Hazard Analysis

UNO-FLIP Remix

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

Table 1: Revision History

Date	Developer(s)	Change
10/25 /2024	Mingyang Xu, Jiahao Wei Kevin Ishak	Wrote section 1-7
10/25 /2024	Andy Liang	Wrote Appendix: Reflection for team and self
...	...	...

Contents

1 Introduction

This document is the hazard analysis of the UNO Flip Remix game. This document analyzes potential hazards within the UNO Flip Remix digital game project. Hazards are identified and assessed to ensure safety, data integrity, and uninterrupted gameplay, particularly in a multiplayer online setting.

2 Scope and Purpose of Hazard Analysis

The analysis focuses on identifying hazards related to user interaction, server reliability, AI behaviors, and multiplayer synchronization, and offers mitigation strategies to maintain gameplay integrity and security

The UNO Flip system includes:

1. The game client (browser-based app for users).
2. The server infrastructure (hosted on Firebase and utilizing WebSockets for real-time communication).
3. AI components for non-player character behaviors.
4. 3D rendering and animation libraries (such as Three.js).

hazard is a property or condition in the system together with a condition in the environment that has the potential to cause harm or damage = loss (From Nancy Leveson's work)

The purpose of the hazard analysis in the UNO Flip game project is to identify potential risks or hazards that could lead to system failure, user dissatisfaction, or even data loss. In this case, the game interacts with multiple users in real time, and improper handling of certain hazards could result in gameplay disruption, incorrect scores, or unexpected errors. The key is to anticipate what could go wrong, understand the consequences, and define strategies to mitigate the impact.

Losses that could be incurred due to identified hazards include:

- **Gameplay Integrity Loss:** Errors during critical moments of the game such as missed turns, improper shuffling of cards, or skipping certain rules, leading to user frustration and unfair results.
- **Data Loss:** Game data such as scores or progress may be lost due to incorrect saving mechanisms or interruptions in communication between the client and server.
- **Security Breach:** Unauthorized users might manipulate the game by altering scores, cheating, or causing denial-of-service attacks to disrupt the gameplay for other users.
- **User Experience Loss:** The game could lag or crash during critical moments due to insufficient handling of concurrent operations, causing a poor user experience.
- **System Failure:** Hardware or network issues could lead to incomplete games, players getting disconnected, or the system not recognizing inputs properly, forcing players to restart the game.

This analysis aims to highlight potential areas of concern in order to create a safer, more robust system that prioritizes smooth gameplay and user satisfaction.

3 System Boundaries and Components

In this UNO Flip game project, we need to define key components. The system consists of several major software components that work together to ensure the functionality and smooth execution of the game. The components include:

1. **User Interface (UI):**
    - The UI is where players interact with the game. It must handle input correctly, provide real-time feedback, and

display the current game state. Any issues in this component could result in incorrect feedback to players or missed actions (e.g., playing a card, flipping the deck).

2. **Game Logic:**
    - This component controls the core gameplay mechanics, such as turn-based actions, deck shuffling, enforcing rules, and tracking game state (e.g., current turn, card count). Bugs or inconsistencies in the game logic could cause incorrect card counts, skipped turns, or illegal moves.
3. **Networking Module:**
    - In the case of multiplayer versions, the networking module facilitates communication between players. Hazards in this module might include lost connections, synchronization issues, or data packet loss, which could lead to players being disconnected or out of sync with the game state.
4. **Database or Game State Storage:**
    - A component responsible for saving game progress, scores, or user profiles (if necessary). Failure in this component could result in loss of player data, game history, or leaderboard standings.
5. **Card Management System:**
    - Handles the cards (e.g., shuffling, drawing, flipping decks). Errors here could cause the wrong cards to be drawn, failure to flip the deck in "Flip" mode, or incorrect deck resets.
6. **Server (if applicable):**
    - In the case of an online version, the server manages player interactions and game sessions. Server failures could result in crashed game sessions, unresponsive behavior, or failure to update player actions.
7. **Audio/Visual Effects Module:**
    - Enhances the user experience with sounds, animations, and visual cues. Malfunctions could lead to misaligned animations or missing sound cues, negatively affecting gameplay flow.

