Wireless Sensor-based Application for Managing Theme Park

Test Plan

**By**

Pabhawee Chuacharoen 552115037

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr.Noppon Choosri

**Document History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Status | Change Context | Viewable | Reviewer | Responsible | Date |
| V.1.0 | Draft | Introduction  Test Plan & Strategy | PC, NC | PC | PC | 05/08/15 |
| V.1.1 | Draft | Unit Test | PC, NC | PC | PC | 19/08/15 |
| V.1.2 | Draft | System Test | PC, NC | PC, NC | PC | 20/08/15 |
| V.2.0 | Release | Change project name | PC, NC | PC | PC | 27/08/15 |
| V.2.0 | Release | New test plan | PC, NC | PC | PC | 21/10/15 |

PC = Pabhawee Chuacharoen , NC = Dr. Noppon Choosri**Table of Content**

[Chapter One | Introduction 3](#_Toc428536160)

[1.1. Introduction 3](#_Toc428536161)

[1.2. Scope of Testing 3](#_Toc428536162)

[1.3. Test Objectives 3](#_Toc428536163)

[1.4. Key Acronyms and Abbreviation 3](#_Toc428536164)

[1.5. Dependency Document 3](#_Toc428536165)

[Chapter Two | Test Plan & Strategy 3](#_Toc428536166)

[2.1. Test Strategy 3](#_Toc428536167)

[2.2. Result of Testing 4](#_Toc428536168)

[2.3. Test Environment 4](#_Toc428536169)

[2.3.1. Hardware 4](#_Toc428536170)

[2.3.2. Software 4](#_Toc428536171)

[Chapter Three | Unit Test 5](#_Toc428536172)

[3.1. Test Attraction Model 5](#_Toc428536173)

[3.1.1. Unit Test Case 1 (UTC-01): 5](#_Toc428536174)

[3.1.2. Unit Test Case 2 (UTC-02): 5](#_Toc428536175)

[3.1.3. Unit Test Case 3 (UTC-03): 6](#_Toc428536176)

[3.1.4. Unit Test Case 4 (UTC-04): 6](#_Toc428536177)

[3.1.5. Unit Test Case 5 (UTC-05): 7](#_Toc428536178)

[3.2. Test Guest Model 7](#_Toc428536179)

[3.2.1. Unit Test Case 6 (UTC-06): 7](#_Toc428536180)

[3.2.2. Unit Test Case 7 (UTC-07): 8](#_Toc428536181)

[3.2.3. Unit Test Case 8 (UTC-08): 8](#_Toc428536182)

[3.2.4. Unit Test Case 9 (UTC-09): 9](#_Toc428536183)

[3.2.5. Unit Test Case 10 (UTC-10): 9](#_Toc428536184)

[3.3. Test RegisterData Model 10](#_Toc428536185)

[3.3.1 Unit Test Case 12 (UTC-12): 10](#_Toc428536186)

[3.3.2 Unit Test Case 13 (UTC-13): 10](#_Toc428536187)

[Chapter Four | System Test 11](#_Toc428536188)

[4.1. System Test Case 01 (STC-01): 11](#_Toc428536189)

[4.2. System Test Case 02 (STC-02): 12](#_Toc428536190)

[4.3. System Test Case 03 (STC-03): 12](#_Toc428536191)

[4.4. System Test Case 04 (STC-04): 13](#_Toc428536192)

[4.5. System Test Case 09 (STC-09): 13](#_Toc428536193)

[4.6. System Test Case 10 (STC-10): 14](#_Toc428536194)

[4.7. System Test Case 11 (STC-11): 15](#_Toc428536195)

[4.8. System Test Case 12 (STC-12): 16](#_Toc428536196)

# Chapter One | Introduction

## Introduction

The purpose of this test plan document is to describe testing plan and methodologies that are used in this project testing in order to guarantee the system works as it expected.

## Scope of Testing

The scope of test plan document is to test activities/events within the WSAT project. And then find the quality of user requirement and system requirement.

