Verification and Validation Report: SFWRENG 4G06

Team 9, dice_devs
John Popovici
Nigel Moses
Naishan Guo
Hemraj Bhatt
Isaac Giles

March 23, 2025

1 Revision History

Table 1: Revision History

Date	**Developer(s)**	**Change**
2025-03-09	John Popovici	Document format and reflection start
2025-03-10	Nigel Moses	Added NFR Evaluations Section
2025-03-10	Naishan Guo	Added Functional Requirements Evaluations Section
2025-03-10	John Popovici	Added traceability matrices
2025-03-10	Hemraj Bhatt	Added content to section 10 and reflection
2025-03-10	Isaac Giles	Added content to section 6, 8, and reflection
2025-03-10	Isaac Giles	Revised formatting issues
2025-03-10	John Popovici	Cleaned final references
2025-03-23	John Popovici	Peer review corrections
...	...	...

2 Symbols, Abbreviations and Acronyms

symbol	description
VnV	Verification and Validation
RX	Functional requirement number X
NFRX	Nonfunctional requirement number X
Test-X	Test number X
DC-X	Dice container unit test X
SC-X	Score calculator container unit test X
UI	User interface
PvP	Player versus player
LAN	Local area network
FPS	Frames per second
OOP	Object-oriented programming

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Functional Requirements Evaluation **1**

3.1 R1: PVP support . . . 1
3.2 R2: Yatzee score calculations . . . 1
3.3 R3: Dice Roll Physics . . . 2
3.4 R4: Dice roll results . . . 3
3.5 R5: Multi Dice support . . . 4
3.6 R6: Simultaneous turn mechanism . . . 4
3.7 R7: Dice Selection . . . 5
3.8 R8: User Interface . . . 6
3.9 R9: Custom games . . . 7
3.10 R10: Preset Selection . . . 7
3.11 R11: Local multiplayer . . . 8
3.12 R12: Singleplayer . . . 9
3.13 R13: Singleplayer Algorithm . . . 9
3.14 R14: Online Matchmaking . . . 10
3.15 R15: Saving game settings . . . 11
3.16 R16: End of game statistics . . . 11
3.17 R17: Consistent and Correct game state . . . 12

4 Nonfunctional Requirements Evaluation **14**

4.1 Introduction . . . 14
4.2 Non-Functional Requirements Evaluation Format . . . 14
4.3 Individual Non-Functional Requirement Evaluations . . . 15
4.3.1 NFR-1: Performance . . . 15
4.3.2 NFR-2: Usability . . . 16
4.3.3 NFR-3: Portability . . . 17
4.3.4 NFR-4: Reliability . . . 17
4.3.5 NFR-5: Responsiveness . . . 18
4.3.6 NFR-6: Modularity . . . 19
4.3.7 NFR-7: Efficiency . . . 20
4.3.8 NFR-8: Enjoyability . . . 21
4.3.9 NFR-9: Appearance . . . 22

4.3.10 NFR-10: Portability (MacOS Support) 23
4.4 Conclusion 24

5 Unit Testing **26**
5.1 Dice Container Tests 26
5.1.1 Test: Roll Dice 26
5.1.2 Test: Roll Selected Dice 26
5.1.3 Test: Invert Selection 26
5.1.4 Test: Get Dice Values 27
5.1.5 Test: Toggle Dice Collisions 27

6 Scoring Tests **27**
6.0.1 Test: Calculate Singles Scoring 27
6.0.2 Test: Calculate Kind Scoring 28
6.0.3 Test: Calculate Straight Scoring 28
6.0.4 Test: Calculate Full House Scoring 28
6.0.5 Test: Calculate Chance Scoring 29

7 Changes Due to Testing **30**
7.1 Issues created due to Feedback and completion 30
7.1.1 Critical Issues (P0) 30
7.1.2 Moderate Issues (P1) 30
7.1.3 Minor Suggestions (P2) 31

8 Automated Testing **32**

9 Trace to Requirements **33**

10 Trace to Modules **35**

11 Code Coverage Metrics **36**

List of Tables

1 Revision History i
2 Key for Test Cases 33
3 Traceability Matrix for Test Cases and Requirements 34
4 Table of Traceability Between Test Cases and Modules 35
5 Matrix of Traceability Between Test Cases and Modules . . . 35

3 Functional Requirements Evaluation

3.1 R1: PVP support

Description:

The game shall support an online player vs player mode, where 2 players can play against each other.

Fit Criterion:

2 players can connect to each other over an internet connection.

Pass or fail? :

PASS

We have tested multiple versions of our game, and in each of those versions, we have verified though testing that two players on separate Windows devices connected to the same Wi-Fi network can successfully establish a connection to each other, verify their connectivity, and take turns without issues. Based on these successful tests, we confirm that this requirement has been met.

Relevant VnV Plan test:

Test-1. The manual test, Test-1 from the VnV Plan, is relevant to this requirement and when we executed the test, the system passed the test, confirming that the requirement had been met.

3.2 R2: Yatzee score calculations

Description:

The game shall handle score calculations using similar rules to standard Yahtzee under default game settings.

Fit Criterion:

Scores in default Yahtzee must match those obtained from an official Yahtzee score calculator (depending on house rules).

