CaBelle: Hotel Booking System Project

Test Case Documentation and Testing Report

Spring 2024

Alyssa Bustos - Quality Assurance and Front-End Developer
Carlos Hernandez - Quality Assurance, Back-End Developer, and Front-End Developer
Chelsea Ogbedeagu - Back-End Developer and Front-End Developer
Klarissa Navarro - UI/UX Designer and Front-End Developer
Michael Baldo - Quality Assurance, Back-End Developer and Front-End Developer
Tina Pham - Project Manager, Technical Writer, Back-End Developer, and Front-End Developer

Key Components of Our System and Interactions

Key components in our system are the login/sign up page, the payment page, the user account information page, the hotel search page, and the booking page.

Testing Process

We began our testing process with a Testing Plan:

- Software Verification
    - Ensuring data can be stored and retrieved in database
- Software Validation
    - Checking that our application conforms to our software requirement specifications and matches the user needs
- Testing Environment
    - Ensuring that our application can be opened in any web browser
- Test Cases
    - Decide on what and how we will be testing our application
    - Plan test cases for functionality, boundary, and error handling based on our key components of our system
    - Decide on a testing framework that is suitable for our code
        - We ran our test script using Puppeteer and Jest.
            - Puppeteer is part of the node.js library and is a tool for automating the process of frontend development and testing without requiring browser extensions.
            - Jest is a texting framework for JavaScript and is designed to ensure correctness of any JavaScript codebase.
- Error Identification
    - Identify a list of potential problems and fix any errors that arise in testing our application

Developing and Implementation of Test Cases

Three main test cases that we will be focusing are:

- Functionality Test Case
    - Ensure each function performs as expected
- Boundary Test Case
    - Test the extremes of the input domains and test input values that are outside of the defined input boundaries
- Error Handling Test Cases
    - Test with inputs that show how well the system handle errors

Functionality Test Cases

Test Case #1:

Description: To verify that users can search for hotels based on their criteria and make a booking after the user is logged in.

Preconditions:
- User has a registered account on the hotel booking system website.
- The hotel booking system is accessible and functional.

Postconditions/Expected Outcome:
- Booking confirmation is sent to the user.
- The booked hotel room is reserved in the system.

Test Execution:
- Pass criteria:
    - Login and hotel booking successfully pass without any errors.
- Fail Criteria:
    - The user does not receive a booking confirmation email.
    - Manager and employe accounts are not fully functional.

Test Case #2:

Description: To verify that users can login and register an account on the website.

Preconditions:
- The user must be on registration and user account information pages of the website.
- The user must click on the register and insert (data) buttons.

Postconditions/Expected Outcome:
- The user's email and user information (i.e. home address) is inputted into the FireBase database.

Test Execution:
- Pass criteria:
    - Users can successfully login and register an account on the website.
- Fail Criteria:
    - No email verification when a user signs up as a registered user.

Test Case #3:

Description: To verify the user has successfully completed the payment process and payment confirmation.

Preconditions:
- The user must be on the payment page and have a valid credit card.

Postconditions/Expected Outcome:
- The user is able to successfully book a hotel room.

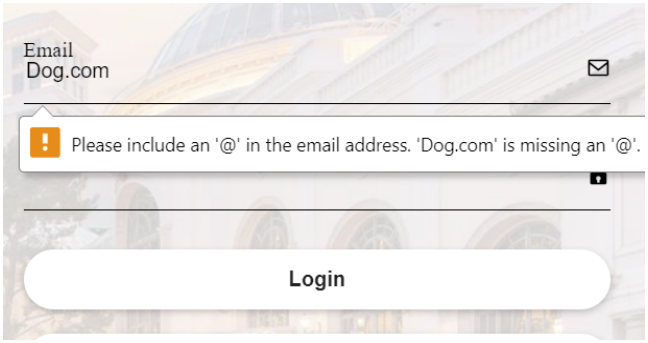
Test Execution:
- Pass criteria:

    - The user was able to complete the payment process.
- Fail Criteria:
    - Doesn't check for proper credit card input on the payment page.

Boundary Test Cases

Test Case #1:

Description: To verify the behavior of the login page when boundary input conditions are met or exceeded.



Precondition:

- The user is on the login page and payment page of the website.

Postcondition/Expected Outcome:

- The user is either successfully logged in or receives appropriate error messages based on input conditions.

Test Execution:

- Pass Criteria:
    - Error messages are displayed appropriately for invalid input.
    - Valid inputs are accepted and the login process does not have any issues.
- Fail Criteria:
    - N/A

Test Case #2:

Description: To verify that the user can only choose future dates for their reservation.

Precondition:

- The user is on the booking page of the website.

Postcondition/Expected Outcome:

- The user cannot book the same day they decide to book their reservation and they cannot book with previous dates. They can only book for the next day and/or future dates.

Test Execution:

- Pass Criteria:
    - It allows users to only choose future dates for their reservation. Arrival date will give an error if you choose a previous date. Departure date will give you an error if you choose before the arrival date.
- Fail Criteria:
    - N/A

Test Case #3:

Description: To verify that the user can input their payment to book a hotel.

Precondition:

- The user needs to sign in or have a registered account.
- The user must confirm a booking date before payment.

Postcondition/Expected Outcome:

- Once a user has input their payment information, the website outputs a payment confirmation message. This does not include sending the user a confirmation email.

