Task 1: Planning the test

**BLACK BOX TESTING**

Our virtual home designing app using black box test is essential to ensure its functionality and friendly user interface. In black box testing, is this method it focuses on testing the application's functionality without knowing its internal Structure. And the are many approaches to do Black boxing.

Here's are few guidelines on how we can perform black box testing:

Understanding the Software Requirements:

Having a clear understanding about the app's requirements, before you begin testing features, and expected performance. This serves as the basis for test cases.

Create Test Cases:

Make a complete set of test cases that will cover all aspects of the prototype functionality. These Test cases should include input data and expected results.

Consider various user roles, such as designing a room, saving designs, sharing designs, and so on.

Execute Test Cases:

Complete test cases step by step, inputting the test data and comparing the expected results with the actual results.

Write down errors and issues you encounter during testing.

**Boundary Value Analysis for Blackbox Testing**

For the black box testing, the boundary value analysis testing will be used. Input values at the boundaries of the different screens will be tested, specifically the minimum and maximum values.

Boundary Value Analysis testing will be applied to the following screens.

* Login Screen and Sign-up screen: Boundary input values will be tested such as the minimum and maximum number of characters for password length. Password lengths exceeding the maximum length as well as no passwords entered at all, will be tested.  
  Expected outcomes for valid passwords should be successfully creating an account, storing the username and password in a database and being able to log in successfully.  
  Expected outcomes for invalid passwords should display an error message and prevent users from creating accounts and logging in.
* Virtual Room creation screen: The maximum and minimum values of the allowed room dimensions will be tested as well as values exceeding the maximum and values preceding the minimum. The maximum number of furniture will be tested and the number of furniture items exceeding the maximum.   
  Expected outcomes should be a room generating correctly for valid room dimensions while room dimensions outside these boundaries should be rejected. When the maximum number of furniture is reached, the interface should prevent the user from adding more furniture by disabling the furniture selection.
* Furniture and Paint Catalogue screen: Furniture items that are sold out will be tested to determine what happens when they are added to the payment interface.  
  Expected outcomes should be that sold-out furniture items should be disabled from being selected.
* Payment interface: Test different card numbers, such as valid and invalid card numbers. Valid and invalid expiry dates and CVV’s should be tested.  
  Expected outcomes should be that the payment button is disabled when invalid card details are entered. It should only be enabled when valid card details are entered. The payment will then be processed when the payment button is clicked.

Test purchasing dates at the boundaries to verify the purchases made. The expected outcome should be an interface that displays the corrected purchasing history for the correct dates selected.

**Whitebox Testing**

Unit testing - is a software testing method which concentrates on examining each of the elements of a software program in independence. In unit testing, every unit is often a small, independent piece of software, such as an operation, technique, or object. The fundamental purpose of unit testing is to guarantee that these separate units of software work properly and generate the desired outcomes for varied inputs.  
  
Concentrate on testing particular features or techniques inside the app's code. For a home interior design software, this could entail testing performs accountable for inserting furniture, adjusting colours, or visualizing 3D models. Check that these routines work as intended and manage various inputs accurately.  
  
TECHNIQUES ARE AS FOLLOWS:

**1. Security Testing** - Examine the app's security procedures, such as how it manages logins for users, confidentiality of information, and manages accessibility. Discover and fix any possible bugs in the source code that may result to privacy violations.  
**2. Data Flow Testing** - Analyse how data flows through the app. Verify that data is properly transmitted among elements, safely kept, and presented properly. Pay close consideration to the data verification, preservation, and querying operations.  
**3. Integration Testing** - Examine the way various sections or parts of the software cooperate with one another. Examine how the furniture positioning component interacts with the 3D visualization component, or how changes in room size effect furniture location.

To ensure the quality and effectiveness of our application, we will be implementing Dynamic Analysis as a form of white-box testing. This approach involves observing the system's behaviour during runtime to determine if the program is functioning properly. We will use various techniques to perform the test and guarantee the app's performance.

The techniques are outlined as follows:

1. **Usability Testing**: The app's functionality, ease of use, and overall satisfaction will be evaluated by a user to provide feedback to identify areas of improvement.

2. **Performance Testing**: To ensure optimal performance of the app, we will assess how the app performs under high user loads, identifying issues that may affect its performance.

3. **Regression Testing**: The method is used to verify if recent changes or updates on the app have not introduced new defects. This involves re-testing the app to ensure that its functionality is continued.

Non-functional testing checks the non-functional attributes of any software performance, usability, security, and other qualities. These tests establish software quality and performance in real user conditions.

Software testing techniques

* Compatibility Testing – to determine whether or not your program is compatible with various browsers, browser versions, devices, operating systems, and OS versions. This involves investigating if the app can be used on multiple devices; for example, can a Samsung user have the same experience or utilize your app as easily as an iPhone user?
* Visual Testing- visual tests check if all elements are rendering as expected, with the right shape, size, background colour of the app, font type, and font size. This also involves testing the appearance of the app and creativity.

* Responsive Testing – responsive tests verify if the app renders well on-screen sizes and resolutions offered by different devices, mobile, desktops, and many other is appropriate. Also, the resolution of the app is very important since we are doing a HOME INTERIOR DESIGN APP.