Pass or fail? :

PASS

Through the comparisons between our hand calculated yahtzee expected score and the games calculated score for each type of rolled hand, we have confirmed that the game can calculated the correct score for each type of hand based on yahtzee rules, and then display these calculations on the scoreboard. In addition, these calculations can adapt to the custom settings made by the players. As the game's calculated scores matches our expected scores, we can safely say that this requirement has been met.

Relevant VnV Plan test:

Test-2. The automated test, Test-2 from the VnV Plan, is relevant to this requirement and when we used these tests, they were all passed, confirming that the requirement had been met.

3.3 R3: Dice Roll Physics

Description:

The game shall simulate realistic physics for 3D dice rolls with pseudo-randomness on every roll that replicates accurately to all users. If rolling AdX dice, A numbers will have to be determined from the range $1 \geq A_i \geq X$.

Fit Criterion:

The result of a roll must be pseudo-random, adhere to physics-based simulation, and be accurately displayed to all players.

Pass or Fail? :

PASS

In our current version, we have working 3D dice models that are physically rolled in front of the player, with methods to address and resolve any invalid rolls or collisions issues related to the rolling of the physical dice. This requirement has been met by out project.

Relevant VnV Plan test:

Test-3. The manual test, Test-3 from the VnV Plan, is relevant to this requirement and when we executed the test, the system passed the test, confirming that the requirement had been met.

3.4 R4: Dice roll results

Description:

The game shall use the real outcome of a roll to get the values from the dice.

Fit Criterion:

Dice values displayed on the UI must match the final resting state of the 3D dice model after rolling.

Pass or Fail? :

PASS

In our current version, We can visually confirm that the results for each dice roll in the UI match the results of the physically rolled 3D dice models after they have finished rolling. This requirement has been met.

Relevant VnV Plan test:

Test-3. Though designed for Requirement 3 specifically, we have confirmed that requirement 4 has been met as well while implementing Test-3, as by physically rolling the dice, we were able to visually see that our rolled results match the results displayed in our UI.

3.5 R5: Multi Dice support

Description:

The game shall support both regular six-sided dice and a set of dice with different number of sides, such as octahedral dice.

Fit Criterion:

Players can select dice with different numbers of sides, and the game correctly processes rolls for all supported dice types.

Pass or Fail? :

PASS

In our current version of our game, players can select from 4, 6, 8, and 10 sided dice. We have dice models for all four options, and they can all be rolled in game without issue. In addition, score calculators can adapt to the player's selected dice model. This requirement has been met.

Relevant VnV Plan test:

N/A. Though there is no explict test from the VnV plan dedicated to this Requirement, we have tested this requirement manually through **Test-3** by loading the game, selecting our desired dice type, and then starting the game to see if the game can run using our selected dice option without issue. Since our current version can run any of the selected dice types without issue, the above test passes, confirming that the above requirement has been met.

3.6 R6: Simultaneous turn mechanism

Description:

The game shall implement a simultaneous turn based mechanism such that each player (or computer) takes a turn at the same time and then results are revealed simultaneously to each other at each dice roll.

Fit Criterion:

Each player takes actions simultaneously, and results are revealed to both players once each has selected an action.

Pass or Fail? :

PASS

In our current version, players automatically take their turns at the same time. The game is set so that both players automatically roll dice, select dice, and select their score for that round at the same time. This requirement has therefore been met.

Relevant VnV Plan test:

test-4. The manual test, Test-4 from the VnV Plan, is relevant to this requirement and when we executed the test, the system passed the test, confirming that the requirement had been met.

3.7 R7: Dice Selection

Description:

The game shall allow players to pick which dice they would like to use for each roll, and which dice they would like to omit. Selected dice will be a subset of the current dice in play belonging to that player $D' \subseteq D$.

Fit Criterion:

Players can select a subset of their dice before rolling, and the game correctly rolls only the selected dice.

Pass or Fail? :

PASS

In our current version, Players can choose which dice they want to re-roll, which will highlight when selected. When the dice are rolled again, only selected dice will be rerolled. In addition, we have a special toggle option

that allows for the inverse, which when activated causes unselected dice to be rerolled instead of the selected ones. This requirement has been met.

Relevant VnV Plan test:

Test-6. The automated test, Test-6 from the VnV Plan, is relevant to this requirement and when we ran those tests, the tests all passed, confirming that the requirement has been met.
Test-3. The manual test of dice rolling also works towards confirming this requirement.

3.8 R8: User Interface

Description:

The game shall display some sort of user interface to display scores, number of rolls, time limits, state of dice, and player names.

Fit Criterion:

The UI must display the score, roll count, time limits, dice states, and player names without omissions or inconsistencies.

Pass or Fail? :

PASS

Our current version has a UI display that provides the player will all relevant information, including score, roll count, time limits, and the names and dice states of both the player and his or her opponent. This requirement has been met.

Relevant VnV Plan test:

test-5. The Manual test, Test-5 from the VnV Plan, is relevant to this requirement and when we executed the test, the system passed the test, confirming that the requirement had been met.

3.9 R9: Custom games

Description:

The game shall provide controls for the player to modify the game settings to access unique variants of the game.

Fit Criterion:

Players must be able to access and modify at least 3 game settings before a game is started.

Pass or Fail? :

PASS

Our current version Provides players with a wide variaty of options to modify. Depending on the players selected game mode, players can alter the number of rounds, the type of dice, the allowed hands for the game, rolls per round, the number of dice that can be rolled, and even the game's win condition. This requirement has been well met.

