Software Requirements Specification

for

Diamond Detonator

Version 1.1

Prepared by Lis Contoli

February 6, 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a detailed envision for the application, “Diamond Detonator”, a desktop puzzle game. It is intended to guide the development process for the stakeholder, Dr. Frank J. Mitropoulos Ph.D., and to be executed by designer and developer, Lis G. Contoli. This document serves as a flexible blueprint for the application which can be modified to adapt to the stakeholder’s needs and is intended for review by all project participants, including future developers, project managers, and quality assurance teams.

1.2 Document Conventions

This SRS document has a prioritized structure where requirements are organized from the highest to the lowest priority. High priority elements are critical to the game’s core functionality, while lower-priority elements include game features that will not affect the game’s core functionality. The Requirements in font color red are for future releases.

1.3 Intended Audience and Reading Suggestions

Each project participant should focus on the sections of this document relevant to their role. The overall design of “Diamond Detonator” is outlined in this document, detailing function and non-functional requirements, graphical user interfaces, and user interactions. To ensure a thorough understanding of the application’s vision, it is recommended that those tasked with reviewing this document proceed sequentially from beginning to end.

1.4 Product Scope

“Diamond Detonator” is a diamond matching puzzle game that is designed to run on both PC and Mac platforms. The game features an 8x8 board filled with colored diamonds that players match to score points. Its purpose is to deliver engaging and exiting entertainment through a gameplay that also enhances cognitive skills. Its objective is to captivate players by stimulating their competitiveness, challenging them to surpass their own high scores. The product will be developed with scalability in mind, allowing for future incorporation of new features and updates.

1.5 References

SRS document template provided by Dr. Frank J. Mitropoulos Ph.D.

2. Overall Description

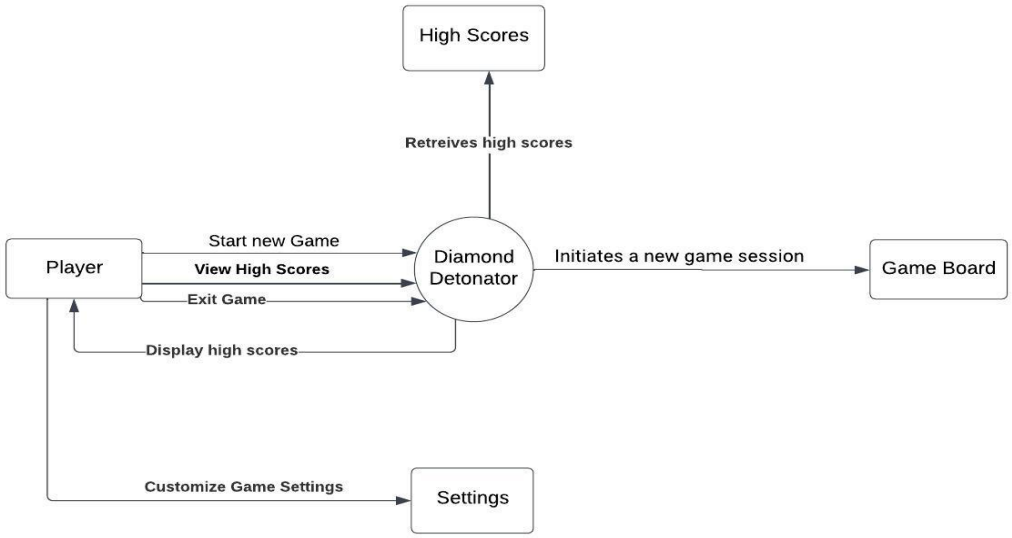
2.1 Product Perspective

"Diamond Detonator" technical foundation will be to utilize the robust capabilities of Python and PyGame for development on PC and Mac platforms. The functionality of this software is to simulate a challenging environment where players strategically match diamonds of the same color to accumulate points.

2.2 Product Features

- Player: Single-player gameplay.
- Diamond Varieties: Four different colored regular diamonds, each with a corresponding special diamond that unleashes a power up. (Detonation of matched and unmatched diamonds of that particular color). Diamonds will each be assigned 100 points, and with the power up it will be doubled.
- Game Board: An [8x8, squared] board that ensures a unique configuration for each new game, and guaranteed to start with at least one playable move. After matching and detonating, the board refills diamonds with equal probability for each color. New board sizes and shapes ([6x6, 10x10, and rectangular]) future release feature.
- Scoring System: Points are accumulated and stored for the player, with a focus on surpassing previous high scores. Player can access previous high scores. Accumulate points for each matched diamond, with higher scores for special diamonds (future release feature).
- Store and display high scores for players to track their progress and compete with others.
- User Interface: Intuitive and user-friendly, ensuring players have a seamless and engaging experience from the main menu to the game board and back.
- Splash screen with the game logo upon launching.
- Main menu for navigating to different sections (Start new game, High scores, Settings, Exit).
- Settings screen for adjusting board size and shape(8x8, square) and match options (3,4,5).
- Game screen displaying the diamond-filled board and score.

Top-Level Data Flow Diagram (DFD)



2.3 User Classes and Characteristics

In the context of the "Diamond Detonator" game, several user classes are identified based on their interaction with the product. Here are some potential user classes along with their characteristics:

Casual Players:
Frequency of Use: May play the game occasionally for entertainment purposes.
Subset of Product Functions Used: Primarily focused on gameplay and scoring.
Technical Expertise: Varied, but likely limited technical expertise in gaming.
Educational Level: Varied, from casual gamers to enthusiasts.
Experience: May have varying levels of experience with puzzle games.

