Project Acceptance Tests

CITS3200 Team 15 – UniPark Booking System

Preface:

This document outlines tests that should be conducted on the system to know that it is functioning as expected. If these tests are passed, then it can be deemed that the project has been successful.

Test Summary:

Lists the requirements whose satisfaction will be demonstrated by the test as well as the expected output. This includes:

- Means of control: How the data is entered
- Data: Input data / output data
- Procedures: how the test will operate
- Outcome: what outcomes will the test case produce

1.0 Tests

1.1 Test 1

Test that UniPark staff can create user accounts for the application that UWA staff can access.

Means of Control:

- This can be tested by means of unit testing the functionality as well as manually testing that it works as intended.
  - Unit Testing
    - A test case class using the Python library *unittest* can be created to automate testing that this functionality works correctly
  - Manual Testing
    - This functionality can also be tested by hand to ensure it is operational

Data

- Input data: User account information to be included in the account creation form

- Output data: A success or failure message from the application

Procedure

- Unit Test
    - The unit test should use environment variables to log in to an instance of the application as an admin user
    - It should then call relevant functions to attempt to insert a new user into the applications database.
    - The unit test should then perform an assertion to test that the user exists in the database.
    - The test case should then attempt to log in as this user in the current application instance

- Manual Test
    - The tester should log in to the admin account
    - Use the application UI to attempt to create a new user account
    - Log out of the admin account and attempt to log in as the new user

Outcomes

- Both procedures will result in one of three outcomes

    1. Logging in to the newly created account was successful
    2. Logging in to the new account was unsuccessful
    3. An error was encountered during the test process

1.2 Test 2

Test that only users created by UniPark can access the application.

Means of Control:

- This can be tested by means of unit testing the functionality.
    - Unit Testing
        - A test case class using the Python library *unittest* can be created to automate testing that this functionality works correctly

Data

- Input data: A set of authorised users and their account information and a set of unauthorised users.
- Output data: A success or failure message from the application depending on whether the user is authorised to access the application or not.

Procedure

- Unit Test
    - The unit test should retrieve a set of authorised users from the current or test database.
    - At least one unauthorised user account should be created to test that this user cannot access the application.
    - The unit test should then attempt to log each of these users into the application testing whether the login was successful or not against their authorisation status.

Outcomes

The procedure described will result in one of three outcomes:

1. All tested users were or were not able to log in with respect to their authorisation.
2. Some unauthorised users were able to log in or some authorised users were not able to log in
3. An error was encountered during the test process

1.3 Test 3

Test that the application provides an A3 pdf reservation sign with booking details to a user's email when they make a booking.

Means of Control:

- This can be tested by means of running a test script that attempts to execute the desired functionality. The outcome of the test script will need to be inspected manually.
    - Test script and manual inspection of output
        - A script to create an instance of the application. The script will then run the relevant functions to simulate a booking for a user that should auto generate a reservation pdf that is sent to that user's email address. The email account can be checked to see if it received the reservation sign and that the reservation sign is correct with respect to the booking that was made.

Data

- Input data: A test user account and a simulated booking

- Output data: A generated email sent to the users account containing a reservation sign attachment for the relevant booking

Procedure

- The script should create an instance of the application and create a test user.

- A booking should then be made by the test user.

- The person running the test should then inspect the inbox of the test user to see whether the correct reservation sign attachment was received.

Outcomes

The procedure described will result in one of three outcomes:

1. The test user received the correct reservation sign attachment in their inbox.

2. The test user did not receive any email from the application, or the attachment included was incorrect.

3. An error was encountered during the test process.

1.4 Test 4

Test that a user can make a booking and that this information is stored correctly by the application.

Means of Control:

- This should be tested by means of unit testing the functionality as well as manually testing that it works as intended.
    - Unit Testing
        - A test case class using the Python library *unittest* can be created to automate testing that this functionality works correctly
    - Manual Testing
        - This functionality can also be tested by hand to ensure it is operational

Data

- Input data: A user account and a booking, specifying bay, time, and date.
- Output data:
    - Unit test: A Boolean value on whether the details of the booking made match those retrieved from the database after the booking has been placed.
    - Manual Test: A visual representation of the booking being available to the user when returning to the application after having first made the booking. This representation should match the original booking made if it has been stored and presented correctly.

Procedure

- Unit Test
    - The unit test should use environment variables to log in to an instance of the application as regular user
    - It should then call relevant functions to attempt to make a booking on an available bay.
    - The unit test should then attempt to retrieve the booking made by this user and see whether the retrieved information matches that of the actual booking that was made.
    - If the retrieved booking matches the actual booking, return True, else False.

- Manual Test
    - The tester should log in to a regular user account.
    - Use the application UI to place a booking for this user.
    - Navigate to the summary of bookings for this user and check whether the listed information contains the booking that was just previously made.

Outcomes

- Both procedures will result in one of three outcomes
    1. Placing a booking was successful
    2. Placing a booking was not successful or information about the booking that was placed was incorrectly stored.
    3. An error was encountered during the test process

1.5 Test 5

Test that a user can delete a booking that they have previously made.

Means of Control:

- This should be tested by means of unit testing the functionality as well as manually testing that it works as intended.
    - Unit Testing
        - A test case class using the Python library *unittest* can be created to automate testing that this functionality works correctly
    - Manual Testing
        - This functionality can also be tested by hand to ensure it is operational

Data

- Input data: A user account and a booking, specifying bay, time, and date.
- Output data:
    - Unit test: A Boolean value on whether the initial booking placed exists after an attempt to delete it by the user has been made.
    - Manual Test: A visual notification displayed to the user that the booking has been deleted, and its non-appearance in the list of bookings made by the user.

Procedure

- Unit Test
    - The unit test should use environment variables to log in to an instance of the application as regular user
    - It should then call relevant functions to attempt to make a booking on an available bay.
    - The unit test should then test whether this booking exists in the application database to ensure the prior step was successful.
    - The unit test should then attempt to run the relevant function to delete this booking.
    - If the initial booking can no longer be retrieved from the database return True, else False.

- Manual Test
    - The tester should log in to a regular user account.
    - Use the application UI to place a booking for this user.
    - Navigate to the summary of bookings for this user and check whether the listed information contains the booking that was just previously made.
    - The tester should now use the UI to attempt to delete this booking.
    - The tester should look out for a visual notification that mentions the successful deletion of the booking and should then navigate to the list of bookings for that user and inspect whether that initial booking still appears in this list.

Outcomes

- Both procedures will result in one of three outcomes

4. Deleting a booking was successful
5. The booking still exists or the wrong booking has been deleted.
6. An error was encountered during the test process