A Real Time and Interactive Dashboard for Tourism Industry

**Test Plan**

By

Junyu Zhou 592115508

Yawei Li 592115518

Department of Software Engineering,

College of Arts, Media and Technology,

ChiangMai University

Project Advisor



Table of Contents

[1. Document History 5](#_Toc21463469)

[2. Introduction 7](#_Toc21463470)

[2.1 Objectives 7](#_Toc21463471)

[2.2 Scope 7](#_Toc21463472)

[2.3 Acronyms and Definitions 7](#_Toc21463473)

[2.3.1 Acronyms 7](#_Toc21463474)

[3. Test Plan and Test Procedure 10](#_Toc21463475)

[3.1 Scope of testing 10](#_Toc21463476)

[3.2 Test Duration 10](#_Toc21463477)

[3.4 Test Strategy 11](#_Toc21463478)

[3.5 Result of Testing 12](#_Toc21463479)

[3.6 Test Environment 12](#_Toc21463480)

[4. Unit Testing 13](#_Toc21463481)

[UTC-01: Front-end: Method name: onUsernameChanged (e) 13](#_Toc21463482)

[UTC-02: Front-end: Method name: onPasswordChanged (e) 14](#_Toc21463483)

[UTC-03: Front-end: Method name: submitLogin (e) 15](#_Toc21463484)

[UTC-04: Front-end: Method name: logout () 16](#_Toc21463485)

[UTC-05: Front-end: Method name: showSummary () 17](#_Toc21463486)

[UTC-06: Back-end: Method name: login () 21](#_Toc21463487)

[UTC-07: Back-end: Method name: admin\_login () 22](#_Toc21463488)

[UTC-08: Back-end: Method name: delete\_account (username) 23](#_Toc21463489)

[UTC-09: Back-end: Method name: register () 24](#_Toc21463490)

[UTC-10: Back-end: Method name: get\_all\_account () 25](#_Toc21463491)

[UTC-11: Back-end: Method name: edit\_username () 26](#_Toc21463492)

[UTC-12: Back-end: Method name: edit\_password () 27](#_Toc21463493)

[UTC-13: Back-end: Method name: approve\_user () 28](#_Toc21463494)

[UTC-14: Back-end: Method name: reject\_user () 29](#_Toc21463495)

[UTC-15: Back-end: Method name: get\_pending () 30](#_Toc21463496)

[UTC-16: Back-end: Method name: add\_user () 31](#_Toc21463497)

[UTC-17: Back-end: Method name: get\_account\_by\_username () 32](#_Toc21463498)

[5. System Testing 33](#_Toc21463499)

[5.1 Super admin 33](#_Toc21463500)

[STC-01: Super admin can login to the system and manage accounts. 33](#_Toc21463501)

[4.2 Admin 39](#_Toc21463502)

[STC-02: Admin can login or register account. 39](#_Toc21463503)

[STC-03: Admin can edit information 41](#_Toc21463504)

[STC-04: Admin can logout from system. 43](#_Toc21463505)

[STC-05: View the summary of data visualization result 44](#_Toc21463506)

[6. Appendix 46](#_Toc21463507)

[1. Account information 46](#_Toc21463508)

[2. Sparkline chart information 46](#_Toc21463509)

[3. Positive comments rate line chart information 47](#_Toc21463510)

[4. Word frequency bar chart information 48](#_Toc21463511)

[5. Number of comments bar chart information 48](#_Toc21463512)

[6. Types of comments stacked column chart information 49](#_Toc21463513)

# 1. Document History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| History | Status | Date | Viewable | Editable | Responsible |
| Project-Test plan\_v1.docx  Create:  - Introduction  - Test Plan and Test Procedure  - Unit Testing  - System Testing  - Reference  - Appendix | Draft | 23 July, 2019 | ZJY, LYW,  AJP | ZJY, LYW | ZJY, LYW |
| Project-Test plan\_v2.docx  Modify:  - Unit Testing  - System Testing  - Appendix | Final | 22  Aug,  2019 | ZJY, LYW,  AJP | ZJY, LYW | ZJY, LYW |

ZJY = Junyu Zhou

LYW = Yawei Li

AJP = Dr. Pree Thiengburanathum

# 2. Introduction

### 2.1 Objectives

The objective of the test plan of A Real Time and Interactive Dashboard in Tourism Industry is to establish test plan of the unit testing and system testing and make sure that the bugs or the defects are discovered and fixed. The unit testing covers all of implemented method in the A Real Time and Interactive Dashboard in Tourism Industry system. The system testing covers the user requirements.

### 2.2 Scope

This test plan describes the white box-testing and black-box testing are activities to defect the defects in the system and describes the system testing activities for testing a completely integrated system to verify that it meets the user requirements.

