step 2.0: Add new staff

The system prompts a new windows form to appear, displaying empty text boxes which indicates the first name, last name, NRIC, DOB, email, contact number, country, address, postal code, duty, salary and bank account number.

The admin proceeds to fill in all the relevant details into the textboxes, and saves the information once they are done adding.

Step 3.0: Update staff information

The admin double clicks on the row in the database they wish to update, and existing staff information would be shown in the text boxes on the homepage of the system, appearing in their respective boxes.

The admin chooses which data to update from the box, before updating and saving the changes.

Step 4.0: Delete staff

The admin double clicks on the on the row in the database they wish to delete.

The system prompts a message saying “Are you sure you want to delete this record”. Admin select yes and the record in the database is deleted.

Step 5.0: View updated information

The admin clicks on the refresh button.

The system would show the updated information in the database.

Step 6.0: Search for staff

The admin types in the search box for the staff ID.

The system shows only the staff record belonging to the staff with that ID.

* **Alternate flow:**

Step 7.0: Help

The system prompts a new form to pop up, and some FAQ questions and answers would pop up. It also displays the contact number of the IT support.

Step 8.0: Quit

The admin clicks on the home button of the page, and the system would bring the user to the home page of the hotel management system.

* **Exception flow:**

Step 2.1: Uncompleted create

In the create (Step 2.0 of the basic flow) if the admin tries click on submit while creating even though it has not filled in all the values from any of the text box, the system prevents the user from submitting.

A message will appear saying “ (Box that is not filled in) is required”, in descending order.

Step 2.2: Postal code and Contact number input non-integer for create

In the create (Step 2.0 of the basic flow) if the admin tries to enter any non-integer into the contact number or postal code box, the system prevents the user from submitting.

A message will appear saying “Postal code should be in integers” or “Contact number should be in integers”.

Step 3.1: Uncompleted update

In the update (Step 3.0 of the basic flow) if the admin tries to update the row even though it has not filled in all the values from any of the text box, the system prevents the user from submitting.

A message will appear saying “Double click on the existing rows you wish to update/delete”.

Step 3.2: Postal code and Contact number input non-integer for update

In the update (Step 3.0 of the basic flow) if the admin tries to enter any non-integer into the contact number or postal code box, the system prevents the user from submitting.

A message will appear saying “Postal code should be in integers” or “Contact number should be in integers”.

Step 3.3: Click update without selecting any rows/clicking once on the row

In the update (Step 3.0 of the basic flow) if the admin clicks on the update button without double clicking on the rows they wish to update, the system prevents the user from submitting.

A message will appear saying “Double click on the existing rows you wish to update/delete”.

Step 4.1: Click delete without selecting any rows/clicking once on the row

In the delete (Step 3.0 of the basic flow) if the admin clicks on the delete button without double clicking on the rows they wish to update or just clicking once, the system prevents the user from submitting.

A message will appear saying “Double click on the existing rows you wish to update/delete”.

Step 6.1: Search input non-staff ID in search bar

In the search (Step 6.0 of the basic flow) if the admin types in a non integer value into the search bar, the system prevents the user from submitting.

A message will appear saying “Please enter the row ID”.

Step 6.2: Click search button while search is empty

In the search (Step 6.0 of the basic flow) if the admin clicks on the search button without typing in any values, the system prevents the user from submitting.

A message will appear saying “Please enter a value into the search box”.

## **Scenario:**

|  |  |  |
| --- | --- | --- |
| **Scenario Name** | **Starting Flow** | **Exception** |
| 1:Successful Creation | Basic flow |  |
| 2:Successful Delete | Basic flow |  |
| 3:Successful update | Basic flow |  |
| 4:Successful Search | Basic flow |  |
| 5: Uncomplete create | Basic flow |  |
| 6: Postal code and Contact number input non integer for create | Basic flow | Step 2.2 |
| 7:Uncompleted update | Basic flow | Step 3.1 |
| 8:Click update without selecting any rows/clicking once on the row | Basic flow | Step 3.3 |
| 9:Search input non staff ID in search bar | Basic flow | Step 6.1 |
| 10:Click search button while search is empty | Basic flow | Step 6.2 |

