Software Requirements Specification for ProgName: subtitle describing software

Team #25, The Crazy Four

Ruida Chen
Ammar Sharbat
Alvin Qian
Jiaming Li

30.09.2025

Contents

1 Purpose of the Project **vii**
- 1.1 User Business . . . vii
- 1.2 Goals of the Project . . . vii

2 Stakeholders **viii**
- 2.1 Client . . . viii
- 2.2 Customer . . . viii
- 2.3 Other Stakeholders . . . viii
- 2.4 Hands-On Users of the Product . . . ix
- 2.5 Personas . . . ix
- 2.6 Priorities Assigned to Users . . . ix
- 2.7 User Participation . . . x
- 2.8 Maintenance Users and Service Technicians . . . x

3 Mandated Constraints **x**
- 3.1 Solution Constraints . . . x
- 3.2 Implementation Environment of the Current System . . . xi
- 3.3 Partner or Collaborative Applications . . . xi
- 3.4 Off-the-Shelf Software . . . xi
- 3.5 Anticipated Workplace Environment . . . xi
- 3.6 Schedule Constraints . . . xi
- 3.7 Budget Constraints . . . xi
- 3.8 Enterprise Constraints . . . xii

4 Naming Conventions and Terminology **xii**
- 4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project . . . xii

5 Relevant Facts and Assumptions **xiii**
- 5.1 Relevant Facts . . . xiii
- 5.2 Business Rules . . . xiv
- 5.3 Assumptions . . . xiv

6 The Scope of the Work **xiv**
- 6.1 The Current Situation . . . xiv
- 6.2 The Context of the Work . . . xv
- 6.3 Work Partitioning . . . xv

6.4 Specifying a Business Use Case (BUC) xvi

7 Business Data Model and Data Dictionary **xvii**
7.1 Business Data Model xvii
7.2 Data Dictionary xviii

8 The Scope of the Product **xix**
8.1 Product Boundary xix
8.2 Product Use Case Table xx
8.3 Individual Product Use Cases (PUCs) xx

9 Functional Requirements **xxiii**
9.1 Game Manager xxiii
9.2 Score Manager xxiv
9.3 Education Support xxiv
9.4 Login Manager xxiv
9.5 Data Manager xxv

10 Look and Feel Requirements **xxv**
10.1 Appearance Requirements xxv
10.2 Style Requirements xxv

11 Usability and Humanity Requirements **xxvi**
11.1 Ease of Use Requirements xxvi
11.2 Personalization and Internationalization Requirements xxvi
11.3 Learning Requirements xxvi
11.4 Understandability and Politeness Requirements xxvi
11.5 Accessibility Requirements xxvii

12 Performance Requirements **xxvii**
12.1 Speed and Latency Requirements xxvii
12.2 Safety-Critical Requirements xxvii
12.3 Precision or Accuracy Requirements xxvii
12.4 Robustness or Fault-Tolerance Requirements xxvii
12.5 Capacity Requirements xxvii
12.6 Scalability or Extensibility Requirements xxviii
12.7 Longevity Requirements xxviii

13 Operational and Environmental Requirements **xxviii**
13.1 Expected Physical Environment xxviii
13.2 Requirements for Interfacing with Adjacent Systems xxviii
13.3 Productization Requirements xxix
13.4 Release Requirements xxix

14 Maintainability and Support Requirements **xxix**
14.1 Maintenance Requirements xxix
14.2 Supportability Requirements xxx
14.3 Adaptability Requirements xxx

15 Security Requirements **xxxi**
15.1 Access Requirements xxxi
15.2 Integrity Requirements xxxi
15.3 Privacy Requirements xxxi
15.4 Audit Requirements xxxi
15.5 Immunity Requirements xxxii

16 Cultural Requirements **xxxii**
16.1 Cultural Requirements xxxii

17 Compliance Requirements **xxxii**
17.1 Legal Requirements xxxii
17.2 Standards Compliance Requirements xxxiii

18 Open Issues **xxxiii**
18.1 Which Supported bases? xxxiii
18.2 Deck creation for supported bases xxxiii
18.3 Counting mechanics vs. player enjoyment and game understandability xxxiv
18.4 Multiplayer mode implementation decision (ISS-04) xxxiv
18.5 Educational feedback granularity and optional hints (ISS-05) . xxxv
18.6 Multiple-round scoring to 100 and base-consistent thresholds (ISS-06) xxxv
18.7 Card deck cardinality and representation discrepancy (ISS-07) xxxv
18.8 Rank ordering and rank semantics (Ace/1/face cards) (ISS-08) xxxvi
18.9 Sum-target rule ambiguity (10 vs 11 / base-dependence) (ISS-09) xxxvi
18.10 Optional zero-card and deck-extension (ISS-10) xxxvii

18.11Queen rule and optional house rules (ISS-11) xxxvii
18.12Base-display and conversion policy (ISS-12) xxxviii
18.13Mobile / phone responsiveness feasibility (ISS-13) xxxviii
18.14Language support clarification (ISS-14) xxxviii
18.15Capacity target re-evaluation (ISS-15) xxxix
18.16Option A ambiguity (ISS-16) xxxix
18.17Animations and hand ordering requirement (ISS-17) xxxix
18.18Reuse of existing domain and cost feasibility (ISS-18) xl
18.19Missing Fit Criteria and placeholder content (ISS-19) xl
18.20Fit criterion: xli

19 Off-the-Shelf Solutions **xli**
19.1 Ready-Made Products xli
19.2 Reusable Components xli
19.3 Products That Can Be Copied xli

20 New Problems **xlii**
20.1 Effects on the Current Environment xlii
20.2 Effects on the Installed Systems xlii
20.3 Potential User Problems xlii
20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product xlii
20.5 Follow-Up Problems xlii

21 Tasks **xliii**
21.1 Project Planning xliii
21.2 Planning of the Development Phases xliii

22 Migration to the New Product **xliii**

23 Costs **xliii**
23.1 Assumptions xliii
23.2 One-time Costs xliv
23.3 Recurring Costs xliv
23.4 Total and Headroom xliv
23.5 Cost Control Plan xliv

24 User Documentation and Training **xliv**
24.1 User Documentation Requirements xliv
24.2 Training Requirements xlv

25 Waiting Room (Stretch Requirements) **xlv**
25.1 Purpose and scope xlv
25.2 Functional Stretch Requirements xlv
25.3 UI / Look-and-Feel Stretch Requirements (mapped to Section 10) xlvi
25.4 Education / Learning Stretch Requirements (mapped to Section 11) xlvii
25.5 Internationalization and Language Stretch Requirements (mapped to Section 11.2) xlvii
25.6 Performance / Capacity Stretch Requirements (mapped to Section 12) xlvii
25.7 Integration / Deck Production Stretch Requirements xlviii
25.8 Urgency Ranking xlviii
25.9 Crazy10s v1.1 (Rank 1) — Priority stretch items xlviii
25.10Crazy10s v1.2 (Rank 2) — Lower priority stretch items xlviii

26 Ideas for Solution **xlix**
26.1 Architecture Overview xlix
26.2 Implementation Options xlix
26.3 Recommended Path l
26.4 Key Non-functional Techniques l

Revision History

Table 1: Revision History

Date	**Developer(s)**	**Change**
9.29	Jiaming Li	1 - Purpose of the Project
9.30	Ruida Chen	Section 3, 10 ,11, 12
9.30	Jiaming Li	8 - Scope of the Product
9.30	Jiaming Li	9 - Functional Requirements
9.30	Alvin Qian	Section 4,6,7
10.1	Alvin Qian	Section 14,16
10.9	Ruida Chen	Section 20,21,22
10.9	Jiaming Li	Section 17,18,19
10.9	Alvin Qian	Appendix and Reflection
10.12	Ammar Sharbat	Section 1.1, 2, 5, 13, 15, 18
10.13	Ammar Sharbat	Section 18, 19, 25

1 Purpose of the Project

1.1 User Business

The purpose is to study, evaluate and facilitate opportunities for gamification in supporting mathematical learning. Our team wants to determine whether concepts in numerical arithmetic (such as base conversions, or divisibility in Dozenal) can be effectively taught through play.

