PineApple

Version 1.0.2

Testing Document

GroupB2

1155173962	DONG Yi
1155173753	SU Yihang
1155173943	YANG Han
1155173774	LUAN Changwen
1155173835	LIANG Leyan

Department of Computer Science and Engineering

The Chinese University of Hong Kong

May 4, 2024

Table of Contents:

1 TEST PLAN

1.1 Test Approaches

Since the shopping mall system is an intricate system that integrates plenty modules and multiple functions, a meticulous testing strategy is essential. Therefore, we will utilize module testing and bottom-up integration. This strategy allows for thorough examination of each individual module or function in isolation before integration. To ensure comprehensive coverage and reliability of the shopping mall system, we will simultaneously adopt white box and black box testing technologies. White box testing will be applied to smaller modules and individual features to ensure each component operates efficiently and correctly. Black box testing will be applied to larger and more complex modules to validate the system's functionality from the user's perspective.

1.2 Features to be Tested

To determine the robustness and functionality of our modules, we adopted a meticulous approach in our testing strategy to identify critical and valuable test cases. For black box testing, we will employ boundary value analysis and error guessing as the main strategies. Boundary value analysis focuses on examining edge cases of the input domain, including minimum and maximum values and values that exceed these limits. This method can effectively monitor the correctness of some functions that accept digital input, and can help find problems in programs under extreme input ranges.

On the other hand, we can use our experience and intuition to make bad guesses and predict and test for potential errors, such as submitting a form with empty inputs or incorrect data. This strategy helps detect unusual errors that may not be identified by more systematic methods.

We will implement statement, branch, and condition coverage criteria for white-box testing. When it comes to statement coverage, we ensure that every statement in the code is executed at least once, helping to pinpoint any parts of the code that are inaccessible or not executed during testing. Branch coverage aims to ensure that both possible outcomes of each branch are executed at least once. That is, the truth and falsehood of each judgment are executed at least once to ensure that all decision points in the code are evaluated. Finally, conditional coverage tests every possible outcome of every logical expression in your code, enhancing testing depth by covering all logical branches and compound conditions.

1.3 Testing Environments

The testing environment should have the necessary hardware and software to run the PineApple website, including an appropriate operating system, web browser, and database software. In our test, we will use the Microsoft Windows OS, the Chrome browser, and WampServer, which is a Windows web development platform for dynamic web applications using the Apache2 server, the PHP scripting language, MySQL database and also MariaDB. We will utilize the PhpMyAdmin page in WampServer, which has a MySQL Workbench GUI, to more easily manage the MySQL database of PineApple.

2. Test Sets

2.1 Case-1: User Registration

2.1.1 Purpose

This set is to test whether new users can successfully register a new account in Pineapple. Specifically, we would like to verify the following corner cases using the black-box testing techniques:

(1) When one or more input data are empty
(2) When the E-mail is invalid
(3) When the E-mail has already been used for another registered account
(4) When the input data for confirm password and password do not match
(5) When the username already exists

2.1.2 Inputs

We assume the initial database contains no user data. This is the format of sample inputs in each test case: {"Username", "E-mail", "Password", "Confirm Password"}, and the data type of every input is string.

Test Cases	Inputs (explanations)
1	{"", "", "", ""} (test all empty inputs)
2	Inputs with at least one place left empty while all the other inputs are legal (test one or more empty inputs, since the number of test cases is too large and trivial, we omit the details)
3	{"Mamba", "@not email", "0824", "0824"} (test invalid email)
4	{"Mamba", "whatcanisay@man.com", "0824", "0824"}, {"Kobe", "whatcanisay@man.com", "2408", "2408"} (register twice to test already existing email)
5	{"Mamba", "whatcanisay@man.com", "0824", "2408"} (test confirm password and password do not match)
6	{"Mamba", "whatcanisay@man.com", "0824", "0824"}, {"Mamba", "manba@out.com", "2408", "2408"} (register twice to test already existing username)
7	{"Mamba", "whatcanisay@man.com", "0824", "0824"} (test all valid inputs)

2.1.3 Expected Outputs & Pass/Fail Criteria

Pass/Fail Criteria (the order of the criteria follows the above testing cases):