## **Test case:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Staff\_ID** | **Contact Number** | **PostalCode** | **Others datatypes**  **E.g**  **FirstName**  **LastName**  **Address**  **NRIC**  **DOB**  **Salary**  **Bank a/n**  **Country**  **Email**  **Duty** | **Expected Results** |
| SC1 | 1: Successful creation | Valid | Valid | Valid | Valid | Value confirmation and saved in database |
| SC2 | 2: Successful delete | Valid | Valid | Valid | Valid | Value confirmation and deleted from database |
| SC3 | 3: Successful update | Valid | Valid | Valid | Valid | Value confirmation and updated in database |
| SC4 | 4:Successful Search | Valid | Valid | Valid | Valid | Value confirmation and selected row shown |
| SC5 | 5: Uncomplete create | Valid | Invalid | Invalid | Invalid | Error message, case ends |
| SC6 | 6: Postal code and Contact number input non integer for create | Valid | Invalid | Invalid | N/A | Error message, go back to step 2.0 |
| SC7 | 7:Uncompleted update | Valid | Invalid | Invalid | Invalid | Error message, go back to step 3.0 |
| SC8 | 8:Click update without selecting any rows/clicking once on the row | Invalid | Invalid | Invalid | Invalid | Error message go back to step 3.0 |
| SC9 | 9:Search input non staff ID in search bar | Invalid | N/A | N/A | N/A | Error message, go back to step 6.0 |
| SC10 | 10:Click search button while search is empty | Invalid | Invalid | Invalid | Invalid | Error message, go back to step 6.0 |

## **Test data:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **Staff\_ID** | **Contact Number** | **PostalCode** | **Others datatypes**  **E.g**  **FirstName**  **LastName**  **Address**  **NRIC**  **DOB**  **Salary**  **Bank a/n**  **Country**  **Email**  **Duty** | **Expected Results** |
| SC1 | 1: Successful creation | 2041 | 91234532 | 435653 | Gideon  Ler  Bedok 123  S123455A  23-08-1998  $50,000  123-45-34  Singapore  [g@gmail.com](mailto:g@gmail.com)  Security | Value confirmation and saved in database |
| SC2 | 2: Successful delete | N/A | N/A | N/A | N/A | Value confirmation and deleted from database |
| SC3 | 3: Successful update | 2041 | 81244343 | 435653 | Gideon  Ler  Bedok 123  S123455A  23-08-1998  $50,000  123-45-34  Singapore  [g@gmail.com](mailto:g@gmail.com)  Security | Value confirmation and updated in database |
| SC4 | 4:Successful Search | 2041 | 91234532 | 435653 | Gideon  Ler  Bedok 123  S123455A  23-08-1998  $50,000  123-45-34  Singapore  [g@gmail.com](mailto:g@gmail.com)  Security | Value confirmation and selected row shown |
| SC5 | 5: Uncomplete create | 2041 | 91234532 | N/A | N/A | Error message,go back to step 2.0 |
| SC6 | 6: Postal code and Contact number input non integer for create | 2041 | john123 | john123 | Gideon  Ler  Bedok 123  S123455A  23-08-1998  $50,000  123-45-34  Singapore  [g@gmail.com](mailto:g@gmail.com)  Security | Error message, go back to step 2.0 |
| SC7 | 7:Uncompleted update | 2041 | N/A | N/A | N/A | Error message, go back to step 3.0 |
| SC8 | 8:Click update without selecting any rows/clicking once on the row | N/A | N/A | N/A | N/A | Error message go back to step 3.0 |
| SC9 | 9:Search input non staff ID in search bar | 5000 | N/A | N/A | N/A | Error message, go back to step 6.0 |
| SC10 | 10:Click search button while search is empty | N/A | N/A | N/A | N/A | Error message, go back to step 6.0 |

# 4.3. Booking and Room availability Module

### 4.3.1. Flow of events - Adding new reservation

* **Basic Flow**
* (Step 1) Logon
* (Step 2) Select “Manage customer”
* (Step 3) Enter the customer information
* (Step 4) Select “add”
* (Step 5) Display completed Customer information in DataGridView Table.
* (Step 6) Select “Back Button”
* (Step 7) Select “Manage Booking”
* (Step 8) Select the Customer ID, Room number from drop down box and select the checkin and checkout date.
* (Step 9) Select “Add”
* (Step 10) Display booking details on DataGridView table.
* **Alternative Flow**
* A1) Quit
* A2) Unfulfilled Prerequisites, no room size available.
* A3) Hotel full, no available room
* **Exception flow**
* Failure to create form, missing information.
* Computer breakdown

**Generate Scenario:**

|  |  |  |
| --- | --- | --- |
| **Scenario Name** | **Starting Flow** | **Alternate** |
| Scenario 1: Successfully created customer information | Basic flow |  |
| Scenario 2: User quit | Basic flow | A1 |
| Scenario 3: Unfulfilled Prerequisites, room size not available | Basic flow | A2 |
| Scenario 4: Hotel full, no room available | Basic flow | A3 |