### 2.3 Acronyms and Definitions

### 2.3.1 Acronyms

URS = User Requirement Specification

UTC = Unit Test Case

STC = System Test Case

**2.3.2 Definitions**

|  |  |
| --- | --- |
| Name | Definition |
| Feature | Transformation of input parameters to output parameters based on a specified algorithm. It describes the functionality of the product in the language of the product. Used for requirements analysis, design, coding, testing or maintenance. [1] |
| Design | The period in the software life cycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements. [2] |
| IEEE | Institute for Electrical and Electronics Engineers. Biggest global interest group for engineers of different branches and computer scientists. [3] |
| Requirement | (1) A condition or capability needed by the user to solve a problem or achieve an objective.  (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. (3) A documented representation of a condition or capability as in definition (1) or (2). [4] |
| Specification | Precise description of an activity or work product that serves as the basis or input for further activities or work product. A specification can comprise requirements to a Product and how they will be solved. Different parts of a specification (e.g. what is to be done, how it will be done) must not be mixed. [5] |
| Unit testing | A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed [6] |
| System testing | A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. [7] |

# 3. Test Plan and Test Procedure

### 3.1 Scope of testing

A Real Time and Interactive Dashboard in Tourism Industry will be tested by white-box and black-box testing techniques that are unit testing and system testing and record the test result in the test record.

### 3.2 Test Duration

|  |  |
| --- | --- |
| Progress | Data and Duration |
| Progress Report I | Perform date:29 July,2019  Duration: 15 day |

**3.3 Test Responsibility**

|  |  |
| --- | --- |
| Item | Responsibility |
| Unit test of dashboard | ZJY, LYW |
| Record unit test of dashboard | ZJY, LYW |
| System test of dashboard | ZJY, LYW |
| Record system test of dashboard | ZJY, LYW |

### 3.4 Test Strategy

A Real Time and Interactive Dashboard in Tourism Industry test strategy will be following:  
1. Design test case for each feature.  
2. Prepare test data for each feature.  
3. Determine expected results.  
4. Perform testing on individual features.  
5. Result of testing will be a record.  
6. All test files will be store in the project repository.

### 3.5 Result of Testing

In the test record the test result will separate into two parts, which are:  
1. Actual output: The actual outputs that are performed by each test case. 2. Pass/Fail criteria:

2.1 Pass: the result of the actual result is same as expected result. 2.2 Fail: the result of the actual result is not same as expected result.

### 3.6 Test Environment

**3.6.1 Hardware**

**Computer:**

**MacBook Pro (Retina, 15-inch, Mid 2015)**

Processor: 2.2 GHz Intel Core i7

Memory: 16 GB 1600 MHz DDR3

OS: MacOS Catalina 0.15 Beta (19A501i)

**MacBook Pro MacBook Pro (15-inch, 2019)**

Processor: 2.3 GHz Intel Core i9

Memory: 32 GB 2400 MHz DDR4

OS: MacOS Mojave 10.14.6

**2.6.2 Software**

- Chrome version 75.0.3770.142 or later

# 4. Unit Testing

**[Progress I]**

### UTC-01: ****Front-end: Method name: onUsernameChanged (e)****

Description: This method will validate format of username.

Test Date: 23 Aug 2019

Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 1 | Input empty username | Username= | - |
| 2 | Input username with special characters | Username= admin;’] | Console displays (“Username cannot contain special characters!”) |
| 3 | Input username within 10 characters | Username= admin001 | Console displays (“Format correct”) |

### UTC-02: ****Front-end: Method name: onPasswordChanged (e)****

Description: This method will validate format of password.

Test Date: 23 Aug 2019

Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 4 | Input password less than 6 digits | Password= 1234 | Console displays (“Password must be more than 6 digits!”) |
| 5 | Input password with 6 digits | Password= admin001 | Console displays (“Format correct”) |
| 6 | Input empty password | Password= | - |

### UTC-03: ****Front-end: Method name: submitLogin (e)****

Description: This method will login to the dashboard.

Test Date: 23 Aug 2019

Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 7 | Inputs correct username and password | Username= admin001  Password= admin001 | Console displays (“login successful”) |
| 8 | Inputs incorrect username or password | Username= admin001  Password= admin000 | Console displays (“login failed”) |

### UTC-04: ****Front-end: Method name: logout ()****

Description: This method will log out from dashboard.

Test Date: 23 Aug 2019

Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 9 | Decision maker logs out from dashboard | - | Console displays(“Log out successfully”) |

### UTC-05: ****Front-end: Method name: showSummary ()****

Description: This method will display summary on the dashboard page.