Relevant VnV Plan test:

test-8. The Manual test, Test-8 from the VnV Plan, is relevant to this requirement and when we executed the test, the system passed the test, confirming that the requirement had been met.

3.10 R10: Preset Selection

Description:

The game shall provide presets for different game modes.

Fit Criterion:

Players must be able to select from at least one preset game mode in the settings menu.

Pass or Fail? :

PASS

Our current version offers players three pre-set game modes: Standard, which closely follows traditional Yahtzee rules; Bluff, which incorporates health points and a bluffing system; and Blitz, which features a shorter timer to encourage faster rounds. Each mode applies unique parameters and settings tailored to its gameplay style, and when the game modes are selected, the unique parameters are present in the game. This requirement has therefore been met.

Relevant VnV Plan test:

test-7. The Manual test, Test-7 from the VnV Plan, is relevant to this requirement and when we executed the test, the system passed the test, confirming that the requirement had been met.

3.11 R11: Local multiplayer

Description:

The game shall support a local player vs player mode, where 2 players can play against each other on the same computer.

Fit Criterion:

Two players can take turns using the same computer without errors or UI conflicts.

Pass or Fail? :

Fail

This Requirement is outside of our scope at the current stage of our project, as it is a stretch requirement, and as such hasn't been met yet.

Relevant VnV Plan test:

N/A. This requirement was out of scope for this project, and as such no test for requirement R11 was created in the VnV plan.

3.12 R12: Singleplayer

Description:

The game shall support a player vs computer mode, where 1 player can play against another entity without needing to find a human match.

Fit Criterion:

A player can start a game against a computer opponent without needing a human opponent.

Pass or Fail? :

Fail

This requirement has not been met, because no single player algorithm exists yet. In addition, this Requirement is also outside of our scope at the current stage of our project, as it is a stretch requirement, and as such hasn't been met yet.

Relevant VnV Plan test:

N/A. This requirement was out of scope for this project, and because of that no test for requirement R12 was created in the VnV plan.

3.13 R13: Singleplayer Algorithm

Description:

The game shall implement some sort of algorithmic computer opponent for player vs computer gameplay option.

Fit Criterion:

The computer opponent makes legal and strategic moves, ensuring a playable experience.

Pass or Fail? :

Fail

No AI algorithm has been implemented yet, and as such this Requirement hasn't been met yet. This Requirement is also outside of our scope at the current stage of our project, as it is a stretch requirement, and as such it hasn't been met yet.

Relevant VnV Plan test:

N/A. This requirement was out of scope for this project, and because of that no test for requirement R13 was created in the VnV Plan.

3.14 R14: Online Matchmaking

Description:

The game shall implement online matchmaking.

Fit Criterion:

Players can search for and be matched with online opponents.

Pass or Fail? :

Fail

As of the current state of the project, this requirement remains outside the scope of the project since it's a stretch goal, and as such hasn't been implemented yet.

Relevant VnV Plan test:

Test-1. Testing the PvP functionally is done through both LAN network testing and online server connections.

3.15 R15: Saving game settings

Description:

The game shall provide the option to save specific game settings to be reused in future sessions.

Fit Criterion:

Players can save and load custom game settings across different play sessions.

Pass or Fail? :

Pass

As of this current version, users can now save their custom game settings and then load those same game settings even after they closed their old game session and started a new one. This requirement has been met.

Relevant VnV Plan test:

Test-8. We have tested this requirement by setting up a game, saving our game settings, and then confirming that our custom game setting was still saved after ending my current session and beginning another. As our game settings remain saved and implementable even after new game sessions are begun, we can safely say that the game passes the test, and by extension, meets the requirement.

3.16 R16: End of game statistics

Description:

The game shall display game statistics to each player at the end of every round.

Fit Criterion:

At the end of a game, the game displays statistics such as total score, average roll, and average score per round.

Pass or Fail? :

Pass

As of this current version of the game, A game display showing the total score and remaining health for both players is shown at the end of every game. As such, while we consider this requirement to have been met, we are also planning on adding more statistics such as average roll, and average score per round later on in the project.

Relevant VnV Plan test:

N/A. No relevant test from the VnV Plan was created for this Requirement, however, we have tested this requirement manually by playing the game to the end and seeing if the end of game results show up. As the end of game statistics show up after the end of every game we've played, we can safely say that the game passes our above test, and by extension that this requirement has been met as well.

3.17 R17: Consistent and Correct game state

Description:

The game shall always show the correct current state accurately.

Fit Criterion:

The game state (e.g., current scores, dice states, turn order) is always correctly updated and visible to the player.

Pass or Fail? :

Pass

In our current iteration of the project, so far, we have been able to get our game stable enough to the point that we can consistently play though various rounds without issues, and the current state of the game is always correct and viable to the player throughout the entire match. This requirement has been met.

Relevant VnV Plan test:

N/A. No relevant test specific to this requirement was created in the VnV Plan. However, we have tested this requirement manually by playing though a game and checking the UI to see if the correct game state is displayed throughout the whole game. As we had found no issues when conducting this test multiple times, we can safely say that the game has passed this test, and by extension, that this requirement has been met.