We aim to accomplish the aformentioned by designing and implementing an educational card game based on the traditional *Crazy 8s* rule set, but with the following differences and additions:

- Switching the wild card from "8" to "10" to improve awareness of number system orders (i.e. 10 in base-10 is a Decimal is 10 in base-12 is a Dozenal).
- Incorporating counting within the game, to improve understanding of number bases and basic arithmetic.
- Integrating the **Dozenal (base-12) number system** using a special Dozenal deck of cards. We also want to add other potential number bases (e.g. Octal, Hexadecimal, etc).

1.2 Goals of the Project

The goals of this project are:

- Educational Integration: Seamlessly incorporate Dozenal concepts (numerical symbols, factorization, arithmetic) into the gameplay, ensuring that players learn by playing without requiring formal prior knowledge.
- Gameplay Design: Deliver a working digital version of this variant of *Crazy 8s* that is intuitive, responsive, and enjoyable, while maintaining the familiar flow of the original game and introducing Dozenal-specific mechanics.
- Accessibility: Create a user-friendly interface that lowers the barrier to learning, accessible for casual users while offering depth for learners who want to explore Dozenal further.

- Testability: Implement robust testing strategies to ensure correctness of game logic, Dozenal calculations, and user interactions.
- Measuring Educational Impact (Stretch Goal): Assessing impact by collecting user data through user engagement, playtesting, progress monitoring and player feedback collection.
- Scalability (Stretch Goal): Explore the potential for extending the system to other educational card or board games, and investigate how different number bases can be taught through similar game mechanics.

2 Stakeholders

2.1 Client

- **Supervisor** Dr. Spencer Smith - primary sponsor of the deliverable; approves acceptance criteria for academic evaluation and EXPO demonstration.
- **Supervisor:** Dr. Paul Rapoport — academic acceptance authority and milestone reviewer.

2.2 Customer

- **Course Instructor / Teaching Staff:** Consumer of the deliverable for grading and course outcomes.
- **End-users (primary customer persona):** Educators and students who may adopt the game as a learning aid; they decide whether the product fits classroom or self-study needs.

2.3 Other Stakeholders

- **Subject Matter Experts:** Dozenal advocates/educators consulted for correctness of dozenal content.
- **Usability / Accessibility Experts:** Provide guidance to meet WCAG guidelines.
- **IT / Hosting Providers:** Maintain hosting infrastructure and CI/CD.

- **Testers / Demonstration Audience:** Provide feedback during PoC and demos.

2.4 Hands-On Users of the Product

- **Casual Learners (novice):** Students or hobbyists with little/no dozenal experience. Low technical skill; require strong onboarding and hints.
- **Educators (journeyman):** Teachers who use the game in lessons; expect configurable sessions and exportable results.
- **Capstone Evaluators / Demo Attendees (master/journeyman):** Technical audience who will assess correctness, stability, and educational value.

2.5 Personas

- **Maya (16, Student)**: Low dozenal familiarity; wants a short tutorial and visual hints to learn while playing.
- **Prof. Singh (45, Educator)**: Wants quick setup for classroom demo, configurable rules, and exportable session results.
- **Jordan (21, Competitive Hobbyist)**: Interested in multiplayer fairness and replay logs; tolerates brief tutorials.

2.6 Priorities Assigned to Users

- **Key users:** Educators and students (primary product purpose is educational adoption).
- **Secondary users:** Demonstration evaluators, hobbyists.
- **Low priority:** Anonymous spectators and infrequent guest users.

2.7 User Participation

- **Educator involvement:** Review of tutorial content and alignment with learning objectives (estimated 2–4 hours over term).
- **Student testing:** Playtest sessions (estimated 3–5 sessions of 20–30 minutes) to validate usability and learning uptake.
- **Developer/Testers:** Ongoing involvement in unit/integration test creation and bug triage.

2.8 Maintenance Users and Service Technicians

- **DevOps / Maintainers:** Responsible for deployment, backups, and monitoring. Require documentation on deployment scripts and DB snapshot procedures.
- **Support Contact (team liaison):** Handles escalations and coordinates fixes with team members.

Fit criterion

Stakeholder list and contact/role assignments exist in the project README and team Kanban; at least one named representative for each stakeholder category is recorded. Evidence: README / Team Roles section and meeting minutes indicating stakeholder reviews.

3 Mandated Constraints

3.1 Solution Constraints

- All source code must be managed using Github, with frequent commits and version control.
- The system must be developed using React for frontend and Node.js for backend.

3.2 Implementation Environment of the Current System

- The backend must operate on Node.js (LTS version, e.g., 18+) with a relational database such as PostgreSQL or MySQL.
- The application must be deployable on modern web browsers (Chrome, Firefox, Edge)

3.3 Partner or Collaborative Applications

- The team must use GitHub for code collaboration and issue tracking.
- Discord are required for real-time communication and coordination.

3.4 Off-the-Shelf Software

- The system must rely primarily on open-source libraries.
- Proprietary software must not be required for end users to run the application.

3.5 Anticipated Workplace Environment

- End users are expected to interact with the application primarily on web browsers.

3.6 Schedule Constraints

- The project must be completed within the academic deadline.
- Progress reports and milestone check-ins with the TA are mandatory.

3.7 Budget Constraints

- Total financial budget is under 500 Canadian Dollars.
- Open source and free tools are primary to be used.

3.8 Enterprise Constraints

- The project must comply with academic integrity and university policies.
- Data collection and storage must follow privacy, security, and ethical guidelines.

4 Naming Conventions and Terminology

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

- **Base-10 (Decimal)**: The standard numerical system using ten digits (0–9), commonly used in arithmetic and everyday calculations.
- **Base-12 (Dozenal)**: A numerical system using twelve digits (0–9, ⅋, Ɛ), where ⅋ represents 10 and Ɛ represents 11 in decimal. Has divisibility advantages (divisors: 2, 3, 4, 6).
- **Base-8 (Octal)**: A numerical system using eight digits (0–7). Each octal digit represents three binary digits (bits). Commonly used in computing and digital systems due to easy conversion with binary.
- **Base-16 (Hexadecimal)**: A numerical system using sixteen digits (0–9, A–F), where A through F represent decimal values 10 to 15. Widely used in programming and computer memory addressing for its compact binary representation.
- **Crazy Eights**: A classic card game where players match cards by suit or rank, with the "8" card acting as a wild card allowing the player to declare a new suit.
- **MVP**: Minimum Viable Product, the initial version of the Crazy Eights software with core functionality, including two-player gameplay, classic rules, and dozenal scoring.
- **Functional Goals**: Features and behaviors the software must implement, such as gameplay mechanics and dozenal score display.

- **Non-functional Goals**: Quality attributes of the software, such as usability, performance, and stability.
- **Stretch Goals**: Optional features or enhancements, such as multiplayer support or advanced dozenal rule variants, to be implemented if time permits.
- **GitHub**: The platform used for version control, issue tracking, and Kanban board management.
- **Discord**: The communication platform for team coordination, quick updates, and voice meetings.
- **Kanban Board**: A project management tool in GitHub Projects, divided into stages (Backlog, In Progress, Review, Done) to track tasks.
- **CI/CD**: Continuous Integration/Continuous Deployment, an automated process for testing and deploying code changes.
- **SRS**: Software Requirements Specification, this document outlining the requirements for the Crazy Eights project.
- **PoC**: Proof of Concept, a prototype demonstrating core gameplay mechanics to validate feasibility.
- **UI**: User Interface, the visual and interactive components of the software, such as the game board and score display.

5 Relevant Facts and Assumptions

5.1 Relevant Facts

- **Prevalence of decimal:** Users are overwhelmingly familiar with base-10; dozenal notation is novel for most.
- **Educational effectiveness:** Game-based learning improves numeracy engagement (backed by pedagogical literature).
- **Resource constraints:** Project budget is capped at 500 CAD and timeline is an academic term.