4 Critical Assumptions

User Inputs:

- It is assumed that the players will provide valid inputs (e.g., correctly play their cards and follow game rules). However, the system will still need to validate inputs to prevent illegal moves, misclicks, or game errors caused by incorrect inputs.

Network Stability (for multiplayer version):

- It is assumed that the network connection remains stable throughout the game session. However, we recognize that disconnections or poor network performance could lead to synchronization issues, and thus, proper error-handling mechanisms should be implemented to allow reconnections or resynchronization when needed.

System Resources:

- The system is assumed to have sufficient memory and processing power to handle the real-time execution of the game, animations, and sound effects. However, stress-testing will be conducted to ensure that the game can handle edge cases like simultaneous actions, card flips, or large numbers of players without crashing.

Server Reliability (if applicable):

- The server is assumed to handle multiple game sessions concurrently without performance degradation. However, in case of high load or server crashes, fallback measures such as session saving and game state restoration will be implemented.

Deck Management:

- It is assumed that the deck of cards is shuffled properly and that the flipping mechanism works without issues. Any failure in shuffling or card handling could disrupt the game, so the system will ensure that the deck is shuffled randomly and that the Flip feature triggers correctly.

Game Logic Accuracy:

- It is assumed that the game's logic engine, which handles turns, rule enforcement, and game progression, is bug-free. However, extensive testing will be required to ensure that no game-breaking bugs or rule misinterpretations occur during gameplay.

Audio/Visual Synchronization:

- It is assumed that all audio and visual effects are triggered at the correct time. If these effects are not synchronized, players may miss critical game cues. The system will be tested to ensure that animations and sounds match gameplay events.

5 Failure Mode and Effect Analysis

Design function	Failure mode	Effects of failure	Causes of failure	Detection	Recommended action	SR	Ref
Multiplayer sync	Loss of connection	Player can’t interact with the game	Internet disconnection	Monitor connection status	Allow player reconnection to ongoing session	MSR1	R1-1
	Desynchronization between players	Players experience different game states	Network latency or packet loss	Discrepancy between player game states	Implement rollback networking to resync	IR1	R1-2
AI Behavior	AI fails to take its turn	Game halts or delays unexpectedly	Processing timeout in AI	Game round timer expires without action	Set AI move timeout and default action	IR3	R2-1
	AI selects an invalid card	Game logic breaks or inconsistencies in card play	AI decision algorithm error	Game logs show invalid card choice by AI	Implement stricter rule validation for AI actions	IR3	R2-2
Authentication	Unauthorized access to game rooms	Unprivileged users access restricted game areas	Weak passwords or vulnerabilities	Failed 2FA attempts or password breaches	Enforce strong passwords and two-factor authentication	AR1	R3-1
	User login failure	Players cannot access	Incorrect credentials	Multiple failed login	Implement password	AR2	R3-2

		multiplayer features	or server issue	attempts logged	recovery and error messages		
Card Flip Mechanic	Incorrect card flip during gameplay	Game state differs from intended player action	Synchronization lag or UI error	Card actions do not match server validation	Validate player actions server-side	IR2	R4-1
	Flip effect does not trigger correctly	Game dynamics are disrupted	Code error in flip functionality	Game state shows inconsistency in card values	Regular testing of flip mechanics	IR2	R4-2
Game Room Access	Room inaccessible after creation	Players cannot join or rejoin the game room	Room ID generation failure	Room code missing or unrecognized by server	Reinitialize room creation logic	GR1	R5-1
	Multiple players assigned the same room ID	Players are joined into incorrect game rooms	Room ID collision due to poor generation logic	Duplicate room IDs detected in database logs	Generate unique room codes with stronger hashing	GR1	R5-2
Server Stability	Server downtime	All players are disconnected mid-game	Overload or Firebase server issue	Error logs showing connection failures	Set up redundant/fall back servers	IM1	R6-1
	Game performance degrades under high load	Slow game response or lag for all players	Insufficient server resources	High CPU or memory usage alerts	Scale server resources during peak usage	IM2	R6-2