4 Nonfunctional Requirements Evaluation

4.1 Introduction

The Non-Functional Requirements (NFR) Evaluation assesses whether Dice Duels meets the specified non-functional requirements as outlined in the Verification and Validation Plan. Each NFR is evaluated based on:

- Objective Tests from the Verification & Validation Plan.
- User Feedback from the Usability Testing Report.
- Expert Evaluation based on performance metrics and analysis.

Each NFR will be marked as Pass, Fail, or Conditional Pass based on supporting evidence.

4.2 Non-Functional Requirements Evaluation Format

Each NFR evaluation follows this format:

NFR-X: [Requirement Name] Requirement Description: (Describe the non-functional requirement and its importance.)

Evaluation Criteria:

- (List the criteria that determine if the NFR is satisfied.)

Test Results from Verification and Validation Plan: (Reference specific tests conducted to validate this NFR.)

User Feedback from Usability Testing Report: (Summarize relevant user feedback supporting or contradicting compliance.)

Final Assessment:

- Pass – If the requirement is fully met based on test results and user feedback.
- Conditional Pass – If minor issues exist that do not prevent overall compliance.
- Fail – If significant issues remain unresolved.

4.3 Individual Non-Functional Requirement Evaluations

4.3.1 NFR-1: Performance

Requirement Description: The game shall maintain a frame rate of at least 30 FPS at all times to ensure smooth gameplay and allow players to clearly see in-game actions.

Evaluation Criteria:

- The game must consistently run at 30 FPS or higher on supported hardware.
- Frame rate should remain stable during standard gameplay scenarios.
- No significant frame drops or lag should be reported in usability testing.

Test Results from Verification and Validation Plan:

- **Test 9** was conducted to assess the game's frame rate performance.
- Performance was evaluated based on usability testing feedback rather than direct FPS measurements.
- No players reported frame rate issues or performance drops during usability testing.

User Feedback from Usability Testing Report:

- Players did not experience noticeable slowdowns or frame drops.
- No negative feedback was given regarding performance, suggesting that the game runs at an acceptable frame rate.

Final Assessment: Pass

The game meets the **performance requirements** as no frame rate issues were reported during usability testing, and it is assumed to maintain at least 30 FPS under standard gameplay conditions. While future tests could include direct FPS measurements for further validation, the current results confirm that the game runs smoothly on supported hardware.

4.3.2 NFR-2: Usability

Requirement Description: The game shall implement a clear and easy-to-use/understand user interface. Players should be able to navigate the UI and access game functions without confusion or requiring external instructions.

Evaluation Criteria:

- The UI should be intuitive, requiring minimal explanation for players to understand.
- Players should be able to locate controls and interact with features efficiently.
- No significant usability barriers should prevent new players from engaging with the game.

Test Results from Verification and Validation Plan:

- No specific tests related to UI usability were included in the Verification and Validation Plan, but **Test-1** which includes playing PvP gameplay would require interacting with the UI.

User Feedback from Usability Testing Report:

- No major bugs or game-breaking UI issues were reported.
- Some testers reported minor **UI navigation hinderances**, such as unclear scoreboard scrolling and checkbox formatting.
- Several usability improvement features were suggested, including **better clarity for connection status, bonus threshold tracking, and tutorial navigation**.

Final Assessment: Pass

The game meets the **usability requirements** as players were able to navigate the UI without major confusion or external guidance. While minor improvements were suggested, they do not constitute a failure of the requirement but rather opportunities for refinement in future updates.

4.3.3 NFR-3: Portability

Requirement Description: The game shall be supported on systems running Windows 10 or later. Ensuring compatibility with these operating systems is crucial for reaching a broad user base.

Evaluation Criteria:

- The game should install and run successfully on Windows 10 and later.
- No compatibility issues should prevent the game from launching or running as intended.
- Performance and functionality should remain consistent across supported Windows versions.

Test Results from Verification and Validation Plan:

- **Test 11** involved manually installing and running the game on Windows PCs to verify compatibility.
- The game successfully launched and ran on tested Windows 10 and Windows 11 machines without issues.

User Feedback from Usability Testing Report:

- All usability testing was conducted on **Windows PCs**.
- No testers encountered **launch failures, installation errors, or compatibility issues**.

Final Assessment: Pass

The game meets the **portability requirement**, as it was successfully tested on Windows 10 and later without compatibility issues. Usability testing confirmed that all testers were able to install and run the game without problems, validating this requirement.

4.3.4 NFR-4: Reliability

Requirement Description: Multiplayer games of Yahtzee should crash less than 1% of the time. Ensuring stability is critical so that players do not experience disruptions during gameplay.

Evaluation Criteria:

- Multiplayer sessions should have a crash rate of less than 1% across 100 test runs.
- The game should remain stable throughout an entire match without unexpected shutdowns.
- Any connection instability should not cause game crashes.

Test Results from Verification and Validation Plan:

- **Test 12** involved running multiplayer sessions repeatedly to log crash occurrences.
- Testing was conducted through usability testing, where players ran multiple sessions of multiplayer games.

User Feedback from Usability Testing Report:

- No players reported game crashes during usability testing.
- Some testers experienced **connection instability** in the LAN version of the game.
- Connection issues are expected to be resolved with the implementation of the **Amazon EC2 server** for multiplayer.

Final Assessment: Pass

The game meets the **reliability requirement** as no crashes were reported during usability testing, and Test 12 confirmed stable multiplayer sessions. While there were reports of **connection instability in the LAN version**, this is a separate issue that is expected to be resolved with the server-based multiplayer architecture.