(1) Pass: The registration failed and an “All the inputs must be filled” warning appears. The user has to register again.
    Fail: The registration is successful, or no warning message is shown.
(2) Pass: The registration failed and an “All the inputs must be filled” warning appears. The user has to register again.
    Fail: The registration is successful, or no warning message is shown.
(3) Pass: The registration failed and an “Invalid E-mail” warning appears. The user has to register again.
    Fail: The registration is successful, or no warning message is shown.
(4) Pass: The first registration is successful. After logging out, register again. The second registration failed and an “E-mail already exists” warning appears. The user has to register again.
    Fail: The first registration is failed, or the second registration is successful, or no warning message is shown after the second registration.
(5) Pass: The registration failed and a “Confirm password and password do not match” warning appears. The user has to register again.
    Fail: The registration is successful, or no warning message is shown.
(6) Pass: The first registration is successful. After logging out, register again. The second registration failed and an “Username already exists” warning appears. The user has to register again.
    Fail: The first registration is failed, or the second registration is successful, or no warning message is shown after the second registration.
(7) Pass: The registration is successful and the user is redirected to the login page.
    Fail: The registration is failed, or the user is not redirected to the login page, or any warning message appears.

Expected Outputs:

Test Cases	Expected Outputs
1	The registration failed and an “All the inputs must be filled” warning appears.
2	The registration failed and an “All the inputs must be filled” warning appears.
3	The registration failed and an “Invalid E-mail” warning appears.
4	The first registration is successful. The second registration failed and an “E-mail already exists” warning appears.
5	The registration failed and a “Confirm password and password do not match” warning appears.
6	The first registration is successful. The second registration failed and an “Username already exists” warning appears.
7	The registration is successful and the user is redirected to the login page.

2.2 Set-2: Login

2.2.1 Purpose

This set is to test both the user login function and the administrator login function. Since both of these two logins require similar inputs which are username and password, the test cases discussed below will be shared for these two testing tasks. In detail, we would like to test the following cases using the black-box testing techniques:

(1) When one or more input data are empty
(2) When the username does not exist
(3) When the password is not correct

2.2.2 Inputs

This is the format of sample inputs in each test case: {“Username”, “Password”}, and the data type of every input is string.

Test Cases	Situation before Input	Inputs (explanations)
1	There is a registered user with username “Mamba” and password “0824”.	{“Mamba”, “0824”} (test valid inputs)
2	There is a registered user with username “Mamba” and password “0824”.	{“”, “”} and {“Mamba”, “”} and {“”, “0824”} (test one or more empty inputs)
3	There is a registered user with username “Mamba” and password “0824”. But no registered user with username “Kobe”.	{“Kobe”, “0824”} (test invalid username)
4	There is a registered user with username “Mamba” and password “0824”.	{“Mamba”, “2408”} (test invalid password)

2.2.3 Expected Outputs & Pass/Fail Criteria

Pass/Fail Criteria (the order of the criteria follows the above testing cases):

(1) Pass: The login is successful and the page is redirected to the user/admin homepage.
Fail: The login is failed, or the page is not redirected to the user/admin homepage, or any warning message is shown.
(2) Pass: The login fails and an “All inputs must be filled” warning message is shown. The user/admin has to login again.
Fail: The login is successful or no warning message is shown.
(3) Pass: The login fails and a “Username does not exist” warning message is shown. The user/admin has to login again.
Fail: The login is successful or no warning message is shown.
(4) Pass: The login fails and an “Incorrect password” warning message is shown. The user/admin has to login again.
Fail: The login is successful or no warning message is shown.

Expected Outputs:

Test Cases	Expected Outputs
1	The login is successful and the page is redirected to the user/admin homepage.
2	The login fails and an “All inputs must be filled” warning message is shown.
3	The login fails and a “Username does not exist” warning message is shown.
4	The login fails and an “Incorrect password” warning message is shown.

