

Software Requirements Specification

for:

UNO GAME

Prepared by Create-gineers

10/20/2025

Table of Contents

Table of Contents.....
.....	ii
Revision History.....
.....	ii
1.	
Introduction.....
.....	1
1.1	
Purpose.....
.....	1
1.2 Document Conventions.....
.....	1

1.3 Intended Audience and Reading Suggestions.....	1
1.4 Product Scope.....	1
1.5 References.....	1
2. Overall Description.....	2
2.1 Product Perspective.....	2
2.2 Product Functions.....	2
2.3 User Classes and Characteristics.....	2
2.4 Operating Environment.....	2

2.5 Design and Implementation	
Constraints.....	2
2.6 User Documentation.....	
..... 2	
2.7 Assumptions and Dependencies.....	
... 3	
3. External Interface Requirements.....	
3	
3.1 User Interfaces.....	
..... 3	
3.2 Hardware Interfaces.....	
..... 3	
3.3 Software Interfaces.....	
..... 3	
3.4 Communications Interfaces.....	
. 3	

4. System Features.....	
..... 4	
4.1 System Feature 1 (Card Dealing Mechanism)	
.....	
4	
4.2 System Feature 2 (Turn Management)	
.....	4
4.3 System Feature 3 (Button & Penalty Rule)	
.....	4
4.4 System Feature 4 (UNO Deck)	
.....	4
4.5 System Feature 5 (Computer Opponent(Sequential Bot))	4
4.6 System Feature 6 (Menu System)	
.....	4
4.7 System Feature 2 (Non-Functional Requirements)	
.....	4
5. Other Nonfunctional Requirements.....	4
5.1 Performance Requirements.....	
..... 4	

5.2 Safety Requirements.....	
..... 5	
5.3 Security Requirements.....	
..... 5	
5.4 Software Quality Attributes.....	
. 5	
5.5 Business Rules.....	
..... 5	
6. Other Requirements.....	
..... 5	
Appendix A: Glossary.....	
..... 5	
Appendix B: Analysis Models.....	
5	
Appendix C: To Be Determined List.....	
..... 6	

Section 1.

1.1 Purpose

This document defines the requirements for *Uno*, a digital version of the traditional card game. The goal is to recreate the gameplay experience virtually, covering user interaction, game logic, and rule enforcement.

1.2 Document Conventions

Highlighting in this document indicates **keywords** that will be referenced later. **Bold text** is used for key topics, while *italicized text* emphasizes important notes or terms.

1.3 Intended Audience and Reading suggestions

The audience includes developers, testers, instructors, and users interested in the system's scope. Readers should begin with section 1 for context, then move to section 2 for an overview, then section 3 for the user interface and last but not least, section 4 for detailed features.

1.4 Product Scope

The scope will be a simple UNO game made for Windows operating systems. This game is made as part of a school project designed to teach us the software engineering process through experience. Our goal is to create a rough version of our game. Anything code and design-related will be done first with placeholder art. Final art will be drawn after that.

1.5 References

Rules of UNO: <https://www.unorules.com/>

Unity Documentation: <https://docs.unity.com/en-us>

System requirements for Unity 2022.3:

<https://docs.unity3d.com/2022.3/Documentation/Manual/system-requirements.html>

Section 2.

2.1 Product Perspective

This product is within the Unity Game Engine. It is built entirely in there and everything to the coding, physics, and anything else branches from the Unity Game Engine.

2.2 Product Functions

A few functions the product will carry out:

- Selection of same color, same number, and wild cards
- Pick up a card if none can be played
- Skipping turns when a special card is played
- Picking up multiple cards when a special card is played
- Randomly choose a color when a special card is played

2.3 User Classes Characteristics

For now, this project's program will have little difficulty as all there will be is just a program that places cards in line with the rules of the game. The strategy level of the computer is not relevant to this project at this phase. We intend to have this be a game accessible for all ages and all skill levels.

2.4 Operating Environment

The game will be built in Unity 2022.3 and run on Windows PCs. It needs a mouse and a keyboard and has a minimum of 1280x720 resolution with 1920x1080 suggested. The game should run smoothly on regular Windows hardware without needing high-end specs.