## **Identify Test case:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **staff\_id** | **Staff**  **password** | **Room**  **Selected** | **Room**  **availability** | **Room size** | **Expected Results** |
| CC1 | 1. Successfully created | valid | valid | valid | valid | valid | Customer profile created and customer\_id generated |
| CC2 | 2. User quit | valid | valid | n/a | n/a | n/a | Login screen appear |
| CC3 | 3.Unable to add new customer, Unfulfilled  prerequisites | valid | valid | invalid | valid | n/a | Error message go back to 2 |
| CC4 | 4. Unable to add customer, no room available | vallid | valid | n/a | n/a | n/a | Error message go back to 2 |

## 

## **Identify Test Data:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **staff\_id** | **Staff**  **password** | **Room**  **Selected** | **Room**  **availability** | **Room size available** | **Expected Results** |
| CC1 | 1. Successfully created | admin | adminpassword | 130 | yes | yes | Customer profile created and customer\_id generated |
| CC2 | 2. User quit | admin | adminpassword | n/a | n/a | n/a | Login screen appear |
| CC3 | 3.Unable to add new customer, Unfulfilled  prerequisites | admin | adminpassword | 130 | Invalid | no | Error message go back to 2 |
| CC4 | 4. Unable to add customer, no room available | admin | adminpassword | 130 | no | no | Error message go back to 2 |

## 

### 4.3.2. Flow of events - Update reservation

* **Basic Flow**
* (Step 1) Logon
* (Step 2) Select “Manage booking”
* (Step 3) Double click the Row you would like to update from DataGridView table.
* (Step 4) Enter the new data.
* (Step 5) Select “Updated”
* (Step 6) Updated data will be displayed on DataGridView table.
* **Alternative Flow**
* A1) Quit
* A2) Original Data is not on the DataGridView table
* A3) Update fail.

**Generate Scenario:**

|  |  |  |
| --- | --- | --- |
| **Scenario Name** | **Starting Flow** | **Alternate** |
| Scenario 1: Successfully Updated booking information | Basic flow |  |
| Scenario 2: User quit | Basic flow | A1 |
| Scenario 3: Unfulfilled Prerequisites, unable to find data | Basic flow | A2 |
| Scenario 4: Update failed | Basic flow | A3 |

## **Identify Test case:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **staff\_id** | **Staff**  **password** | **Room**  **No.** | **Customer**  **ID** | **Expected Results** |
| CC1 | 1. Successfully Updated | valid | valid | valid | valid | Successfully updated booking details |
| CC2 | 2. User quit | valid | valid | n/a | n/a | Login screen appear |
| CC3 | 3.Unable to find room record | valid | valid | invalid | n/a | Error message go back to 2 |
| CC4 | 4. Unable to find Customer profile | vallid | valid | n/a | invalid | Error message go back to 2 |

## 

## **Identify Test Data:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **staff\_id** | **Staff**  **password** | **Room**  **No.**  **selected** | **Customer**  **ID**  **created** | **Expected Results** |
| CC1 | 1. Successfully Updated | admin | adminpasswod | 130 | valid | Successfully updated booking details |
| CC2 | 2. User quit | admin | adminpasswod | n/a | n/a | Login screen appear |
| CC3 | 3.Unable to update, No room record found | admin | adminpasswod | 500 | valid | Error message go back to 2 |
| CC4 | 4. Unable to find Customer profile | admin | adminpasswod | 130 | invalid | Error message go back to 2 |

## 

### 4.3.3. Flow of events - Delete reservation

* **Basic Flow**
* (Step 1) Logon
* (Step 2) Select “Manage booking”
* (Step 3) Double click the Row that you want to delete from DataGridview table
* (Step 4) Select “Delete”
* (Step 5) Select “Yes” for confirmation.
* (Step 6) Record removed from DataGridView table and db.
* **Alternative Flow**
* A1) Quit

**Generate Scenario:**

|  |  |  |
| --- | --- | --- |
| **Scenario Name** | **Starting Flow** | **Alternate** |
| Scenario 1: Successfully deleted booking information | Basic flow |  |
| Scenario 2: User quit | Basic flow | A1 |

## **Identify Test case:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **staff\_id** | **Staff**  **password** | **Booking ID**  **Selected** | **Expected Results** |
| CC1 | 1. Successfully created | valid | valid | valid | Record removed from db and view. |
| CC2 | 2. User quit | valid | valid | n/a | Login screen appear |