2.3 Case-3: Search Product

2.3.1 Purpose

To test the functionality of searching for products in our online mall, and to ensure that users can filter their favorite products based on keywords and a variety of conditions, we combine keywords and search conditions together to make the search function more powerful.
In the search by the combination of the keywords and conditions, we test the following controls by using black-box testing:

(1) Whether it will display correct product list when the user searches by empty string. (Our search design is if user inputs empty string, search page will display all products)
(2) Whether it will display correct product list when the user searches by a random string like “xyzwww”.
(3) Whether it will display correct product list when the user searches by valid inputs. Anything other than special characters and random string will be considered as valid input.
(4) Whether it can display correct product when the user searches by the product ID. (Our search design is the product can be searched by inputting product ID)
(5) Whether the search function is case-sensitive. (Our search design is not case-sensitive)
(6) Whether it can display correct product when the user searches by conditions.
(7) Whether it can display correct product when the user searches by using the combination of the keyword and valid conditions.
(8) Whether it can display correct product when the user searches by using the combination of the keyword and contradictory conditions.

2.3.2. Inputs

Note that there are several assumptions in this part. Users cannot search for the product information using the following: (a) the sales of the product, (b) the rate of the product or (c) the availability of the product

This is a sample format of input in each test case:
{“keywords”, “condition”}, where the data type of the above inputs is string and there may be more than one condition in the condition part, separated by a separator character ‘/’.

Test Cases	Input (explanations)
1	{“”} (test with empty string)
2	{“xyzwww”} (test with random string)
3	{“Mamba”} (test with valid string)
4	{“24”} (test with product ID)
5	{“mamba”} (test with the case sensitivity of the function)
6	{“Mamba”, “Price:100-200”} (test with the combination of keyword and single condition)
7	{“Price:100-200”} (test with single condition)
8	{“Price:100-200, Category: sports”} (test with multiple conditions)
9	{“Mamba”, “Price:100-200, Category: sports”} (test with the combination of keyword and multiple conditions)
10	{“Mamba”, “Price:200-100”} (test with the combination of keyword and contradictory condition)

2.3.3. Expected Outputs & Pass/Fail Criteria

Pass/Fail Criteria (note the order follows the above testing purpose):

(1) Pass: All the products will be displayed since all our products contain the empty string and for the overview requirement.
Fail: Display nothing or some error messages.

(2) Pass: The statement “No available product is found” will be displayed since there are no products including the random keyword.
Fail: Display a list of n product information, where 1<= n < max number of products.

(3) Pass: Display a list of n product information containing the keyword, where 1<= n < number of products retrieved by the backend server.
Fail: Display nothing but the statement “No available product is found” or other error messages.

(4) Pass: Display the product information which has corresponding product ID
Fail: Display wrong product information or the statement “No available product is found”.

(5) Pass: Display a list of n product information, where 1<= n < number of products retrieved by the backend server. For example, two inputs “mamba” and “Mamba” should give the same results.
Fail: Display nothing, the statement “No available course is found” or even the server is down since it may not be able to switch between upper and lower case.

(6) Pass: Display a list of n product information containing the keyword and fulfilling

the condition, where 1<= n < number of products retrieved by the backend server.
Fail: Display nothing, the statement “No available course is found” or a product information list that does not contain the keyword or does not fulfill the conditions

(7) Pass: Display a list of n product information fulfilling the condition, where 1 <= n < number of products retrieved by the backend server.
Fail: Display nothing, the statement “No available course is found” or a product information list which not fulfill the conditions.

(8) Pass: Display a list of n product information fulfilling the conditions, where 1 <= n < number of products retrieved by the backend server.
Fail: Display nothing, the statement “No available course is found” or a product information list which not fulfill the conditions.

(9) Pass: Display a list of n product information containing the keyword and fulfilling the conditions, where 1<= n < number of products retrieved by the backend server.
Fail: Display nothing, the statement “No available course is found” or a product information list that does not contain the keyword or does not fulfill the conditions

(10) Pass: Display the statement “You should input valid condition” since the conditions are contradictory.
Fail: Display a list of product information retrieved by the confused backend server.