## Test Objectives

The objectives of test plan are:

* Exceed user requirements.
* Reduce the rejecting project.
* Test the functional of system that includes Software Requirement Specification and Software Design Document.
* The Functional systems are correctness.

## Key Acronyms and Abbreviation

|  |  |
| --- | --- |
| Name | Explain |
| UTC | Unit test case |
| STC | System test case |
| UID | User Identification |
| SID | Station Identification |

## Dependency Document

This Test Plan is depended on the following documents

* Software Project Management plan
* Software Requirement Specification
* Software Design document
* Traceability Record

# Chapter Two | Test Plan & Strategy

## Test Strategy

This project test strategy will be followed by:

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. Results of testing will be recorded.
6. All test file will be stored in repository.

## Result of Testing

In the test record, the test results will be separated into two categories, which are:

Passed: The actual result is the same as the expected result.

Failed: The actual result is not the same as the expected result.

## Test Environment

### Hardware

#### Computers

* + **Asus**

Processor: Intel(R) Core i5-2450m CPU @2.50Ghz

Memory: Installed Memory 8.00 GB

Operating system: 64-bit Operating System

#### Arduino

* + **UNO R3**

ATmega328 microcontroller

Input voltage - 7-12V

14 Digital I/O Pins (6 PWM outputs)

6 Analog Inputs

32k Flash Memory

16Mhz Clock Speed

#### WIFI module ESP8266-01

#### RFID module RC522

### Software

#### NetBeans IDE 8.0.2

#### Mozilla Firefox 39.0

# Chapter Three | Unit Test

## Test Attraction Model

### Unit Test Case 1 (UTC-01):

public function getAttractions()

* **Description:**

This test case tests whether the method can get the all Attraction objects from the database or not.

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that the method should return the all existed Attraction objects in database | array(  array(     'aid' =>'9',      'aname' =>'twister',      'description' =>'beware of the storm',      'height\_limit' =>'120'),  array(     'aid' =>'10',      'aname' =>'magic carpet',      'description' =>'a whole new world',      'height\_limit' =>'0'),  array(     'aid' =>'11',      'aname' =>'coaster',      'description' =>'like a rolling stone',      'height\_limit' =>'120')          ); |

### Unit Test Case 2 (UTC-02):

public function getAttraction($id)

* **Description:**

This test case tests whether the method can get the specific Attraction object by valid id in the database or not.

* **Test Cases:**

| Test# | Description | Input | Expected Result |
| --- | --- | --- | --- |
| 1 | Test that the method should return the right Attraction object (has Id equals to input) with valid id | $id= ’9’ | array(  'aid' =>'9',  'aname' =>'twister',  'description' =>'beware of the storm',  'height\_limit' =>'120'); |
| 2 | Test that the method should return null with invalid id | $id = ‘444’ | null |

### Unit Test Case 3 (UTC-03):

public function insertAttraction($data)

* **Description:**

This test case tests whether the method can add the Attraction object to the database or not.

* **Test Data:**

$data = array(

'aname' =>'jet coaster',

'description' =>'not for senior citizen and child',

'height\_limit' =>'140'

);

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | Test that the method should return Attraction object which is same as input | $data | array(  'aname' =>'jet coaster',  'description' =>'not for senior citizen and child',  'height\_limit' =>'140'  ); |

### Unit Test Case 4 (UTC-04):

public function editAttraction($id,$data)

* **Description:**

This test case tests whether the method can update the Attraction object or not.

* **Test Data:**

$data = array(

'aname' =>'jet coaster',

'description' =>'better than the viking',

'height\_limit' =>'140'

);

* **Test Cases:**

| Test# | Description | Input | Expected Result |
| --- | --- | --- | --- |
| 1 | Test that the method should return Attraction object which is same as input | $id=’11’,$data | array(  'aname' =>'jet coaster',  'description' =>'better than the viking',  'height\_limit' =>'140'  ); |
| 2 | Test that the method should return null when the Id does not exist | $id=’777’,$data | null |

### Unit Test Case 5 (UTC-05):

public function deleteAttraction($id)

* **Description:**

This test case tests whether the method can delete the specific Attraction object by valid id to the database or not.

