Development Plan
The Crazy Eights

Ruida Chen
Alvin Qian
Ammar, Sharbat
Jiaming Li

September 2025

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sep 21	Ruida Chen	Worked on section 8-12 of dev plan and first half of appendix
Sep 22	Alvin Qian	Worked on section 1-7 of dev plan and second half of appendix
...	...	...

1 Introduction

This Development Plan defines how the Crazy Eights project will be organized and executed. It establishes scope, objectives, team structure, communication methods, workflow conventions, schedule, risk mitigation focus, technology stack, and coding standards.

2 Confidential Information

There is no confidential or proprietary (industry) information in this project. No NDAs or confidentiality agreements are in place.

3 IP to Protect

There is no external partner IP and the team does not plan to pursue patents. All work is collectively authored by the team and will be released under the selected open source license.

4 Copyright License

The project is released under the MIT License. All source code and documents may be used, modified, and redistributed in accordance with that license.

5 Team Meeting Plan

Meeting Frequency

- Regular weekly meetings:
  - One in-person meeting during scheduled tutorial time
  - One online meeting via Discord (time TBD by team availability)
- Additional ad hoc meetings scheduled as needed for deadlines or unblocking issues
- Bi-weekly supervisor meetings with Dr. Paul Rapoport (virtual or in-person)

Meeting Format

- Primary format: Virtual meetings via Discord (voice channel)
- Secondary format: In-person meetings during class/tutorials
- Meeting roles (rotating):

  - **Chair**: Responsible for
    * Preparing and sharing agenda (12+ hours in advance)
    * Managing time allocation
    * Facilitating inclusive discussion
  - **Note-taker**: Records
    * Key decisions
    * Action items with owners and deadlines
    * Risks and open questions

Attendance Policy

If unable to attend:

- Notify team via Discord in advance
- Review meeting notes independently
- Complete assigned tasks asynchronously

6 Team Communication Plan

We will use multiple coordinated channels to ensure clear, persistent, and efficient communication:

- **GitHub**: Version control, issue tracking, pull request review, and Kanban board management.
- **Discord**: Primary platform for quick text and voice communication. Daily informal updates, coordination of short-term tasks, and quick problem-solving occur here. Urgent blockers are first raised in Discord.
- **In-Person (Tutorial / Scheduled Check-ins)**: Used for structured milestone work, live demonstrations, and supervisor touchpoints.

7 Team Member Roles

Roles are rotational to ensure balanced workload distribution, broaden experience, and encourage shared ownership. Anticipated roles include:

- **Team Liaison (Ammar Sharbat)**: Primary point of contact with the supervisor and course staff; coordinates meeting requests and relays external feedback.
- **Developer (All members)**: Implements assigned features, writes and maintains tests, and updates related documentation.

- **Reviewer (All members)**: Performs code reviews for pull requests, checks alignment with coding standards, and leaves comments if needed before approval.
- **Meeting Chair/Note-taker (All members rotate)**: Facilitates meetings, ensures agenda is followed, and documents key discussion points and action items.

8 Workflow Plan

1. **Update Local Repository**: Pull the latest changes from the 'main' branch to ensure the local repository is up to date.
2. **Branching**: Create a new feature (or fix) branch from 'main'. Follow a consistent naming convention such as: feature/short-description or fix/issue-id.
3. **Coding**: Implement modules and functions according to the design and requirements. Follow the agreed coding standards and document code with comments where necessary.
4. **Unit Testing**: Write and execute unit tests for the newly implemented modules/functions to check expected behaviour. Tests must pass locally before pushing to the branch.
5. **Commit and Push**: Commit changes with descriptive messages. Push to the remote feature branch.
6. **Review and Merge**: Create a pull request into 'main'. A team member reviews for feature logic, style, test coverage, and documentation updates. The reviewer will leave comments and suggestions if needed. After at least one approval, the branch is merged into the main branch.

All project tasks are tracked through GitHub Issues and a Kanban board using GitHub Projects (Backlog / In Progress / Review / Done).

