***TEST SPECIFICATION***

***Group 11***

***Group Members:***

***Keaton Dinger,***

***Nathan Sikon,***

***LiLou DeRivera,***

***Hitesh Kukreja***

**1.0 Introduction**

This section provides an overview of the entire test document. This document describes both the test plan and the test procedure.

**1.1 Goals and objectives**

The primary goal for our test plan is to verify that our blackjack game behaves as expected under a variety of conditions, ensuring proper logic handling, state conditions, a responsive user interface, and ensuring that all edge cases are properly addressed. The objectives are to validate that the game follows the rules of a traditional Blackjack game which delivers consistent user experience.

**1.2 Statement of scope**

The test plan focuses on functional testing of the game, including features such as: bet placing, starting a new game, player actions (hit/stay), evaluation of game results (win/lose/draw), and game settings (music control).

What is not within the score of our project is performance testing (speed and memory usage), and multiplayer gameplay, as these are currently outside the bounds of the current implementation.

**2.0 Test Plan**

This section describes the overall testing strategy and the project management issues that are required to properly execute effective tests.

**2.1 Software to be tested**

Our software that we’re testing is called “Domino’s Casino”. The software operates on web browsers and includes game logic, user interface, login/account system, betting system, and settings functionality.

This version of our software currently supports single-player, one-on-one against a dealer (computer), following standard Blackjack rules.

Exclusions include multiplayer functionality and performance & load testing as it is outside the scope of the testing process.

**2.3 Testing tools and environment**

Development Platform: JavaScript used for game logic and interactivity, HTML and CSS used for webpage structure and styling

Web Browsers for Testing: Chrome, Firefox, Edge

Version Control & Code Editor: GitHub

Hosting Environment: Netlify, used for live deployment and testing of software

Testing Methodology: manual testing through direct interaction with the UI

**2.4 Test schedule**

**Betting Functionality (Day 1)**

.TC01: Place valid bet ($50)

. TC02: Place bet more than bank ($2000)

. TC03: Invalid bet amount ($0)

**Game Mechanics (Day 2)**

. TC04: Deal cards to player and dealer

. TC05: Player hits and stays

. TC06: Player hits and busts

**Game Outcome Tests (Day 3)**

. TC07: Player wins (20 vs dealer 18)

. TC07: Player loses (16 vs dealer 20)

. TC09: Tie (18 vs 18)

**Auxiliary Function Tests (Day 4)**

. TC10: Music plays during game

**Integration Testing (Day 5)**

. Run all tests in sequence to verify complete game flow

. Test transitions between different game states

**3.0 Test Cases**

This section enumerates a complete list of test cases for the software. A template for test cases is as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Description | Input | Expected Output | Pass/Fail |
| TC01 | Place valid bet | $50 | Game starts with 50$ dedcuted | Functional |
| TC02 | Place bet more than bank | 2000$ | Alert shown, no deduction | Functional |
| TC03 | Invalid bet amount | 0$ | Alert shown, no deduction | Functional |
| TC04 | Deal cards to player and dealer | Start game | 2 cards for player, dealer starts with 1 visible | Functional |
| TC05 | Player hits and stays | Hit once, then stay | Dealer plays, results displayed | Functional |
| TC06 | Player hits and busts | Hit repeatedly | Player score > 21, can’t hit again | Functional |
| TC07 | Player wins | 20 vs dealer 18 | Win message, bank updated | Functional |
| TC08 | Player loses | 16 vs dealer 20 | Loss message, bank updated | Functional |
| TC09 | Tie | 18 vs 18 | Tie message, refund bet | Functional |
| TC10 | Music plays | Turns on and stays in | Music playing in game | Functional |