Test Date: 23 Aug 2019

Test Data:

sparklineData = [47, 45, 54, 38, 56, 24, 65, 31, 37, 39, 62, 51, 35, 41, 35, 27, 93, 53, 61, 27, 54, 43, 19, 46]

expectedData2 = {

seriesSpark1: [{data: [25, 66, 41, 89, 63, 25, 44, 12, 36, 9, 54]}],

seriesSpark2: [{data: [47, 45, 74, 14, 56, 74, 14, 11, 7, 39, 82]}],

seriesSpark3: [{data: [12, 14, 2, 47, 42, 15, 47, 75, 65, 19, 14]}],

seriesSpark4: [{data: [15, 75, 47, 65, 14, 2, 41, 54, 4, 27, 15]}],

}

expectedData3 = {

total\_comments: '135,965',

positive\_comments: '99,821',

negative\_comments: '10,212',

neutral\_comments: '25,932',

percent\_total: '100%',

percent\_positive: '73.4%',

percent\_negative: '7.5%',

percent\_neutral: '19.1%',

}

expectedData4 = [

[ 1483203600000, 5], [ 1483808400000, 22],

[ 1484413200000, 52], [ 1485018000000, 87],

[ 1485622800000, 57], [ 1486227600000, 84],

[ 1486832400000, 87], [ 1487437200000, 43],

[ 1488042000000, 29], [ 1488646800000, 69],

[ 1489251600000, 81], [ 1489856400000, 46],

[ 1490461200000, 14], [ 1491066000000, 79],

[ 1491670800000, 9], [ 1492275600000, 51],

[ 1492880400000, 76], [ 1493485200000, 9],

[ 1494090000000, 51], [ 1494694800000, 77],

[ 1495299600000, 22], [ 1495904400000, 96],

[ 1496509200000, 18], [ 1497114000000, 67],

[ 1497718800000, 24], [ 1498323600000, 51],

[ 1498928400000, 41], [ 1499533200000, 83],

[ 1500138000000, 56], [ 1500742800000, 42],

[ 1501347600000, 76], [ 1501952400000, 20],

[ 1502557200000, 73], [ 1503162000000, 14],

[ 1503766800000, 67], [ 1504371600000, 8],

[ 1504976400000, 22], [ 1505581200000, 64],

[ 1506186000000, 44], [ 1506790800000, 1],

[ 1507395600000, 34], [ 1508000400000, 34],

[ 1508605200000, 70], [ 1509210000000, 16],

[ 1509814800000, 20], [ 1510419600000, 17],

[ 1511024400000, 86], [ 1511629200000, 78]

]

expectedData5 = [400, 430, 448, 470, 540, 580, 690, 1100, 1200, 1380]

expectedData6 = [{name: 'Neutral',

data: [44, 55, 57, 56, 61, 58, 63, 60, 66]}, {

name: 'Positive',

data: [76, 85, 101, 98, 87, 105, 91, 114, 94]}, {

name: 'Negative',

data: [35, 41, 36, 26, 45, 48, 52, 53, 41]}];

expectedData7 = [{name: 'Neutral',

data: [44, 55, 41, 67, 22, 43, 21, 49, 39]}, {

name: 'Positive',

data: [13, 23, 20, 8, 13, 27, 33, 12, 14]}, {

name: 'Negative',

data: [11, 17, 15, 15, 21, 14, 15, 13, 9]}];

Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 10 | Decision maker could view the summary page. | - | Console displays (“Summary is showing!”) |
| 11 | Decision maker could view the sparkline chart. | - | {sparklineData}  {expectedData2}  {expectedData3} |
| 12 | Decision maker could view the positive comments rate line chart. | - | {expectedData4} |
| 13 | Decision maker could view the word frequency bar chart. | - | {expectedData5} |
| 14 | Decision maker could view the number of comments bar chart. | - | {expectedData6} |
| 15 | Decision maker could view the types of comments stacked column chart. | - | {expectedData7} |

### [Progress II]

### UTC-06: ****Front-end: Method name: componentDidMount ()****

Description: This method is initial method when component render.

Test Date: 27 Sep 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 16 | Test when the browser accesses the view hotel page. | - | Console displays (“View component render!”) |

### UTC-07: ****Front-end: Method name: getAllHotel ()****

Description: This method is used to get all hotels.

Test Date: 27 Sep 2019

expectedData8 = {

"results": [{"count": 2,

"hotel": "Agriturismo Le Grotte"},

…

{"count": 282,

"hotel": "Park Hotel Villa Grazioli"}]}

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 17 | Test when connect to database. | - | expectedData8 |
| 18 | Test when database is empty | - | [] |
| 19 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-08: ****Front-end: Method name: getHotel ()****

Description: This method is used to get data of a hotel.

