**Temasek Polytechnic**

**School of Informatics and IT**

**Diploma in Information Technology (IT)**

Software Test Specifications (STS)

**Project Particulars**

|  |  |
| --- | --- |
| **Tutor** | Mr. Qi Yutao |
| **Class** | C17P05 |
| **Project Title** | Delonix Regia Hotel Management System |

**Project Team’s Particulars**

|  |  |
| --- | --- |
| **Matric Number** | **Student Name** |
| 1705905F | Umar Bin Othman |
| 1706362I | Tan Peng Siong Jasper |
| 1702347D | Gerald Doh Ting Feng |
| 1702530E | Haemon barat |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 15/01/2019 | 1.0 | Initial planning | All members |
| 18/01/2019 | 1.1 | First revision | Umar, Haemon |
| 21/01/2019 | 1.2 | Second revision | All members |
| 01/02/2019 | 1.3 | Third revision | Jasper, Gerald |
| 10/02/2019 | 2.0 | Final revision | All members |

***Table of Contents***

[***1.***](#_17dp8vu)***DISTRIBUTION OF WORKLOAD*** *3*

[***2.***](#_30j0zll)***MODULE DEVELOPMENT AND UNIT TESTING*** *3*

[*2.1*](#_2et92p0) *Room Availability and Booking Module 3*

[*2.2*](#_tyjcwt) *Housekeeping and Staff Management Module 3*

*2.3 Reporting Module 4*

*2.4 User Account and Login Creation Module 5*

[***3***](#_3dy6vkm)***SYSTEM INTEGRATION 6***

[***4***](#_1t3h5sf)***TEST LOG 6***

[*4.1*](#_4d34og8) *Getting list of staff particulars and searching for staff particulars by ID 6*

[*4.2*](#_2s8eyo1) *Getting list of staff duty type and searching staff duty type by ID 7*

[*4.3*](#_2s8eyo1) *Navigating from “Generate Module” page to Login Page and back 8*

[*4.4*](#_2s8eyo1) *Generating a Report 9*

[*4.5 Change Report Generation Schedule*](#_2s8eyo1) *10*

[*4.6 Viewing Latest Reports*](#_2s8eyo1) *11*

[*4.7 Editing a Report*](#_2s8eyo1) *12*

[*4.8 Searching for a Report*](#_2s8eyo1) *12*

[*4.9 Create a new User Account*](#_2s8eyo1) *13*

# **DISTRIBUTION OF WORKLOAD**

|  |  |
| --- | --- |
| **Construction & Testing** | **Members** |
| Module Development and Unit Testing | Gerald, Jasper, Umar, Haemon |
| System Integration | Jasper, Gerald, Umar |
| Test Log | Gerald, Jasper, Umar, Haemon |
| Database Creation, Linking, and Seeding | Gerald, Jasper, Umar, Haemon |
| Configuration Management | Gerald |

# 

# 

# **MODULE DEVELOPMENT AND UNIT TESTING**

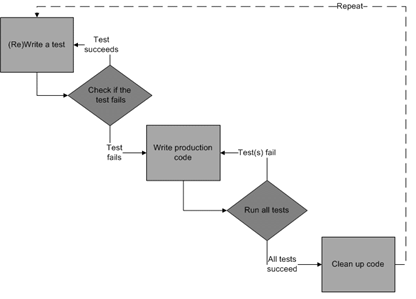
## 2.1 Room Availability and Booking Module

Approach

The method that will be used for this test will be similar to the waterfall model flow, and I will be using the test-driven development (TDD) method. Test-driven development is a software development process that relies on the repetition of a very short development cycle:

* Firstly, I will write a test cases that defines the function and the expected results
* After that the unit testing will be done by experienced testers to find errors or bugs
* Testers will make sure there is no connectivity between the modules and testing begins
* Finally, after the unit testing is completed, I will refactor new code to satisfactory standards if errors and bugs are found in the module

This approach can be defined as reiteration of steps shown in the picture given below.



1. I will state which feature have been applied

2. Create a test for the use cases

3. Compile, initiate tests and check if there is any error

4. Program the code, make it work

5. Restructure the existing codes

6. Rerun tests and fix it, if there are any failed tests

7. Execute code that has been refactor

8. Move on to another feature and repeat all the process from the beginning

Conducting the test with this approach will be cost and time efficient. After this process is completed, it will be integrated to the system with the other modules. Integration testing will commence before the software application is officially launched.