- **Technology stack:** Planned stack is React (frontend), Node.js (backend), PostgreSQL (data).
- **Browser target:** Modern desktop browsers (Chrome, Firefox, Edge) are the primary runtime environment.

5.2 Business Rules

- **Scoring rule:** Round score equals sum of opponents' remaining card values (displayed in both decimal and dozenal).
- **Account rule:** Registered usernames must be unique; guest mode creates impermanent sessions.
- **Open-source rule:** All deliverables must use appropriately licensed assets; avoid proprietary-only dependencies.

5.3 Assumptions

- **Connectivity:** Players typically have a stable internet connection for multiplayer sessions.
- **Device:** Primary devices are desktop/laptop; mobile support is a stretch-goal.
- **Play familiarity:** Players understand basic card game mechanics (matching suit or rank).
- **Team availability:** Team members will perform peer review before merges as per Development Plan.
- **Third-party components:** Standard open-source libraries (React, Socket.io, PG client) will be available and compatible.

6 The Scope of the Work

6.1 The Current Situation

The current numerical system used around the world is predominantly decimal (base-10). This system, while widely adopted, has limitations in representing fractions cleanly, as its prime factors (2 and 5) result in recurring

decimals for simple ratios 1/3 or 1/6. On the other hand, the dozenal (base-12) system, with divisors 2, 3, 4, and 6, offers a more intuitive and concise fraction representations, historically used in trade and measurement systems (e.g., dozens, hours). However, dozenal is underutilized in education and practice, leaving students, educators, and professionals reliant on decimal despite its inefficiencies for certain calculations. There is no current fun and engaging way to demonstrate the practical advantages of dozenal, such as a card game like Crazy Eights, to promote its adoption.

6.2 The Context of the Work

The project focuses on the software implementation of the Crazy Eights card game that incorporates dozenal (base-12) scoring and display to highlight the benefits of the dozenal system. The context includes:

- **Educational Context**: The project will simplify mathematical understanding for the user by showing dozenal's advantages in a familiar and fun game format.
- **Technical Context**: The software will be developed using a modern tech stack for web applications (JavaScript, TypeScript, Node.js, React, PostgreSQL) and use GitHub CI/CD pipelines for version control and testing.
- **Stakeholder Context**: Key stakeholders include students, educators, mathematicians, computer scientists, and the general public, all of whom could benefit from clearer fraction representations and easier mental arithmetic.

6.3 Work Partitioning

The workload is divided into the following major phases, aligned with the development plan:

1. **Problem Definition and Planning (Weeks 3–4)**: Draft problem statement, development plan, and initial proof of concept (PoC) to establish scope and feasibility.
2. **Requirements and Hazard Analysis (Week 6)**: Develop the SRS and identify potential risks and mitigation strategies.

3. **Verification and Validation Planning (Week 8)**: Define testing strategies to ensure the software meets functional and non-functional requirements.

4. **System Design (Weeks 10–16)**: Create and refine architecture diagrams, decompose the system into modules (frontend, backend, database, API), and document extensibility.

5. **Implementation and Testing (Weeks 11–19)**: Develop the MVP (two-player game with classic rules and dozenal scoring), conduct unit and integration testing, and prepare for demonstrations.

6. **Demonstrations and Refinement (Weeks 19–24)**: Conduct Revision 0 and final demonstrations, incorporating feedback to improve functionality and performance.

7. **Final Documentation and EXPO (Week 26)**: Finalize documentation and present a polished product at the EXPO.

Each task is tracked via GitHub Issues and the Kanban board, with responsibilities assigned to team members based on rotating roles (developer, reviewer, meeting chair, note-taker).

6.4 Specifying a Business Use Case (BUC)

BUC: Play a Game of Crazy Eights with Dozenal Scoring

- **Actors**: Two players, the software system.
- **Trigger**: A player creates a new game session using the UI.
- **Description**: Two players take turns matching cards by suit or rank, with the "10" card acting as a wild card that allows the player to declare a new suit. If no valid move can be made, the player draws from the stock pile. The game ends when one player discards all their cards, and the score is calculated and displayed in dozenal notation.
- **Preconditions**: The user is logged in. The software is running on a web interface, with a functional UI displaying the hand, discard pile, stock pile, and score tracker.

- **Postconditions**: The game concludes with a winner, points are displayed in dozenal and must be counted by the users to see the final score. The user can log Off or start a new game.
- **Basic Flow**:
    1. The system deals cards to both players and initializes the discard pile with a starter card.
    2. Players take turns, selecting a card to play (matching suit or rank) or drawing from the stock pile.
    3. If an "10" is played, the player selects a new suit via the UI.
    4. The system checks moves and gives immediate feedback for invalid moves.
    5. The game keeps going until one player has no cards left, triggering user score counting and calculation challenge in dozenal.
    6. The system displays the final score.
- **Exceptions**:
    - Invalid move attempted: The system highlights the error and prompts the player to select a valid card or draw.
    - Stock pile exhausted: The system reshuffles the discard pile to replenish the stock pile.
- **Assumptions**: Players are familiar with basic card game mechanics and the system supports a stable internet connection for play.

7 Business Data Model and Data Dictionary

7.1 Business Data Model

These are the key entities and relationships involved in the Crazy Eights game with dozenal scoring.

- **Entities**:
    - **Player**: Represents a game participant, holding a hand of cards and a score.

  - **Card**: Represents a single playing card with a suit and rank.
  - **Deck**: A collection of cards, divided into the stock pile and discard pile.
  - **Game Session**: Tracks the state of a single game, including players, current turn, discard pile, and scores.
  - **Score**: Tracks points in dozenal notation, calculated based on game rules.

- **Relationships**:
  - A **Game Session** has 2 **Players** (MVP) or 2–4 **Players** (stretch goal).
  - Each **Player** has a **Hand** (subset of **Cards**).
  - A **Deck** consists of 52 **Cards**, split into **Stock Pile** and **Discard Pile**.
  - A **Game Session** produces a **Score** for each **Player** in dozenal notation.
  - A **Card** played in a **Game Session** affects the **Discard Pile** and may trigger a suit change (if a "10").

7.2 Data Dictionary

- **Player**:
  - **ID**: Unique identifier for a player (integer).
  - **Name**: Display name for the player (string).
  - **Hand**: List of cards held by the player (array of Card objects).
  - **Score**: Player's cumulative score in dozenal notation (string, such as "1Ɛ" for 22 in decimal).

- **Card**:
  - **Suit**: One of four suits (Hearts, Diamonds, Clubs, Spades) (string).
  - **Rank**: Card value (2–10, J, Q, K, A) (string).
  - **IsWild**: Boolean indicating if the card is an "10" (true/false).

- **Deck**:
  - **Stock Pile**: List of cards available for drawing (array of Card objects).
  - **Discard Pile**: List of played cards (array of Card objects).
- **Game Session**:
  - **Session ID**: Unique identifier for the game session (string).
  - **Current Turn**: ID of the player whose turn it is (integer).
  - **Status**: Game state (Active, Completed) (string).
  - **Starter Card**: The first card in the discard pile (Card object).
  - **Last Played Suit**: The active suit after an "10" is played (string).
- **Score**:
  - **Player ID**: Links to the player (integer).
  - **Dozenal Value**: Score in base-12 notation (string, such as "14" for 16 in decimal).
  - **Calculation Method**: Rules for scoring (sum of remaining cards in the opponents hands) (string).

8 The Scope of the Product

8.1 Product Boundary

The product to be developed is a digital card game application based on the traditional *Crazy 8s* rules, modified to integrate the **Dozenal (base-12) number system**. The system boundary includes:

- A game engine that supports core Crazy 8s mechanics (drawing, discarding, turn-taking, winning conditions).
- Adaptations of rules, card values, and scoring to incorporate Dozenal arithmetic and representations.
- A user interface allowing players to interact with the game (play cards, view scores, receive feedback).

- Educational prompts or visual aids to help players understand Dozenal concepts.

External systems not included in the boundary are: general learning platforms, multiplayer servers beyond basic peer-to-peer/local play, and integrations with unrelated educational tools.