Test Date: 27 Sep 2019

expectedData9 = [

{"c\_id": 1627,"content": "The user only left a rating.","date": "2017-11-12", "rating": 50},{"c\_id": 1640, "content": "The user only left a rating.", "date": "2017-02-05","rating": 40}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 20 | Test when connect to database. | hotelname=” Agriturismo Le Grotte” | expectedData9 |
| 21 | Test when database is empty | - | [] |
| 22 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-09: ****Front-end: Method name: submit(e)****

Description: This method is used to submit a comment.

Test Date: 27 Sep 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 23 | Test when connect to database. | comment = {username: “user”, content: “good”, hotel: “Agriturismo Le Grotte”} | {“message”: “comment added”} |
| 24 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-10: ****Front-end: Method name: getAll ()****

Description: This method is used to get data of log information.

Test Date: 27 Sep 2019

expectedData10 = [

[{"content": "super admin logged in","creator": "super admin",= "log\_id": 1,"time":"2019-10-08 15:40:37","type": "Login"}, {"content": "super admin logged in","creator": "super admin","log\_id": 2, "time": "2019-10-08 15:40:52", "type":"Login"},{ "content": "super admin logged in","creator": "super admin","log\_id": 3,"time": "2019-10-08 15:40:57","type": "Login}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 25 | Test when connect to database. | - | expectedData10 |
| 26 | Test when database is empty | - | [] |
| 27 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### ****[Progress I]****

### UTC-11: ****Back-end: Method name: login ()****

Description: This method will receive http POST request “/users/login”

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 28 | Inputs correct username and password | Username= admin001  Password= admin001 | {“message": "Login successfully", "token": "eyJ…g34GY", "user": "admin001”} |
| 29 | Inputs incorrect username or password | Username= admin001  Password= admin000 | {“error": "Invalid username and password"} |

### UTC-12: ****Back-end: Method name: admin\_login ()****

Description: This method will receive http POST request “/admin/login”

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 30 | Inputs correct username and password | Username=super  Password=supersuper | {“message": "Super admin login successfully", "token": "eyJ0…TEjCzJln\_wXR79yR-s”} |
| 31 | Inputs incorrect username or password | Username=super  Password=supersuper | {“error": "Invalid username and password"} |

### UTC-13: ****Back-end: Method name: delete\_account (username)****

Description: This method will receive http DELETE request “/admin/delete/<username>”

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 32 | Inputs exist username | Username= admin001 | {“result": {“message": "User deleted”}} |
| 33 | Inputs non-exist username | Username= admin002 | {“result”: {  "message": "No user found"}  } |

### UTC-14: ****Back-end: Method name: register ()****

Description: This method will receive http POST request “users/register”

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 34 | Inputs new user to register | Username= admin001  Password=admin001 | {"result": {  "message": “admin001 registered”}  } |

### UTC-15: ****Back-end: Method name: get\_all\_account ()****

Description: This method will receive http POST request “users/get-all”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 35 | Database has users | - | [{“status”: "super admin added",  "trial\_time": "Fri, 06 Sep 2019 07:05:07 GMT",  "username": "test"},  {"status": "super admin added",  "trial\_time": "Fri, 06 Sep 2019 07:05:46 GMT",  "username": "test02"}] |
| 36 | Database has no users | - | [] |

### UTC-16: ****Back-end: Method name: edit\_username ()****

Description: This method will receive http POST request “users/edit/username”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 37 | Inputs exist username to edit | {"username":"test@test.com",  "info":"feiji@test.com"} | {"result": {  "message": "test is changed to feiji@test.com"}} |
| 38 | Inputs non-exist username to edit | {"username":"test123@test.com",  "info":"feiji123@test.com"} | {  "result": {  "message": "No user found"  }  } |

### UTC-17: ****Back-end: Method name: edit\_password ()****

Description: This method will receive http POST request “users/edit/password”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 39 | Inputs correct old password | {  "username":"test@test.com",  "password":"tttttt",  "new\_password":"123456"  } | {  "message": "Password changed"  } |
| 40 | Inputs incorrect old password | {"username":"test@test.com",  "password":"kkkkkk",  "new\_password":"123456"} | {"error": "Invalid username and password"} |

### UTC-18: ****Back-end: Method name: approve\_user ()****

Description: This method will receive http POST request “admin/approve”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 41 | Super admin approves the request | {"username": “testagain”} | {"result": {"message": "testagain's request is approved"}} |
| 42 | Cannot find the user | {"username": “testttagain”} | {"result": {"message": "No user found"}} |

### UTC-19: ****Back-end: Method name: reject\_user ()****

Description: This method will receive http POST request “admin/reject”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 43 | Super admin rejects the request | {"username": “testagain”} | {"result": {"message": "testagain's request is rejected"}} |
| 44 | Cannot find the user | {"username": “testttagain”} | {"result": {"message": "No user found"}} |