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | Test that the method should return null when the input does not exist | $id = ‘11’ | null |

## Test Guest Model

### Unit Test Case 6 (UTC-06):

public function getGuests()

* **Description:**

This test case tests whether the method can get the all Guest objects from the database or not.

* **Test Cases:**

| Test# | Description | Expected Result |
| --- | --- | --- |
| 1 | Test that the method should return the all existed Guest objects in database | array(  array(  'gid' =>'4',  'guid' =>'00ee4535',  'gname' =>'Faifai',  'gmail' =>'fai@mail.com',  'gtel' =>'1234567890',  'status' =>'0'),  array(  'gid' =>'5',  'guid' =>'3153f052',  'gname' =>'Biabia',  'gmail' =>'bia@mail.com',  'gtel' =>'765443210',  'status' =>'0')  ); |

### Unit Test Case 7 (UTC-07):

public function getGuest($id)

* **Description:**

This test case tests whether the method can get the specific Guest object by valid id in the database or not.

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | Test that the method should return the right Guest object (has Id equals to input) with valid id | $id=’4’ | array(  'gid' =>'4',  'guid' =>'00ee4535',  'gname' =>'Faifai',  'gmail' =>'fai@mail.com',  'gtel' =>'123456789',  'status' =>'0'); |
| 2 | Test that the method should return null with invalid id | $$id=’666’ | null |

### Unit Test Case 8 (UTC-08):

public function insertGuest($data)

* **Description:**

This test case tests whether the method can insert the Guest object with valid input and cannot update the object with invalid input or not.

* **Test Data:**

$data = array(

'guid' =>'33ee5f00',

'gname' =>'pakapol',

'gmail' =>'duangdee',

'gtel' =>'0952238845'

);

* **Test Cases:**

| Test# | Description | Input | Expected Result |
| --- | --- | --- | --- |
| 1 | Test that the method should return Guest object which is same as input | $data | $data = array(  'guid' =>'33ee5f00',  'gname' =>'pakapol',  'gmail' =>'duangdee@mail.com',  'gtel' =>'0952238845'  ); |

### Unit Test Case 9 (UTC-09):

public function editGuest($id,$data)

* **Description:**

This test case tests whether the method can update the Guest object or not.

* **Test Data:**

$data = array(

'guid' =>'00ee4535',

'gname' =>'FaiFai',

'gmail' =>'duangdee@mail.com',

'gtel' =>'0952238845'

);

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | The method should return Guest object which is same as input | $id=’4’,$data | array(  'guid' =>'00ee4535',  'gname' =>'FaiFai',  'gmail' =>'duangdee@mail.com',  'gtel' =>'0952238845'  ); |
| 2 | The method should return null when the Id does not exist | $id=’’,$data | null |

### Unit Test Case 10 (UTC-10):

Public function deleteGuest($id)

* **Description:**

This test case tests whether the method can delete the specific Guest object by valid id to the database or not.

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | The method should return null when the input does not exist | $id=’5’ | null |

## Test RegisterData Model

### Unit Test Case 12 (UTC-12):

public function insertUid($uid)

* **Description:**

This test case tests whether the method can insert the UID value to the database or not.

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | Test that the method should return uid which is same as input | $uid=’a3cf8x99’ | $uid = ‘a3cf8x99’; |

### Unit Test Case 13 (UTC-13):

public function getLastInserted()

* **Description:**

This test case tests whether the method can get the last inserted UID from the database or not.

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that the method should return uid which is lasted insert | $uid = ‘a3cf8x99’; |

## Test Activity Stat Model

### 3.4.1 Unit Test Case 14 (UTC-14):

public function insertThemeParkStat($data)

* **Description:**

This test case tests whether the method can insert the UID received from postdata class into the 'tm\_stat' table.