Expected Outputs:

Test Cases	Expected Outputs
1	All the products will be displayed since all our products contain the empty string and for the overview requirement.
2	The statement “No available product is found” will be displayed since there are no products including the random keyword.
3	Any products that contain the string “Mamba” will be displayed
4	The product with the Product ID “24” will be displayed
5	Any products that contain the string “mamba” will be displayed and the output will be the same to input “Mamba”
6	Any products that contain the string “Mamba” and its price lies in the range [100, 200] will be displayed
7	Any products whose price lies in the range [100, 200] will be displayed
8	Any products whose price lies in the range [100, 200] and category is “sports” will be displayed
9	Any products that contain the string “Mamba” and its price lies in the range [100, 200] and its category is “sports” will be displayed
10	The statement “You should input valid condition” will be displayed because the condition is invalid

2.4 Case-4: Add Product to Cart

2.4.1. Purpose

To test the add product to cart functionality in product detail page, ensuring that user can successfully add the product to cart, valid and displayable on the website. In specific, we would like to test the following controls using black-box testing:

(1) Whether the process can correctly handle add a product into cart.
(2) Whether the process can correctly handle the case when the product is out of stock.
(3) Whether the process can correctly handle the case when 0 product is selected

2.4.2. Input

The input follows the following format:
{“product ID”: int, “number”: int}.

Test Cases	Input (explanations)
1	{“1”, “140”} (test with insufficient inventory due to excessive purchase quantity)
2	{“2”, “”} (test with no product quantity selected)
3	{“3”, “1”} (test with the product is already out of stock)
4	{“4”, “1”} (valid case)

2.4.3. Pass/Fail Criteria & Expected Outputs

(1) Pass: The system will check whether the number of products in stock is larger or equal to the number. If not, return an error message and reject the request.
Fail: Accept the request.
(2) Pass: Since the select quantity is 0, the system will reject the add to cart request and return an error message.
Fail: Accept the request.
(3) Pass: The system will check whether the number of products in stock is 0. If yes, return an error message and reject the request.
Fail: Accept the request.
(4) Pass: Accept the add to card request, add the product to the Cart Database, and redirect to shopping cart page.
Fail: Reject the request.

Expected Outputs:

Test Case	Expected outputs
1	The statement “Due to excessive purchase quantity, the product is out of stock, please reduce the quantity!” will be displayed in red color. The webpage will still be the product info page, waiting for new input.
2	The statement “Select a quantity!” will be displayed in red color.
3	The statement “Product out of stock!” will be displayed in red color. The webpage will still be the add to cart page, waiting for

	new input.
4	Redirect to shopping cart page, and a new record is inserted to the Cart Database.

2.5 Case-5: Checkout Shopping Cart

2.5.1. Purpose

To test checkout shopping cart item functionality in shopping cart page, ensuring that user can successfully pay products and generate orders. In specific, we would like to test the following controls using black-box testing:
(1) Whether the process can correctly handle pay a product in shopping cart.
(2) Whether the process can correctly handle the case when more than one product is selected
(3) Whether the process can correctly handle the case when the product is out of stock.

2.5.2. Input

The input follows the following format:
{“product ID and number”: list, “payment method”: string}. “product ID and number” is a list of dictionary with keys of product ID and number.

Test Case	Input (explanations)
1	{“[4, 10]”, “Alipay”} (test with insufficient inventory due to excessive purchase quantity)
2	{“”} (test with no product selected)
3	{“[4, 1]”, “WeChat”} (valid case)
4	{“[1, 1], [2, 1]”, “WeChat”} (valid case)

2.5.3. Pass/Fail Criteria & Expected Outputs

Pass/Fail Criteria (the order of the criteria follows the above testing cases):
(1) Pass: The system will check whether the number of products in stock is larger or equal to the number. If not, reject the request.
Fail: Accept the request.
(2) Pass: As there is nothing in the shopping cart, the system will reject the request.
Fail: Accept the request.
(3) Pass: Accept the pay shopping cart request, add the product and number to the Order Database, and redirect to order page.
Fail: Reject the request.
(4) Pass: Accept the pay shopping cart request, add the product and number to the Order Database, and redirect to order page.
Fail: Reject the request.

Expected Outputs:

Test Case	Expected Outputs
1	The webpage will be redirected to the shortage page to inform the

	user of the short supply.
2	The webpage will be redirected to the homepage.
3	Redirect to the order page, and a new record is inserted to the Order Database.
4	Redirect to the order page, and two new records is inserted to the Order Database.