### UTC-20: ****Back-end: Method name: get\_pending ()****

Description: This method will receive http GET request “admin/get-pending”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 45 | Database has users that their status is “pending” | - | {"results": [{"status": "pending",  "trial\_time": "pending",  "username": "hello"}]} |
| 46 | Cannot find the user | - | {  "results": []  } |

### UTC-21: ****Back-end: Method name: add\_user ()****

Description: This method will receive http POST request “admin/add”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 47 | Database has users that their status is “pending” | {"username": "test\_admin\_add@test.com",  "trial\_time": 14,  "status": "approved"  } | {"result": {"message": "test\_admin\_add@test.com added"}} |

### UTC-22: ****Back-end: Method name: get\_account\_by\_username ()****

Description: This method will receive http GET request “/users/get-by-username”.

Test Date: 23 Aug 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 48 | Inputs exist user | {"test\_admin\_add@test.com"} | {"result": {"message": "User found",  "status": "approved",  "trial\_time ": "Fri, 06 Sep 2019 07:34:06 GMT",  "username": "test\_admin\_add@test.com"}} |
| 49 | Inputs non-exist user | {"username":"junyu"} | {  "result": {  "message": "No user found"  }  } |

### [Progress II]

### UTC-23: ****Back-end: Method name: get\_all ()****

Description: This method will receive http GET request and get comment data from database.

Test Date: 27 Sep 2019

expectedData11 =[ {'\_id': ObjectId('5d84c401361d71734497ae6a'), 'c\_id': 11, 'date': '2017-03-06', 'hotel': 'One Park Hotel', 'rating': 10,'content': "Would defo not recommend this hotel it's...'"},…]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 50 | Test when connect to database | - | expectedData11 |
| 51 | Test when database is empty | - | [] |
| 52 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-24: ****Back-end: Method name: get\_count\_by\_date ()****

Description: This method will receive http GET request and get the comment count by date from database.

Test Date: 27 Sep 2019

expectedData12 = [{"count": 1, "date": "2003-05-31"}, …

{"count": 1, "date": "2004-04-25"}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 53 | Test when connect to database | - | expectedData12 |
| 54 | Test when database is empty | - | [] |
| 55 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-25: ****Back-end: Method name: get\_count\_by\_date\_positive ()****

Description: This method will receive http GET request and get the positive comment count by date from database.

Test Date: 27 Sep 2019

expectedData13 = [{"count": 0, "date": "2003-05-31"}, …

{"count": 0, "date": "2004-04-25"}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 56 | Test when connect to database | - | expectedData13 |
| 57 | Test when database is empty | - | [] |
| 58 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-25: ****Back-end: Method name: get\_count\_by\_date\_negative ()****

Description: This method will receive http GET request and get the negative comment count by date from database.

Test Date: 27 Sep 2019

expectedData14 = [{"count": 1, "date": "2003-05-31"}, …

{"count": 1, "date": "2004-04-25"}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 59 | Test when connect to database | - | expectedData14 |
| 60 | Test when database is empty | - | [] |
| 61 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-26: ****Back-end: Method name: get\_count\_by\_date\_neutral ()****

Description: This method will receive http GET request and get the neutral comment count by date from database.

Test Date: 27 Sep 2019

expectedData15 = [{"count": 0, "date": "2003-05-31"}, …

{"count": 0, "date": "2004-04-25"}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 62 | Test when connect to database | - | expectedData15 |
| 63 | Test when database is empty | - | [] |
| 64 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-27: ****Back-end: Method name: get\_hotel ()****

Description: This method will receive http GET request and get the number of comments of each hotel from database.

Test Date: 27 Sep 2019

expectedData16 = "results": [{"count": 2,"hotel": "Agriturismo Le Grotte"}, {

"count": 2,"hotel": "Locanda Specchio di Diana Albergo Diffusso"}, {"count": 2, "hotel": "Hotel Nespolo D'Oro"},

... ,

{"count": 3,"hotel": "Il Giardino di Diana Bed and Breakfast"},]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 65 | Test when connect to database | - | expectedData16 |
| 66 | Test when database is empty | - | [] |
| 67 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-28: ****Back-end: Method name: get\_hotel\_by\_name ()****

Description: This method will receive http GET request and get the comment of inputting hotel from database.

Test Date: 27 Sep 2019

expectedData9 = [{"c\_id": 1627,"content": "The user only left a rating.","date": "2017-11-12", "rating": 50},{"c\_id": 1640, "content": "The user only left a rating.", "date": "2017-02-05","rating": 40}]

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 68 | Test when connect to database | name = “Agriturismo Le Grotte” | expectedData9 |
| 69 | Test when database is empty | name = “Agriturismo Le Grotte” | [] |
| 70 | Test when cannot connect to database | name = “Agriturismo Le Grotte” | alert: “Cannot connect to database, please try again” |
| 71 | Test when input nonsexist hotel name | name = “AAA” | [] |

### UTC-29: ****Back-end: Method name: add\_comment ()****

Description: This method will receive http POST request and store the comment to database.

