Software Requirements Specification for UNO Flip

Team 24
Mingyang Xu Kevin Ishak, Zain-Alabedeen Garada, Jianhao Wei, Andy Liang

October 11, 2024

Contents

1 Purpose of the Project

1.1 User Business

The UNO Flip Remix project is designed to bring the classic UNO Flip card game into the digital gaming landscape, catering to the growing demand for online and interactive multiplayer experiences. By targeting casual gamers, students, and board game enthusiasts, the project aims to replicate the traditional game's excitement while incorporating modern features such as AI opponents and real-time synchronization. As part of a capstone initiative, the project not only provides an engaging product for the market but also serves an educational purpose, allowing the development team to apply software engineering principles and tackle real-world challenges.

1.2 Goals of the Project

The UNO Flip Remix project is part of a capstone design initiative undertaken by Team 24. The objective of this project is to develop a multiplayer card game based on the popular UNO Flip game, integrating real-time synchronization and basic AI functionalities to simulate player interactions. The project will focus on both front-end and back-end development to ensure a seamless user experience, using a combination of JavaScript for core functionalities and TensorFlow for reinforcement learning to implement AI behaviors.

2 Stakeholders

2.1 Client

The primary client for the UNO Flip Remix project is McMaster University, as the game is being developed as part of a capstone design project. The university's role is to evaluate the project based on predefined academic and technical criteria, ensuring that the final product meets educational goals while demonstrating the team's proficiency in software development, real-time system design, and AI integration. Feedback from the university's faculty, including course instructors and teaching assistants, will be essential for guiding the project's progress and determining its success.

2.2 Customer

The primary customers for the UNO Flip Remix game are the end users, specifically students, casual gamers, and board game enthusiasts who enjoy digital versions of classic card games. These users seek an engaging online experience that captures the essence of the traditional UNO Flip game, providing entertainment and social interaction in a multiplayer environment. The feedback and satisfaction of these customers will be crucial in guiding future updates and feature expansions.

2.3 Other Stakeholders

Other stakeholders include the development team (Team 24), system administrators, and potential future contributors. The development team is responsible for the end-to-end creation of the game, including coding, testing, and delivering the final product. System administrators will manage the server-side operations, addressing any technical issues to ensure smooth performance and availability. Future developers or contributors, such as open-source enthusiasts, may also be involved in extending the game's features, enhancing AI, or scaling the system.

2.4 Hands-On Users of the Project

Hands-on users consist of the players who will interact with the game, system administrators who ensure that the game servers run efficiently, and developers involved in both the initial development and any future maintenance. Players will experience the game's features firsthand, providing valuable feedback on usability and gameplay mechanics, while administrators will handle technical aspects like server maintenance and troubleshooting.

2.5 Personas

The main personas for this project include casual gamers seeking a fun, online card game, students who play during breaks or as a social activity, and board game enthusiasts who enjoy traditional card games in a digital format. These

personas help guide design choices to cater to different user expectations, such as ease of use for casual players and strategic gameplay options for board game enthusiasts.

2.6 Priorities Assigned to Users

High priority is assigned to casual gamers, who are expected to form the largest user base. Medium priority is given to students who may play during leisure time, while low priority is assigned to potential future developers who could contribute to the project's open-source aspects or further development.

2.7 User Participation

Players will provide feedback through gameplay experiences, which will be used for future improvements. Users will participate in the development process by providing feedback during alpha and beta testing phases. Their input on gameplay, user interface, and any encountered issues will help shape the final product and guide future iterations. Players' suggestions may influence the addition of new features, refinement of existing mechanics, or improvements in AI behavior.

2.8 Maintenance Users and Service Technicians

System administrators will perform maintenance tasks such as server monitoring, software updates, and troubleshooting technical issues to ensure uninterrupted gameplay. Service technicians may also assist with more complex issues involving server configuration or scaling, particularly when the user base increases or when adding new features requires infrastructure changes.

3 Mandated Constraints

3.1 Solution Constraints

The UNO Flip Remix project is constrained by the technologies chosen for development, including JavaScript and Node.js for both front-end and back-end implementations. Socket.IO will be used to manage real-time communication for multiplayer gameplay, while TensorFlow is employed for integrating AI functionalities. These technologies set limitations on how the system can be built, as they must work together seamlessly to ensure cross-platform compatibility and efficient performance. Additionally, all code must adhere to standard best practices for modularity and maintainability.