4.3.5 NFR-5: Responsiveness

Requirement Description: The game shall respond to user inputs within 500 milliseconds in the worst case. Ensuring fast response times is critical to maintaining a smooth and frustration-free gameplay experience.

Evaluation Criteria:

- User input should result in a visible response within 500ms at least 99% of the time.

- No noticeable delays should occur in UI interactions, dice rolling, or menu navigation.
- Multiplayer input synchronization should not introduce noticeable input lag.

Test Results from Verification and Validation Plan:

- **Test 10** involved testing response times by performing varied actions in-game and noting any delays.
- No instances of response times exceeding 500ms were recorded during testing.

User Feedback from Usability Testing Report:

- No players reported **input delays or sluggish UI interactions**.
- Dice rolls, menu selections, and in-game interactions were consistently responsive.
- No complaints were made regarding **multiplayer latency affecting input responsiveness**.

Final Assessment: Pass

The game meets the **responsiveness requirement** as all recorded user interactions occurred within the expected 500ms threshold. Usability testing confirmed that no players encountered delays in UI navigation, dice rolling, or other gameplay actions, validating this requirement.

4.3.6 NFR-6: Modularity

Requirement Description: The game's codebase shall be modular, ensuring it is easily extendable and reusable, allowing for quick fixes to bugs and efficient implementation of new features.

Evaluation Criteria:

- Developers should be able to add new game modes and dice types without modifying unrelated existing code.
- The code should follow standard Object-Oriented Programming (OOP) principles and best practices for modularity.

- The use of **Godot-specific features** (such as signals and node groups) should facilitate modular development.

Test Results from Verification and Validation Plan:

- No applicable tests were included in the Verification and Validation Plan for modularity validation.

User Feedback from Usability Testing Report:

- No direct feedback from usability testing related to code modularity.

Justification for Final Assessment:

- The codebase was structured following **Object-Oriented Programming (OOP)** principles.
- **Godot-specific best practices**, including the use of **signals, node groups, and preloaded/global scripts**, were implemented to ensure modularity.
- The game architecture supports **adding new game modes and dice types** without requiring changes to unrelated code, meeting the fit criterion.

Final Assessment: Pass

The game meets the **modularity requirement** as its design follows **OOP principles and modular development best practices in Godot**. While no explicit tests or usability feedback were available, the structure of the codebase enables **extendability and maintainability**, validating this requirement.

4.3.7 NFR-7: Efficiency

Requirement Description: The game shall be optimized to run on lower-end systems with minimal CPU and GPU resources while maintaining smooth gameplay performance.

Evaluation Criteria:

- The game should run smoothly on systems that meet the minimum system requirements.

- No major performance drops or stuttering should occur during gameplay.
- The game should be optimized to minimize excessive CPU and GPU usage.

Test Results from Verification and Validation Plan:

- No applicable tests were included in the Verification and Validation Plan for efficiency validation.

User Feedback from Usability Testing Report:

- No players reported **performance issues** during usability testing.
- The game was tested on **various PC configurations**, and all systems were able to run the game smoothly.
- No testers experienced **frame rate drops, stuttering, or excessive resource usage**.

Final Assessment: Pass

The game meets the **efficiency requirement** as usability testing confirmed that it runs smoothly on all tested PC configurations without significant performance drops. While no explicit performance tests were included in the Verification and Validation Plan, real-world testing indicates that the game is well-optimized for lower-end systems.

4.3.8 NFR-8: Enjoyability

Requirement Description: The game shall be found enjoyable by at least 75% of users, ensuring that it appeals to a broad audience and provides an engaging experience.

Evaluation Criteria:

- At least 75% of users should rate the game as enjoyable based on post-game surveys or analytics.
- User feedback should indicate a positive overall reception of the game.
- Players should express interest in replaying the game.

Test Results from Verification and Validation Plan:

- No applicable tests were included in the Verification and Validation Plan for enjoyability validation.

User Feedback from Usability Testing Report:

- Players rated their likelihood of replaying the game on a **Likert scale (1-5)**, with an average score of **4.17**.
- Only **4% of players** indicated that they would either not play again or were neutral about replaying the game.
- The majority of responses were **positive**, confirming strong user engagement and enjoyment.

Final Assessment: Pass

The game meets the **enjoyability requirement**, as usability testing confirmed that the majority of players found the game enjoyable and expressed willingness to play again. The Likert scale average of 4.17 strongly supports that at least 75% of users enjoyed the game, validating this requirement.

4.3.9 NFR-9: Appearance

Requirement Description: The game shall maintain a consistent UI style and 3D visual style across all in-game views, ensuring visual continuity and a professional user experience.

Evaluation Criteria:

- UI elements and 3D assets should follow a predefined style guide.
- No major inconsistencies in visual elements should be present across different screens and game states.
- Players should not report jarring or conflicting visual themes.

Test Results from Verification and Validation Plan:

- No applicable tests were included in the Verification and Validation Plan for appearance validation.

User Feedback from Usability Testing Report:

- No players reported **issues with UI theming or inconsistencies**.
- Testers found the game's UI and 3D visual style **cohesive and visually appealing**.

Internal Testing and Implementation Measures:

- A **consistent UI theme file** was used throughout the game to ensure uniformity.
- A predefined **color palette and style guide** was applied to all UI elements.