* **Test Data:**

$data = array(

'uid' => ‘a3cf8x99’,

'location' => ‘en’,

'date' => 2015-10-16'',

'time' => ' 16:37:21'

);

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that the method should insert the received data into the database | array(  'uid' => ‘a3cf8x99’,  'location' => ‘en’,  'date' => 2015-10-16'',  'time' => ' 16:37:21'  ) |

### 3.4.2 Unit Test Case 15 (UTC-15):

public function insertAttractionStat($data)

* **Description:**

This test case tests whether the method can insert the UID received from postdata class into the 'at\_stat' table.

* **Test Data:**

$data = array(

'uid' => ‘a3cf8x99’,

'location' => ‘en’,

'date' => 2015-10-16'',

'time' => ' 21:17:33'

);

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that the method should insert the received data into the database | array(  'uid' => ‘a3cf8x99’,  'location' => ‘ex’,  'date' => 2015-10-16'',  'time' => ' 21:17:33'  ) |

### 3.4.3 Unit Test Case 16 (UTC-16):

public function getThemeParkStat()

* **Description:**

This test case tests whether the method can get all the data that was inserted within range of one week count from the current date.

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that the method should return a set of valid data within range of one week count from the current date | array(  array(  'uid' => ‘a3cf8x99’,  'location' => ‘en’,  'date' => 2015-10-16'',  'time' => ' 11:17:55'  ),  array(  'uid' => ‘00ee4535',  'location' => ‘en’,  'date' => 2015-10-16'',  'time' => ' 09:57:33'  ),  array(  'uid' => ‘qqwwer45’,  'location' => ‘en’,  'date' => 2015-10-16'',  'time' => ' 14:11:44'  )  ); |
|  |  |  |

### 3.4.3 Unit Test Case 17 (UTC-17):

public function getAttractionStat($id)

* **Description:**

This test case tests whether the method can to get specific set of data that was inserted within range of one week count from the current date.

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | Test that the method should return a set of valid data within range of one week count from the current date according to the input $id | $id=’11’ | array(  array(  'uid' => ‘cd089j4n’,  'id' => ‘11’,  'date' => 2015-10-16'',  'time' => ' 11:11:33'  ),  array(  'uid' => ‘00ee4535',  'id' => ‘11’,  'date' => 2015-10-17'',  'time' => ' 13:47:66'  ),  array(  'uid' => ‘qqwwer45’,  'id' => ‘11’,  'date' => 2015-10-17'',  'time' => ' 08:01:29'  )  ); |

# Chapter Four | System Test

## System Test Case 01 (STC-01):

Ticket seller can register guest into the system.

* **Description:**

Ticket seller can register a guest with valid name, email and mobile no. into the system.

* **Prerequisites:**

Users select role as a ticket seller.

* **Test procedure:**

1. Scan RFID tag at the reader
2. Choose ‘Register Guest’
3. Fill in guest information
4. Click submit button
5. View the guest information page

* **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Input | Expected Result |
| 1 | Test that ticket seller can register a guest with correct name, email and mobile no. formats | Guest name = ‘Thong Dee’  Guest email = ‘[thong@mail.com](mailto:thongsuk@mail.com)’  Guest mobile no. = ‘0976352211’ | System shows message ‘Inserted Data Successful’ |
| 2 | Test that ticket seller cannot register a guest with empty name, email and mobile no. fields | Guest name = ‘’  Guest email = ‘’  Guest mobile no. = ‘’ | System shows error messages ‘Please fill out this field' |
| 3 | Test that ticket seller cannot register a guest with a name that has less than 5 character length | Guest name = ‘som’  Guest email = ‘[thong@mail.com](mailto:thongsuk@mail.com)’  Guest mobile no. = ‘097635221’ | System shows error messages ‘Name must be 5-30 characters in length’ |
| 4 | Test that ticket seller cannot register a guest with invalid email format | Guest name = ‘Thong Dee’  Guest email = ‘thong’  Guest mobile no. = ‘097635221’ | System shows error messages ‘Please enter an email address’ |
| 5 | Test that ticket seller cannot register a guest with invalid mobile no. format | Guest name = ‘Thong Dee’  Guest email = ‘[thong@mail.com](mailto:thongsuk@mail.com)’  Guest mobile no. = ‘zero-five-one-one’ | System shows error messages ‘Invalid mobile no..’ |

## System Test Case 02 (STC-02):

Ticket seller can view guest information from the system.