## 

## **Identify Test Data:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **staff\_id** | **Staff**  **password** | **Booking\_ID** | **Expected Results** |
| CC1 | 1. Successfully deleted | admin | adminpassword | 1 | Record removed from db and view |
| CC2 | 2. User quit | admin | adminpassword | n/a | Login screen appear |

## 

## 

# 4.4. Login and User management Module

Flow of event(User Management)

(Basic flow)

1.0 Create new user with role

2.0 Search for user based on (id, First name, Last Name, Username and Position)

3.0 Update user information

4.0 View updated user information

5.0 Delete user information

6.0 Login with user credentials

(Alternate flow)

1.0 Exit function

2.0 Back function

(Exception flow)

1.0 Search function will not return any value if user did not specify any condition

* Basic flow

Step 1.0: Create new user with role

Only Administrator position will have the authority to create another user. Once the account has been created, the program will display “User is registered”.

Step 2.0: Search for user based on (id, First name, Last Name, Username and Position)

Administrator will be able to search for users based on the employee’s id, First name, Last Name, Username and Position

Step 3.0: Update user information

Administrators can modify other user particulars on the field displayed on the spot. Once updates has been made, the program will display “update successful.”

Step 4.0: View updated user information

Once the database has been modified by the Administrator, the administrator will be able to view the updated data immediately.

Step 5.0: Delete user information

If an employee is no longer with the company, the administrator will be able to use this function to delete any employee’s data

Step 6.0: Login with user credentials

After creating a new user, the staff can login to the program with the given username and password

* Alternate flow

1.0: Exit function

When any user click this function, the program will be close.

2.0: Back function

When a user click the back button, the user will be directed back to the previous page.

* Exceptional flow

1.0: Search function will not return any value if user did not specify any condition

if the search box is left empty, the search function will display a blank data.

2.0: Login function will return “Username or Password is incorrect” if the user enter their credentials wrongly.

**Unit Test**

|  |  |  |
| --- | --- | --- |
| **Scenario name** | **Starting Flow** | **Alternate** |
| Scenario 1:  Successful account creation | Basic flow |  |
| Scenario 2:  Displaying of user data | Basic flow |  |
| Scenario 3  Updating of user data | Basic flow |  |
| Scenario 4  Deleting of user data | Basic flow |  |
| Scenario 5  Login new account | Basic flow |  |

**Test Cases**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Scenario** | **ID** | **First Name** | **Last Name** | **Username** | **Password** | **Position** | **Expected result** |
| UC1 | 1. Successful registration | valid | valid | valid | valid | valid | valid | Account created |
| UD1 | 2.  Display user by ID | valid | n/a | n/a | n/a | n/a | n/a | Account display by ID |
| UD2 | 3. Display user by First Name | n/a | Valid | n/a | n/a | n/a | n/a | Account display by First Name |
| UD3 | 4. Display user by Last Name | n/a | n/a | Valid | n/a | n/a | n/a | Account display by Last Name |
| UD4 | 5. Display user by Username | n/a | n/a | n/a | Valid | n/a | n/a | Account display by Username |
| UD5 | 6. Display user by Position | n/a | n/a | n/a | n/a | n/a | Valid | Account display by Position |
| UU1 | 7. Update user credentials | Valid | Valid | Valid | Valid | Valid | Valid | Account updated |
| UD1 | 8. Delete  User credentials | Valid | Valid | Valid | Valid | Valid | Valid | Account deleted |
| UL1 | 9. Login with new credentials | Valid | Valid | Valid | Valid | Valid | Valid | Enter homepage |

**Test data**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Scenario** | **ID** | **First Name** | **Last Name** | **Username** | **Password** | **Position** | **Expected result** |
| UR1 | 1. Successful registration | 1 | eugene | sim | eugene | 12345 | Administrator | Account created |
| UD1 | 2.  Display user by ID | 1 | n/a | n/a | n/a | n/a | n/a | Account display by ID |
| UD2 | 3. Display user by First Name | n/a | eugene | n/a | n/a | n/a | n/a | Account display by First Name |
| UD3 | 4. Display user by Last Name | n/a | n/a | sim | n/a | n/a | n/a | Account display by Last Name |
| UD4 | 5. Display user by Username | n/a | n/a | n/a | eugene123 | n/a | n/a | Account display by Username |
| UD5 | 6. Display user by Position | n/a | n/a | n/a | n/a | n/a | Administrator | Account display by Position |
| UU1 | 7. Update user credentials | 1 | eugene | sim | eugene1 | 12346 | Desk manager | Account updated based on column changed |
| UD1 | 8. Delete  User credentials | 1 | eugene | sim | eugene | 12345 | Administrator | Account deleted |