3.2 Implementation Environment of the Current System

The game is designed to operate as a web-based application, running on various web browsers such as Chrome, Firefox, Safari, and Edge. The development

environment utilizes a combination of JavaScript for the client-side, Node.js for the server-side, and Socket.IO for managing real-time data synchronization. The implementation environment does not include support for mobile apps or offline gameplay in the initial phase, though the system is designed with potential future expansion to mobile platforms in mind.

3.3 Partner or Collaborative Applications

There are no partner or collaborative applications required for the initial release of the UNO Flip Remix game.

3.4 Off-the-Shelf Software

The project will use off-the-shelf software components such as TensorFlow for AI capabilities and Socket.IO for real-time communication. Additionally, standard libraries for web development (e.g., React or Express.js) may be used to accelerate development. These off-the-shelf components will be integrated into the custom codebase to provide essential functionality like AI decision-making and low-latency multiplayer synchronization.

3.5 Anticipated Workplace Environment

The UNO Flip Remix game is intended to be played on standard desktop and laptop computers, using modern web browsers. Players are expected to access the game through stable internet connections, as multiplayer synchronization requires continuous online connectivity.

3.6 Schedule Constraints

The project follows a Scrum development process with weekly sprints, which dictates the pace and structure of the project. The capstone project's academic timeline also constrains the schedule, requiring the game to be completed by a predetermined deadline for evaluation.

3.7 Budget Constraints

The project is constrained by a limited budget provided by the Capstone course, which covers essential development tools and technologies.

3.8 Enterprise Constraints

The development of the UNO Flip Remix game must comply with McMaster University project guidelines and meet academic criteria. In addition, adherence to open-source standards where applicable is required to allow future community involvement.

4 Naming Conventions and Terminology

4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders Involved in the Project

- **UNO** – A card game with specific rules and card actions.
- **AI** – Artificial Intelligence.
- **ML** – Machine Learning.
- **RL** – Reinforcement Learning.
- **3D Rendering** – Three-dimensional graphics rendering system.
- **NPC** – Non-Player Character.
- **CI/CD** – Continuous Integration / Continuous Deployment.
- **DLC** – Downloadable Content.
- **GDPR** – General Data Protection Regulation.

5 Relevant Facts and Assumptions

5.1 Relevant Facts

The project is based on the popular UNO Flip card game, which has well-established gameplay mechanics. JavaScript and Node.js are being used as the primary languages for front-end and back-end development, ensuring cross-platform compatibility. Socket.IO is employed to handle real-time communication, enabling multiplayer gameplay with low-latency synchronization. TensorFlow is used for the AI implementation, which will control non-player characters (NPCs) during the game.

5.2 Business Rules

- **Game Integrity:** The rules of UNO Flip must be strictly followed. The server will enforce game logic, including player turns, card actions, and the application of special card effects like "Flip," "Skip," or "Draw."
- **Player Limits:** The game will allow a minimum of 2 players and a maximum of 4 players in multiplayer mode. Additional player capacities could be explored in future updates.
- **Card Distribution:** Each player will receive 7 cards at the start of the game. The server will randomize the deck to ensure fairness in card distribution.

- **Game Modes:** Only the "standard" and "flip" sides of the game will be available in the initial version. Future iterations may add variations to the gameplay or more modes.
- **No Cheating Policy:** Any attempts to manipulate or cheat the system (e.g., unauthorized access to player card data) will result in immediate disqualification from the match and possibly a temporary or permanent account ban.
- **Matchmaking:** Players will be automatically matched with opponents based on skill levels or allowed to invite friends to private games.

5.3 Assumptions

- Players are expected to have reliable internet connections, ensuring smooth gameplay and minimal latency issues.
- Players will use modern browsers (e.g., Chrome, Firefox) that support JavaScript and WebSockets.
- The AI will initially be rule-based, with the potential to evolve into more complex AI using machine learning techniques in future iterations.
- The game will initially support up to 4 players, with scalability planned for future updates.
- Players have a basic understanding of the rules of UNO Flip. Tutorials will be available for new users but not mandatory for experienced players.

6 The Scope of the Work

6.1 The Current Situation

Currently, UNO Flip is a popular physical card game enjoyed by small groups of players. Players must be physically present to engage in gameplay, shuffling decks, and manually handling game mechanics like drawing cards and flipping the deck. The game lacks an official online platform, limiting its accessibility to in-person gatherings. Additionally, tracking scores and enforcing rules is dependent on the players themselves, which may lead to errors or misunderstandings.

6.2 The Context of the Work

The project aims to develop an online, multiplayer version of UNO Flip that not only mimics the physical game but also enhances the user experience with engaging and dynamic features. This digital version will allow players to connect and play in real-time, regardless of location, and will automatically enforce all game rules to ensure fairness. Visually stunning animations and dynamic features will further enhance the excitement, especially during special moves

such as “Flip,” “Draw,” and “Reverse,” making the game more immersive and enjoyable.