Test Date: 27 Sep 2019

expectedData17 = {"c\_id": 1675,"content": "Hate it", "date": "2019-10-08", "rating": 10}

expectedData18 = {"c\_id": 1676, "content": "Love it", "date": "2019-10-08", "rating": 50}

expectedData19 = {"c\_id": 1677, "content": "It’s ok", "date": "2019-10-08", "rating": 30}

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 72 | Test when connect to database and input a negative comment | {"content": "Hate it", "hotel": "Test", "user": "Peter"} | expectedData17,  “comment added” |
| 73 | Test when connect to database and input a positive comment | {"content": "Love it", "hotel": "Test", "user": "Peter"} | expectedData18,  “comment added” |
| 74 | Test when connect to database and input a neutral comment | {"content": "It’s ok", "hotel": "Test", "user": "Peter"} | expectedData19,  “comment added” |
| 75 | Test when database is empty | {"content": "It’s ok", "hotel": "Test", "user": "Peter"} | expectedData19,  “comment added” |
| 76 | Test when cannot connect to database | - | alert: “Cannot connect to database, please try again” |

### UTC-30: ****Back-end: Method name: record\_log ()****

Description: This method will record log information and store to database.

Test Date: 27 Sep 2019

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 77 | Test when super admin login | - | {"content": "super admin logged in",  "creator": "super admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Login" }, |
| 78 | Test when admin login | - | {"content": "admin logged in",  "creator": "admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Login" }, |
| 79 | Test when admin write a comment | - | {"content": "admin wrote a comment", "creator": "admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Comment" }, |
| 80 | Test when admin edit username | - | {"content": "admin edit username", "creator": "admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Edit" }, |
| 81 | Test when admin edit password | - | {"content": "admin edit password", "creator": "admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Edit" }, |
| 82 | Test when admin login but failed | - | {"content": "admin login failed", "creator": "admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Login" }, |
| 83 | Test when super admin deletes an account | - | {"content": "super admin delete an account", "creator": "super admin", "log\_id": 1,"time": "2019-10-08 15:40:37", "type": "Delete" }, |
| 84 | Test when super admin approves all register request. | - | {"content": "all user approved.", "creator": "super admin", "log\_id": 1,"time": "201910-08 15:40:37", "type": "Approve" }, |
| 85 | Test when super admin rejects all register request. | - | {"content": "all user rejected.", "creator": "super admin", "log\_id": 1,"time": "201910-08 15:40:37", "type": "Reject" }, |

# 5. System Testing

### 5.1 Super admin

### STC-01: Super admin can login to the system and manage accounts.

**Description:** The system testing for URS-01 – URS07. The system should provide an interface. The interface includes “All account”, “Waiting List”, “Dashboard”, and “Logout” buttons. Super admin can log in to system, view all accounts or waiting list page, edit account information, delete account, add admin account, accept or reject admin account register request.

**Prerequisite:**

- The browser is not logged in to the system.

- Test data in the Appendix 1.