8.2 Product Use Case Table

The following table summarizes the primary product use cases (PUCs):

PUC #	**Description**
PUC-1	Player logs into the system or creates a new account to enable personalized features and multiplayer access.
PUC-2	Player views their personal profile, including gameplay history, win/loss record, and achievements.
PUC-3	Player starts a new Crazy 8s game with Dozenal-enabled deck.
PUC-4	Player takes a turn by drawing or discarding a card.
PUC-5	System validates whether the played card is legal (same suit, same Dozenal value, or sum = 12).
PUC-6	Player views scores and progress, displayed in both decimal and Dozenal.
PUC-7	System provides hints or explanations to support Dozenal learning.
PUC-8	Game ends when a player wins; final scores are calculated and displayed.

8.3 Individual Product Use Cases (PUCs)

UC1: Player Login

1. Player opens the game application and selects "Login" or "Create Account".

2. The system prompts the player to enter credentials (username and password) or choose guest mode.

3. The system validates login information against the stored database.

4. If credentials are valid, the player is granted access to personalized data (game progress, scores, and settings).

5. If the player selects guest mode, a temporary profile is generated for the session.

6. The system displays the main menu or lobby after login.

UC2: See Player Profile

1. The player selects "View Profile" from the main menu.

2. The system retrieves player data (username, past game records, win/loss ratio, and achievements) from the database.

3. The system displays profile information on screen.

4. If the player is logged in as a guest, the system displays a message such as "Profile unavailable for guest mode".

5. The player may choose to return to the main menu or edit available settings (if logged in).

UC3: Start a New Game

1. Player opens the application and selects "New Game".

2. The system initializes a Dozenal-enabled deck (0–B, 10).

3. The system shuffles the deck.

4. The system deals cards to each player.

5. The game state is displayed on the interface.

UC4: Take a Turn

1. The system indicates that it is the player's turn.

2. The player chooses either to play a card or to draw from the deck.

3. If the player chooses a card, the system goes through **UC5** to check its validity against the discard pile.

4. If valid, the card is placed onto the discard pile.

5. If invalid, the system notifies the player and the card remains in the hand.

6. If the player chooses to draw, the system gives one card from the deck to the player.

UC5: Validate Move

1. The player selects a card to play.
2. The system retrieves the top card from the discard pile.
3. The system checks if the move is legal under Dozenal rules:
   (a) same suit, or
   (b) same Dozenal value, or
   (c) sum of the two values equals 12 (base-12).
4. If the move is legal, the system accepts the card and updates the discard pile.
5. If not, the system rejects the move and notifies the player.

UC6: View Scores

1. A round ends or the player requests to view scores.
2. The system calculates points for each player.
3. The system converts the scores into both decimal and Dozenal.
4. The system displays the results on screen.

UC7: Provide Hints

1. The player hovers over or selects a card.
2. The system analyzes the current game state.
3. The system provides a hint (e.g., "This card is valid because its sum with the top card equals 12 (base-12).").
4. The hint is displayed as text or a visual highlight.

UC8: End Game

1. A player discards their last card.

2. The system checks if the game-ending condition is satisfied.
3. If satisfied, the system declares the winner.
4. The system calculates final scores in both decimal and Dozenal.
5. The results are displayed on the final game screen.

9 Functional Requirements

9.1 Game Manager

1. **Start new game:** Game manager shall allow the player to start a new Crazy 8s game with a Dozenal-enabled deck. (FR-1)
   Rationale: Players need a way to initialize the game state; starting a new game is the foundation for all other gameplay functions.

2. **Turn management:** Game manager shall manage player turns, ensuring that each player either discards a valid card or draws a card. (FR-2)
   Rationale: Turn-taking enforces fairness and ensures game flow consistency.

3. **Rule validation:** Game manager shall validate that each played card is legal under Dozenal rules. A valid move is defined as either: (a) same suit as the previous card, (b) same Dozenal value as the previous card, or (c) the sum of the two card values equals 12 in Dozenal. (FR-3)
   Rationale: Prevents illegal moves, guarantees consistency, and introduces the core educational mechanic of Dozenal arithmetic.

4. **Special cards:** Game manager shall implement special card effects (e.g., 10s are wild) while supporting extensions with Dozenal-specific effects. (FR-4)
   Rationale: Special cards increase engagement and add flexibility in teaching Dozenal-based rules.

5. **End of game:** Game manager shall determine when the game ends (e.g., when a player runs out of cards) and declare the winner. (FR-5)
   Rationale: A clear end condition is required for meaningful gameplay and reinforcement of learning objectives.

9.2 Score Manager

1. **Calculate score:** Score manager shall calculate points for each round in both decimal and Dozenal. (FR-6)
   Rationale: Displaying scores in both systems reinforces learning by encouraging comparison between familiar decimal and new Dozenal formats.

2. **Display score:** Score manager shall display both decimal and Dozenal results on the user interface. (FR-7)
   Rationale: Visual feedback supports player understanding and helps users internalize Dozenal representations.

9.3 Education Support

1. **Hints:** System shall provide hints or explanations when a player performs an action involving Dozenal arithmetic. (FR-8)
   Rationale: On-demand guidance lowers the learning curve and supports players with varying levels of familiarity.

2. **Highlight valid moves:** System shall visually highlight valid moves based on Dozenal rules. (FR-9)
   Rationale: Reduces frustration, ensures players stay engaged, and reinforces Dozenal rules through visual learning.

9.4 Login Manager

1. **Account creation:** The system shall allow players to create a new account with a unique username and password. (FR-10)

2. **Login/Logout:** The system shall allow players to log in and log out at any time without losing progress. (FR-11)

3. **Guest mode:** The system shall allow players to start a temporary session without login. (FR-12)

4. **Credential validation:** The system shall validate player credentials against stored records before granting access. (FR-13)

5. **Rationale:** Ensures users can have persistent profiles for tracking progress and enables multiplayer authentication.

9.5 Data Manager

1. **Data storage:** The system shall securely store user data, including usernames, game history, and Dozenal scores. (FR-14)
2. **Data retrieval:** The system shall allow retrieval of stored user data when logging in or viewing profiles. (FR-15)
3. **Data update:** The system shall update stored data after each game or when profile information changes. (FR-16)
4. **Data deletion:** The system shall allow players to delete their data permanently upon request. (FR-17)
5. **Rationale:** Supports persistent user experience, analytics, and compliance with data privacy expectations.

10 Look and Feel Requirements

10.1 Appearance Requirements

- **Layout**: The user interface must have a clean and minimalistic layout, ensuring readability and ease of navigation.
- **Responsiveness**: The system must be responsive, adapting automatically to different screen sizes.
- **Visual Hierarchy**: Key elements such as active player indicators, playable cards, and turn timers must be visually distinguishable using contrast, color, or animation.
- **Color Palette**: The color scheme should promote clarity and engagement.

10.2 Style Requirements

- **Consistency**: The application must follow a unified design language, maintaining consistent typography, iconography, and button styles across all screens.

- **Animations**: Smooth 2-D animations should be implemented for card movements, dealing, and discarding actions. These animations should be realistic but not distracting, maintaining short transition times. When a wild card is played, a distinct animation and color shift should visually emphasize its effect while maintaining overall style consistency.

- **In-Game Hints**: Hints should appear contextually (e.g., glowing borders, pulsing icons) rather than as intrusive pop-ups. They should guide the player without interrupting the flow of the game.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

- **Simplicity**: The interface must minimize the number of steps needed to perform core tasks.

- Data collection and storage must follow privacy, security, and ethical guidelines.

11.2 Personalization and Internationalization Requirements

- **Language Options**: The system must support at least English and one additional language for international users.

11.3 Learning Requirements

- **Onboarding**A short interactive tutorial must explain the rules of the game and how counting in different bases works.

11.4 Understandability and Politeness Requirements

- **Friendly Wording**: Prompts like "Invalid Move" must be displayed as polite guidance.

- **Clear Instructions**: Rules and base-conversion explanations must be phrased in simple and non-technical terms.