6.3 Work Partitioning

Front-End: The front-end development includes all user interface (UI) and user experience (UX) tasks, such as designing a visually engaging game interface, implementing animations for card actions, and ensuring responsiveness across devices (e.g., desktops, tablets, mobile). Front-end developers will also handle real-time interactions, including drag-and-drop functionality for card movements, interactive menus, and chat integration for multiplayer communication.

Back-End: The back-end development will focus on building and maintaining the server-side infrastructure. This includes implementing game logic, managing player connections, handling real-time synchronization of game states, user authentication, matchmaking, and turn-based systems. Node.js and WebSockets will be used for real-time communication, while AI logic will be implemented for single-player or AI-assisted multiplayer games.

6.4 Specifying a Business Use Case (BUC)

Player Login: Players must log into the platform using a unique ID or credentials. Upon logging in, the system retrieves the player’s profile, game history, and scores. The system ensures the player’s session is secure and maintains continuity between different gaming sessions.

Game Setup: Once logged in, players can create a game room or join an existing room. The game setup includes customizable options like the number of players, game mode (e.g., standard or flip mode), and AI opponents. The host of the game room has administrative rights to set and adjust these configurations.

Turn Management: The system automatically manages turn-taking during the game, enforcing the rules to ensure that each player performs valid moves. The system also ensures the correct order of play is followed and handles the transition between players smoothly.

AI Opponents: The AI players will follow predefined strategies, making decisions in real-time. The AI’s behavior will initially be rule-based, with future potential for more advanced strategies using machine learning techniques.

Scoring: At the end of each game, the system calculates scores based on the cards remaining in each player’s hand. These scores will be stored in the player’s profile, and a leaderboard will track the highest scores. Additionally, player statistics, such as the number of games played and win/loss ratio, will be updated after each game.

7 Business Data Model and Data Dictionary

7.1 Business Data Model

The following entities and attributes represent the core of the game's business data model:

- **Player:**
  - **PlayerID:** Unique identifier for the player.
  - **Username:** The name displayed during the game.
  - **Wins:** Total number of games won.
  - **Losses:** Total number of games lost.
  - **CurrentHand:** A list of cards the player holds.
- **Game Room:**
  - **RoomID:** Unique identifier for the game room.
  - **HostID:** PlayerID of the host.
  - **PlayerList:** List of players currently in the room.
- **Game:**
  - **GameID:** Unique identifier for each game session.
  - **GameState:** The current state of the game (e.g., "In Progress," "Completed").
  - **TurnOrder:** The order of players in the game.
  - **CurrentDeck:** A shuffled list of cards.
  - **DiscardPile:** The cards that have been played and discarded.
- **Card:**
  - **CardID:** Unique identifier for the card.
  - **Color:** The color of the card (e.g., red, yellow, green, blue).
  - **Value:** The value of the card (e.g., 1-9, Draw 2, Skip, Reverse, Flip).
- **Deck:**
  - **DeckID:** Unique identifier for each deck.
  - **Cards:** List of all cards in the deck, including normal and flip sides.

7.2 Data Dictionary

- **PlayerID:** Integer, Unique identifier for each player in the game.
- **Username:** String, The name chosen by the player displayed during the game.
- **Wins:** Integer, Tracks the total number of games won by the player.
- **Losses:** Integer, Tracks the total number of games lost by the player.
- **CurrentHand:** List of Card objects, Represents the cards currently held by the player.
- **RoomID:** Integer, Unique identifier for each game room.
- **HostID:** Integer, The PlayerID of the player who created the game room.
- **PlayerList:** List of PlayerID, A list representing the players in the game room.
- **GameID:** Integer, Unique identifier for each game session.
- **GameState:** Enum, (In Progress, Completed), The current state of the game.
- **TurnOrder:** List of PlayerID, An ordered list of PlayerIDs representing the order of players' turns.
- **CurrentDeck:** List of Card objects, A shuffled list of cards used in the current game.
- **DiscardPile:** List of Card objects, A list of cards that have been played and discarded.
- **CardID:** Integer, Unique identifier for each card.
- **Color:** Enum, (Red, Yellow, Green, Blue), The color of the card.
- **Value:** Enum, (1-9, Draw 2, Skip, Reverse, Flip), The value or action of the card.
- **DeckID:** Integer, Unique identifier for each deck used in a game.
- **Cards:** List of Card objects, A list of all cards present in a deck, including both normal and flip sides.