**Test Script:**

**1. Login to the system**

1.1 Input username and password.

1.2 Click “Login” button.

**2. Add admin account**

2.1 Click “All accounts” button.

2.2 Click “Add” button

2.3 Input username, trial time and status then click “Save” or “Cancel” button.

**3. Edit account information**

3.1 Click “All accounts” button.

3.2 Click “Edit” button.

3.3 Input username and trial time then click “Save” or “Cancel” button.

**4. Delete account**

4.1 Click “All accounts” button.

4.2 Click “Delete” button

4.3 Click “Save” or “Cancel” button.

**5. Accept or reject admin account register request**

5.1 Click “Waiting List” button.

5.2 Select “Approval” or “Reject” then click “Save” or “Cancel” button.

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 1 | Super admin login by inputting correct username and password | Username= super  Password= supersuper | The system redirects to super admin page. |
| 2 | Super admin login by inputting incorrect username and password. | Username= super  Password= super | Show an error message “Login failed” |
| 3 | View accounts | - | Show all accounts information |
| 4 | View waiting list | - | Show waiting list information |
| 5 | View accounts when database is empty | - | Show a message “No record to display” |
| 6 | View accounts when cannot connect to database | - | Show a message “Cannot connect to database” then provide refresh and cancel button |
| 7 | View waiting list when database is empty | - | Show a message “No record to display” |
| 8 | View waiting list when cannot connect to database | - | Show a message “Cannot connect to database” and provide refresh and cancel button |
| 9 | Edit user information and click “Save” | Username= super  Trial Time = 30 | User’s trial time increased 30 days and system saves the new information to database. |
| 10 | Edit user information and click “Cancel” | Username= super  Trial Time = 30 | System cancel the edit action. |
| 11 | Edit user information and click “Save” | Username= super  Trial Time = -30 | User’s trial time decreased 30 days and system saves the new information to database. |
| 12 | Delete account and click “Save” button | - | Remove user account from database. |
| 13 | Delete account and click “Cancel” button | - | System cancel the delete action. |
| 14 | Add admin account and click “Save” button | username = admin  trial time = 30  status = approved | System records account information to database. |
| 15 | Add admin account and click “Cancel” button | username = admin  trial time = 30  status = approved | System cancel the add admin account action. |
| 16 | Accept admin account register request and click “Save” button | - | System records account information to database. |
| 17 | Accept admin account register request and click “Cancel” button | - | System cancel the accept admin account register request action. |
| 18 | Reject admin account register request and click “Save” button | - | System records account information to database. |
| 19 | Reject admin account register request and click “Cancel” button | - | System cancel the reject admin account register request action. |

### STC-02: Super admin can view system log[Progress II]

**Description:** The system testing for URS-21 – URS-23. The system should provide an interface. The interface includes a table which shows “ID”, “User”, “Time”, “Type” and “Content” of each log. Super admin view log information and account information.

**Prerequisite:**

- The super admin is logged in.

- Test data in the Appendix 1.

**Test Script:**

**1. View System log and account information.**

1.1 Click “System Log” button.

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 20 | Super admin can view system log | - | The system displays the table which shows “ID”, “User”, “Time”, “Type” and “Content” of logs. |

### 4.2 Admin

### STC-03: Admin can login or register account.

**Description:** The system testing for URS-08 –URS09. The system should provide an interface. The interface includes “Login”, “Register” buttons. Admin can log in and register.

**Prerequisite:**

- The browser is not logged in to the system.

- Test data in the Appendix 1.

**Test Script:**

**1. Login to the admin page.**

1.1 Input username and password.

1.2 Click “Login” button.

**2. Register account**

2.1 Input username and password.

2.2 Click “Register” button.

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 21 | Admin login by Inputting correct username and password | Username= admin001  Password= admin001 | The system redirects to admin page. |
| 22 | Admin login by Inputting incorrect username and password. | Username= admin001  Password= admin | Show an error message “Login failed” |
| 23 | Admin register account by inputting username and password. | Username= admin001  Password= admin001 | The system records account information to database |

### STC-04: Admin can edit information

**Description:** The system testing for URS-10. The system should provide an interface. The interface includes “Edit” buttons. Admin can edit their username and password.

**Prerequisite:**

- The browser is logged in to the system.

- Test data in the Appendix 1.

**Test Script:**

**1. Edit username.**

1.1 Input username.

1.2 Click “Save” or “Cancel” button.

**2. Edit password.**

2.1 Input password.

2.2 Click “Save” or “Cancel” button.

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 24 | Edit username and click “Save” button. | Username= admin002 | The system stores information to database. |
| 25 | Edit username and click “Cancel” button. | Username= admin002 | System cancel the edit action. |
| 26 | Edit password and input old password then click “Save” button. | Old password=admin001  New password=admin002 | The system stores information to database. |
| 27 | Edit password and input old password then click “Cancel” button. | Old password=admin001  New password=admin002 | System cancel the edit action. |

### STC-05: Admin can logout from system.

**Description:** The system testing for URS-11. The system should provide an interface. The interface includes “Logout” buttons. Admin can logout system.

**Prerequisite:**

- The browser is logged in to the system.

- Test data in the Appendix 1.

**Test Script:**

**1. Click logout button.**

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 28 | Admin logout. | - | System remove user token from local storage. |

### STC-06: View the summary of data visualization result

**Description:** The system testing for URS-12 – URS-16. The system should provide an interface. The interface includes all the summary sparkline chart, positive comments rate line chart, word frequency bar chart, number of comments bar chart, types of comments stacked column chart.

**Prerequisite:**

- Admin is logged in.

- Test data in the Database.

**Test Script:**

**1. Access the homepage.**

**2. Access to the dashboard.**

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 29 | View all the summary sparkline chart. | - | System displays the data in Appendix expectedData3, expectedData4 |
| 30 | View positive comments rate line chart. | - | System displays the data in Appendix expectedData4 |
| 31 | View word frequency bar chart. | - | System displays the data in Appendix expectedData5 |
| 32 | View number of comments bar chart. | - | System displays the data in Appendix expectedData6 |
| 33 | View types of comments stacked column chart. | - | System displays the data in Appendix expectedData7 |
| 34 | View charts when cannot connect to database | - | Show a message “Cannot connect to database” then provide refresh and cancel button |

### STC-07: View the comments[Progress II]

**Description:** The system testing for URS-17 – URS-19. The system should provide an interface. The interface includes a list of all hotels and the user could click to access each hotel page to view comments.

