Gin Rummy, With a Twist
Development Process & Implementation

Team 17

Qiwen Jiao

Yunze (Figo) Li

Zizeng Li

Qixuan Zhong

Chenrui Hao

October 25th, 2024

Content

1 Title

Gin Rummy, With a Twist

2 Revision History

Version Number	Author	Description	Date
Rev 0	All	Initialized Development Plan	October 25th, 2024

3 Team Meeting and Communication Plan

Our team will hold weekly meetings to discuss project progress and ensure everyone is on track with their responsibilities. These meetings will take place either through Teams or WeChat group calls. If screen sharing is required, we will switch to Discord for the group calls. Attendance at these meetings is mandatory for all team members, and any requests to skip a meeting must be made as early as possible.

For daily commination, we will rely on WeChat group chat to raise issues and schedule additional meetings as necessary. In addition to the weekly meetings, we plan to meet with our advisor, Dr. Paul Rapoport, at least once a month to discuss our progress, seek feedback, and address any questions. These meetings will be scheduled in advance via email and will be conducted either in person or through Teams, depending on availability.

Task and issue tracking will be managed through Github's project dashboard. Each team member will be assigned specific tasks, though some tasks may involve multiple team members. We aim to follow Agile workflow, ensuring that each task is clearly tracked in stages: to-do, in-progress and done. GitHub will also serve as our platform for code collaboration and version control. For documentation collaboration, we will use SharePoint or Google Docs to co-author any necessary documents.

4 Team Member Roles

Advisor Professor: Dr. Paul Rapoport

Project Coordinator (Product Manager): Yunze (Figo) Li

Team Member	Email	Role
Qixuan Zhong	zhongq7@mcmaster.ca	Developer
Zizeng Li	li124@mcmaster.ca	Backend Developer
Qiwen Jiao	jiaoq2@mcmaster.ca	Frontend Developer
Chenrui Hao	haoc3@mcmaster.ca	Full Stack Developer
Yunze(Figo) Li	li561@mcmaster.ca	Product Manager & Developer

4.1 Functional Requirements

Functional Requirements	Developers
P0: Implement basic game logic in dozenal scheme (**Backend**)	Zizeng Li, Qixuan Zhong, Yunze (Figo) Li
P0: Implement login feature and user management system (**Backend**)	Yunze (Figo) Li, Chenrui Hao
P0: Basic interaction (Login, Card Animation, Home Pages etc) (**Frontend)**	Qiwen Jiao, Chenrui Hao
P1: Implement a local network server that connects two clients for a match (**Backend**)	Zizeng Li, Qixuan Zhong
P1: Implement basic game logic in the regular (decimal) scheme and allow the user to select between the schemes (**Backend**)	Chenrui Hao, Zizeng Li, Yunze (Figo) Li, Qixuan Zhong
P1: Implement a lobby feature for inviting two users to the same match (**Backend**)	Zizeng Li, Qixuan Zhong
P1: Implement a GUI for the game, preferably a web-based game **(Frontend)**	Qiwen Jiao
P2: Pushing the server to the internet, allowing two remote clients to match **(Backend)**	Zizeng Li, Qixuan Zhong
P2: Implement a matchmaking feature to pair players of similar skill levels. **(Backend)**	Yunze (Figo) Li, Chenrui Hao
P2: Highlight the changed cards in dozenal scheme **(Frontend)**	Qiwen Jiao, Yunze (Figo) Li
P2: Implement animation for dealing cards, sorting cards and other necessary actions **(Frontend)**	Qiwen Jiao, Chenrui Hao
P2: Ensure the game is playable across web browsers on various devices (desktop, mobile, tablets). **(Frontend)**	Qiwen Jiao, Zizeng Li
P3: Implement a leaderboard and ranking system **(Backend)**	Zizeng Li, Yunze (Figo) Li, Qixuan Zhong
P3: Implement game rules in other number bases (octal for example) **(Backend)**	Chenrui Hao, Qixuan Zhong
P3: Implement In-game chat and communication system. **(Frontend)**	Zizeng Li, Yunze (Figo) Li
P3: Implement a verifying process for users to recover their account **(Frontend)**	Qixuan Zhong
P3: Implement skin and customization for user preferences **(Frontend)**	Qiwen Jiao, Yunze (Figo) Li

4.2 Non-Functional Requirements

Non-Functional Requirements	Developers
Look and Feel Requirements	Qiwen Jiao, Yunze (Figo) Li

Usability and Humanity Requirements	Zizeng Li, Qixuan Zhong
Performance and Speed Requirements	Chenrui Hao, Qixuan Zhong
Operational and Environmental Requirements	Chenrui Hao, Qiwen Jiao
Maintainability and Support Requirements	Zizeng Li, Yunze (Figo) Li
Security Requirements	Chenrui Hao

5 Workflow Plan

5.1 Use of GitHub

Our team will utilize GitHub as the central platform for code collaboration and version control. We will adopt a branching strategy where each new feature or bug fix is developed in its own branch derived from the main branch. Before any changes are merged into the main branch, the developer must open a Pull Request (PR). This PR will require peer review by at least one other team member before merging. The review process will consider code style, adherence to coding standards, and functionality to ensure code quality and consistency across the project. Only after the PR has been approved can it be merged into the main branch.