Testers

This module will be tested by experienced testers outsourced from a legitimate company, the testers will be following a list of test cases that I have created in the Room Availability and Booking Module test log section. This is to prevent biased opinion on the system module created giving it honest and purest results of the module tested, this will allow me to refactor (process of restructuring the codes) the given errors and bugs found in the module.

Test isolation

An important topic to take note of is the test isolation, this is to make sure that the unit testing are not mixed up with the other modules, in order to maintain integrity of the unit tested and the independency of the results during the test.

Testing of navigation

The navigation of this module will be tested throughout the entire software pages. This will consist of navigating from the page itself to other pages, such as the homepage/index, login page or the generate report page. When success is achieved on the navigating test, it is safe to proceed with other function tests.

Testing of the functions

Room availability and booking module testing will commence when the module prerequisites have been met. This is to prevent any unnecessary hurdle throughout the process, examples like putting the wrong information in the input field.

Another important process to take note of, is to do regression testing after every integration of a function, this can prevent integration defects and lessen risk exposure to bugs and errors to the complete software.

Integration of modules

When modules are fully tested and working as expected, the implementation process will commence. This is when this Room Availability and Booking Module will combine with the rest of the modules created for the hotel management system, before the integration happens it is important to take note if there are any duplicate functions or similar naming between the other modules.

Integration will be done through GitHub as it is easier for the team to change any of the codes if there is any problem surfacing during the integration, examples like duplicate functions etc.

## 2.2 Housekeeping and Staff Management module

* Testing of navigation

For the housekeeping and staff management module, the testing will began by testing the navigation to the housekeeping and staff management page starting from the homepage. Once the test result is successful the test will change from different starting pages as the objective is whether the user is able to navigate to the housekeeping and staff management page regardless of where the user is. Once the objective is completed, another test will began to test the authorization of the user. If the user is the receptionist staff, an error will be displayed that he/she does not have access to the module.

* Testing of the functions

The test prior conditions are that the user logged in must be either from the management or the owner, else the page will display an error stating that the user does not have authorised access to the page to use the function. The test will begin once all of the prior conditions are met. It starts by testing the function individually for any errors. After that it starts another test for validation by entering the wrong parameters in the function. This process repeats for the other functions individually.

-Integrating the functions

After successfully testing all the functions individually with no errors, the functions will then be combined together into bigger chunks and will be tested as a whole so if the integration of the functions will result in any errors. This is to ensure that the implementation of the interface of the functions are working successfully with no errors. Once the integrated testing of the functions is complete, the module is ready to be integrated with other modules for a complete system testing.

**2.3** **Reporting Module**

* Navigation

The main navigation elements will be the same for all the pages so the testing will generally consist of the same use cases with some exceptions for specific on-page content navigation.

The navigation for the reporting module would involve accessing the other module pages from the current page through links in the navigation bar and sidebar for black box testing and simple code reviews to check statement coverage regarding navigational elements such as ‘<a>’ tags.

The black box test will go through each of the links as use cases, and recording down the success rate for each test. However, as the use cases are so simple, there will not be any alternative flows for them to test.

Some links would require a pre-condition to be met before working, such as being logged in or having an account with greater privileges. Therefore each of the use cases that involve these links would have the tester meet the conditions before conducting the test to ensure validity of the results.

These tests would be conducted for all suitable navigation elements and the results would be recorded down and sent to the development team to improve upon the product.

* Functionality

The functions of the module consist of generating reports on demand (including editing and previewing before generation), changing the schedules of the report generation, as well as displaying the most recent reports. These functions are interacted with using user inputs and drop-down lists, with buttons being used for actuation. Therefore, there will be many possible results that the module will output depending on the user inputs entered.

For white box testing, since there is little to no conditional statements for this module’s functions, the focus will be on general statement coverage rather than branch coverage. The black box testing will involve techniques such as equivalence partitioning to reduce the number of use cases needed since many of the options entered would give similar outputs anyways. The tester would enter values into each of the user inputs and actuate the function, checking the output for success. This would be repeated for each partition until sufficient coverage is obtained.