8 The Scope of the Product

8.1 Product Boundary

The UNO Flip project will deliver a multiplayer card game focused on strategic depth and AI-driven gameplay. The primary environment is a web-based platform with potential future expansion to mobile. Real-time multiplayer synchronization and dynamic card interactions are central to the experience, as well as AI opponents that can adapt and learn from gameplay. The project will not initially support offline play or mobile platforms.

8.2 Product Use Case Table

ID Goal	Use Case Name	Primary Actor
UC-01 Create a new multiplayer game	Create Game Room	Player
UC-02 Join an existing multiplayer game	Join Game Room	Player
UC-03 Play a card during the game	Play Card	Player
UC-04 Perform AI action in response	AI Turn	System (AI)
UC-05 Declare winner and finish the game	End Game	System

8.3 Individual Product Use Cases (PUC's)

UC-01: Create Game Room
Primary Actor: Player
Preconditions:

- The player is logged into the system.
- The system is connected to the server and has access to multiplayer services.

Main Flow:

1. The player clicks on "Create Game Room."
2. The system presents a configuration menu for the game (e.g., number of players, AI opponents).
3. The player configures the game and clicks "Start."
4. The system creates a game room, generates a unique room code, and displays it to the player.
5. The system waits for other players to join using the room code.

Postconditions:

- A game room is created, and the player is in the lobby, waiting for others to join.

UC-02: Join Game Room
Primary Actor: Player
Preconditions:

- The player has a valid game room code.
- The player is logged into the system.

Main Flow:

1. The player selects "Join Game Room."
2. The system prompts the player to enter a valid room code.
3. The player enters the code and submits the form.
4. The system validates the room code.
5. Upon successful validation, the player is placed in the game lobby with other participants.

Postconditions:

- The player successfully joins the game room.

UC-03: Play Card
Primary Actor: Player
Preconditions:

- It is the player's turn in the game.
- The player has valid playable cards in their hand.

Main Flow:

1. The system displays the player's hand and valid card options.
2. The player selects a card to play.
3. The system validates the card choice against the current game state (e.g., matching color or number).
4. The system updates the game state to reflect the played card.
5. The system notifies the next player to take their turn.

Postconditions:

- The player's card is successfully played, and the turn passes to the next player.

UC-04: AI Turn
Primary Actor: System (AI)
Preconditions:

- It is the AI's turn in the game.

Main Flow:

1. The AI analyzes the current game state and evaluates its hand of cards.
2. The AI selects the best card based on predefined strategy rules or adaptive learning.
3. The system validates the AI's card choice.
4. The AI plays the selected card.
5. The system updates the game state and proceeds to the next player's turn.

Postconditions:

- The AI successfully plays its turn, and the game state is updated.

UC-05: End Game
Primary Actor: System
Preconditions:

- The game has reached a win condition (e.g., a player has no more cards).

Main Flow:

1. The system detects the end game condition (e.g., no cards remaining for a player).
2. The system displays the results, showing the winning player and the scores of all participants.
3. The system offers players options to play again or exit.

Postconditions:

- The game ends, and players are provided with results and options to continue.

9 Functional Requirements

9.1 Functional Requirements

Authentication:

- AR1 - Players must log in or create an account to access multiplayer modes.

- AR2 - User profiles will store game statistics and preferences.

Game Setup:

- GSR1 - Players can create game rooms and invite others or join existing rooms.
- GSR2 - Options for choosing game rules and difficulty settings.

Turn Management:

- TMR1 - The system will manage player turns, ensuring synchronization across devices.
- TMR2 - Notifications will alert players when it is their turn.

Chat Functionality:

- CFR1 - Real-time chat within the game room for players to communicate.
- CFR2 - Chat can be enabled or disabled based on player preferences.

Scoring System:

- SSR1 - A detailed scoring mechanism to track and display player scores at the end of each game.
- SSR2 - The system will keep a record of player stats, including wins, losses, and average score.

End Game Handling:

- EGHR1 - The game will automatically declare a winner based on the rules and display a summary of the results.
- EGHR2 - Players can choose to play another round or exit the game.

Multiplayer Synchronization:

- MSR1 - A state synchronization mechanism will ensure that all players see the same game state at all times, regardless of network conditions.
- MSR2 - The system will manage latency and update the game state in real time.

10 Look and Feel Requirements

10.1 Appearance Requirements

- AR1 - The game interface should have a modern design that is visually appealing.
- AR2 - Consistency in color schemes, fonts, buttons, and animation effects should be maintained throughout the game.
- AR3 - Card designs should be easily recognizable, with clarity maintained during 3D rendering.