* **Description:**

Ticket seller can view all guest information from the database in a table.

* **Prerequisites:**

None

* **Test procedure:**

1. Select role as a ticket seller
2. Choose ‘View Guest Info’
3. View the information of the guest

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that ticket can view the information of all guests | System shows the information of guests:  ID = ‘4’  UID = ‘00ee4535’  Name = ‘Faifai’  Email = ‘fai[@mail.com](mailto:thongsuk@mail.com)’  Tel. = ‘123456789’  Status = ‘false’  ID = ‘5’  UID = ‘3153f052’  Name = ‘Biabia’  Email = ‘bia[@mail.com](mailto:thongsuk@mail.com)’  Tel. = ‘76544321’  Status = ‘false’ |
| 2 | Test that ticket seller can view the information when the guest information is not exist | System shows empty table |

## System Test Case 03 (STC-03):

Ticket seller can edit guest information from the system.

* **Description:**

Ticket seller can edit name, email and mobile no. from the selected guest information.

* **Prerequisites:**

None

* **Test procedure:**

1. Select role as a ticket seller
2. Choose ‘View Guest Info’
3. Click at ‘edit’ at the row of the selected record
4. Edit the guest information
5. View the guest information

* **Test Cases:**

| Test# | Description | Input | Expected Result |
| --- | --- | --- | --- |
| 1 | Test that ticket seller can edit the selected guest information | Select guest id = 4 :  Guest name = ‘Ann Living’  Guest email = ‘hoocurr[@mail.com](mailto:thongsuk@mail.com)’  Guest mobile no. = ‘5524567345’ | System shows the updated guest information :  ID = ‘4’  UID = ‘00ee4535’  Name = ‘Ann Living’  Email = ‘hoocurr[@mail.com](mailto:thongsuk@mail.com)’  Tel. = ‘5524567345’  Status = ‘false’ |

## System Test Case 04 (STC-04):

Ticket seller can delete guest information from the system.

* **Description:**

Ticket seller can delete the selected guest information from the system.

* **Prerequisites**

None

* **Test procedure**

1. Select role as a ticket seller
2. Choose ‘View Guest Info’
3. Click at ‘edit’ at the row of the selected record
4. View the guest information

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that ticket seller can delete the selected guest information | System shows all guest information without the deleted guest information |

## System Test Case 05 (STC-05):

Admin can view attraction information from the system.

* **Description:**

Admin can view all attraction information from the database in a table.

* **Prerequisites:**

None

* **Test procedure:**

1. Select role as an admin
2. Choose ‘View Attraction Info’
3. View the information of the guest

* **Test Cases:**

| Test# | Description | Expected Result |
| --- | --- | --- |
| 1 | Test that can view the information of all attractions | System shows the information of attractions:  ID=’9’  Name=’twister’  Description =’ beware of the storm’  Height limit=’120’ ID=’10’  Name=’magic carpet’  Description=’a whole new world’  Height limit=’0’  ID= ‘11’  Name=’coaster’ Description=’like a rolling stone’ Height limit=’120’ |
| 2 | Test that admin chooses to view the information when the attraction information is not exist | System shows empty table |

## System Test Case 06 (STC-06):

Admin can add attraction information into the system.

* **Description:**

Admin can add an attraction with valid name, description and height limit into the system.