2.5 Design and Implementation Constraints

Development will use C# in Unity. Art and sound will start as a place-holder and are to be revised later. The basic-rules will be played according to the official UNO and some house-rules may be introduced, but only if they don't change the main logic. Multiplayer is possible but depends on time. Accessibility options like color-blind indicators should be included.

2.6 User Documentation

The game will include an in-game rule guide, tooltips for card interactions, and a short tutorial to explain gameplay. If needed, a PDF or online manual will be provided for additional reference.

2.7 Assumptions and Dependencies

Unity 2022.3 and Visual Studio are assumed to be stable and be available during the process of development. The project relies on members of the team accomplishing their tasks in time. Until finished assets are ready, placeholders of art and sound will be utilized. Other options like multiplayer, additional bots, or customization are dependent on time and time resources.

Section 3: External Interface Requirements

3.1 User Interfaces

- User Interface Characteristics and Components
- Logical Characteristics of User Interfaces

User Roles

- Player – The user interacts with the game by joining a lobby, drawing cards, playing cards, and passing turns. One player also acts as the admin, responsible for starting the game.
- Bot Player – An automated player that follows the same rules as a human player. The bot does not require a user interface, but its actions are displayed to the user in the same way as a human player's moves (IE “Bot played a Red 2”).

Screen Layouts

- Main Menu
 - Buttons: Start Game, Rules/Help, Exit.
- Lobby Screen
 - Displays connected bot(s).
 - Admin has a Start Game button.
- Gameplay Screen
 - Top section: Game status bar (current turn, cards remaining, discard pile top card).
 - Center area: Discard pile + draw deck.
 - Bottom section: Player's hand with selectable cards, and action buttons (Play, Draw, Pass).
 - Side area: List of players (human + bot) with card counts.
- End-of-Game Screen
 - Displays winner and a summary of results.
 - Buttons: Play Again, Exit to Menu.

Common Controls & Standards

- Navigation: Consistent Back button where applicable; Help button on all major screens.
- Style Guide: UNO-inspired color palette (red, yellow, green, blue). Consistent font family and button style across all screens.
- Error Messages: Invalid moves display a red-highlighted alert (e.g., “That card cannot be played”). Connection or rule errors appear in modal pop-ups.

Interaction Methods

- Mouse/Touch: Select a card by clicking/tapping, then click/tap Play.
- Keyboard Shortcuts (optional for future use):

- o D = Draw card
 - o P = Pass turn
 - o Esc = Pause/Options menu
-

- Software Components Requiring a User Interface

The following software components require user interfaces:

1. Main Menu Component
 - o Navigation to start a new game, view rules/help, or exit the application.
2. Lobby Component
 - o Displays connected participants (human + bot).
 - o Provides admin player control to start the game.
3. Gameplay Component
 - o Displays the player's hand, the discard pile, the draw deck, and game status.
 - o Provides action controls (Play Card, Draw Card, Pass).
 - o Display bot moves to the player.
4. Pause/Options Component
 - o Accessible during gameplay.
 - o Provides options such as Resume, Rules/Help, and Exit to Main Menu.
5. Rules/Help Component
 - o Displays game rules, card descriptions, and instructions for controls.
6. End-of-Game Component
 - o Shows results (winner, final standings).
 - o Provides navigation to Play Again or return to Main Menu.

3.2 Hardware Interfaces

- Hardware Interface Characteristics

Supported Device Types

- Desktop or laptop computer running Microsoft Windows (Windows 10 or higher).
- Standard monitor/display.
- Mouse for primary input (point-and-click interaction).
- Keyboard and other devices are not required at this stage.

Logical Characteristics

- Input: All user interactions are performed using the mouse (clicking on buttons, selecting cards, drawing cards).
- Output: Visual feedback is displayed on the screen, including cards, game state, turn indicators, and error messages. No audio output is currently planned.

Physical Characteristics

- Target system: Windows-based PC.
- Minimum hardware requirements: Dual-core CPU, 4 GB RAM, integrated graphics capable of 1280×720 resolution or higher.
- Tested primarily on standard mouse-and-monitor setups.

Data and Control Interactions

- Mouse click events are captured through the operating system's input handling (Windows API).
- Rendered graphics are displayed via the system's graphics subsystem (Dedicated/integrated GPU)

- The game logic communicates with the UI layer to update the screen after each player action or bot move.