Similar to the navigation testing, some pre-conditions would need to be met for the functions to perform normally, such as being logged in or the presence of certain privileges. Before conducting each use case, the conditions would be met by the tester.

The results would be recorded down for each use case and sent to the developers to implement any necessary fixes or changes.

* Implementation

When unit testing for the module is done, it will be integrated with the other modules as a single application and unit testing will be conducted on the application as a whole. Many of the previously mentioned techniques and instructions will be applied when testing the report module again, with greater emphasis on the navigation and integration aspects.

The results of the test are likewise recorded down and used as reference to fix issues with the system before it can be rolled out.

**2.4 User Account and Login Creation Module**

* Navigation

Navigation for the user account and login creation module were tested and developed by making sure all the different pages links that led to them properly worked. All the pages were also tested to navigate to and from each other.

* Functionality

Functionality was developed as per our software design specifications. Functions all had specific names so as to prevent clashing functions in the full integration. After a function was added, it was tested to see if it worked. I would then carry out regression testing to make sure no other functions were affected by the new one.

* Integration

Retest all navigation and functionality. Check ability to replicate exact results. Make sure functions from different modules do not have the same names or data, and do regression testing after each fix to see if anything else might have been affected.

# 3. **SYSTEM INTEGRATION**

# System integration is the process of combining the modules together into one complete system. This will be then followed by system testing to ensure that all the modules are functioning together without causing any errors or crashing the entire system. Once the system testing has been completed successfully, the system will be ready for deployment. Our teamwork system integration will be based on github as it is completely reliable and fullproof, and the process of the system integration is self explanatory.

To integrate all of the individual modules together, firstly we will have to implement navigation between each module through adding links in each module’s page.

The segregated modules will have to be connected in order to avoid crashes when launch. Considering that each module will utilise similar styling and functions that the team has agreed on using and implementing, all of them will source those elements from one external CSS and JavaScript file respectively. Similarly, they are all connected to a single database, from which they will access their data, with individual tables containing their corresponding data. Finally, these modules are kept together in a repository on GitHub for version control and configuration management.

# 4. **TEST LOG**

## Room Availability and Booking Module

**<Testing response functionality when navigating from pages**>

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | |  | TC0001 | **Test Case Description:** | | Check functionality response when navigating from Index/home page to login page and booking page | | | | | |  |
| **QA Tester Log** | |  |  |  |  |  |  |  |  |  |  |  |
| **Tester's Name** | | | This part will be left empty for the tester use | **Date Tested** | | This part will be left empty for the tester use | | **Test Case (Pass/Fail/Not Executed)** | | This part will be left empty for the tester use | |  |
| **Test Conditions** |  |  |  |  |  |  |  |  |  |  |  |  |
| S # | | Prerequisites: | | |  | S # | Test Data Requirement | | | | |  |
| 1 | | Nil | | |  | 1 | Nil | | | | |  |
| **Step #** | **Step Details** | | **Test data** | **Expected Results** | | **Actual Results** | | | **Pass / Fail / Not executed / Suspended** | | |  |
|  |
| 1 | From the home/index page user clicks on the “**Login**” button at the top right corner of the navigation bar | | - | User is redirected to the login page | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 2 | From the login page, user will click the “**Room Booking**” button at the sidebar navigation | | - | User is redirected to Room Booking page | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |

**<Testing response functionality when booking room>**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | |  | | TC0002 | **Test Case Description:** | | Check functionality response when booking a room | | | | | |  |
| **QA Tester Log** | |  | |  |  |  |  |  |  |  |  |  |  |
| **Tester's Name** | | | | This part will be left empty for the tester use | **Date Tested** | | This part will be left empty for the tester use | | **Test Case (Pass/Fail/Not Executed)** | | This part will be left empty for the tester use | |  |
| **Test Conditions** |  |  | |  |  |  |  |  |  |  |  |  |  |
| S # | | Prerequisites: | | | |  | S # | Test Data Requirement | | | | |  |
| 1 | | User is registered in the database as Staff or Admin | | | |  | 1 | Customer particulars | | | | |  |
| 2 | | User is logged in the Delonix regia application | | | |  | 2 | Country code | | | | |  |
| 3 | | User has valid data for input | | | |  | 3 | Phone number | | | | |  |
| 4 | | - | | | |  | 4 | Country of residence | | | | |  |
|  | | |  | | |  | 5 | Credit Card detail | | | | |  |
|  | | |  | 6 | No. of Adult or Child guests | | | | |  |
|  | | |  | 7 | Additional remarks | | | | |  |
|  | |  | |  |  |  |  |  |  |  |  |  |  |
|  |  |  | |  |  |  |  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | | **Test data** | **Expected Results** | | **Actual Results** | | | **Pass / Fail / Not executed / Suspended** | | |  |
|  |
| 1 | From the home/index page, user will click the “**Room Booking**” button at the sidebar navigation | | | **-** | User is redirected to Room Booking page | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 2 | User will “**choose room type**” from the drop-down list | | | **“Deluxe” room** | Selected room type “Deluxe” from the drop-down list will appear in the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 3 | User will Key in the “**First name**” | | | **Pete** | The name “Pete” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 4 | User will Key in the “**Last name**” | | | **Sakes** | The name “Sakes” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 5 | User will Key in the “**Date of birth**” | | | **10.10.1994** | The date of birth “10.10.1994” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 6 | User will Key in the “**Home address**” | | | **"Simei Ave 111"** | The home address “Simei Ave 111” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 7 | User will Key in the “**Phone Number**” | | | **“91823836”** | The phone number “91823836” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 8 | User will Key in the “**Additional Remarks**” | | | **“Nil”** | The additional remark “Nil” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 9 | User will Key in the “**No. of Child Guests**” | | | **“4”** | The no of child guests “4” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 10 | User will Key in the “**No. of Adult Guests**” | | | **“2”** | The no of adult “2” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 11 | User will Key in the “**Check-in Date**” | | | **“31/12/18”** | The check-in date “31/12/18” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 12 | User will Key in the “**Check-out Date**” | | | **“01/01/19”** | The check-out date “01/01/19” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 13 | User will “**choose Payment type**” from the drop-down list accordingly | | | **“Credit Card”** | Selected payment “Credit card” from the drop-down list will appear in the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 14 | User will Key in the “**Card Name**” | | | **“Pete Sakes”** | The card name “Pete Sakes” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
| 15 | User will Key in the “**Card Number**” | | | **“123456784321”** | The card number “123456784321” will appear inside the input field | | This part will be left empty for the tester use | | | This part will be left empty for the tester use | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**<Testing response functionality when search for a room by Id>**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | | TC0003 | Test Case Description: | | Check response functionality when searching for a room by Id at the navigation search bar | | | | | |  |
| **QA Tester Log** | | | | | | | | | | | |
| Tester's Name | | This part will be left empty for the tester use | Date Tested | | This part will be left empty for the tester use | | Test Case (Pass/Fail/Not Executed) | | This part will be left empty for the tester use | |  |
| **Test Conditions** |  |  |  |  |  |  |  |  |  |  |  |
| S # | Prerequisites: | | |  | S # | Test Data Requirement | | | | |  |
| 1 | User is registered and logged in the database as Staff or Admin | | |  | 1 | Room Id | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step #** | **Step Details** | **Test data** | **Expected Results** | **Actual Results** | **Pass / Fail / Not executed / Suspended** |  |
|  |
| 1 | User clicks on the the search bar at the top navigation bar | - | User is able to type in the input field | his part will be left empty for the tester use | This part will be left empty for the tester use |  |
| 2 | User will then type in the room id and search pressing enter | **07** | User is redirected to search page with outcome | This part will be left empty for the tester use | This part will be left empty for the tester use |  |

## Housekeeping and Staff Management Module

Prerequisites for testing both functions for the module: The user is logged in as either the management staff or the owner.