Data privacy	Data leak in game chat or history	Player information is exposed to unauthorized users	Insufficient encryption or session management	Alerts for unauthorized access attempts	Encrypt all user data and implement session timeouts	PR1	R7-1
	Unauthorized access to player profiles	Players' stats and personal data compromised	Weak session controls or API vulnerability	Failed login attempt logs	Strengthen session control and require re-authentication	PR2	R7-2
Leaderboard	Score or ranking is inaccurate	Players lose track of progress or ranking	Score calculation error or database sync failure	Discrepancies in displayed scores or ranks	Regular validation of score data in the database	IR4	R8-1
	Leaderboard updates delayed	Player standings not immediately reflected	High server load during peak times	Delayed updates in leaderboard records	Optimize database queries and refresh intervals	IR4	R8-2

Table 1: FMEA

6 Safety and Security Requirements

6.1 Access Requirements

AR1. Only server administrator can modify basic game features, such as layout, implementing new rules

AR2. Only server administrators are able to modify other user's accounts. But this has to be done under the consent of the user

6.2 Integrity Requirements

IR1. The software system should not be modify the user information and game data unintentionally

IR2. The software system should conduct authentication checks for all of the users within a certain period.

IR3. User's app should store all the data that are not been uploaded, and make sure to upload to the server whenever possible

IR4. The system should alert user about the unauthorized access

IR5. The system should automatically timeout when there is no activities within a certain period

6.3 Privacy Requirements

PR1. The app should not track and save conversation between users and make sure the conversation has been encrypted

PR2. The app should not provide user's personal information to third parties anonymous

6.4 Stability Requirements

SR1. The system should shut down in case of the detection of system instabilities to protect user's data

SR2. The system should have back-up servers in case of the system failure to make sure user's data not getting lost

6.5 Audit Requirements

AR1. The system documents should be easy to read and understand for third parties to provide their service and integrate them into the system.

7 Roadmap

Sept. 11, 2024 - Team formed, and brainstorms the basic idea for project

Sept. 23, 2024 - Initial idea rejected, professor suggest add more feature to make the project more challenging

Sept. 24, 2024 - Our projects getting approved after adding desired feature, and the problems, statements and goal has been identified for this document, but need further clarification

Oct. 11, 2024 - The SRS document has analyzed the scope of our projects and requirements. There are still some part of our project been vague and needs more clarifications, and this will be done in the future documents

Oct. 25, 2024 - The hazard analysis has identified several safety and security requirements, some of which will be implemented in the final application. However, due to time constraints, not all requirements may be addressed within the project timeframe. As the project progresses, we will regularly consult the hazard analysis to evaluate which risks have been mitigated and to identify any areas that still require further attention. The final review will assess the effectiveness of the implemented measures and guide future improvements.

Appendix — 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?

   Our team collaborated effectively by leveraging each member's strengths to brainstorm potential hazards in the card game and online multiplayer context. Breaking down the game into specific modules (e.g., networking, game logic, user interface) allowed us to focus on each area methodically. This organized approach made it easier to identify relevant risks and implement initial mitigation strategies.

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

   The most significant challenge was identifying hazards for a card game, as it differs from other, more complex software. Translating the risks into technical hazards required creativity. We addressed this by discussing examples of game disruptions, such as incorrect shuffling or data loss, and categorizing them under gameplay integrity, data security, and user experience.

3. Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?

   Initially, our team considered risks like desynchronization in multiplayer, loss of data during gameplay, and server downtimes. During the deliverable, we identified additional hazards, such as AI decision failures and user input issues in the UI, as we delved deeper into potential failure modes and examined the unique characteristics of the card game setup. These hazards emerged through role-play scenarios and visualizing unexpected game interactions.

4. Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?

    1. Data Privacy Risk: It is essential to protect user information and prevent unauthorized access, as data breaches can damage reputation and lead to legal issues.

    2. System Performance Risk: Ensuring the game performs well under high load or network instability is crucial for user experience. If the game lags or crashes frequently, players may become frustrated and leave the platform.

Andy:

1. What went well while writing this deliverable?

    From my perspective, one of the positives was gaining a better understanding of hazard analysis through collaboration. The structured breakdown of each risk category was particularly helpful in learning how to think critically about potential issues, even in a seemingly straightforward project like a card game.

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

    Thinking of relevant hazards was definitely a challenge. Since this was my first experience with hazard analysis, I initially struggled to identify risks that seemed realistic or impactful. Working with the team, I found examples of user frustrations (like gameplay disruptions) helpful for grounding abstract risks in real-world scenarios, making it easier to contribute ideas.