Problem Statement and Goals
ProgName

Team #, Team Name
Student 1 name
Student 2 name
Student 3 name
Student 4 name

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

1 Problem Statement

1.1 Problem

1.2 Inputs and Outputs

1.3 Stakeholders

1.4 Environment

2 Goals

2.1 Minimum Viable Product (MVP) Goals

2.1.1 Functional Goals

Table 2: MVP Functional Goals

Goal	**Explanation**	**Reasoning**
Two-Player Core Loop	Support a one-versus-one match with turn-taking, drawing, discarding, and win checks. Includes starter card, discard pile, and reshuffling the stock pile.	Two players are the smallest playable unit; finishing this proves the core loop.
Classic Rules Engine	Implement standard Crazy Eights: match by suit or rank; an "8" is wild and lets the player declare a suit; draw if no valid move exists.	Correct classic rules establish a solid baseline before adding variations.
Dozenal (Base-12) Scoring/Display	Display scores, counters, or thresholds in base-12 notation while keeping classic rules unchanged.	Introduces dozenal in a simple, non-disruptive way that highlights novelty while retaining accessibility.

2.1.2 Non-functional Goals

Table 3: MVP Non-functional Goals

Goal	**Explanation**	**Reasoning**
Move Validation and Feedback	Provide immediate invalid-move feedback, clear suit selection UI after playing an "8," and real-time state indicators.	Reduces errors and learning curve; improves usability.
Testability and Determinism	Support seeded shuffling and provide basic logs or replays for each session.	Enables reproducible unit/integration testing and debugging.
Stability and Performance	Ensure responsive UI $< 200\,ms$, no crashes, no deadlocks, and correct reshuffling.	Reliability is the baseline for acceptance and live demonstration.
Minimal UI	Provide a desktop or web interface showing hand, discard pile, current state, and a dozenal score tracker.	Covers essential interactions while limiting complexity at the MVP stage.

Table 4: MVP Non-functional Goals

Goal	**Explanation**	**Reasoning**
Move Validation and Feedback	Provide immediate invalid-move feedback, clear suit selection UI after playing an "8," and real-time state indicators.	Reduces errors and learning curve; improves usability.
Testability and Determinism	Support seeded shuffling and provide basic logs or replays for each session.	Enables reproducible unit/integration testing and debugging.
Stability and Performance	Ensure responsive UI, no crashes, no deadlocks, and correct reshuffling.	Reliability is the baseline for acceptance and live demonstration.
Minimal UI	Provide a desktop or web interface showing hand, discard pile, current state, and a dozenal score tracker.	Covers essential interactions while limiting complexity at the MVP stage.

2.1.3 Non-functional Goals

3 Stretch Goals

3.1 Functional Stretch Goals

Table 5: Functional Stretch Goals

Goal	**Explanation**	**Reasoning**
3–4 Player Matches	Extend gameplay to three or more players with clear turn rotation and visualization.	Increases replayability; closer to common play.
Online Multiplayer	Create/join rooms and synchronize game state; provide basic reconnection.	Demonstrates system design capability and supports real usage.
AI Opponent	Computer-controlled players with simple heuristics for card choice and suit declaration.	Enables single-player testing and showcases extensibility.
Advanced Dozenal Variants	Optional rule packs (e.g., effects for 12-related cards; base-12 scoring thresholds such as 60_{12}).	Deepens the dozenal theme while keeping the classic mode intact.
Rule Configurator	Toggles for house rules (draw-until-playable, stacking eights, scoring methods in decimal or dozenal).	Shows variability management; supports experimentation and future product-line thinking.

3.2 Non-functional Stretch Goals

Table 6: Non-functional Stretch Goals

Goal	**Explanation**	**Reasoning**
Tutorial and Visual Guidance	First-game tutorial, invalid-move highlights, and play suggestions.	Lowers the learning curve and improves usability.
Save/Replay System	Save completed games and replay event sequences.	Aids debugging, user study, and documentation.
Cross-Platform Packaging	Deploy as a web app or desktop executable (Windows/macOS).	Lowers the barrier for evaluators and external users to try the system.

4 Extras

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?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?