9 Project Decomposition and Scheduling

- Our team will use Github Projects as our core tool for task tracking and project management. Each feature will be decomposed into specific, verifiable small tasks, these tasks will be created in the form of Issues in Github Kanban Board. The Kanban board is divided into 4 stages: Todo, In Progress, Review and Done to visualize the workflow and ensure accountability. Team members will be assigned to different tasks, related pull request will be tied to these issues to maintain traceability.
- Link to our Kanban project: https://github.com/orgs/The-Crazy-Four-Games/projects

- Decomposed Schedule:
  - Week 03: Team Formed, Project Selected
    - Create Github repo
    - Assign initial role
    - Decide on final project selection
  - Week 04: Problem Statement, POC, Development Plan
    - Draft initial problem statement and development plan
    - Set up CI/CD pipeline and Github branch strategy
    - Discuss the final programming languages and frameworks
  - Week 06: SRS + Hazard Analysis (Rev. 0)
    - Draft System Requirements Specification (SRS)
    - Identify hazards and mitigation strategy
  - Week 08: V&V Plan (Rev. 0)
    - Define V&V strategy
    - Break down into different tests and assigned responsibilities
  - Week 10: Design Document (Rev.-1)
    - Decompose system into major modules (frontend, backend, db, api, etc.)
    - Document architecture diagrams
  - Week 11-12: Proof of Concept Demonstration
    - Implement minimum working prototype
    - Prepare slides and live demo for demo presentation
  - Week 16: Design Document (Rev.0)
    - Refine design based on the feedback of POC Demo
    - Add details for extensibility and scalability
  - Week 18-19: Revision 0 Demonstration
    - Implement key features and workflows
    - Conduct internal testing and bug fixing
    - Prepare live demo
  - Week 22: V&V Report Revision 0
    - Execute test plan and record results
    - Analyze test coverage and traceability
  - Week 24: Final Demonstration (Rev.1)
    - Finalize all core features
    - Optimize performance
    - Conduct mock presentation and user feedbacks
  - Week 26: EXPO Demonstration
    - Prepare polished project presentation for EXPO
  - Week 26: Final Documentation (Rev.1)
    - Finalize all documents

10 Proof of Concept Demonstration Plan

Main Risk

- Implementation Complexity: Correctly enforcing the gameplay mechanism (e.g. turn-taking, rule variation, base-12)
- Ensuring required libraries and frameworks install and integrate smoothly across team members' environments.

PoC Demonstration Goals
In our PoC demonstration, we will address these risks by showing:

- A working prototype of Crazy Eights where players can take turns, match cards by suit/rank, and play an "8" as a special
- Successful integration between frontend and backend for real-time gameplay.

11 Expected Technology

Table 2: Expected Technologies and Their Roles

Technology	**Purpose / Usage**	**Category**
React	Builds the user interface for the game, including card rendering, score display, and animations. Provides strong typing and maintainability.	Frontend
Node.js	Handles backend logic, API endpoints, and rule validation. Responsible for communication between frontend and database.	Backend
PostgreSQL	Stores user accounts, scores, and game history in a relational database for persistent data management.	Database
WebSocket / Socket.io	Enables real-time synchronization of turns and game state between players.	Networking
GitHub + GitHub Actions	Provides version control, issue tracking, and CI/CD automation for testing and deployment.	Version Control / CI-CD
Jest / React Testing Library	Frameworks for automated unit and integration testing of frontend and backend components.	Testing
ESLint / Prettier	Ensures consistent coding style and detects syntax or logic errors early in development.	Code Quality
Discord	Used for real-time team communication, coordination, and meeting discussions.	Collaboration
LaTeX	Used for professional documentation, including SRS, VnV Plan, and Development Plan deliverables.	Documentation

12 Coding Standard

- Quality-Oriented Development: Code should be clear, maintainable and consistent across the whole project
- Requirement and Specification-Based: All implementation will be tied to the requirements and specifications
- Defensive Programming: Follow practices that reduce errors and improve robustness

Appendix — Reflection

1. Why is it important to create a development plan prior to starting the project?
   A clear development plan provides the team with structure and direction, it ensures that all team members have a common understanding of the project's goals, responsibilities and timelines. It can also reduce uncertainty, help the team identify risks early, and make it easier to track progress.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?
   Advantages: CI/CD automates testing and deployment, it reduces human errors and improves code quality. Since it encourages frequent integration, problems are detected early.
   Disadvantages: Setting up CI/CD pipelines can be time-consuming, especially for small teams, it may introduce overhead if the project scope is small or team members are not familiar with this tool.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?
   One disagreement our group had was whether to focus solely on developing the Crazy Eights card game, or to make it into a product line that could support multiple card games/number systems, some members felt that the product line idea would make the project impressive and ambitious, while others were concerned about the limited timeline and feasibility. After a team discussion and seeking advice from the professor and supervisor, we resolved this by agreeing to prioritize Crazy Eights as the core deliverable, ensuring we can deliver a complete and functional game. At the same time, we left the product line concept as a stretch goal that could be pursued if time and resources permit. This compromise allowed us to balance ambition with practicality, while keeping the team aligned.