11.5 Accessibility Requirements

- **Keyboard Support**: All main actions must be accessible via keyboard shortcuts

12 Performance Requirements

12.1 Speed and Latency Requirements

- **Low latency**: Game actions must update across all players' screens within 300 ms.
- **Fast loading**: The game lobby and first match must load within 5 seconds on a stable internet connection.

12.2 Safety-Critical Requirements

- **Cheat Prevention**: The system must prevent unauthorized manipulation (e.g., directly altering game state through client-side tools).
- **Data Integrity**: No game state (e.g. deck composition, player hand) should be lost or corrupted due to refresh or reconnect.

12.3 Precision or Accuracy Requirements

- **Card Rules**: Card-matching and counting rules must be enforced with 100% accuracy according to the chosen numeral base.

12.4 Robustness or Fault-Tolerance Requirements

- **Reconnection**: If a player disconnects, they must be able to rejoin within 30 seconds without losing progress.

12.5 Capacity Requirements

- **Concurrent Players**: The system must support at least 200 concurrent users during testing.

12.6 Scalability or Extensibility Requirements

- **Game Modes**: The system must allow for adding new rule variations (e.g., different numeral bases or wild card effects) with minimal code changes.
- **Server Scaling**: The backend must be deployable in a scalable environment (e.g., Docker, cloud hosting) to handle larger user bases.

12.7 Longevity Requirements

- **Maintainability**: The codebase must be modular and documented, so future developers can update rules easily.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

- The product will be used on desktop/laptop computers in classrooms, homes, and demo booths under normal indoor lighting and noise conditions.
- Typical input devices: keyboard and mouse; mouse will be primary input device; keyboard interaction considered a stretch goal.
- No special physical environment (temperature, humidity, dust) requirements are expected.

13.2 Requirements for Interfacing with Adjacent Systems

- **Authentication API:** Integrate with a simple backend-auth service (REST endpoints) for account creation and login.
- **Telemetry / Analytics (optional):** Provide an HTTP endpoint to export anonymized session metrics for research (GDPR-compliant).

- **Hosting / CI:** Integration with GitHub Actions for tests and deploy; target hosting platform must accept standard Docker containers or static web app hosting.

13.3 Productization Requirements

- Distribution as a web application (hosted) and a developer-local deployment (Docker-compose) for evaluation.
- Installation for in-person demos should require at most three documented steps (clone, install, run).

13.4 Release Requirements

- Release cadence: a demonstration-ready milestone (PoC) and a final-revision release for EXPO.
- Releases must be tagged in GitHub with release notes and a reproducible build artifact (Docker image or static bundle).
- **Fit Criterion:** Acceptance when: (a) core flows run on supported browsers; (b) documented REST endpoints respond to auth and telemetry; (c) dev deployment runs end-to-end using provided Docker-compose with README steps validated by at least one tester.

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

- **Code Maintainability**: The codebase must be modular, with clear separation of concerns to facilitate updates and debugging. Inline comments for non-obvious logic and updated README/module documentation are required, as specified in the development plan.
- **Version Control**: All changes must be tracked via GitHub, with descriptive commit messages and pull requests requiring at least one peer review to ensure maintainability.

- **Extensibility**: The system must support adding new features without requiring significant refactoring. This is achieved by using a modular architecture and Object oriented Design principles.
- **Documentation Updates**: Any change in behavior, interfaces, or setup instructions must be reflected in the documentation to ensure future developers can maintain the system. Local deployment instructions documented in the README to simplify setup for developers and maintainers.

14.2 Supportability Requirements

- **Error Reporting**: The system must provide clear, user-friendly feedback for invalid moves. Logs or replays of game sessions must be available for debugging, as specified in the non-functional goals.
- **User Support**: A tutorial or visual guidance (stretch goal) must be provided to assist new players in understanding gameplay and dozenal scoring. This reduces the need for extensive manual support.
- **Cross-Platform Support**: The software must run on different web environments (Windows, macOS, or browsers).

14.3 Adaptability Requirements

- **Rule Flexibility**: The system must support toggling house rules (e.g., draw-until-playable, stacking eights) via a rule configurator (stretch goal), allowing adaptation to different play styles without code changes.
- **Scalability for Players**: The architecture must accommodate extending from two-player (MVP) to 3–4 player games (stretch goal) by modifying session management and turn logic.
- **Numeric System Extensibility**: The scoring system must allow switching between dozenal and decimal displays, ensuring adaptability for users unfamiliar with base-12.

15 Security Requirements

15.1 Access Requirements

- Role model: **Guest**, **Registered Player**, and **Administrator/Team Maintainer**. Define permitted APIs for each role.
- Registered players may access persistent profile data and match history for their account only.
- Administrators may access system logs and anonymized telemetry; no arbitrary user personal data access without consent.

15.2 Integrity Requirements

- Server-authoritative game state: all moves validated server-side for any online session to prevent client tampering.
- Persistent data (user profiles, match results) must be stored atomically; partial writes are unacceptable.
- Backups: periodic DB snapshots for recovery as defined in deployment docs.

15.3 Privacy Requirements

- Collect minimal PII: username and optional display name; e-mail only if explicitly provided for account recovery.
- Users must be informed (privacy notice) before data collection; opt-out option for telemetry must be available.

15.4 Audit Requirements

- Maintain tamper-evident logs for administrative actions and system errors for at least 30 days.
- For online sessions, optionally retain replay logs (seed and sequence) to permit investigation of disputes; stored replays must be anonymized when used outside the originating account.

15.5 Immunity Requirements

- The product shall be resilient to basic web threats (e.g., injection, CSRF) by following standard secure-coding practices and OWASP guidance.
- External penetration testing is a stretch goal; at minimum, use linting and dependency-audit tools in CI.

16 Cultural Requirements

16.1 Cultural Requirements

- **Numeric System Accessibility**: The software must present dozenal (base-12) in an intuitive, non-disruptive way. This includes clear UI elements for dozenal scores (using Ɛ and Ɛ for 10 and 11) and optional tutorials (stretch goal) to explain dozenal notation to users unfamiliar with it.
- **Inclusivity**: The system must avoid cultural biases in its design, ensuring that gameplay and terminology (suits, ranks) are universally recognizable across cultures familiar with standard playing cards. No culturally specific references or imagery should be used in the UI to maintain broad accessibility.
- **Educational Alignment**: The software must align with educational goals by demonstrating the practical benefits of dozenal in a game context, making it appealing to students and educators. This supports the cultural shift toward exploring alternative numeric systems, as advocated by stakeholders like dozenal enthusiasts.

17 Compliance Requirements

17.1 Legal Requirements

The game must comply with intellectual property laws and digital content distribution regulations. All assets (images, fonts, and sounds) used in the game must be open-source, royalty-free, or properly licensed.

17.2 Standards Compliance Requirements

The software shall follow standard software development practices such as version control (Git), documentation standards (LaTeX-based SRS), and accessibility guidelines (WCAG 2.1 Level AA). The codebase should be compliant with modern C++/Python style conventions (PEP8 or equivalent).

18 Open Issues

18.1 Which Supported bases?

- **Description:** Decide whether to support only Decimal and Dozenal, or to include multiple other bases (Base-8: Octal, Base-16: Hexadecimal, Base-6: Senary, Base-9: Nonary).
- **Stakeholders:** Educators; Students.
- **Action:** 1. Fix inconsistencies within SRS documented about which number bases to implement. Discussion required.
- **Status:** Open.
- **Target resolution:** Educational stakeholders will give guidance on curriculum fit and scope.

18.2 Deck creation for supported bases

- **Description:** Create physical or print-ready decks for additional number systems. (Decimal and Dozenal decks exist: https://www.thegamecrafter.com/games/k6t.)
- **Stakeholder:** Dr. Paul Rapoport
- **Action:** Design and create decks for Octal and Hexadecimal; estimate cost and lead time.
- **Resolution:** Dr. Rapoport will inquire and report back to the team with feasibility info.
- **Status:** Pending response.