# 4.5. Guest management Module

Flow of event (Guest Management)

**(Basic flow)**

1.0: View list of guest

2.0: Create a new guest

3.0: Update a guest’s information

4.0: Delete a guest’s information

**(Alternate flow)**

5.0: Refresh the guest’s list

6.0: Search the guest’s list by Room No

7.0: Quit

**(Exception flow)**

1.1: Double Clicking a field cause errors

2.1: A field is missing it’s value

2.2: A field is the wrong data type eg: int > string

3.1: A field is missing it’s value

3.2: A field is the wrong data type eg: int > string

6.1: Searching a string

6.2: Searching while textbox is empty

6.3: Error while searching an incorrect room number

* Basic Flow
  + 1.0: List of guests will be retrieved from the database using connection string and will be displayed onto the grid datatable in the windows form. The grid datatable will show all details and information of the guests from their ID to their name,country to the Room Number that they are staying in.
  + 2.0: When creating a new guest, staff members will be required to fill in all the fields provided above the grid datatable. Fields that are compulsory to be filled are labelled with an asterisk ,\* , at the end.   
    After the fields are filled up, staff members can click on the create button to complete the creation. A notification will be displayed if creation is successful.
  + 3.0: When updating a guest, staff members are required to double click on the grid datatable the guest that they wish to update first. After updating the fields with the appropriate data types, staffs will be able to click on the Update button to complete the update. A notification will be displayed if update is successful. Click the refresh button to see the new changes.
  + 4.0: When deleting a guest, staff members are required to first double click on the grid datatable the guest that they wish to delete. After doing so, staffs will just need to click on the delete table and a notification will be displayed that the delete is successful. Click the refresh button to see the new changes.
* Alternate Flow
  + 5.0: staffs can click on the refresh button to reset any searches made or to refresh the grid datatable if changes were made to them.
  + 6.0: Staffs can input a room number and click on the search button to search for all guests with the same room number. This allows them to find guests within a room faster.
  + 7.0: This button is to exit the form and to continue on in another form.
* Exception Flow
  + 1.1: Double clicking a field in the grid datatable will cause the software to crash. This can be avoided by clicking on the whole row instead.
  + 2.1: If a field is missing while creating a new guest, an alert will be provided with the fields that are missing and letting staffs know that they are required to be filled.
  + 2.2: If a field is the wrong data type, an alert will be provided with the fields that are wrong and letting staffs know that they should be only in integers.
  + 3.1: If a field is missing while updating a guest, an alert will be provided with the fields that are missing and letting staffs know that they are required to be filled.
  + 3.2: If a field is the wrong data type, an alert will be provided with the fields that are wrong and letting staffs know that they should be only in integers.
  + 6.1: If the search button is clicked when the Room number is a string value, it will send a warning that it is the wrong data type.
  + 6.2: If the search button is clicked when the Room number is empty, it will reset the Grid datatable similar to clicking the Refresh button.
  + 6.3: If the room number inputted is not available, it will cause the system to crash.

Scenario:

|  |  |  |
| --- | --- | --- |
| **Scenario name** | **Starting Flow** | **Alternate** |
| Scenario 1:  Creating a new guest | Basic flow: 2.0 |  |
| Scenario 2:  Leaving a field empty on Update | Basic flow: 3.0 | 3.2 |
| Scenario 3:  Deleting a guest | Basic flow: 4.0 |  |

Test Case/ Scenario 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Input fields above the grid table with valid data | Guest\_ID: 1  Guest\_FirstName: Brandon  Guest\_LastName: Chew  Guest\_Age: 19  Guest\_DOB: 05/05/1998  Guest\_Address: Tampines  Guest\_Country: Singapore  Guest\_Postal:  527491  Guest\_Email:  Nil  RoomID:130 | N/A | N/A |  |
| 2 | Click on the Create Button | N/A | An Alert is shown that creation is successful | An Alert is shown that creation is successful |  |

Scenario 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Double click on a row in the grid table |  | N/A | N/A | The row clicked is the guest created in Scenario 1. |
| 2 | Change the value for postal code to a string | Postal Code: 527491 to  abc | An Alert is shown that it is the wrong datatype | An Alert is shown that it is the wrong datatype |  |

Scenario 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Double click on a row in the grid table |  | N/A | N/A | The row clicked is the guest created in Scenario 1. |
| 2 | Click on the Delete Button |  | An Alert is shown that deletion is successful | An Alert is shown that deletion is successful |  |