Appendix — Team Charter

External Goals

- Deliver a polished product at the EXPO with hope of receiving positive feedback
- Build a project that can be showcased in future interviews and portfolios
- Aim for a strong course grade by following best practices and meeting all deliverable expectations
- Strengthen our knowledge of modern frameworks so that the project also contributes to our long-term career growth.

Attendance

Expectations

Team members are expected to attend scheduled meetings(in-person or via Discord) on time and stay until the meeting is concluded. Consistent attendance is essential to maintain good communication and progress.

Acceptable Excuse

Acceptable excuses for missing a meeting or a deadline include illness, family emergencies, or unavoidable academic conflicts.Unacceptable excuses include forgetting, oversleeping, etc.

In Case of Emergency

If a team member experiences an emergency, they should notify the team as soon as possible through Discord. They should also provide updates on the status of their assigned tasks and, if necessary, delegate or share their work so that the team can adjust and continue meeting deadlines.

Accountability and Teamwork

Quality

- **Meeting Preparation:**
  - Read the agenda and relevant issues/PRs prior to the meeting.
  - Come prepared with (a) a concise progress update, (b) explicit blockers, (c) questions to ask.
  - For design/architecture discussions, read any documentation shared at least 12 hours before the meeting.
- **Expectation for a code task / issue:**

  - Code compiles/runs locally without new warnings or linter errors.
  - Automated tests added or updated: cover new logic and edge cases.
  - All tests pass locally.
  - Documentation updated: inline comments for non-obvious logic; README/module documentation updated if behaviour, interfaces, or run steps change.
  - Peer review completed (minimum one approval) with review comments addressed or explicitly deferred via a follow-up issue.
- **Timeliness:** Work items should be completed within their originally estimated iteration window.

Attitude

- **Respect and Inclusion:** Listen actively and avoid interrupting. Credit ideas to originators; disagreements focus on the idea, never the person.
- **Communication Responsiveness:** Weekday Discord messages acknowledged within 24 hours (a reaction or short reply). If unavailable (midterms, travel), communicate ahead of time if possible.
- **Constructive Feedback:** Use specific, friendly, actionable language and pair it with reasoning.
- **Conflict of Ideas:** Healthy debate is encouraged, and once a decision is recorded, team members support it unless new evidence emerges.
- **Inclusivity:** Zero tolerance for harassment, discrimination, or disparaging language. Maintain a commitment to inclusivity across backgrounds and experience levels.

Conflict Resolution Process

1. *Direct Discussion:* Involved members attempt a private, respectful conversation to clarify intent and desired outcome.
2. *Mediated Conversation:* If unresolved, bring the issue to the next meeting or request a teammate to facilitate a short discussion.
3. *Escalation:* If behaviour breaches the Code of Conduct, escalate to the supervisor / TA. Persistent issues may be elevated to the instructor.

Stay on Track

The team will use clear metrics and regular check-ins to ensure progress and accountability.

Metrics Tracked

- Meeting attendance (tutorials, scheduled meetings, supervisor check-ins)
- Number of pull requests made and contributions to reviewing PRs

Recognition Positive, on-time, high-quality contributions are acknowledged verbally in weekly meetings. Reusable good practices or discoveries are shared for team-wide adoption.

Managing Underperformance

1. Check-in to clarify blockers (scope, skill gap, time constraints).
2. Adjust: refine issue scope, pair program, or provide targeted resources.
3. If no improvement and no proactive communication: document an action plan with achievable concrete milestones.
4. Continued gaps will need to be escalated to TA / instructor for guidance.

Consequences If deadlines are repeatedly missed without notice:

- Action plan logged (issue comment or team log) with dates.
- Escalation path: (1) team check-in, (2) written plan, (3) supervisor/TA, (4) instructor if unresolved.

Team Building

Lightweight cohesion practices:

- Open casual discussion on any topic for the first 5 minutes of each meeting.
- Celebrate small wins (merged PRs, resolved issues) in meetings.
- Share interesting articles, tools, or tips related to the project in Discord.

Decision Making

1. Open discussion aiming for consensus.
2. If no consensus, move on to a simple majority vote.
3. Record decision (issue comment, PR, or decision log); set as the final decision until new evidence justifies a change.