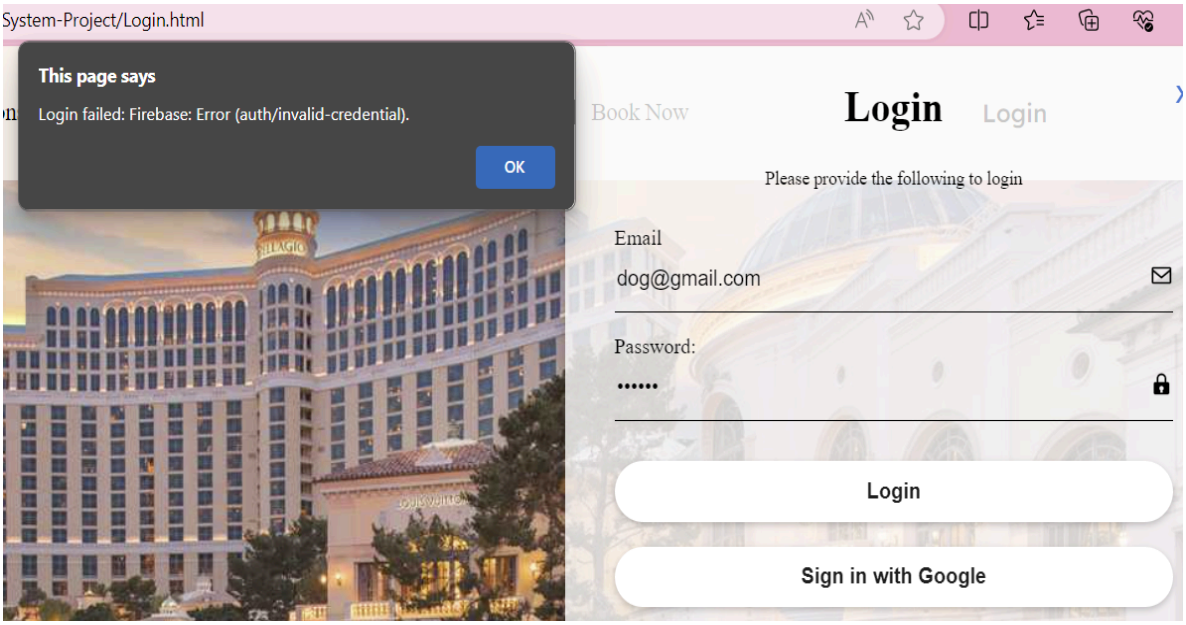
Test Execution:

- Pass criteria:
    - N/A
- Fail criteria:
    - Payment verification does not send an email to the user.
    - No limit on the amount of numbers or characters for user input.

Error Handling Test Cases

Test Case #1:

Description: To verify that appropriate error handling occurs when the user enters an incorrect password during login.



Precondition:

- User has registered an account on the hotel booking system website.
- User is on the login page of the website.

Postcondition/Expected Outcome:

- The user is logged into their account.
- An error message is displayed for incorrect password entry.

Test Execution:

- Pass criteria :
  - The correct error message is displayed when an incorrect password is entered.
  - The user is able to successfully log in after entering the correct password.
- Fail Criteria:
  - N/A

Test Case #2:

Description: To verify that the email that the user enters is not an already existing email.

Precondition:

- The user is on the email registration page.

Postcondition/Expected Outcome:

- The user is able to log into the website with an email that has not been inserted into the database.

Test Execution:

- Pass criteria:
  - The user can successfully register an account with their email.
- Fail Criteria:
  - If they used an email that is already registered, then there will be an error

message.

Test Case #3:

Description: To verify valid Hotel ID in the hotel database

Precondition:

- The hotel that the user chooses must already be in the hotel database.

Postcondition/Expected Outcome:

- The user is able to successfully book with a hotel that is in the hotel database.

Test Execution:

- Pass criteria:
    - Verifies that the Hotel ID is in the database. If the user searches for the incorrect Hotel ID, it displays an error message. If the user inputs the correct Hotel ID, the website displays the correct hotel with pricing for rooms.
- Fail Criteria:
    - If the hotel does not exist in the database, then it will not be displayed.

Findings

This Test script goes through all functionality for login, booking, payment, and confirmation. We tested for error handling with registered emails, invalid passwords, and invalid hotel IDS that are not listed within the database. Finally, the script tests the boundaries with entering an email, booking dates, and credit card number. This test script is run using Puppeteer and Jest. Before launching the script, we must run the source files with Live Server so we can get proper html urls and edit the script accordingly.

We were very ambitious with our UI design which conflicted with our time constraint. Things we were able to accomplish were allowing the user to sign in/sign and allow error messages to be displayed appropriately for invalid input. Things we were unable to complete were:

- The payment method doesn't check for valid payment
- Unable to connect to other online payment forms
- Incomplete expected booking process (i,e. you don't see the booking email)

Recommendations

For our third boundary test case, we failed our test for verifying that the user can input their payment to book a hotel. A way we could fix this would be to create code that can verify the digits on a valid credit card. The first six to eight digits of a credit card number is the Issuer

Identification Number (IIN). These numbers uniquely identify the card issuer so merchants and payment processing networks know who issued the card. By having code that verifies these six to eight digits, we can allow the user to successfully input their payment to book a hotel while verifying if the credit card is validated and properly approved by their bank.

The following is a list of additional ways to improve the quality of our application:

- Adding a word/number limit to the text boxes
- Add a key sensitive feature to password
- Ensuring a way that managers/employees have direct and easy access to the FireBase database to retrieve user information
- Adding more interactive features (i.e. clicking on interactive features of newsletters to receive additional information about events or redirecting to external links about local tourist attractions)
- Generate an email template that will be sent out to users that matches the booking receipt that users can view on our application