2.6 Case-6: User Edit Information

2.6.1. Purpose

To test the edit & change user information functionality in personal info page, ensuring that user can successfully change their information, valid and displayable on the website. In specific, we would like to test the following controls using black-box testing:
(1) Whether the process can correctly handle incomplete personal information.
(2) Whether the process can correctly handle invalid input format.
(3) Whether the process can correctly handle invalid input email and phone number.

2.6.2. Input

The input follows the following format:
{“name”: str, “email”: str, “phone number”: int, “password”: str}.

Test Case	Input (explanations)
1	{“”, “”, “”, “”} (test with empty input)
2	{“Mamba”, “”, “56783443”, “”} (test with random empty entries)
3	{“CAI Xukun”, “user@@domain.com”, “69559858”, “Liangleyanjuangou666”} (test with invalid email)
4	{“Kunkun”, “user@gmail.com”, “68779”, “changwenjuanGou666”} (input with invalid length of phone number which is less than 8)
5	{“LIANG Leyan”, “Juangou@gmail.com”, “46534221”, “666”} (Test with a weak password)
6	{“LIANGLeyanshizhendejuan,zenmekeyi”, “wuming@163.com”, “78669900”, “Linytre5563”} (Test with a long string input)
7	{{“Kunkun”, “user@gmail.com”, “68779wwe”, “leyanjuanGou666”} (input with invalid type of phone number)
8	{“Kobee”, “Laoda@163.com”, “2556344334”, “TTssd99099”} (input with invalid length of phone number which is longer than 8)
9	{“CAI Xukun”, “usedomaincom”, “69559858”, “Liangleyanjuangou666”} (test with invalid email)
10	{“Luan changwen”, “wule@163.com”, “46938021”, “Woshijuan666”} (valid input)

2.6.3. Pass/Fail Criteria & Expected Outputs

Pass/Fail Criteria (the order of the criteria follows the above testing cases):
(1) Pass: Since the information is not complete, the system will reject the edit request

and return an error message.
Fail: Accept the request.

(2) Pass: Since the information is not complete, the system will reject the edit request and return an error message.
Fail: Accept the request.

(3) Pass: Since the email is not correct, the system will reject the edit request and return an error message.
Fail: Accept the request.

(4) Pass: Since the phone number is not correct, the system will reject the edit request and return an error message.
Fail: Accept the request.

(5) Pass: Since the password is too weak, the system will reject the edit request and return an error message.
Fail: Accept the request.

(6) Pass: Since the username is too long, the system will reject the edit request and return an error message.
Fail: Accept the request.

(7) Pass: The system will check whether the phone number entry is an integer number. If not, return an error message and reject the request.
Fail: Accept the request.

(8) Pass: Similar to case 3, since the email format is wrong, the system will reject the edit request and return an error message.
Fail: Accept the request.

(9) Pass: Accept the request.
Fail: Reject the request.

Expected Outputs:

Test Case	Expected outputs
1	The statement "Please fill in the information, all information has not been filled in!" will be displayed in red color. The webpage will still be the personal info page, waiting for input.
2	The statement "Please fill in the information, email and password are required!" will be displayed in red color. The webpage will still be the personal info page, waiting for input.
3	The statement "Please enter correct email!" will be displayed in red color. The webpage will still be the personal info page, deleting the invalid email waiting for new input.
4	The statement "Please enter correct phone number with length 8!" will be displayed in red color. The webpage will still be the personal info page, deleting the invalid phone number waiting for new input.
5	The statement "Please enter a stronger password with at least one lowercase letter, one uppercase letter and a number with length no less than 8 digits!" will be displayed in red color. The webpage will still be the personal info page, deleting the already input

	password waiting for new input.
6	The statement "Username too long, please reinput!" will be displayed in red color. The webpage will still be the personal info page, deleting the invalid username waiting for new input.
7	The statement "Invalid type of phone number!" will be displayed in red color. The webpage will still be the personal info page, deleting the invalid phone number waiting for new input.
8	The statement "Please enter correct email!" will be displayed in red color. The webpage will still be the personal info page, deleting the invalid email waiting for new input.
9	Redirect to login-and-security page, and the corresponding user information will be updated to the User Database.