Issues will be managed using GitHub's issue tracker, where tasks and bugs will be logged, prioritized, and assigned to team members. Labels and milestones will be employed to categorize and track the progress of these issues effectively.

5.2 General Outline of the Workflow

1. Pull any new changes from the master branch
2. Create a new branch named [issue number]_[target feature/bug]_[branch owner] based on develop
3. Develop, commit and push to the new branch created, for the target feature/bug
4. Perform unit testing on the modules and functions
5. After the target feature/bug is completed and tested, submit a pull request for review
6. After at least one approval, merge the code into develop branch

5.3 Agile Methods and Sprint Planning

We will implement Agile methodologies to enhance our workflow, specifically by adopting Scrum practices. Our sprint design includes three stages.

The first stage is the **MVP Stage (Minimum Viable Product)**, where our objective is to complete essential tasks before the demo day. During this phase, we will focus on developing a functional prototype that demonstrates the core features of the game, including basic gameplay mechanics, user registration, and simple animations. Sprints will be carefully planned to ensure that all MVP features are developed, tested, and ready for demonstration.

The second stage is the **MMP Stage (Minimum Marketable Product)**, aiming to achieve full functionality with all intended features implemented. The focus here is to integrate advanced game logic, multiplayer synchronization, comprehensive UI/UX elements, and additional functionalities outlined in our requirements. Subsequent sprints will build upon the MVP, adding features and refining existing ones to reach a marketable product level.

The final stage is the **Production Stage**, where our objective is to deploy the application to a server and conduct closed testing. In this phase, we will perform performance optimization, bug fixing, and user experience enhancements based on feedback from closed testing. We will actively seek feedback to make improvements and prepare the application for public release. The final sprints will concentrate on deployment tasks, rigorous testing, and incorporating valuable feedback.

5.4 Data Storage

Our project will utilize PostgreSQL as the primary database for storing all game-related data, including user information, game states, and card images. The database will be deployed on the Google Cloud Platform (GCP), leveraging its robust infrastructure and scalability. By hosting our database on GCP, we ensure secure, reliable, and efficient data management. User data, such as login credentials and gameplay history, will be stored securely with appropriate encryption measures to protect user privacy. Image assets for the cards and other game elements will also be stored in the cloud, allowing for seamless access and updates as needed.

5.5 Compute-Heavy Tasks

Our project does not involve computer-heavy tasks like training machine learning models. The computational requirements are primarily centered around real-time game logic processing, user authentication, and data retrieval, which are efficiently handled by the Spring Boot backend application and the PostgreSQL database. The backend services will be hosted on Google Cloud Platform, ensuring that we can scale resources as needed to handle multiple concurrent users without performance degradation.

5.6 Tools and Methods for Requirements and Performance Metrics

To fulfill the requirements specified in our Software Requirements Specification (SRS) and meet the desired performance metrics, we will utilize a combination of technologies and methodologies. We will employ the Spring Boot framework for building a robust and scalable backend, and Next.js for developing a high-performance front-end application. Our data will be securely stored in a PostgreSQL database hosted on the Google Cloud Platform, ensuring efficient data retrieval and scalability. For the user interface and state management, we will use Shadcn/ui and Redux to create intuitive and responsive interactions. GitHub will serve as our platform for version control and project management, enabling effective collaboration and issue tracking. We will implement automated testing using JUnit and Jest to ensure code reliability and set up Continuous Integration/Continuous Deployment (CI/CD) pipelines with GitHub Actions to automate testing and deployment processes. Performance monitoring tools will be employed to track key indicators such as response times and server load, allowing us to make data-driven optimizations. Finally, we will follow Agile methodologies with Scrum practices, including sprint planning and daily stand-ups, to keep our development process aligned with the performance metrics and ensure timely delivery of all project requirements.

6 Proof of Concept Demonstration Plan

To mitigate potential risks in our development process, we will conduct a proof-of-concept demonstration focusing on key functionalities essential to the project's success. We will develop a functional user authentication system that allows users to register and securely log in. This

involves coding both the front-end user interface and the back-end processes for handling authentication requests, verifying credentials, and managing user sessions. By implementing this feature, we aim to demonstrate our capability to handle user authentication securely and efficiently.