**Prerequisite:**

- Test data in the Database.

**Test Script:**

**1. View the hotel list.**

**2. View comments.**

2.1 User clicks a hotel.

2.2 User view comments of clicked hotel.

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 35 | View all the comments of selected hotel. | - | System displays the data in Appendix exceptedData11. |

### STC-08: Write the comments[Progress II]

**Description:** The system testing for URS-20. The system should provide an interface. The interface includes an input area for inputting comment.

**Prerequisite:**

- Admin is logged in.

**Test Script:**

**1. View the hotel list.**

**2. Write a comment.**

2.1 Admin clicks a hotel.

2.2 Admin accesses the hotel page.

2.3 Admin inputs a comment.

2.4 Admin clicks “Submit” button.

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Description | Input | Expect Result |
| 36 | Write a comment for a hotel | “This hotel is good.” | System displays new comment at hotel page, the comment includes content, date, ID and rating. |

# 6. Appendix

### 1. Account information

Username: admin001

Password: admin001

### 2. Sparkline chart information

sparklineData = [47, 45, 54, 38, 56, 24, 65, 31, 37, 39, 62, 51, 35, 41, 35, 27, 93, 53, 61, 27, 54, 43, 19, 46]

expectedData2 = {

seriesSpark1: [{data: [25, 66, 41, 89, 63, 25, 44, 12, 36, 9, 54]}],

seriesSpark2: [{data: [47, 45, 74, 14, 56, 74, 14, 11, 7, 39, 82]}],

seriesSpark3: [{data: [12, 14, 2, 47, 42, 15, 47, 75, 65, 19, 14]}],

seriesSpark4: [{data: [15, 75, 47, 65, 14, 2, 41, 54, 4, 27, 15]}],

}

expectedData3 = {

total\_comments: '135,965',

positive\_comments: '99,821',

negative\_comments: '10,212',

neutral\_comments: '25,932',

percent\_total: '100%',

percent\_positive: '73.4%',

percent\_negative: '7.5%',

percent\_neutral: '19.1%',

}

### 3. Positive comments rate line chart information

expectedData4 = [

[ 1483203600000, 5], [ 1483808400000, 22],

[ 1484413200000, 52], [ 1485018000000, 87],

[ 1485622800000, 57], [ 1486227600000, 84],

[ 1486832400000, 87], [ 1487437200000, 43],

[ 1488042000000, 29], [ 1488646800000, 69],

[ 1489251600000, 81], [ 1489856400000, 46],

[ 1490461200000, 14], [ 1491066000000, 79],

[ 1491670800000, 9], [ 1492275600000, 51],

[ 1492880400000, 76], [ 1493485200000, 9],

[ 1494090000000, 51], [ 1494694800000, 77],

[ 1495299600000, 22], [ 1495904400000, 96],

[ 1496509200000, 18], [ 1497114000000, 67],

[ 1497718800000, 24], [ 1498323600000, 51],

[ 1498928400000, 41], [ 1499533200000, 83],

[ 1500138000000, 56], [ 1500742800000, 42],

[ 1501347600000, 76], [ 1501952400000, 20],

[ 1502557200000, 73], [ 1503162000000, 14],

[ 1503766800000, 67], [ 1504371600000, 8],

[ 1504976400000, 22], [ 1505581200000, 64],

[ 1506186000000, 44], [ 1506790800000, 1],

[ 1507395600000, 34], [ 1508000400000, 34],

[ 1508605200000, 70], [ 1509210000000, 16],

[ 1509814800000, 20], [ 1510419600000, 17],

[ 1511024400000, 86], [ 1511629200000, 78]

]

### 4. Word frequency bar chart information

expectedData5 = [400, 430, 448, 470, 540, 580, 690, 1100, 1200, 1380]

### 5. Number of comments bar chart information

expectedData6 = [{

name: 'Neutral',

data: [44, 55, 57, 56, 61, 58, 63, 60, 66]

}, {

name: 'Positive',

data: [76, 85, 101, 98, 87, 105, 91, 114, 94]

}, {

name: 'Negative',

data: [35, 41, 36, 26, 45, 48, 52, 53, 41]

}];

### 6. Types of comments stacked column chart information

expectedData7 = [{

name: 'Neutral',

data: [44, 55, 41, 67, 22, 43, 21, 49, 39]

}, {

name: 'Positive',

data: [13, 23, 20, 8, 13, 27, 33, 12, 14]

}, {

name: 'Negative',

data: [11, 17, 15, 15, 21, 14, 15, 13, 9]

}];