Communication Protocols

- None is required; the system is designed for local play only.
- Potential future extension: Network protocols (e.g., TCP/IP sockets) may be introduced to support multiplayer functionality if time permits.

3.3 Software Interfaces

- Software Interface Characteristics

Connections to Other Software Components

- Operating System: Microsoft Windows 10 (or higher). Provides process management, windowing, and low-level device drivers for mouse and display.
- Game Engine: Unity (version X.X, e.g., Unity 2022 LTS). Provides rendering, input handling, UI system, asset management, and game loop services.
- Programming Language: C# (supported by Unity runtime/Mono).
- Development Tools: Unity Hub and Unity Editor, Visual Studio (C# IDE and debugger), GitHub or other version control (if used).
- Databases: None required. The game is handled entirely in memory.

Data Items / Messages

- Incoming Data:
 - o Mouse click events captured by Unity's Input System (e.g., selecting cards, clicking buttons).
 - o Menu navigation selections (e.g., Start Game, Exit).

- Outgoing Data:
 - o Graphical rendering of cards, discard pile, draw deck, and player hands via Unity UI canvases.
 - o Status messages displayed as UI text (e.g., “Invalid Move”, “Bot played a Red 7”).
 - o End-of-game results displayed on the results screen.
-

Services Needed & Nature of Communication

- Rendering: Provided by Unity’s rendering engine, handles 2D sprite display for cards and UI elements.
 - Input Handling: Managed by Unity’s Input System for mouse events.
 - Game Loop: Unity manages the update loop that drives the rules engine, bot actions, and UI refresh.
 - Audio (optional): Unity Audio System may be used for effects (card played, error alert).
 - Networking (future extension): Unity’s Netcode for GameObjects or Photon could be integrated if multiplayer functionality is added.
-

Shared Data Between Internal Components

- Deck Manager ↔ Gameplay UI: Deck logic supplies cards to UI for display in the player’s hand and discard pile.
 - Rules Engine ↔ Gameplay UI: Validates card plays; UI updates with success or error messages.
 - Bot AI ↔ Gameplay UI: Bot’s selected moves are passed to UI for display.
 - Game State Manager ↔ End-of-Game Screen: Provides results for display (winner, remaining cards).
-

Implementation Constraints

- All data sharing occurs via Unity-managed C# scripts and in-memory objects (GameObjects, ScriptableObjects, serialized fields).
- No external databases or global data stores are required.
- Inter-component communication follows Unity patterns (IE event systems, method calls, GameObject references).
- Input/output handled entirely within Unity's APIs — no direct OS-level API calls needed.

3.4 Communications Interfaces

For the current version of our Uno Game, the application will run entirely on a local machine thus no external communication functions (IE e-mail, browser integration, network protocols, electronic forms, etc) will be required.

There might be an option for future scalability with multi-player and cross platform but for the project/semester, we will not be doing this.

Section 4: System Features

4.1 System Feature 1: Card Dealing Mechanism

4.1.1 Description and Priority

This mechanism will implement the automatic dealing of cards to players at the start of the game. Ensures that each player begins with an equal number of cards (7) and that the game starts with a properly initialized draw pile and discard pile. It also covers edge cases such as reshuffling when the draw pile becomes empty during play.

Priority: High

Benefit: 9

Penalty: 9

Cost: 4

Risk: 3

4.1.2 Stimulus/Response Sequences

Sequence A: Game start

Stimulus: The player will start a new game

Response: The system shuffles the deck and deals 7 cards to each player

Sequence B: Player cannot play card

Stimulus: On their turn, if a player has no valid card to play

Response: The system requires the player to draw one card from the draw pile

Sequence C:

Stimulus: A player plays a "+2" card

Response: The next player in turn order automatically draws 2 cards, then their turn is skipped

Sequence D: Wild Draw Four (+4)

Stimulus: A player plays a "Wild +4" card

Response: The next player in turn order automatically draws 4 cards, then their turn is skipped

Sequence E: Empty Draw pile

Stimulus: The draw pile is empty, and a player needs to draw.

Response: The system shuffles the discard pile (EXCEPT THE TOP CARD) to create a new draw pile.