10.2 Style Requirements

- SR1 - The interface should be adaptive, ensuring proper display on different screen resolutions for mobile, tablet, and desktop devices.
- SR2 - Actions in the game (e.g., flipping a card, playing a card) should feature smooth and responsive animations.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

- EOUR1 - New players should be able to quickly grasp the gameplay, with the system providing simple and clear tutorials.
- EOUR2 - The game menu and control buttons should be easy to navigate, allowing players to easily create or join rooms.
- EOUR3 - Drag-and-drop actions for playing cards should be intuitive and highly responsive.

11.2 Personalization and Internationalization Requirements

- PALR1 - The game should support language localization, offering options for multiple languages (e.g., English, French, Spanish).
- PALR2 - Players can customize personal settings, such as sound effects, background music, and interface themes, for a personalized experience.

11.3 Learning Requirements

- LR1 - The system will offer interactive tutorials for new players to understand the rules of UNO Flip and the effects of special cards.
- LR2 - AI will guide players to familiarize themselves with advanced strategies and dynamically adjust difficulty based on player performance.

11.4 Understandability and Politeness Requirements

- UAFR1 - All system notifications and error messages should use concise, friendly language without technical jargon.
- UAFR2 - Detailed game help and rule explanations should be available for players to consult at any time.

11.5 Accessibility Requirements

- AR1 - The game should support accessibility features such as colorblind modes, text enlargement options, and a soundless mode for hearing-impaired users.
- AR2 - For mobile devices, the game should support multi-touch controls and be compatible with accessibility devices.

12 Performance Requirements

12.1 Speed and Latency Requirements

- SALR1 - The average response time for the game should be within 50 milliseconds to ensure smooth interaction.
- SALR2 - Network latency in multiplayer mode should be kept under 200 milliseconds to avoid inconsistencies in the game state.

12.2 Safety-Critical Requirements

- SCR1 - Changes in the card and game state must be synchronized in real-time to ensure all players see the same status.
- SCR2 - Disaster recovery mechanisms should be in place to save game progress and resume play in case of server failure.

12.3 Precision or Accuracy Requirements

- POAR1 - When calculating scores and game state, the system should ensure high precision to avoid incorrect scorekeeping or errors in the game logic.

12.4 Robustness or Fault-Tolerance Requirements

- ROFR1 - The system should be fault-tolerant, allowing players to reconnect quickly after a network interruption without losing game progress.
- ROFR2 - The server should handle a surge in user logins without crashing.

12.5 Capacity Requirements

- CP1 - The system should support at least 1,000 concurrent users, with each game room running independently.
- CP2 - Each game room should support up to 4 to 8 players.

12.6 Scalability or Extensibility Requirements

- SOER1 - The system should be scalable, allowing for future additions of new game modes, features, or AI difficulty levels.
- SOER2 - As the player base grows, the server should be able to scale dynamically to handle more concurrent users.

12.7 Longevity Requirements

- LR1 - The game's code should be maintainable for long-term use, supporting future updates, performance optimizations, and feature expansions.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

- EPER1 - The system will primarily run on desktop and laptop platforms (Windows, macOS) using modern web browsers.

13.2 Wider Environment Requirements

- WER1 - The game must perform under varying network speeds, implementing latency mitigation techniques such as rollback networking to ensure smooth gameplay even in slower or unstable connections.
- WER2 - The game must detect internet interruptions and allow players to reconnect without losing game progress.
- WER3 - The game must be compatible with all major web browsers (Chrome, Firefox, Safari, and Edge).

13.3 Requirements for Interfacing with Adjacent Systems

- RFIWASR1 - The game must integrate with third-party authentication providers (e.g., Google, Facebook) to allow users to log in using existing accounts.
- RFIWASR2 - The game should integrate with social media platforms for sharing gameplay results or achievements.

13.4 Productization Requirements

- PR1 - The game must undergo thorough testing across multiple devices, browsers, and network conditions to ensure consistent performance and identify any compatibility issues.
- PR2 - The system must undergo performance optimization to handle at least 1,000 concurrent users across different multiplayer rooms.

13.5 Release Requirements

- RR1 - The initial release must include a robust help system, including in-game tutorials and access to FAQs or troubleshooting guides.
- RR2 - The system must support post-launch patches and updates without requiring users to reinstall or experience significant downtime.
- RR3 - The release version must undergo comprehensive regression testing to ensure that all core functionalities work as expected across devices and browsers.

