|  |
| --- |
| Team Alpaca |
| Software Quality Assurance |
| Task 1: Planning the test |

Contents

[Planning the test 2](#_Toc115448029)

[Test Strategy 2](#_Toc115448030)

[The objectives of the test 2](#_Toc115448031)

[Context of the test 2](#_Toc115448032)

[Scope 2](#_Toc115448033)

[Techniques 2](#_Toc115448034)

[Key performance Indicators (KPIs) 3](#_Toc115448035)

[Black-box testing guide: 3](#_Toc115448036)

[Cause-Effect Graph 5](#_Toc115448037)

[White-Box testing guide: 6](#_Toc115448038)

[Unit testing 6](#_Toc115448039)

[Integration testing 6](#_Toc115448040)

[Figure 1: Causes of a possible failed purchase 5](file:///C:\Users\Max%20Naidoo\Desktop\BIT\Modules\IITP102\Practicals\Group%20work\Task1_Planning_the_test.docx#_Toc115447653)

|  |
| --- |
|  |

# Planning the test

# Test Strategy

## The objectives of the test

This document describes the plan for testing the Drinks2You prototype. It will support the following objectives:

* Identifying the software that will be tested.
* Recommending the testing techniques that will be used.
* Using the different levels of software testing requirements.
* Listing the elements that will be tested.
* Determining appropriate outcomes with the necessary input

## Context of the test

The user/development team will be testing the mobile application Drinks2You.

## Scope

The applications interface and functions will be tested. Testing of the source code will help determine how well integrated the system is. The login system will be tested separately from the rest of the app. Navigation system as well as the finance system will be the interfaces which will be tested.

This will be a high level of testing which includes:

**Unit Testing:** This is the first level of testing. Each unit of the code will be tested to see if it performs correctly. Inputs being tested with a desired output and how often errors will occur.

**Integration Testing:** The flow of data from each screen in the app will be tested. Checking if user data is stored correctly. Testing if the code for the billing and payment modules operates correctly as a whole.

**System Testing:** Application is tested and whether it is functional or not as a user. The GUI is tested on whether the buttons operate correctly as planned. If the application works according to the functional and non-functional requirements.

**Acceptance testing:** The users and testers will confirm if the system is ready to use. Once the test is compared to the initial requirements, and if expectations are met then feedback from the user is given. This allows the software to receive more corrections.

## Techniques

The testing will follow the black-box and white-box testing methodology. Techniques include :

* Boundary Value Analysis (BVA)
* Cause-Effect testing
* State Transition

## Key performance Indicators (KPIs)

The KPIs that we will be using to measure the growth of the product will mainly be the growth of monthly users who make use of the application.

# Black-box testing guide:

The user will initially be presented with the “Phone screen before sign up” screen, where the actual phone’s home screen is shown with different app icons, including Drinks2You. He/she will be instructed to select the “Drink2You” app icon.

The user will be taken to the “Welcome screen” screen. This is where he/she will create a new account on the Drinks2You app. He/she will be instructed to select the “Sign up” button next to the “Don’t have an account?” text.

The user will be taken to the “Sign up button selected” screen. Theoretically, the user will then enter his/her first name, surname, ID number and email address. He/s will then create a secure password, and then confirm the new password by retyping it in the “Confirm Password” text box. For this test, the user will be instructed to simply select the “Next” button at the bottom centre of the screen.

The user will be taken to the “Account address insert” screen. Theoretically, the user will then enter his/her address (home address, city/town and zip code). For this test, the user will be instructed to simply select the “CREATE” button at the bottom right corner of the screen.

The user will be taken to the “Home screen” screen. This screen will show the user the current day’s deals as well as the current best sellers. He/she will be instructed to select the “Browse” button at the bottom of the screen.

The user will be taken to the “Browse screen”. The user is presented with a list of all the different types of drinks available on the app. He/she will be instructed to simply select the “Beer” category.