4.1.3 Functional Requirements

REQ-1: The system shall shuffle the deck of cards before dealing.

REQ-2: The system shall deal 7 cards to each player at the start of the game.

REQ-3: The system shall place the remaining cards in a draw pile

REQ-4: The system shall draw the top card from the draw pile to start the discard pile

REQ-5: The system shall enforce draw mechanics when a player cannot play a card (1 card drawn)

REQ-6: The system shall enforce action card penalties (IE +2, +4)

REQ-7: The system shall reshuffle the discard pile (except the top card) to recreate a draw pile when the draw pile is empty.

REQ-8: The system shall validate that only legal cards may be played, preventing invalid plays.

REQ-9: The system shall display the updated card count for each player after dealing and drawing events.

4.2 System Feature 2: Turn Management

4.2.1 Description

Manages the flow of turns between players in accordance with official UNO rules. Ensures the current player is identified and actions are executed in the correct order. Especially the turn based mechanic, in special cases such as Skip and Reverse.

4.2.2 Action/Response Sequences

Sequence A: Normal Play

Action: A player successfully plays a valid card

Response: The system places the card on the discard pile, applies any card effects (if applicable), and updates the turn to Player B

Sequence B: Skip Card

Action: A player successfully plays a “Skip” card

Response: The next player’s turn is lost and the system advances to the following player

Sequence C: Reverse Card

Action: A player successfully plays a “Reverse” card

Response: The turn order changes direction, clockwise or counterclockwise

Sequence D: Last Card Rule

Action: A player has only one card in their hand

Response: The system prompts the player to declare "UNO". If failed to do so, the player is given a penalty in which they must draw two cards

Sequence E: End of Game

Action: A player plays the last card in their hand

Response: The system declares that player as the winner and ends the game

Sequence F: Wild or Wild +4

Action: A player plays a Wild or Wild +4 card

Response: The system prompts the user to select a new color. If Wild +4, the next player must also draw four cards before play continues.

Sequence G: No playable cards

Stimulus: A player has no valid cards to play on their turn

Response: The system requires the player to draw one card from the draw pile. If that card is playable, the player may play it immediately; otherwise, their turn ends.

Sequence G: Invalid Play attempt

Stimulus: A player attempts to play an invalid card (IE Wrong color/number, or does not follow special card rules)

Response: The system rejects the action, displays an error message, and prompts the player to try again or draw a card

4.2.3 Functional Requirements:

REQ-10: The system shall enforce turn order among players.

REQ-11: The system shall validate that only the active player when a turn is completed.

REQ-12: The system shall update the active player when a turn is completed.

REQ-13: The system shall apply effects of special cards (Skip, Reverse, Wild, +2, +4).

REQ-14: The system shall detect when a player has no cards remaining and declare them the winner

4.3 System Feature 3: UNO BUtton & Penalty Rule

4.3.1 Description and Priority:

This mechanism enforces the UNO rule requiring a player to declare “UNO” when they have only one card remaining. If the player fails to declare before another player notices, they receive a penalty of drawing two cards.

Priority: Medium (Core to official rules but game can function without it)

Benefit: 7 (Adds fairness to gameplay)

Penalty: 6 (Gameplay feels weird or incomplete)

Cost: 4 (Requires turn checking and button interaction)

Risk: 3 (Low risk, straightforward to implement)

4.3.2 Stimulus & Response Sequences:

Sequence A: Declaring ‘UNO’ successfully

Stimulus: A player plays their second-to-last card

Response: The system prompts the player to click the UNO button. If clicked before the next player begins their turn, the system accepts the declaration

Sequence B: Failing to Declare UNO

Stimulus: A player plays their second-to-last card but does not click the UNO button.

Response: If another player clicks the penalty button before the next turn begins, the system penalizes the first player by forcing them to draw two cards.

Sequence C: False Penalty Attempt

Stimulus: A player attempts to penalize another player who has already declared UNO.

Response: The system rejects the penalty attempt and play continues normally.

4.3.3 Functional Requirements:

REQ-15: The system shall prompt a player to declare UNO when they have one card remaining

REQ-16: The system shall provide an UNO button for declaration

REQ-17: The system shall enforce a two-card penalty if a player fails to declare UNO and is caught before the next player's turn

REQ-18: The system shall prevent false penalties if the player has already declared UNO

4.4 System Feature 4: UNO Deck

4.4.1 Description & Priority

This mechanism will implement the full standard UNO deck including number, action, and wild cards.