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

- MR1 - The game must maintain a frame rate of at least 30 FPS across all supported platforms, ensuring smooth animations and interactions.
- MR2 - The system must support up to 1,000 concurrent users in multiplayer mode without significant degradation in performance or user experience.
- MR3 - Actions taken by players, such as playing or drawing a card, must occur in under 500 milliseconds to avoid noticeable delays in gameplay.

14.2 Supportability Requirements

- SR1 - The system should offer online technical support, and players should be able to report issues in-game or via the website.
- SR2 - The system must detect and recover from unexpected disconnections, allowing players to resume gameplay within 10 seconds of a reconnection.

14.3 Adaptability Requirements

- AR1 - The system must ensure 24/7 availability for users worldwide, with planned downtime limited to maintenance windows of no more than 2 hours per month.
- AR2 - Maintenance and updates must be rolled out during off-peak hours to minimize disruption to users.
- AR3 - The game must include an offline maintenance mode where players can still practice against AI when the server is unavailable.

15 Security Requirements

15.1 Access Requirements

- AR1 - Players must securely log in to the game, with support for two-factor authentication (2FA) for added security.
- AR2 - The game should have a password recovery mechanism to allow users to recover their passwords in case they forget them.

15.2 Integrity Requirements

- IR1 - All player data (such as game progress and personal stats) should be encrypted to prevent unauthorized modification.
- IR2 - The game should not allow weak passwords (e.g., passwords composed only of numbers or letters) for login credentials.

15.3 Privacy Requirements

- PR1 - Player data (such as personal information and game records) should comply with privacy policies to prevent data breaches.
- PR2 - The game should notify users that they should not share their personal information with others in the game room.
- PR3 - The game should automatically log users out after a certain period of inactivity to prevent unauthorized access.

15.4 Audit Requirements

- AR1 - The system should log all critical operations in the backend to allow for auditing and troubleshooting.
- AR2 - In case of frontend crashes, the game should provide crash information to the backend for future improvements (with user consent).

15.5 Immunity Requirements

- IR1 - The system should be immune to common network attacks (e.g., DDoS, SQL injection) to ensure the safety of player data and gameplay.
- IR2 - The system should be resilient to network issues, such as fluctuations in connection speed.

16 Cultural Requirements

16.1 Cultural Requirements

- CR1 - The game should be adapted to global markets, avoiding cultural sensitivities and contentious topics.
- CR2 - The content and language shown in the game should not be offensive to any particular culture.
- CR3 - The game should respect the cultural customs and religious doctrines of the countries where it is released.

17 Compliance Requirements

17.1 Legal Requirements

The UNO Flip Remix project must comply with data protection regulations such as the General Data Protection Regulation (GDPR) for European users and the California Consumer Privacy Act (CCPA) for users in California. This requires the game to implement robust data protection measures, including encryption for personal data, secure storage practices, and mechanisms to obtain explicit user consent for data collection and processing. User rights, such as the ability to access, rectify, or delete their personal information, must be respected, with clear procedures for users to exercise these rights. Privacy policies must be transparent, detailing the types of data collected, purposes for data processing, and third-party data sharing practices.

17.2 Standards Compliance Requirements

- SCR1 - The game must comply with international standards and local laws, especially regarding data protection regulations like GDPR.
- SCR2 - The system's development should adhere to industry best practices for code quality and testing.

18 Open Issues

Several open issues need to be resolved as the project progresses:

- The decision between a rule-based AI and reinforcement learning remains unresolved and will directly impact both the performance and development timeline.
- Ensuring smooth real-time synchronization in multiplayer mode as the number of concurrent players increases will require further testing, particularly with network latency.

- Cross-platform support for mobile devices presents challenges, particularly in optimizing the user interface and game performance for mobile networks.

19 Off-the-Shelf Solutions

19.1 Ready-Made Products

The UNO Flip Remix project will utilize several existing software solutions to expedite the development process and ensure a high-quality gaming experience:

- **TensorFlow:** Employed for integrating AI functionalities, TensorFlow will power the decision-making process for AI opponents, providing a more challenging and dynamic gameplay experience.
- **Socket.IO:** Used for real-time communication, this framework ensures smooth synchronization across players during multiplayer sessions, minimizing latency.
- **Firebase:** This cloud-based solution will be leveraged for data storage and user management, including game data, player profiles, and session management.
- **3D Graphics Libraries (e.g., Three.js):** Libraries like Three.js will facilitate the rendering of 3D elements, animations, and lighting effects, enhancing the visual experience of the game.

19.2 Reusable Components

To improve efficiency and ensure high-quality outcomes, the following reusable components will be incorporated:

- 3D Game Frameworks such as Three.js for creating 3D objects, animations, and scene management.
- Authentication Libraries (e.g., Passport.js) for implementing user authentication.
- AI Algorithms for Card Games for card decision-making, providing a foundation for initial AI functionalities.
- UI/UX Frameworks (e.g., React) for designing a polished user interface.