18.3 Counting mechanics vs. player enjoyment and game understandability

- **Description:** How to add meaningful counting mechanics (teaching) while keeping game fun and understandable? Current ideas include:
  - For non-winners, sum remaining cards at end of round to reduce point tally.
  - Trigger an arithmetic task each time a wild card (10) is played.
  - Other lightweight counting mini-games embedded in rounds.
- **Stakeholders:** Dr. Paul Rapoport, UI/UX Designers, Educators, Students.
- **Action:** Continue ideation and small-scale playtests to evaluate engagement.
- **Status:** Ongoing.

18.4 Multiplayer mode implementation decision (ISS-04)

- **Description:** Decide whether online multiplayer (server-authoritative WebSocket rooms, reconnection, scaling) is in-scope for MVP or deferred to a post-MVP release. Identify minimal online feature set if approved.
- **Stakeholders:** Course Instructor; Dr. Paul Rapoport; Developers; QA.
- **Action:** Convene a design decision meeting; produce an options analysis (local hot-seat vs server-authoritative) with estimated engineering effort and acceptance criteria.
- **Status:** Open.
- **Target resolution:** Decision recorded in GitHub Issue and linked to an implementation milestone or deferral note.

18.5 Educational feedback granularity and optional hints (ISS-05)

- **Description:** Determine the default level of educational feedback (none, minimal, optional per-game setting). Rapoport recommended hints be optional to avoid over-assistance.
- **Stakeholders:** Dr. Paul Rapoport; Educators; Students; UX Designers.
- **Action:** Draft an "Hints/pedagogy" matrix for modes (Off / Minimal / Full), map to UI options, and prepare a small A/B playtest.
- **Status:** Open.
- **Target resolution:** Hints implemented as opt-in per-player or per-session; documented in Settings and Instructor Notes.

18.6 Multiple-round scoring to 100 and base-consistent thresholds (ISS-06)

- **Description:** Clarify multi-round scoring rules (target score e.g., 100) and whether thresholds are expressed and accumulated in the active base (decimal or dozenal).
- **Stakeholders:** Dr. Paul Rapoport; Course Instructor.
- **Action:** Specify scoring rules and sample round calculations in both decimal and dozenal; include acceptance tests.
- **Status:** Open.
- **Target resolution:** SRS updated with explicit scoring calculation examples and unit tests in repo.

18.7 Card deck cardinality and representation discrepancy (ISS-07)

- **Description:** Resolve inconsistency: SRS currently states $\ddot{5}2$ cards$\ddot{(}$decimal) and elsewhere implies a different card count for dozenal (e.g., 64). De-

cide canonical deck construction per base and document mapping of ranks (0..?).

- **Stakeholders:** Developers; Dr. Paul Rapoport; Game Designers.
- **Action:** Produce a deck specification per base (ranks, face-cards, zero inclusion) and update Data Dictionary and Business Data Model.
- **Status:** Open.
- **Target resolution:** Deck spec reviewed and accepted; design files / test vectors added to repo.

18.8 Rank ordering and rank semantics (Ace/1/face cards) (ISS-08)

- **Description:** Formalize rank ordering (is Ace low or high?) and whether face-cards participate in arithmetic rules; clarify how face cards map to numeric values when arithmetic is required.
- **Stakeholders:** Game Designers; Dr. Paul Rapoport.
- **Action:** Document rank-to-value mapping for each base and add example scenarios demonstrating validity checks (UC5).
- **Status:** Open.
- **Target resolution:** Data Dictionary updated to include numeric mapping for face cards and Ace.

18.9 Sum-target rule ambiguity (10 vs 11 / base-dependence) (ISS-09)

- **Description:** Rapoport suggests using sum-to-11 (in base) rather than sum-to-10 to avoid duplication and to make face cards usable. Determine canonical "sum-target" and implications for move validity and pacing.
- **Stakeholders:** Dr. Paul Rapoport; Game Designers; Educators.

- **Action:** Evaluate both options via short gameplay simulations; choose target (10 or 11) and document trade-offs and acceptance tests.
- **Status:** Open.
- **Target resolution:** Update FR-3 and UC5 to reflect chosen rule; add test cases.

18.10 Optional zero-card and deck-extension (ISS-10)

- **Description:** Evaluate whether to include a zero card to support arithmetic rules involving face-card pairings (as suggested in review).
- **Stakeholders:** Dr. Paul Rapoport; Designers.
- **Action:** Prototype a deck variant and test for balance and educational clarity.
- **Status:** Open.
- **Target resolution:** Decision recorded; deck files added or rejected with rationale.

18.11 Queen rule and optional house rules (ISS-11)

- **Description:** Include Queen-rule and other house-rule toggles (e.g., 2-rule) as optional settings in the Rule Configurator.
- **Stakeholders:** Players; Educators.
- **Action:** Add to Stretch Goals / Rule Configurator and document expected behaviour and UI toggle locations.
- **Status:** Open.
- **Target resolution:** Rule configurator UI/FRs added in a future sprint.

18.12 Base-display and conversion policy (ISS-12)

- **Description:** Clarify whether UI will show both decimal and dozenal simultaneously or provide optional conversion toggles; Rapoport recommended making conversion optional.
- **Stakeholders:** UX Designers; Educators.
- **Action:** Add an options matrix in SRS: (A) Dozenal-only UI, (B) Dual-display (Dozenal+Decimal), (C) Optional conversion toggle. Select default.
- **Status:** Open.
- **Target resolution:** SRS updated and acceptance tests added verifying display modes.

18.13 Mobile / phone responsiveness feasibility (ISS-13)

- **Description:** Rapoport doubts phone support is feasible; determine whether mobile support is stretch or omitted.
- **Stakeholders:** Developers; Dr. Paul Rapoport.
- **Action:** Classify "mobile" as stretch-goal or out-of-scope for MVP and document reasoning.
- **Status:** Open.
- **Target resolution:** Decision and update to §10.responsiveness and Release Notes.

18.14 Language support clarification (ISS-14)

- **Description:** Rapoport advised English-only for now; confirm language scope and mark translations as stretch.
- **Stakeholders:** Dr. Paul Rapoport; Course Instructor.
- **Action:** Set default language list (English) and mark other languages as stretch goals in SRS.

- **Status:** Open.
- **Target resolution:** SRS updated; internationalization backlog item created if required.

18.15 Capacity target re-evaluation (ISS-15)

- **Description:** Reassess the 200 concurrent users testing target; confirm whether this is practical within the budget and hosting constraints.
- **Stakeholders:** Team Lead; DevOps; Dr. Paul Rapoport.
- **Action:** Produce an estimate of server costs and a scaled test plan; consider lower initial target (e.g., 20–50) for MVP.
- **Status:** Open.
- **Target resolution:** Update Capacity Requirements and CI load plan.

18.16 Option A ambiguity (ISS-16)

- **Description:** Clarify "Option A: Local-first 2P hot-seat" — whether it implies two players on same device or AI opponent; Rapoport flagged confusion.
- **Stakeholders:** Team; Course Instructor.
- **Action:** Update Implementation Options with precise definitions and demo acceptance criteria for Option A and Option B.
- **Status:** Open.
- **Target resolution:** Implementation Options section clarified and accepted.

18.17 Animations and hand ordering requirement (ISS-17)

- **Description:** Define card ordering animation spec (how cards are sorted and animated) and ensure it is included in Look-and-Feel Requirements.

- **Stakeholders:** UI/UX; Frontend Developers.
- **Action:** Draft a brief spec and add to §10 Style Requirements; include accessibility considerations for motion.
- **Status:** Open.
- **Target resolution:** Animation spec committed and demoed in PoC.

18.18 Reuse of existing domain and cost feasibility (ISS-18)

- **Description:** Investigate possibility of reusing existing dozenal-domain names or prior projects (Rapoport note) to reduce brand/setup costs.
- **Stakeholders:** Team Liaison; Dr. Paul Rapoport.
- **Action:** Research available domains and prior dozenal projects; report cost/availability and licensing constraints.
- **Status:** Open.
- **Target resolution:** Domain decision and purchase plan or deferral documented.