Priority: High

Benefit: 9 (Core game mechanics)

Penalty: 9

Cost: 4

Risk: 2

4.4.2 Stimulus & Response Sequences:

Sequence A: Deck initialization

Stimulus: Player starts a new game.

Response: The system generates a complete UNO deck with the correct distribution of cards

Sequence B: Shuffle

Stimulus: Game starts or a new draw pile is needed.

Response: The system shuffles the deck randomly

Sequence C: Deck Validation <optional>

Stimulus: Debug or test mode is triggered

Response: The system verifies that the deck contains the correct number of each card and type.

4.4.3 Functional Requirements:

REQ-19: The system shall generate a full UNO deck consisting of 108 cards.

REQ-20: The system shall include number cards 0-9 in four colors (Red, Blue, Green, Yellow), with the correct duplicates (two of each number except 0, which has one per color)

REQ-21: The system shall include two Skip, two, Reverse, and two +2 cards per color.

REQ-22: The system shall include four Wild and four Wild +4 cards.

REQ-23: The system shall allow shuffling of the deck before dealing.

REQ-24: The system shall support regeneration of a draw pile by reshuffling the discard pile when the draw pile is empty.

REQ-25: The system shall prevent invalid cards from entering the deck (error handling for corrupted game state)

4.5 System Feature 5: Computer Opponent (Sequential Bot)

4.5.1 Description & Priority

This mechanism provides automated players that participate in the UNO game when human/real players are absent or for single players. The computer opponent(s) should follow the same rules as the human player(s), making decisions based on available playable cards and game state.

Priority: High

Benefit: 8

Penalty: 7

Cost: 5

Risk: 4 (Logic bugs may cause invalid moves/deadlocks or other errors)

4.5.2 Stimulus & Response Sequences:

Sequence A: Bot turn with playable card

Stimulus: It is the bot's turn and it has at least one valid card to play.

Response: The bot selects the first valid card according to its programmed logic (IE preferred matching number then color, then wild). The system plays that card onto the discard pile

Sequence B: Bot turn with no playable card

Stimulus: It's the bot's turn and it has no valid cards.

Response: The system forces the bot to draw one card. If that card is playable, the bot immediately plays it; otherwise, the bot's turn ends.

Sequence C: Bot plays +2 / +4

Stimulus: The bot plays a card that forces the next player to draw.

Response: The system enforces draw penalties for the next player and skips their turn.

Sequence D: Bot chooses color with wild card.

Stimulus: The bot plays a wild or wild +4 card

Response: The system requires the bot to choose a new color. By default, the bot selects the color it has the most cards of.

Sequence E: Bot calls UNO!

Stimulus: The bot has one card left.

Response: The system automatically declares "UNO" on behalf of the bot.

4.5.3 Functional Requirements:

REQ-26: The system shall implement at least one bot player (scalable up to N bots)

REQ-27: The system shall ensure that the bot follows the same rules as human players.

REQ-28: The system shall allow the bot to automatically select a valid card to play from its hand.

REQ-29: The system shall enforce that the bot draws a card if it has no valid moves.

REQ-30 The system shall ensure that the bot immediately plays a drawn card if it's valid.

REQ-31: The system shall implement logic for Wild card color selection based on the bot's current hand.

REQ-32: The system shall automatically call "UNO" on behalf of the bot when it has one card left.

4.6 System Feature 6: Menu System

4.6.1 Description & Priority

This mechanism will provide the main menu interface that allows the player to start a new game, exit the program, and access future placeholder options such as settings or house rules. It acts as the user's primary navigation hub before game begins or during game.

Priority: High (Players must be able to start/exit game)

Benefit: 8

Penalty: 7

Cost: 3

Risk: 2

4.6.2 Stimulus & Response Sequences:

Sequence A: Launch game

Stimulus: Player starts the application

Response: The system displays the main menu screen with available options (Start, Exit, Placeholder)

Sequence B: Start Game

Stimulus: Player selects "Start Game" from the menu

Response: The system initializes game setup (deck creation, dealing cards, etc) and transitions to the gameplay screen.