Concurrently, we will construct the core game mechanics by creating a terminal-based version of the game. This includes implementing the dozenal number system, managing the dozen-based deck of 64 cards, and enabling basic gameplay interactions such as shuffling, dealing, and validating moves according to the new game rules. This step will prove that we can successfully translate the game's unique mechanics into a functioning codebase. By coding and demonstrating actual interactions between the front end and back end, we will confirm that our technical infrastructure is sound, ensuring seamless communication between the client and server.

Additionally, we will develop preliminary card animations to illustrate the visual aspect of the game. While these animations may not be final, they will provide insight into the user experience we intend to deliver. Visual elements are essential for user engagement and showcasing them early helps validate our design approach. By presenting these implemented features, aligned with the MVP stage of our development plan, we aim to provide tangible evidence of our progress and capabilities, thereby building confidence in our ability to complete the project successfully.

7 Technology

The technology stack for our project falls into two main categories: front-end and back-end development. For the front end, we will use JavaScript with the Next.js framework, which offers server-side rendering and enhances performance. We will incorporate Shadcn/ui for user interface components and Redux for state management, ensuring a responsive and interactive user experience. The backend will be developed in Java using the Spring Boot framework, facilitating the creation of robust backend services. PostgreSQL will serve as our database management system, hosted on the Google Cloud Platform for scalability and reliability. Our coding environment will primarily be Visual Studio Code for the frontend and IntelliJ IDEA or Eclipse for the backend, providing comprehensive tools for development and debugging.

We will implement unit testing frameworks to ensure code quality and reliability. For the front end, we will use Jest and React Testing Library, which integrate well with React and require minimal configuration. For the backend, we will use JUnit and Mockito to test our Java code, allowing us to validate backend logic effectively. Since our project is primarily software development, following software engineering best practices like unit testing is essential. We will not use any machine learning libraries or GPUs, as our project does not involve machine learning or computer-intensive tasks. Additional technologies include WebSocket communication for real-time multiplayer synchronization and Docker for containerization, ensuring consistency across development, testing, and production environments.

8 Project Scheduling

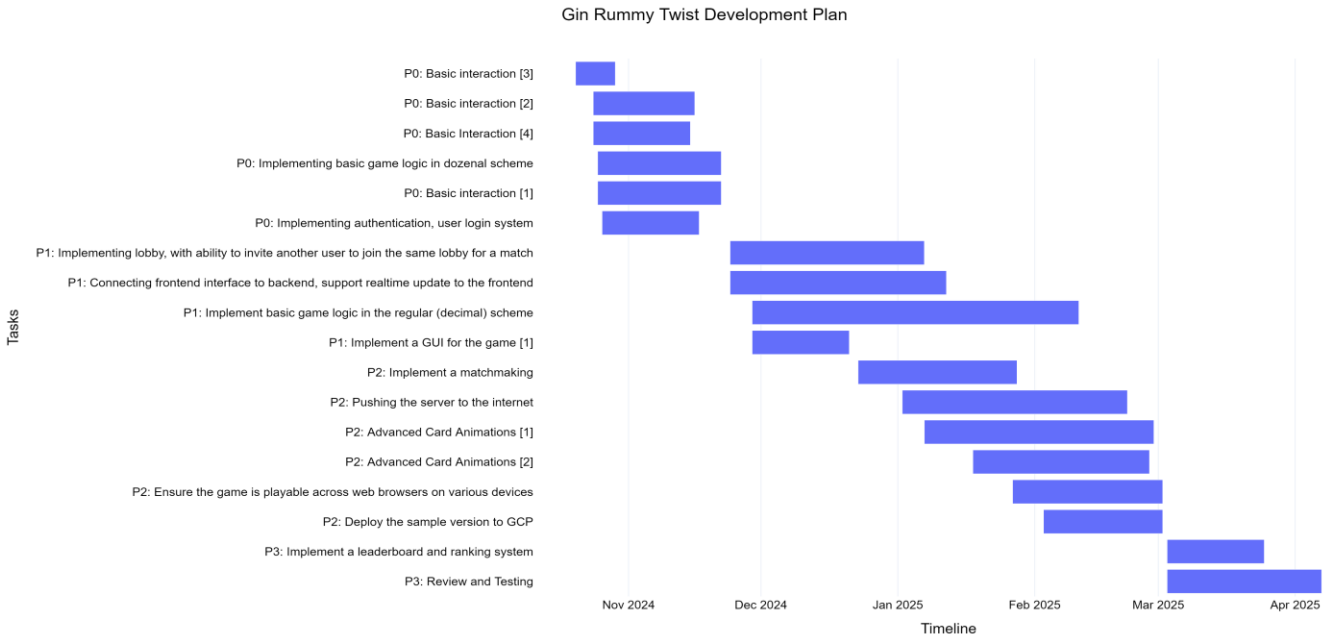

Figure 1Gantt Chart