2.7 Case-7: Admin Add Product

2.7.1. Purpose

To test the add product functionality in admin page, ensuring that the product information is complete, valid and displayable on the website. In specific, we would like to test the following controls using black-box testing:

(1) Whether the process can correctly handle incomplete product information.
(2) Whether the process can correctly handle invalid input format.
(3) Whether the process can correctly handle buggy html content in the input.

2.7.2. Input

The input follows the following format:
{"name": str, "category": select box with default choice as "Empty", "description": str written in html, "price": float, "stock": int, "image": image file}.

Test Cases	Input (explanations)
1	{"Su 7", "Car", "<div>This is a good car. </div>", "299999.00", "5000", "image.png"} (valid input)
2	{"", "", "", "", "", ""} (all empty)
3	{"Su 7", "Empty", "<div>This is a good car. </div>", "299999.00", "5000", "image.png"} (category is not selected)
4	{"Su 7", "Car", "", "299999.00", "5000", ""} (input with random empty entries)
5	{"Su 7", "Car", "<div>This is a good car.", "299999.00", "5000", "image.png"} (description has incorrect html grammar so that it cannot be displayed on the product details page)
6	{"Su 7", "Car", "<div>This is a good car. </div>", "299bb9.0a", "5000", "image.png"} (price is not a float type)
7	{"Su 7", "Car", "<div>This is a good car. </div>", "299999.00", "5000.5", "image.png"} (stock is not an int type)
8	{"Su 7", "Car", "<div>This is a good car. </div>", "299999.00",

	“5000”, “wrong_file.txt”} (wrong image file type)
9	{“Su 7”, “Car”, “<div>This is a good car. </div>”, “299999.00”, “5000”, “high_res.png”} (image file too large)

2.7.3. Pass/Fail Criteria & Expected Outputs

(1) Pass: Accept the new product request, add the product to the Product Database, and return to admin main page.
Fail: Reject the request.

(2) Pass: Since the information is not complete, the system will reject the add request and return an error message.
Fail: Accept the request.

(3) Pass: The system will check whether the category entry is selected. If it is still the default value (i.e., no category is selected), the system will reject the request and return an error message.
Fail: Accept the request.

(4) Pass: Similar to case 2, if any of the entries are empty, the system will reject the request and return an error message.
Fail: Accept the request.

(5) Pass: The system should check the grammatical correctness of the description content. As the description content will be directly inserted into the product detail html page, it should contain no grammatical mistakes to be displayed correctly. If it has grammatical errors, reject the request and return an error message.
Fail: Accept the request even if there are mistakes in the description content.

(6) Pass: The system will check whether the price entry is a positive float number. If not, return an error message and reject the request.
Fail: Accept the request.

(7) Pass: Similar to case 6, the stock should be positive integer value. If not, return an error message and reject the request.
Fail: Accept the request.

(8) Pass: The system should check whether the uploaded image is of supported type such as jpg and png, which can be displayed on most of the web browsers. If not, return an error message and reject the request.
Fail: Accept the request.

(9) Pass: The system should ensure that the uploaded image doesn’t exceed the size limit to reduce the storage consumption in the database. If the image is larger than 5 Mb, the system rejects the request and returns an error message.
Fail: Accept the request even if the image is very large.

Expected Outputs:

Test Cases	Expected Outputs
1	Redirect to admin main page, and a new product item is inserted to the Product Database.

2	The statement “Information incomplete!” will be displayed in red color.
3	The statement “Category is not selected!” will be displayed in red color. The webpage will still be the add product page, and the filled information about the product will be kept.
4	The statement “Information incomplete!” will be displayed in red color. All the filled content will be kept in the add product page, and user needs to complete the missing entries and submit again.
5	The statement “Description has grammatical errors!” will be displayed in red color. All the filled content, including the buggy description content, will be kept in the add product page.
6	The statement “Invalid price value!” will be displayed in red color. All the filled content, except the price, will be kept in the add product page.
7	The statement “Invalid stock value!” will be displayed in red color. All the filled content, except the stock, will be kept in the add product page.
8	The statement “Wrong image type!” will be displayed in red color. All the filled content, except the image file, will be kept in the ad product page.
9	The statement “Image file too large!” will be displayed in red color. All the filled content, except the image file, will be kept in the ad product page.