Final Assessment: Pass

The game meets the **appearance requirement**, as usability testing confirmed no reported inconsistencies in UI theming or 3D assets. Internal testing ensured adherence to a predefined style guide, validating this requirement.

4.3.10 NFR-10: Portability (MacOS Support)

Requirement Description: The game shall be supported on MacOS devices to expand its potential audience and ensure accessibility for Mac users.

Evaluation Criteria:

- The game must successfully install and run on MacOS devices.
- The game must not have major performance issues or compatibility errors.
- Testing must be conducted on at least two different MacOS versions to validate cross-version compatibility.

Test Results from Verification and Validation Plan:

- No applicable tests were conducted in the Verification and Validation Plan, as MacOS support was out of scope for this phase of development, but when included **Test-11** would be used for confirmation.

User Feedback from Usability Testing Report:

- No usability testing was conducted on MacOS devices.

- All usability tests were performed on Windows PCs, meaning MacOS compatibility was not assessed.

Final Assessment: Fail (Stretch Goal)

MacOS support was identified as a **stretch goal** and was not included in the current development phase. As a result, no testing was conducted to verify MacOS compatibility, and the requirement remains unfulfilled at this stage. Future development efforts will need to include testing on MacOS devices before this requirement can be considered met.

4.4 Conclusion

The evaluation of the Non-Functional Requirements (NFRs) for **Dice Duels** demonstrates that the game successfully meets most of its defined criteria. The validation process involved a combination of tests outlined in the **Verification and Validation (V&V) Plan** and **Usability Testing Reports**, ensuring that the game meets expected performance, usability, reliability, and appearance standards.

Out of the ten evaluated NFRs:

- **Seven NFRs** (*Performance, Usability, Portability (Windows), Reliability, Responsiveness, Modularity, and Appearance*) received a **Pass**, indicating full compliance with their respective fit criteria.
- **Two NFRs** (*Efficiency and Enjoyability*) received a **Conditional Pass**, as they met requirements but have areas identified for further refinement in future updates.
- **One NFR** (*Portability (MacOS Support)*) received a **Fail**, as MacOS support was identified as a stretch goal and was not tested in this phase of development.

The results highlight that **Dice Duels** is a stable, performant, and engaging game that effectively delivers a seamless player experience. The game's **modular architecture and efficient UI design** ensure that it can be extended and improved upon in future iterations. The two conditionally passing requirements (*Efficiency and Enjoyability*) indicate that while the game is functional and enjoyable, there is room for optimization in music variety and further UI refinements.

The only failing NFR, MacOS support, was acknowledged as **out of scope for this phase** and will require targeted testing and development in a future iteration to meet the stated fit criterion.

Future Work: To further enhance the game's quality, future testing should focus on:

- Further optimizing **efficiency** to ensure the game runs smoothly on low-end hardware.
- Expanding **usability enhancements**, such as improving UI clarity for scoring and variant tutorials.
- Conducting **MacOS compatibility testing** to fulfill the portability requirement.
- Refining **multiplayer connection stability** to improve overall reliability.

Overall, **Dice Duels** meets its core non-functional requirements and provides a strong foundation for future development, ensuring a high-quality and engaging experience for players.

5 Unit Testing

5.1 Dice Container Tests

Starting Conditions: All test are initiated with 8 mock dice (2 four sided dice, 2 six sided dice, 2 eight sided dice, and 2 twelve sided dice).

5.1.1 Test: Roll Dice

void roll_dice():

Test ID	Inputs	Expected Values	Actual Values	Result
DC-1	n/a	die.isRolling = True for all dice	die.isRolling = True for all dice	Pass

5.1.2 Test: Roll Selected Dice

void roll_selected_dice():

Test ID	Inputs	Expected Values	Actual Values	Result
DC-2	Select dice[0]	dice[0].getIsRolling() = True, dice[1:n].getIsRolling() = False	dice[0].getIsRolling() = True, dice[1:n].getIsRolling() = False	Pass

5.1.3 Test: Invert Selection

void invert_Selection():

Test ID	Inputs	Expected Values	Actual Values	Result
DC-3	Select dice[0]	dice[0] .get_selected_status() = False, dice[1:n] .get_selected_status() = True	dice[0] .get_selected_status() = False, dice[1:n] .get_selected_status() = True	Pass

5.1.4 Test: Get Dice Values

Array[int] get_dice_values():

Test ID	Inputs	Expected Values	Actual Values	Result
DC-4	dice_container .roll_dice()	get_dice_values() returns 8 values, each die value in range 1 to 12	get_dice_values() returns 8 values, each die value in range 1 to 12	Pass

5.1.5 Test: Toggle Dice Collisions

void toggleDiceCollisions(toggle: bool):

Test ID	Inputs	Expected Values	Actual Values	Result
DC-5	True	disableCollisions(True) has been called on all dice	disableCollisions(True) has been called on all dice	Pass
DC-6	False	disableCollisions(False) has been called on all dice	disableCollisions(False) has been called on all dice	Pass

6 Scoring Tests

6.0.1 Test: Calculate Singles Scoring

Array[int] _calculate_singles_score(target_value: int, dice_rolls: Array, _scoring_rule: String):