The user will be taken to the “Beer category” screen. Here the user is presented with different beers to buy. He/she will be instructed to simply select the first tile, right beneath the “Beer” header, i.e. the first listed beer.

The user will be taken to the “Beer selected” screen. Here the user is presented with a pop-up window that shows the selected item, in this case beer, its name, description and price. He/she will be instructed to select the “Add” button in the centre of the pop-up window.

The user will be taken to the “Beer added to cart” screen. The user is now presented with the beer category again. He/she will be instructed to select the “search” bar at the top of the screen.

The user will be taken to the “Select search bar” screen. Here the phone’s keyboard is presented for the user to type. He/she will be instructed to simply select the “search” bar again.

The user will be taken to the “Type drink in search bar” screen. Theoretically, the user will then type any desired drink’s name. He/she will be instructed to simply select the ”X” symbol/button in the right of the “search” bar, with the text “Flying Fish” theoretically typed in.

The user will be taken to the “Back to beer category (pressed X)” screen. He/she will be instructed to simply select the cart icon at the top right corner of the screen.

The user will be taken to the “Shopping cart” screen. Here the user will be presented with the contents of his/her cart in detail, including the drink’s name, description, and price. He/she will be instructed to select the “Proceed to checkout” button at the bottom right corner of the screen.

The user is taken to the “Proceed to checkout button selected (No address)” screen. Theoretically, the user will then have to insert his/her name, as well as the delivery address. For this test, the user theoretically only inserted his/her name. He/she will be instructed to simply select the “GO TO PAYMENT” button at the bottom centre of the screen.

The user will be taken to the “Go to payment button selected” screen. Theoretically, the user will then have to insert his/her bank card (credit or debit) details. For this test, the user will be instructed to simply select the “Place order” button at the bottom centre of the screen.

The user will be taken to the “Shopping cart purchase error” screen. Here, the user will be presented with a pop-up error message saying “Deliver address is empty” and a button below saying “Go back”. He/she will be instructed to simply select the “Go back” button.

The user will be taken to the “Proceed to checkout button selected (Address inserted)” screen. For this test, the user theoretically enters his/her delivery address as well as his/her name. The user will be instructed to simply select the “GO TO PAYMENT” button once again.

The user will be taken to the “Go to payment button selected success” screen. Here, the user will theoretically enter his/her card details again. He/she will be instructed to simply select the “Place order” button at the bottom centre of the screen again.

The user will be taken to the “Successful purchase” screen. Now, instead of the previous error message, the user will be presented with a success message saying, “Success! Your order has been placed and will be on its way soon!”. He/she will be instructed to simply select the “Back to home” button on the pop-up success message.

The user will now be taken to the “Home screen after purchase” screen. He/she will now be freely allowed to explore the app.

## Cause-Effect Graph

Figure 1: Causes of a possible failed purchase

# White-Box testing guide:

## Unit testing

On the sign-in (and sign-up) screen when the user is logging in, we will look if the relative error messages are added in a case the user inputs the wrong email and check if the user entered an existing domain name on the username/email input space. For example, xxxxxx@gmail.com/mandela.ac.za etc. Additionally, we will also check if the password entered is the same as the one currently saved password in the app database.

When the user gets to the payment and billing screen, he or she will be required to fill in their payment information. They will find text boxes that require them to enter their account name, number, and card number just to name a few.

•Account Name: The user needs to fill in their details with string data input, or else, it’ll display an error message alerting the user of the error they have made. The tester will then have to check if the string data input is saved and is stored.

•Account Number: The user will be required to enter their account number using the integer data type. The required length needs to be equal to ten, if it’s less than or greater than ten, it will not be accepted. If the user also includes alphanumeric characters, it will display an error message.

## Integration testing

When the user is creating account there will be checking of the datatype (string and integers) used for that particular input textbox and the length of the input needed like the ID number and passwords (initial & confirmation entry).

On the payment and billing page, the customer can provide complete details and when the payment is done, it will take the customer back to the site page.

All the stored information from each unit carries over to the payment screen or is saved,