18.19 Missing Fit Criteria and placeholder content (ISS-19)

- **Description:** Several sections still contain `\lips` placeholders and lack explicit Fit Criteria (measurable acceptance criteria).
- **Stakeholders:** All authors; Document Owner.
- **Action:** Replace placeholders with substantive requirements and add Fit Criteria for each section before submission; track in issue tracker.
- **Status:** Open.
- **Target resolution:** All placeholders replaced and Fit Criteria added; document review pass.

18.20 Fit criterion:

- Each open item has a GitHub Issue.
- Each Issue lists owner, stakeholders explicit next-action, and a tentative resolution date or review milestone.
- Closure requires documented decision, implementation plan, or formal deferral with rationale.

19 Off-the-Shelf Solutions

19.1 Ready-Made Products

Existing Crazy 8s card games (e.g., UNO or digital clones) offer partial gameplay models, but none integrate an educational Dozenal arithmetic component. No direct off-the-shelf solution currently fulfills both entertainment and mathematical learning objectives.

19.2 Reusable Components

- Card rendering engines from open-source projects such as *Pygame* or *Godot* assets.
- Math libraries for number-base conversion (decimal $\leftrightarrow$ Dozenal).
- UI component libraries for game menus, card animations, and scoreboards.

19.3 Products That Can Be Copied

Some structural aspects (turn management, shuffling algorithms) can be adapted from existing open-source card game repositories, provided licensing terms (e.g., MIT, GPL) are respected. However, educational Dozenal mechanics must be developed in-house to align with project-specific learning goals.

20 New Problems

20.1 Effects on the Current Environment

- The system is a newly devleoped web-based game and does not replace or interfere with any existing enviroment, minimal impact is expected aside from standard hosting and network usage.

20.2 Effects on the Installed Systems

- No existing software systems are modified or replaced by this product. The game operates independently on standard browsers and web servers.

20.3 Potential User Problems

- Users may experience confusion if base-conversion rules are misunderstood. These issues will be mitigated through clear feedback, tutorials, and consistent UI design.

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

- The main limitation is dependency on stable internet connectivity. Poor network conditions may degrade real-time synchronization or cause delayed responses.

20.5 Follow-Up Problems

- As the system evolves, maintaining compatibility with browser updates and third-party libraries (e.g., React, Socket.io) may introduce future maintenance challenges.

21 Tasks

21.1 Project Planning

- The project life cycle will follow the deliverable outline for SFWRENG 4G06: Capstone. By April the project must be completed in its entirety.

21.2 Planning of the Development Phases

- Requirements and Design Phase - Gather functional and non-functional requirements, identifying key components.
- Prototyping Phase - Develop an early functional prototype focusing on core interactions and user interface flow.
- MVP Phase - Implement the main gameplay logic, user authentication, and basic scoring system as our MVP.
- Finalize Phase - Polish the user interface, improve performance, deploy the final version to the hosting environment and prepare presentation.

22 Migration to the New Product

Not applicable.

23 Costs

23.1 Assumptions

- Academic, non-commercial deployment targeting a small user group during the term.
- Free tiers are acceptable for hosting and CI where reliability is sufficient for demos.
- Budget ceiling: 500 CAD (see Section “Budget Constraints”).

23.2 One-time Costs

23.3 Recurring Costs

- App hosting utilize free tier where possible
- DB hosting and Free tier Postgres, small instance
- Domain name renweal cost could be recurring if we plan on hosting it longer term

23.4 Total and Headroom

Estimated total spend for the term: **100 CAD**. This stays within the 500 CAD cap and leaves headroom for risk mitigation or stretch features.

23.5 Cost Control Plan

- Prefer free tiers for hosting, CI, and CDN. Upgrade only if load or reliability requires it.
- Reuse open-source UI kits and icon packs with permissive licenses.
- Cap incentives to pre-approved number of sessions.

24 User Documentation and Training

24.1 User Documentation Requirements

- **In-app help**: Contextual tooltips for controls, hover help for Dozenal terms, and an always-available "How to play" panel.
- **Quick Start**: Setup, start a match, play a turn, view scores. Written for first-time users.
- **Rules and Dozenal guide**: Clear rules of Crazy Eights plus a simple Dozenal primer with examples for A and B, and conversions between base 10 and base 12.
- **FAQ**: Troubleshooting (cannot play a card, reconnect flow, what counts as a valid move).

- **In-app help**: Contextual tooltips for controls, hover help for Dozenal terms, and an always-available "How to play" panel.
- **Quick Start**: Setup, start a match, play a turn, view scores. Written for first-time users.
- **Rules and Dozenal guide**: Clear rules of Crazy Eights plus a simple Dozenal primer with examples for A and B, and conversions between base 10 and base 12.
- **FAQ**: Troubleshooting (cannot play a card, reconnect flow, what counts as a valid move).

24.2 Training Requirements

- **Onboarding tutorial**: 3 to 5 steps covering play a card, draw a card, wild eight, and reading Dozenal scores. Skippable and repeatable.
- **Practice mode**: Solo mode against a simple bot to learn pace and rules without pressure.

25 Waiting Room (Stretch Requirements)

25.1 Purpose and scope

This section lists unimplemented features (stretch goals) relating to the Crazy 10s game project. Each item below is intentionally classified as a post-MVP requirement; these will be considered for Crazy10s v1.1 (Rank 1) or v1.2 (Rank 2) per the urgency ranking in the very bottom of this section.

25.2 Functional Stretch Requirements

1. **4-player match support (Stretch-L2)**
   Description: Extend session logic to allow 3–4 players, clear turn rotation, and UI to display additional hands.
   Form (FR-style): Add FR: "Multi-player session management must support up to 4 players with deterministic turn order and correct scoring."
   Motivation: Increased replayability and social gameplay.

Considerations: Complexity in balancing learning vs. play-time; may increase server load.
Fit Criterion: A 4-player match can be started, played to completion, and score computed correctly.

2. **Invite and classroom session tools (Stretch-L1)**
Description: Teachers create class sessions, bulk-invite students, and view join status.
Form: Teacher dashboard extension FR.
Motivation: Simplify classroom management and improve adoption.
Fit Criterion: Teacher creates a session and sees join statuses in real time in the lobby.

25.3 UI / Look-and-Feel Stretch Requirements (mapped to Section 10)

1. **In-game optional hints (Stretch-U1)**
Appearance / Style Form: Hints shall be available as an opt-in toggle in the lobby settings (per-player or per-session). Visual hint styles: glowing border, subtle tooltip.
Motivation: Supports novice players during early learning but avoid over-assistance for experienced users.
Considerations: Hints must be accessible (ARIA) and optional; do not clutter the interface by default.
Fit Criterion: A player toggles hints in the lobby and receives contextual, non-blocking hints during gameplay when enabled.

2. **Detailed onboarding preview in lobby (Stretch-U1)**
Appearance Form: Lobby displays a short interactive primer about dozenal or the active base; includes a "Quick Try" button to demo the counting mechanic.
Motivation: Lowers barrier to entry for classroom audiences.
Fit Criterion: Primer launches from lobby and completes without entering a full match.

25.4 Education / Learning Stretch Requirements (mapped to Section 11)

1. **Advanced arithmetic mini-challenges (Stretch-E1)**
   Description / Form: When enabled in lobby, certain wild-card plays trigger short arithmetic mini-challenges (e.g., compute sum in active base). These are optional and configurable.
   Motivation: Deepen learning for engaged classes.
   Considerations: Must be skippable and have accessibility support; avoid disrupting game flow.
   Fit Criterion: When enabled, mini-challenges appear at correct trigger points and accept valid answers in the active base.

25.5 Internationalization and Language Stretch Requirements (mapped to Section 11.2)

1. **Optional language support (Stretch-I2)**
   Description: Add support for one additional language (e.g., French) as a stretch feature.
   Form: UI strings externalized; language selectable in lobby settings.
   Motivation: Broaden potential evaluators/users beyond English.
   Considerations: Translation quality is critical for pedagogical text; may require SME review.
   Fit Criterion: UI renders in selected language and sample dozenal primer is translated without truncation.

