syst 17796 Deliverable 1

design document template

# Overview

## Project Background and Description

**Project Background and Description:**

The project aims to develop a digital implementation of the classic card game "Go Fish". The final vision is to create a user-friendly application where players can enjoy playing Go Fish against computer-controlled opponents or other players online.

**Game Description:** Go Fish is a simple card game typically played with a standard deck of 52 playing cards. The objective of the game is to collect sets of four cards of the same rank. Players take turns asking opponents for cards of a specific rank. If the opponent has any cards of the requested rank, they must hand them over. If not, the player must "go fish" by drawing a card from the deck. The game continues until all sets of four cards have been collected. The player with the most sets at the end of the game wins.

**Reference to Rules:** Detailed rules for playing Go Fish can be found in various sources, including online card game websites, rulebooks for card games, and instructional videos. A commonly referenced set of rules can be found on the Bicycle Cards website: Bicycle Cards - How to Play Go Fish.

**Current Starting Base Code:** The starting base code for the project includes a set of Java classes structured to support the implementation of the Go Fish game. The code is written in Java, a widely-used object-oriented programming language known for its platform independence and readability. The code follows standard coding conventions, such as meaningful class and method names, appropriate use of access modifiers, and adherence to object-oriented design principles.

The codebase includes classes for representing the game components, such as cards (**Card**), groups of cards (**GroupOfCards**), players (**Player**), and the game itself (**Game**). These classes are designed to provide a foundation for implementing the game logic and user interactions. Additionally, the code may include design patterns such as the Singleton pattern for managing game instances or the Factory pattern for creating card objects.

Overall, the starting base code provides a solid framework for building the Go Fish game application, allowing for easy extension and customization to meet the project goals and final vision.

## Project Scope

**Team Members:**

1. Olumuyiwa Ogunniyi - Lead Developer: Responsible for overall project coordination, task assignment, and timeline management. Leads the development efforts, designs the system architecture, and implements core game functionalities.

**Technical Scope:**

The project aims to develop a digital implementation of the Go Fish card game, which includes the following technical components:

1. **Game Logic:** The core game logic will be implemented to handle the rules and mechanics of Go Fish. This includes functionalities for dealing cards, player turns, card requests, checking for sets, scoring, and determining game outcomes.
2. **Testing:** Comprehensive testing will be conducted to ensure the reliability, functionality, and performance of the game application. This includes unit testing, integration testing, regression testing, and user acceptance testing.

**Project Completion Criteria:**

The project will be considered complete when the following criteria are met:

1. **Functional Requirements:** All core functionalities of the Go Fish game are implemented and working as intended. This includes dealing cards, player interactions, set collection, scoring, and determining the winner.
2. **Bug-Free:** The game application is thoroughly tested, and all identified bugs and issues are resolved. The application should be stable and free from crashes or unexpected behaviors.
3. **Documentation:** Comprehensive documentation is provided, including user guides, developer guides, and technical documentation. This ensures that users and developers have the necessary information to understand and use the application effectively.
4. **Optional Features (If applicable):** If networking capabilities are implemented, they should be tested and functional. The multiplayer feature should allow players to connect, play Go Fish games together, and communicate in real time.

By meeting these criteria, the project will be considered successfully completed, and the Go Fish game application will be ready for deployment and use by players.

## High-Level Requirements

The high-level requirements for the project include:

1. **Player Registration:** Each player must be able to register with the game before starting a match. This allows the game to keep track of individual players and their progress.
2. **Win/Loss Communication:** The game should be able to communicate to players when they win or lose a match. This feedback is essential for providing a satisfying gaming experience.
3. **Score Tracking:** Players should be able to know their current score at all times during the game. This helps players gauge their progress and make strategic decisions accordingly.

## Implementation Plan

* **Git Repository URL:** [Your Git repository URL here]
* **Description:** Developers will check in code at the end of each workday. The repository will be organized with separate directories for different types of files:
  + Code: Java source files
  + UML Diagrams: UML diagrams in appropriate formats (e.g., PNG, PDF)
  + Documentation: Project documentation in Markdown or other appropriate formats
* **Coding Standards:** The project will follow industry-standard coding conventions, including:
  + CamelCase naming convention for classes and methods
  + Proper indentation and spacing
  + Descriptive variable and method names
  + Javadoc comments on documenting classes and methods
* **Tools:** Development may be done using IDEs such as IntelliJ IDEA, Eclipse, or NetBeans. Version control will be managed using Git, and collaborative development will be facilitated using platforms like GitHub or Bitbucket. Testing will be conducted using JUnit for unit testing and possibly other testing frameworks for integration and system testing.

## Design Considerations

* **Encapsulation:**
  + The **Player** class encapsulates player-specific data such as the player's name and score.
  + The **GroupOfCards** class encapsulates card-related functionalities such as adding/removing cards and shuffling.
* **Delegation:**
  + The **Game** class delegates responsibilities to the **Player** and **GroupOfCards** classes for handling player actions and managing the deck of cards.
  + The **Player** class delegates the responsibility of determining a win or loss to the **Game** class.
* **Flexibility/Maintainability:**
  + The code is structured to allow for easy extension and modification. For example, additional functionalities can be added to the **Game** class without affecting other components.
  + The use of abstraction and modularization helps in maintaining and modifying the codebase efficiently. For instance, the game logic can be easily modified without impacting the UI components.