19.3 Products That Can Be Copied

The existing version of the UNO Flip game available at unoonlinegame.io serves as a reference for this project. While it provides a functional basis, the UNO Flip Remix project aims to improve its visuals, gameplay flow, and features.

- **Visual Overhaul:** The current game's simplistic graphics will be replaced with a more dynamic 3D presentation, including card animations and interactive game boards.
- **Enhanced Gameplay Flow:** The game will be improved with smoother transitions, refined animations, and interactive elements for a more responsive and exciting gameplay experience.
- **Additional Features:** Modern features like real-time lighting effects, dynamic sound design, and fluid animations will enhance the overall user experience.

20 New Problems

20.1 Effects on the Current Environment

The transition from a physical card game to a digital interface introduces several challenges:

- Players who are accustomed to the physical version may face a learning curve with the digital interface.
- The game's dependency on stable internet connections means that network interruptions can disrupt gameplay, unlike the offline nature of physical card games.

20.2 Effects on Installed Systems

- Ensuring cross-platform compatibility across devices such as desktops, laptops, and mobile devices will require system optimizations and updates.
- Security measures like anti-cheating systems may require updates to prevent manipulation of online scores or game states.

20.3 Potential User Problems

- Users unfamiliar with the digital interface might struggle with game controls, such as flipping cards.
- In-game cheating could become a problem, requiring robust security measures to ensure fair play.
- Disconnected sessions due to network issues might cause frustration, especially in competitive gameplay.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

- Players with slow or unstable internet connections may experience lag or interruptions in gameplay.
- Low-end devices may struggle with handling the game's 3D graphics and real-time multiplayer synchronization.
- Synchronization across devices with varying performance capabilities may lead to discrepancies in game experience.

20.5 Follow-Up Problems

- Regular updates and patches may be required to address bugs, introduce new features, or balance the game mechanics, without breaking the existing functionality.
- Long-term player retention may depend on the addition of new content and features to keep the game engaging and competitive.

21 Tasks

21.1 Project Planning

Our team will hold weekly meetings to track progress and align on project tasks. Each team member is expected to attend these meetings. Additionally, we will organize informal work sessions throughout the week to collaborate on specific tasks and document our progress.

Weekly Scrum meetings:

- Weekly Scrum meetings will be conducted with all team members attending.
- Each meeting will last between 15 and 30 minutes and will occur on Microsoft Teams.
- Members will discuss progress and any blockers encountered.

Additional work sessions:

- At least one additional work session per week will be held where coding and design tasks will be conducted collaboratively.

Primary communication will occur on Microsoft Teams, supplemented by cell phone coordination when necessary. The team will also utilize GitHub for version control and code reviews.

21.2 Planning of the Development Phases

The project will follow a Git-flow branching model to streamline development and ensure effective collaboration among team members. In this model, each new feature or bug fix will be developed in its own dedicated branch. This approach allows for focused work on individual components without disrupting the main codebase.

Once a feature is completed, team members will submit a pull request to merge their changes into the master branch. Each pull request will undergo a review process where other team members can provide feedback, suggest improvements, and identify potential issues before integration. This collaborative review process enhances code quality and promotes knowledge sharing within the team.

In addition to code reviews, we will implement unit testing to ensure the reliability and functionality of our code. Each module or feature will include a set of unit tests that validate its behavior and performance against specified requirements. These tests will be written alongside the code and run regularly to catch any regressions early in the development process. By emphasizing unit testing, we aim to maintain high standards of code quality and reduce the likelihood of bugs in the final product.

22 Migration to the New Product

22.1 Requirements for Migration to the New Product

- Data Integration: Existing player data, including usernames, scores, and progress from any prototype or beta versions, must be seamlessly integrated into the new system.
- System Compatibility: The new digital platform should be compatible with various devices (desktop, mobile, and tablet) and operating systems to ensure a consistent user experience across all platforms.
- User Authentication and Data Security: Secure authentication mechanisms must be in place to protect user accounts and personal information. Existing authentication data may need to be encrypted or reformatted to meet new security standards.
- AI Implementation Adaptation: The AI behavior from the initial rule-based prototype needs to be adapted and optimized for the TensorFlow-based reinforcement learning model planned for the new system.
- Rollback Networking Technique: The implementation of rollback networking techniques must be tested and adjusted to maintain real-time synchronization in multiplayer sessions, ensuring that latency issues are minimized during gameplay.