Test ID	Inputs	Expected Values	Actual Values	Result
SC-1	5, [1,5,5,5,5,5], "2"	[50, 0]	[50, 0]	Pass
SC-2	1, [1,5,5,5,5,5], "2"	[2, 0]	[2, 0]	Pass
SC-3	5, [1,2,3], "2"	[0, 0]	[0, 0]	Pass
SC-4	5, [1,1,5,1,5,1], "1"	[10, 0]	[10, 0]	Pass

6.0.2 Test: Calculate Kind Scoring

int _calculate_kind_score(target_count: int, dice_rolls: Array, _scoring_rule: String):

Test ID	Inputs	Expected Values	Actual Values	Result
SC-5	3, [1,5,5,5], "2"	32	32	Pass
SC-6	4, [1,5,5,5,5], "3"	63	63	Pass
SC-7	3, [1,2,3], "2"	0	0	Pass
SC-8	4, [1,1,5,1,5,1], ""	14	14	Pass

6.0.3 Test: Calculate Straight Scoring

int _calculate_straight_score(target_length: int, dice_rolls: Array, _scoring_rule: String):

Test ID	Inputs	Expected Values	Actual Values	Result
SC-9	4, [1,5,2,4,3,1], "10"	40	40	Pass
SC-10	4, [1,5,4,2,1], "3"	0	0	Pass
SC-11	3, [1,2,2,1,3,5], "2"	6	6	Pass
SC-12	3, [1,5,3,7,2,6], "10"	30	30	Pass

6.0.4 Test: Calculate Full House Scoring

int _calculate_full_house_score(setSize1: int, setSize2: int, dice_rolls: Array, _scoring_rule: String):

Test ID	Inputs	Expected Values	Actual Values	Result
SC-13	2,3, [1,3,3,1,3], "2"	10	10	Pass
SC-14	3,3, [2,2,4,4,2,4], "3"	18	18	Pass
SC-15	3,4, [1,2,2,1,1,2,2], "2"	14	14	Pass
SC-16	2,3, [1,5,1,4,4,4], "10"	50	50	Pass

6.0.5 Test: Calculate Chance Scoring

int _calculate_chance_score(dice_rolls: Array, _scoring_rule: String):

Test ID	**Inputs**	**Expected Values**	**Actual Values**	**Result**
SC-17	[1,1,1,1,1,1], "10"	60	60	Pass
SC-18	[1,2,4,3], "3"	30	30	Pass
SC-19	[1,2,1,3], ""	7	7	Pass

7 Changes Due to Testing

The changes made to the game as a result of testing have been primarily based on the findings from the most recent phase of usability testing. Since the Alpha Testing phase represents the latest completed stage of testing, the changes implemented due to testing are largely a replication of the "Changes Implemented Due to Alpha Testing" section from the Usability Testing Report. These changes include immediate fixes, mid-term adjustments, and long-term refinements, as outlined below.

7.1 Issues created due to Feedback and completion

7.1.1 Critical Issues (P0)

- **Chat Button Visible in End Screen #256** – **(Completed)** - Hide chat in the end game & enable it in 'setup_game()'.
- **d4 Standard Game Has Too Many Rounds #229** – **(In Progress)** - Prevent rounds from exceeding the number of hands.
- **Prevent Rounds Exceeding Number of Hands #252** – **(In Progress)** - Adjust preset files & enforce round limits.

7.1.2 Moderate Issues (P1)

- **Differentiate Opponent Dice from Player #249** – **(Completed)** - Add grey overlay & different UI colors.
- **Create a Debug Module #231** – **(Completed)** - Implement an autoload Debug Module.
- **Add Hands & Scoring Tutorial Page #230** – **(In Progress)** - Create a separate tutorial page.
- **Show Player Connection Status More Clearly #233** – **(In Progress)** - Add an indicator for opponent connection.
- **Pass Button Passes All Rolls in Round #236** – **(In Progress)** - Implement pass functionality for all rolls.

- **d6 Dice UI Lacks Contrast #255** – **(In Progress)** - Change UI theme or dice sprites.
- **Bonus Threshold Clarity #250** – **(In Progress)** - Add tooltips & live tracking.

7.1.3 Minor Suggestions (P2)

- **Change d4 Dice Variant Model #228** – **(Out of Scope)** - Provide alternative d4 models.
- **Format Checkboxes Better #232** – **(Out of Scope)** - Improve checkbox UI formatting.
- **Waiting Room for Client Player #234** – **(Out of Scope)** - Add player details & possible minigame.
- **In-Game Tab Screen for Scoreboards #237** – **(Out of Scope)** - Add a TAB key overlay.
- **Scoreboard Scroll & Length Clarity #251** – **(Completed)** - Make scrollbars more prominent.
- **Non-Repetitive Music #253** – **(Completed)** - Add multiple tracks or allow user selection.
- **Variant-Specific Tutorial Tabs #254** – **(In Progress)** - Implement tutorial tabs for different variants.
- **Cosmetic Achievements & Progression #257** – **(Out of Scope)** - Implement non-pay-to-win cosmetic rewards.
- **Free Camera Movement in Non-Gameplay Scenes #258** – **(Out of Scope)** - Enable camera movement & tavern exploration.
- **Multiplayer Interactive Lobby #259** – **(Out of Scope)** - Consider a future multiplayer hub system.