25.6 Performance / Capacity Stretch Requirements (mapped to Section 12)

1. **Higher concurrent capacity (Stretch-P2)**
   Description: Scale hosting to support ¿200 concurrent users for studies or demonstrations.
   Form: Performance requirement with load-test acceptance criteria.
   Motivation: Useful for large demo sessions or class-wide studies.
   Considerations: Higher cost and CI load-testing required.
   Fit Criterion: Simulated load test demonstrates system remains responsive per p95 latency target at target concurrency.

25.7 Integration / Deck Production Stretch Requirements

1. **Physical deck production workflow (Stretch-D2)**
   Description: Provide printable/print-on-demand assets and ordering instructions to create physical dozenal, octal, or hex decks.
   Form: Documentation and asset package in repo.
   Motivation: Supports tactile classroom use and outreach.
   Considerations: Licensing and cost to be estimated.

2. **Fit Criterion:** Provided print assets produce a proof print and match on-screen representations.

25.8 Urgency Ranking

Each unimplemented stretch requirement above is assigned an urgency rank:

- **Rank 1 (v1.1)**: High-priority stretch items likely to be scheduled immediately after MVP if resources permit.
- **Rank 2 (v1.2)**: Lower-priority stretch items for later releases or exploratory work.

25.9 Crazy10s v1.1 (Rank 1) — Priority stretch items

- Invite & classroom session tools (Stretch-L1)
- In-game optional hints (Stretch-U1)
- Advanced arithmetic mini-challenges (Stretch-E1)
- Physical deck production workflow (Stretch-D2) — planning/asset creation phase

25.10 Crazy10s v1.2 (Rank 2) — Lower priority stretch items

- 4-player match support (Stretch-L2)

- Detailed onboarding preview in lobby (additional polish) (Stretch-U1)
- Optional language support (Stretch-I2)
- Higher concurrent capacity (Stretch-P2)

26 Ideas for Solution

26.1 Architecture Overview

- **Frontend**: React with TypeScript. State managed via a small store (Zustand or Redux Toolkit). WebSocket client for live turns. Component library and keyboard-first interactions.
- **Backend**: Node.js with TypeScript. HTTP API for auth and lobby. WebSocket for game sessions and events. Server-authoritative game state.
- **Data**: Postgres for user profiles, match history, and telemetry. In-memory room state with periodic snapshots to DB.
- **Game engine**: Pure TypeScript module with deterministic rules, validators, scoring, and base conversion utilities. No UI concerns.
- **CI/CD**: GitHub Actions for lint, tests, type-checks, and deploy to a free-tier host.

26.2 Implementation Options

Option	Summary, Pros, Cons
A. Local-first 2P hot-seat	*Summary*: Single browser, two players, no backend. *Pros*: Fast MVP, minimal infra. *Cons*: No remote play, no persistence.
B. Server-authoritative WebSocket	*Summary*: Lobby, rooms, turns via WS. *Pros*: Fair play, replay logs, reconnect support. *Cons*: More code and testing.

26.3 Recommended Path

1. Build the game engine as a pure library with full tests.
2. Ship Option A for fast validation and UI polish.
3. Upgrade to Option B for sessions, persistence, and reconnection.

26.4 Key Non-functional Techniques

- **Fairness**: Server-side validation for all moves. Shuffle with seeded RNG and record seed for audits.
- **Resilience**: Rejoin token and state rehydrate on reconnect.
- **Accessibility**: Semantic HTML, focus order, ARIA labels, and keyboard shortcuts for all core actions.

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?

   - One thing that went well while writing the SRS was our team's ability to clearly define functional and non-functional requirements based on our project scope. We divided the document sections efficiently and maintained consistent formatting and terminology. Our discussions on user needs and system goals also helped refine the main features early, making later sections like use cases and functional requirements much easier to complete.

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

   (a) Discussing and validating requirements with the Professor during the middle of midterm season.
   We resolved this by trusting the requirements and feedback from our previous meetings with the Professor, along with course lecture notes, the SRS-Volere template, and other stakeholders to aid us in generating appropriate and relevant requirements.

   (b) Reviewing SRS and ensuring all parts are consistent and complete. This is still unresolved, as many sections in the SRS are inconsistent and still incomplete (like the card game mechanics in Sections 8-9, which appear as complete, but are still being discussed among

the team and Project Supervisor Paul Rapoport).
One thing that worked to aid in resolving this pain point, was one of our team members (Ammar) doing an individual review of SRS (before he added all of his sections), to ensure the document is consistent and includes everything important. Upon review, Ammar found several missing subsections, inconsistencies, and suggested improvements, and created a GitHub issue for each one, which was then delegated to the appropriate team member, and resolved by each team member (as written in the created issue).
Another thing that helped was teammate Ammar getting a 3-day extension (which turned into a 4-day extension because of personal circumstances) on his Sections of the SRS. He used this extra time to schedule a meeting with Paul Rapoport and available team members to present our team's SRS, and open it for review to the Professor.
Shortly after the meeting, Supervisor Paul Rapoport sent us an email with his feedback on our SRS, in a written document. This document of Professor Rapoport's review was added to GitHub as an issue (see Project Issue 47). Teammate Ammar reviewed the document during his extension time, and tried to resolve what he could within the SRS document. The rest of the open issues that were there prior to the meeting, and the ones presented by Paul Rapoport in his review, were added to Section 18: Open Issues. These issues will eventually be added to the Project Repo as individual Issues, and resolved appropriately.

(c) Poor time management of teammate Ammar, last minute rush and missing GitHub skills to merge SRS and HA branch commits to main.
Teammate Ammar has apologized and stated any penalties for late submission should go to him and not other teammates. He also created pull requests for branches SRS and HA, as he was unable to merge these to main himself. To resolve the immediate issue of unmerged commits and branches, teammate Ruida stepped in the day after the deadline to fix branch diversions, resolve merge conflicts and merge teammate Ammar's commit history to main.
This being said, the problem of constant late submission by team-

mate Ammar still persists, and requires resolution. The three clear options are:

- Teammate Ammar improves his time management and meets deadlines in the future.
- Teammate Ammar is penalized for the late submission for Deliverable 2 (SRS + HA), and the team continues as is.
- Professor Smith, Professor Rapoport and/or TA Chris come up with some other resolution/compromise for this pain point.

Written by Ammar Sharbat, 2024-10-15

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

    - Most of our system requirements were influenced by feedback from peers and proxy users who represent our target audience—students learning numeral base conversions, additional requirements, such as real-time multiplayer functionality, clear in-game hints, and visual base conversion indicators are added.

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

    - SFWRENG 3A04: Software Design III - Large System Design (Git, designing software architecture, UML diagrams)
    - SFWRENG 4HC3: Human Computer Interfaces (User-Centered Design, Usability Testing)
    - SFWRENG 3RA3: Software Requirements (Git, Github Issues, Writing SRS, Requirements Elicitation)
    - SFWRENG 3S03: Software Testing (Test Case Design, Automated Testing Frameworks)
    - SFWRENG 4C03: Computer Networks and Security (Network Protocols, Security Best Practices)
    - SFWRENG 2AA4: Software Design I (Git, Kanban Board, Object-Oriented Design, Design Patterns)

5. 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, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

    - Ruida: Frontend Development: Deepening our understanding of React component architecture, animation design, and state management.
    - Alvin: Backend Development: Gaining hands on experience in Node.js, Express, and database management.

6. 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?

    - Ruida: Frontend Development (React & Animation) Approaches: (1) Following online React tutorials and official documentation; (2) Experimenting through prototype iterations and peer code reviews. I will choose the first approach - follow online react tutorials, since the official tutorial documentation of react is really straightfoward and easy to learn, lots of code examples are provided.

    - Alvin: Backend Development (Node.js, Express, Database Management) Approaches: (1) Completing online courses and tutorials on Node.js and Express (2) Reviewing course notes and project repositories from past relevant courses.

Table 2: One-time Purchases (CAD)

Item	**Purpose**	**Unit**	**Total**	
Domain name (1 year)	Simple memorable URL for demos	1	20	
			Subtotal	**20**