Testing of getting list of staff particulars and searching for staff particulars by ID

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Staff Records” button at the side navigation bar | - | System changes to the staff html page | - | - |
| 2 | System auto generate list of staff particulars and display them | - | System displays list of staff particulars | - | - |
| 3 | Enter Staff ID at search bar | Staff ID: 0001 | System displays value entered in the search bar | - | - |
| 4 | Click on the “Search” button at the navigation bar | - | System displays list of staff particulars by that ID | - | - |

## 

Alternate flow of getting list of staff particulars and searching for staff particulars by ID but user did not enter value in the search bar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Staff Records” button at the side navigation bar | - | System changes to the staff html page | - | - |
| 2 | System auto generate list of staff particulars and display them | - | System displays list of staff particulars | - | - |
| 3 | Enter Staff ID at search bar | Empty Test Data | System displays value entered in the search bar | - | - |
| 4 | Click on the “Search” button at the navigation bar | - | System displays error | - | - |

## 

Testing of getting list of staff duty type and searching staff duty type by ID

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Staff Duties” button at the side navigation bar | - | System changes to the staff duties html page | System changes to the staff duties html page | - |
| 2 | System auto generate list of staff duty types and display them | - | System displays list of staff duty types | System displays list of staff duty types | - |
| 3 | Enter Staff ID at search bar | Staff ID: 0001 | System displays value entered in the search bar | - | - |
| 4 | Click on the “Search” button at the navigation bar | - | System displays list of staff duty types by that ID | - | - |

## 

Alternate flow of getting list of staff duty type and searching for staff duty type by ID but user did not enter value in the search bar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Staff Duties” button at the side navigation bar | - | System changes to the staff duties html page | System changes to the staff duties html page | - |
| 2 | System auto generate list of staff duty types and display them | - | System displays list of staff duty types | System displays list of staff duty types | - |
| 3 | Enter Staff ID at search bar | Empty Test Data | System displays value entered in the search bar | - | - |
| 4 | Click on the “Search” button at the navigation bar | - | System displays error | - | - |

## 

## Report Module

Navigating from “Generate Module” page to Login Page and back

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Login” button at the navigation bar | - | The page changes to the login page | - | - |
| 2 | Click on the “Generate Report” button at the sidebar | - | The page changes to the reports page | - | - |

## 

## Generating a report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Generate Report” button at the sidebar | - | The page changes to the reports page | - | - |
| 2 | Choose the type of report to generate from the dropdown list at the “Generate a Report” section | “Total Room Occupancy - Statistics” | The chosen report type is selected | - | - |
| 3 | Enter a starting and ending entry date for the data to use in DD/MM/YY format | Starting: “31/12/18”  Ending: “01/01/19” | The entered data is shown in the user input | - | - |
| 4 | Choose a printer to use from the dropdown list | “Printer 1-C” | The chosen printer is selected | - | - |
| 5 | Click the “Print” button | - | An alert is shown on the page saying: “A report for Total room occupancy - Statistics containing data entered between 2018-12-31 and 2019-01-01 has been generated and is being printed at Printer 1-C” | - | - |
| Alternate flow 1 - Preview Report | | | | | |
| 5 | Click the “Preview” button | - | The report is shown in its entirety in full screen | - | - |
| Alternate flow 2 - Edit Report | | | | | |
| 5 | Click the “Edit” button | - | The page is changed to the edit page | - | - |
| 6 | Change the contents of the text area | New content: “Hello World!” | The test data appears in the text area | - | - |
| 7 | Click the “Save Changes” button | - | An alert appears saying “Your edits have been saved!” and the page changes back to the reports page | - | - |

Change report generation schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Generate Report” button at the sidebar | - | The page changes to the reports page | - | - |
| 2 | Choose the the report type whose schedule you want to change from the dropdown list at the “Report Generation Schedules” section | “Staff housekeeping duties and schedules” | The chosen report type is selected | - | - |
| 3 | Choose the regularity you want the report to be generated | “Monthly” | The chosen regularity is selected | - | - |
| 4 | Enter the number of copies of the report to be printed each time | “5” | The entered date is shown in the user input | - | - |
| 5 | Click the “Save Changes” button | - | An alert shows up saying “Change in generation schedule: 5 copies of reports regarding Staff housekeeping duties and schedules will be printed Monthly” | - | - |

Viewing Latest Reports

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Generate Report” button at the sidebar | - | The page changes to the reports page | - | - |
| 2 | Click the Report #xxxx link at the “Latest Reports” section | “Report #0001” | The report is shown in its entirety in full screen | - | - |

Edit a report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Edit Reports” button at the sidebar | - | The page changes to the reportsDB page | - | - |
| 2 | Click the card of the report you want to edit | “Report #0003” | The page changes to the edits page | - | - |
| 3 | Change the contents of the text area | New content: “Hello Again World!” | The test data appears in the text area | - | - |
| 4 | Click the “Save Changes” button | - | An alert appears saying “Your edits have been saved!” and the page changes back to the reports page | - | - |