* **Prerequisites:**

None

* **Test procedure:**

1. Select role as an admin
2. Choose ‘Add Attraction’
3. Fill in attraction information
4. Click submit button
5. Choose ‘View Attraction Info’

* **Test Cases:**

| Test# | Description | Input | Expected Result |
| --- | --- | --- | --- |
| 1 | Test that admin can add the attraction information with correct format | Name=’magic carpet’  Description=’This is for children’  Height limit=’0’ | System shows the added attraction information within the table among others attraction information :ID=’10’  Name=’magic carpet’  Description=’This is for children’  Height limit=’0’ |
| 2 | Test that admin cannot add the attraction information with Name, Description and Height limit fields empty | Name=’’  Description=’’  Height limit=’’ | System shows error messages 'Please fill out this field.’ |
| 3 | Test that admin cannot add the attraction information with name field shorter than 5 characters | Name=’magic carpet’  Description=’ ’This is for children’  Height limit=’0’ | System shows error messages ‘Name must be 5-30 characters in length.’ |
| 4 | Test that admin cannot add the attraction information with description field shorter than 5 characters | Name=’magic carpet’  Description=’easy’  Height limit=’0’ | System shows error messages ‘Description must be 5-60 characters in length.’ |
| 5 | Test that admin cannot add the attraction information with non-numeric data in a height limit field | Name=’magic carpet’  Description=’This is for children’  Height limit=’zero’ | System shows error messages ‘Please enter the number.’ |
| 6 | Test that admin cannot add the attraction information height limit field longer than 3 digits | Name=’magic carpet’  Description=’This is for children’  Height limit=’1200’ | System shows error messages ‘Height limit must not be more than 3 digits.’ |

## System Test Case 07 (STC-07):

Admin can edit attraction information from the system.

* **Description:**

Admin can edit name, description and height limit from the selected attraction information.

* **Prerequisites:**

Admin viewed attraction information.

* **Test procedure:**

1. Browse to participators page
2. Choose specific advisor
3. View the information of the advisor

* **Test Cases:**

| Test# | Description | Input | Expected Result |
| --- | --- | --- | --- |
| 1 | Test that admin can edit the attraction information | select attraction id= 11  Name=’coaster’  Description=’who care about the description’  Height limit=’120’ | System shows the updated attraction information :  ID= 11  Name=’coaster’  Description=’who care about the description’  Height limit=’120’ |

## System Test Case 08 (STC-08):

Admin can delete attraction information from the system.

* **Description:**

Admin can delete selected attraction information from the system.

* **Prerequisites:**

1. Admin viewed attraction information.
2. The records in the database is not empty.

* **Test procedure:**

1. Choose delete button at the end of an attraction
2. Click 'OK' button at the confirm dialog box
3. View the information of attractions

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that admin can delete the selected attraction from the system | System shows all attraction information without the deleted attraction information |

## System Test Case 09 (STC-09):

Admin can view a report on numbers of guest visiting the theme park within one week range.

* **Description:**

Admin can view a line chart that dedicates the number of guest visiting the theme park within one week range.

* **Prerequisites:**

The records in the database is not empty.

* **Test procedure:**

1. Choose 'Activity' at the main page
2. View the information of attractions

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that gets the correct set of information | System shows a line chart that contains number of guests and date |
| 2 | Test that gets the message when there is no record in database | System shall display the message "The activity record is empty". |

## System Test Case 10 (STC-10):

Admin can view a report on numbers of guest attending each ride within one week range.

* **Description:**

Admin can view a line chart that dedicates the number of guest attending a selected attraction in the theme park within one week range.

* **Prerequisites:**

Admin viewed attraction information.

* **Test procedure:**

1. Choose 'View Attractions'
2. Click 'Activity' button at the end of the selected attraction
3. View the information of attractions

* **Test Cases:**

|  |  |  |
| --- | --- | --- |
| Test# | Description | Expected Result |
| 1 | Test that gets the correct set of information | System shows a line chart that contains number of guests and date |
| 2 | Test that gets the message when there is no record in database | System shall display the message "The activity record is empty". |