Sequence C: Exit Game

Stimulus: Player selects "Exit" from the menu

Response: The system closes the application.

Sequence D: Placeholder Option (for future features)

Stimulus: Player selects an option that has not been implemented yet (IE “Settings” or “House Rules”)

Response: The system displays a message such as “Feature coming soon!” and returns the player to the menu.

4.6.3 Functional Requirements:

REQ-33: The system shall display a main menu upon app launch.

REQ-34: The system shall allow the player to start a new game from the menu.

REQ-35: The system shall allow the player to exit the game from the menu.

REQ-36: The system shall provide at least one placeholder option for future features

REQ-37: The system shall transition smoothly from the menu screen to the gameplay screen.

REQ-38: The system shall handle invalid or unintended input gracefully (IE pressing a disabled option does not crash game)

4.7 System Feature 7: Non-Functional Requirements

- House Rules + Toggle Menu

- The system shall provide a toggle menu to enable/disable house rules
- Additional Bots (3-7)
- Bot Names
 - The system shall generate/display names for each bot
- PvP (Local Multiplayer)
 - The system shall allow 2 or more human players to play on the same device/lobby
- Customization (Art Style/Skins/Profile pictures)
 - The system shall allow players to choose color palettes and card backs.
- Sound Effects
 - The system shall provide optional audio feedback for card plays and game events
- Accessibility
 - The system shall support color-blind friendly symbols on cards.
- UI Usability
 - The system shall support mouse-based interactions for card selection.
 - The system shall clearly display the card pile and active game state.
- Performance Constraints
 - The system shall run smoothly on a standard Windows OS with minimal lag
 - Future versions shall support MacOS and cross-platform play
- Extensibility for House Rules
 - The system shall support adding new house rules without modifying core game logic.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system should respond to user requests within 2 seconds
- Database operations should complete transactions within 1 second
- Scheduled background tasks shall be expected during peak hours.
 - Rationale : guarantees acceptable user experience and system scalability

5.2 Safety Requirements

- Local data protection and offline operation notes
- System shall not cause unsafe states
 - All transactions must be acceptable and recoverable
- Emergency stop functions must override all active process immediately
 - For physical devices and robotic integration

5.3 Security Requirements

- User authentication shall use multi-factor authentication
- Data in transit and in rest are to be encrypted using TLS 1.3 and AES-256
- Access control for the following role-based authorization, enforcing least-privilege principles
- Audit logs should record login attempts, configuration changes, and data exports.
 - Data for this information should be retained for at least 12 months

5.4 Software Quality Attributes

Reliability	99% uptime excluding maintenance
Maintainability	Modular architecture and short mean time repair
Usability	Consistent UI standards and short click to task completion
Portability	Compatible with major browsers such as Chrome, edge, Ios, ect.
Testability	Unit-test coverage and automated integration testing
Scalability	Scaling supported via containerization or cloud deployment

5.5 Business Rules

- Users who are assigned to the administrator role are the only users who may create, modify, or delete other users
- Regular users may access their own data such as amount of wins/loses, etc.
- All administrator actions must be logged for audit purposes
- All operations must align with the organization's goal
 - Must strictly follow all IT policies and compliance regulations

6. Other Requirements

Database Requirements: None; all game state is managed in memory

Internationalization: The current version supports English only.

Legal: the UNO name and rules are used for academic, non-commercial purposes

Reuse Objectives: Deck Logic, turn management, and rule validation components may be reused in future Unity projects.

Appendix A: Glossary

UNO - A color and number matching card game.

Deck - The set of cards used in a play.

Draw Pile - Stack of undealt cards.

Discard Pile - Stack of played cards.

Bot - Automated player following pre-set logic.

Wild Card - A card that changes color.

Appendix B: Analysis Models

Include UML or flow diagrams for

Use Case: Start Game, Play Card, Draw Card, Declare UNO, End Game

Sequence Diagram: Turn Cycle and Rule Enforcement

Activity Diagram: Gameplay Flow

Appendix C: To Be Determined List

TBD-1: Final Art and Sound assets

TBD-2: Multiplier network functionality

TBD-3: Extended house rules configuration