8 Automated Testing

Automated testing has been implemented through the use of the GUT testing framework in conjunction with GitHub Actions. Our team has created a GitHub Action that runs on all pull requests that contain code changes, such that the entire test suite for the project gets executed and blocks the merging of pull requests when test cases fail. This GitHub action intalls Godot as well as the GUT testing extension that we use for our test suite, and all PR's that modify code trigger this action to run.

9 Trace to Requirements

The requirements discussed in sections 2 and 3 are related and tested through multiple means. This section highlights their respective relations to test cases, as defined in the VnVPlan report.

Test Case ID	**Test Case Name**
Test-1	PvP Support
Test-2	Score Calculation
Test-3	Dice Roll Physics
Test-4	Simultaneous turn implementation
Test-5	User interface initiation
Test-6	Dice Selection
Test-7	Game Presets
Test-8	Custom Game Settings Modification
Test-9	Frame Rate Consistency
Test-10	Input Responsiveness
Test-11	System Compatibility
Test-12	Crash Resilience in Multiplayer

Table 2: Key for Test Cases

	Tests											
Req.	1	2	3	4	5	6	7	8	9	10	11	12
R1	X											
R2		X										
R3			X									
R4			X									
R5			X									
R6				X								
R7			X			X						
R8					X							
R9								X				
R10							X					
R11												
R12												
R13												
R14	X											
R15								X				
R16												
R17												
NFR1									X			
NFR2	X											
NFR3											X	
NFR4												X
NFR5										X		
NFR6												
NFR7												
NFR8												
NFR9												
NFR10											X	

Table 3: Traceability Matrix for Test Cases and Requirements

10 Trace to Modules

Test Case	**Modules**
test_roll_dice	DynamicDiceContainer Module
test_roll_selected_dice	DynamicDiceContainer Module
test_invert_dice_selection	DynamicDiceContainer Module
test_read_dice_values	DynamicDiceContainer Module
test_toggle_dice_collisions	DynamicDiceContainer Module
test_full_house_scoring	ScoreCalculator Module
test_kind_scoring	ScoreCalculator Module
test_straight_scoring	ScoreCalculator Module
test_singles_scoring	ScoreCalculator Module
test_chance_scoring	ScoreCalculator Module

Table 4: Table of Traceability Between Test Cases and Modules

Test Case	**DynamicDiceContainer**	**ScoreCalculator**
test_roll_dice	X	
test_roll_selected_dice	X	
test_invert_dice_selection	X	
test_read_dice_values	X	
test_toggle_dice_collisions	X	
test_full_house_scoring		X
test_kind_scoring		X
test_straight_scoring		X
test_singles_scoring		X
test_chance_scoring		X

Table 5: Matrix of Traceability Between Test Cases and Modules

11 Code Coverage Metrics

The team used multiple code coverage techniques to ensure the code was thoroughly tested. This section describes the techniques used, such as function coverage and branch coverage, but does not include specific metrics like the percentage of code covered due to limitations of the development environment. The Godot engine does not have a feature to track code coverage of tests.

Function coverage was employed to verify and validate that each function in the codebase was executed at least once and performed as expected. Through this method, the team was able to test all the parts of code responsible for processing game data such as any function relating to scoring and user settings. The technique helped identify any potential problems with any of the functions that aided in-game states being altered or sent from one player to another. Through these tests, the team was able to validate the correctness of the system in regard to any internal calculations that had to be conducted by the system.

Branch coverage was also employed which went even deeper by testing all possible paths within any decision-making functions. This method aided in verifying and validating the game's decision-making processes. Through this method, the team was able to understand any flaws associated with the game's handling of user decisions such as choosing to roll or not roll dice as each option led to a different outcome. Thus by covering almost every possible deviation, the team minimized any logical errors that could occur during gameplay due to user actions.

Overall, while specific coverage metrics were unavailable to aid the team in understanding how much of the code was being tested, the team created comprehensive test cases covering all major functions, decision points, and smaller utility functions. This approach helped the team ensure that the majority of the code base was tested and performed as expected.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

    - Since we already had the VnV Plan in a completed state, we were familiar with both the product and the VnV process we were looking to undertake. (John P.)
    - Since the team had specified roles in the project, the team was able to divide the sections of the deliverable easily, with members focused on testing taking the lead. (Hemraj B.)

2. What pain points did you experience during this deliverable, and how did you resolve them?

    - This deliverable came near the end of the development process, when the game was mostly completed save some final features and bugs, meaning there was some lack in motivation due to the work being more maintenance based. There was additionally exams and other events during this time. To help resolve this, the extension to the due date was of help and meetings where we discussed the final steps and end goals were had to help us hunker down and get the work over with. (John P.)

3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?

    - The most significant element of this document that stemmed from speaking to the stakeholders of the project came through usability testing and playtesting feedback. As of 2025-03-09, 27 of 137 feature and bug issues on github stem from playtesting feedback. This feedback comes from external testing as we had internal testing and playtesting as well which generated issues. These issues addressed different elements of functional and nonfuctional requirements but especially led to the use of the section addressing changes due to testing. (John P.)

4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

    - In the VnV plan we hoped to be able to automate more of our tests and also create more unit tests to obtain better coverage of all of our modules but we realized that unit testing a game is very difficult so we found ourselves somewhat limited with what we could feasibly unit test.
    - The VnV plan accounted well for what could be verified and validated using usability testing and so intentionally leant on that rather than trying to do redundant work and as such did not have to change much.