- Training and Support: Detailed documentation and tutorials should be provided to users to guide them through the features of the new digital version, emphasizing the differences between the physical and digital gameplay experiences.

22.2 Data That Has to be Modified or Translated for the New System

- Data Format Transformation: Data from the original or prototype version of the UNO Flip game, including card decks, player profiles, and game history, may need to be reformatted to match the new system's data schema.
- User Profile Data: Existing user data, such as login credentials, player stats, and in-game achievements, must be translated into a format compatible with the new authentication and storage standards used by the digital platform.
- AI Training Data: Historical gameplay data should be converted into training datasets for the TensorFlow-based AI. This data must be processed to suit the specific input format required for reinforcement learning algorithms.
- Card Interaction Logic: The logic governing card interactions and game rules from the traditional version should be modified to fit the digital framework, ensuring that the new system accurately replicates the physical gameplay mechanics.
- Multiplayer Session Data: Information related to multiplayer sessions, such as game states and player interactions, must be adapted to function within the Socket.IO framework to enable real-time communication and synchronization.

23 Costs

Item	Details	Cost
Cloud Providers	We need servers to host player data and manage game logic.	$100
Data Storage	Firebase for data storage and user management.	$50
Graphics Support (API)	Possible need for graphic APIs (e.g., OpenGL) for 3D effects.	$20-$70

24 User Documentation and Training

24.1 User Documentation Requirements

- UDR1 - A visual guide to the game's user interface, covering controls for card selection, flipping between sides, and accessing in-game settings.

- UDR2 - A section addressing common issues, such as resolving connection problems, understanding game mechanics, and how scoring works.
- UDR3 - Basic troubleshooting steps for resolving common issues, such as server disconnects or game freezes.

24.2 Training Requirements

- TR1 - New players will be guided through a hands-on, in-game tutorial that explains game rules, mechanics, and card effects in real-time.
- TR2 - A help section within the game, accessible at any time, will provide reminders of the rules and descriptions of card functions.
- TR3 - Players can practice in solo mode against AI opponents before entering multiplayer matches to get familiar with the interface and rules.

25 Waiting Room

The following features and functionalities may be considered as future stretch goals as outlined in the development plan:

- Advanced AI using reinforcement learning for more sophisticated gameplay.
- Mobile App Version: Focus on a web-based platform in the current phase, with mobile app development to be explored later.
- Cross-Platform Play: Potential future support for players on different devices to play together.
- In-Game Purchases: Monetization through in-game purchases or microtransactions is not planned for this phase.
- Offline Play: The game will require a continuous internet connection; offline play will be considered in later versions.
- Custom Game Modes: Future iterations may introduce customizable or additional game modes.
- Voice Chat Integration: Text-based chat functionality will be implemented, with voice chat considered for future releases.
- Network Latency Optimization: Advanced latency management techniques (e.g., rollback networking) will be explored in future updates.
- Scalability for Larger Games: The system will initially support a limited number of players, with larger capacities to be considered later.

26 Ideas for Solution

The UNO Flip Remix project will include the following features and functionalities:

- Player Authentication: Users can create accounts, log in, and manage their profiles.
- Multiplayer Functionality: Players can create or join game rooms to compete in real-time against other human players.
- Turn Management: The game will follow the turn-based rules of UNO Flip, ensuring proper synchronization and validation of each player's moves.
- Basic AI Opponents: AI-controlled players will be available to fill game rooms or provide an alternative to multiplayer.
- Chat Functionality: Players will have access to an in-game chat system for communication during matches.
- Scoring System: A scoring mechanism will track player performance throughout each match.
- End Game Handling: The system will handle end-game scenarios, announcing winners and displaying game statistics.
- Multiplayer Synchronization: The game state will be synchronized across all clients in real-time, ensuring consistent gameplay for all players.
- Non-functional Requirements: Performance, security, and usability requirements, such as data protection, latency minimization, and ease of use, will be addressed in Section 6.

27 Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

27.1 Reflection Questions

1. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain-specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge, or computer science knowledge. Skills may be related to technology, writing, presentation, team management, etc. You should identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

28 Revision History

Revision Number	Name	Notes
Revision 0	Kevin Ishak and Mingyang Xu	Created first draft
Revision 1	Mingyang Xu and Jianhao Wei	Added functional and non-functional requireme
Revision 2	Mingyang Xu and Jianhao Wei	Added functional and non-functional requireme
Revision 3	Andy Liang	Updated Terms, Acronyms and Abbreviations
Revision 4	Zain-Alabedeen Garada	Updated sections 8, 13, 14 and 18.