Search for a report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| 1 | Click on the “Generate Report” or “Edit Reports” button at the sidebar | “Generate Report” | The page changes to the reports page | - | - |
| 2 | Enter a search query in the search input at the navigation bar | “Hello123” | The page changes to the search results page | - | - |

**User Account and Login Creation Module**

## Create New User Account (from the Account Creation page and logged in a Hotel Administrator account)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| Scenario 1: Creating an Account | | | | | |
| 1 | Enter in user details. | Name: “John Johnson”  Password: “Password123”  E-mail: “[jojohnson@gmail.com](mailto:jojohnson@gmail.com)”  Access level: ”End-User” | - |  | - |
| 2 | Click “Create Account”. | - | Details are accepted and account is created. |  | - |
| Scenario 2: Missing fields | | | | | |
| A1 | In step 1, do not input in a name. | Password: “Password123”  E-mail: “[jojohnson@gmail.com](mailto:jojohnson@gmail.com)”  Access level: ”End-User” | - |  | - |
| A2 | Click “Create Account”. | - | Error message informs user to input missing fields. |  | - |
| A3 | Repeat steps 1 and 2 without inputting a password. | Name: “John Johnson”  E-mail: “[jojohnson@gmail.com](mailto:jojohnson@gmail.com)”  Access level: ”End-User” | Error message informs user to input missing fields. |  | - |
| A4 | Repeat steps 1 and 2 without inputting an email. | Name: “John Johnson”  Password: “Password123”  Access level: ”End-User” | Error message informs user to input missing fields. |  | - |
| Scenario 3: Already Existing Email | | | | | |
| B1 | In step 1, input an already existing email. | Name: “John Johnson”  Password: “Password123”  E-mail: “wangwang[@gmail.com](mailto:jojohnson@gmail.com)”  Access level: ”End-User” | - |  | - |
| B2 | Click “Create Account”. | - | Error message informs user that the email address is already being used. |  | - |

## Edit a User Account (from the Edit Account Details page and logged in a Hotel Administrator account)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| Scenario 1: Changing an Name | | | | | |
| 1 | Select an account’s email address to edit. | E-mail: “wangwang@gmail.com” | The account’s details appear in the editing fields. |  | - |
| 2 | Edit the account’s name. | Name: “Johnson Wang” | - |  | - |
| 3 | Click “Save Changes”. | - | Changes are saved to the account. |  | - |
| Scenario 2: Resetting a Password | | | | | |
| A1 | After step 1, click the “Reset Password” checkbox button. | - | Checkbox is ticked. |  | - |
| A2 | Click “Save Changes”. | - | Password is reset with the new temporary password sent to the account’s email. |  | - |
| Scenario 3: Deleting an Account | | | | | |
| B1 | After step 1, click “Delete Account”. | - | Account is deleted. |  | - |

## Logging in a User Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S/No** | **Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Remarks** |
| Scenario 1: Valid Login | | | | | |
| 1 | Enter in user details. | E-mail: “johnsmith@gmail.com”  Password: “Password123” | - |  | - |
| 2 | Click “Log In”. | - | Account is logged in. |  | - |
| Scenario 2: Invalid Password | | | | | |
| A1 | At step 1, input an invalid password. | E-mail: “johnsmith@gmail.com”  Password: “Password124” | - |  | - |
| A2 | Click “Log In”. | - | Error message tells user that inputted details are wrong. |  | - |
| Scenario 3: Invalid Email | | | | | |
| B1 | At step 1, input an invalid email. | E-mail: “johnsmith1@gmail.com”  Password: “Password123” | - |  | - |
| B2 | Click “Log In”. | - | Error message tells user that inputted details are wrong. |  | - |
| Scenario 4: Missing Fields | | | | | |
| C1 | At step 1, do not enter in an email. | Password: “Password123” | - |  | - |
| C2 | Click “Log In”. | - | Error message indicating missing fields. |  | - |
| C3 | Repeat step 1, but do not enter in a password this time. | E-mail: “johnsmith@gmail.com” | - |  | - |
| C4 | Click “Log In”. | - | Error message indicating missing fields. |  | - |