Enthusiast Gamers:
Frequency of Use: Play the game regularly as a leisure activity or to challenge themselves.
Subset of Product Functions Used: Utilize all available features, including different board sizes and match options.
Technical Expertise: Comfortable with gaming interfaces and mechanics.
Educational Level: Can vary but may include individuals with a deeper understanding of game mechanics.
Experience: Likely have extensive experience with puzzle games and gaming in general.

Competitive Players:
Frequency of Use: Play frequently with the goal of achieving high scores and competing with others.
Subset of Product Functions Used: Focus on maximizing score and achieving high scores.
Technical Expertise: Proficient in gaming strategies and tactics.
Educational Level: Varied, but likely includes individuals who enjoy strategic challenges.
Experience: Experienced gamers who actively seek out competitive gaming experiences.

Younger Players:
Frequency of Use: Play the game casually or for entertainment, potentially with parental supervision.
Subset of Product Functions Used: May focus more on visual appeal and simplicity of gameplay.
Technical Expertise: Limited technical expertise, but comfortable with basic gaming interfaces.
Educational Level: School-aged children or teenagers.
Experience: Varying levels of experience with puzzle games, likely influenced by age and exposure to gaming.

Senior Players:
Frequency of Use: Play the game for leisure and mental stimulation.
Subset of Product Functions Used: May prefer simpler gameplay options and larger board sizes for easier visibility.
Technical Expertise: Limited technical expertise, but comfortable with basic gaming interfaces.
Educational Level: Varied but may include individuals with limited experience with digital games.
Experience: May have limited experience with digital gaming, but enjoy the challenge and relaxation provided by puzzle games.

2.4 Operating Environment

Diamond Detonator will run on Windows and MacOS environments.

2.5 Design and Implementation Constraints

In the development of "Diamond Detonator" game, the following design and implementation constraints should be considered:

Technology Stack:
The game must be developed using Python and Pygame as specified by the technical foundation.
Developers must adhere to the capabilities and limitations of these technologies throughout the development process.

Platform Compatibility:
The game must be compatible with both PC and Mac platforms as per the requirements.
Developers need to ensure that the game functions seamlessly on both platforms and addresses any platform-specific issues that may arise.

Hardware Limitations:
Memory requirements must be optimized to ensure smooth performance on a wide range of hardware configurations.
Timing requirements should be met to prevent lag or delay in gameplay interactions.

Scalability:
The design should allow for future incorporation of new features and updates, as scalability is a key consideration.
Developers need to implement a flexible architecture that can accommodate future enhancements without significant rework.

User Interface Design:
The user interface should be intuitive and user-friendly, adhering to design conventions and standards to ensure a seamless user experience.
Design elements and layout should be consistent across different screens and resolutions.

Security Considerations:
While not explicitly mentioned, basic security measures should be implemented to protect user data, such as high scores.
Developers need to ensure that the game does not expose vulnerabilities that could be exploited by malicious actors.

Development Process:
Adherence to project timelines and milestones may be constrained by resource availability and project management considerations.
Developers need to follow any specific development methodologies or coding standards mandated by the client or organization.

2.6 User Documentation

For the "Diamond Detonator" game, the following user documentation components will be delivered along with the software:

1. **User Manual**
   - A comprehensive user manual will be provided, detailing the gameplay mechanics, controls, settings, and scoring system.
   - Installation Instructions will be included.
   - The manual will include step-by-step instructions on how to navigate the game menus, adjust settings, and play the game effectively.
   - It may also contain troubleshooting tips and frequently asked questions (FAQs) to assist users in resolving common issues.
   - Descriptions of different game modes and options (board size, match length, etc.)
2. **Quick Start Guide:**
   - This concise guide should provide a basic overview of the game, installation, and essential gameplay mechanics for new players.
3. **Online Help**
   - Online help resources will be available within the game interface, accessible through the main menu or settings screen.
   - These resources will provide contextual assistance to users while they are playing the game, offering guidance on specific features or actions.
4. **Tutorials:**
   - Tutorials will be included to guide new players through the basic gameplay mechanics and controls.
   - These tutorials may be integrated into the game as interactive guides or presented as standalone videos accessible from the main menu.
5. **Known User Documentation Delivery Formats or Standards:**
   - The user documentation may be delivered in digital formats such as PDF files for the user manual and HTML files for online help.
   - Tutorials may be provided as video files (e.g., MP4) or interactive tutorials embedded within the game interface.
   - The documentation will adhere to standard formatting and language conventions to ensure clarity and ease of understanding for users.

2.7 Assumptions and Dependencies

Assumptions:

- Pygame Compatibility: It is assumed that Pygame will be compatible with the target platforms (PC and Mac) and will provide the necessary functionality to develop the game as outlined in the specification.
- Stable Development Environment: It is assumed that the development environment for Python and Pygame will remain stable throughout the project timeline, without significant updates or changes that could impact development.
- User Familiarity with Gaming Interfaces: The user base is assumed to have basic familiarity with gaming interfaces and concepts, allowing for intuitive interaction with the game without extensive tutorials or guidance.

Dependencies:

- Python and Pygame: The project is dependent on the availability and functionality of Python and Pygame libraries for game development. Any changes or updates to these libraries could impact the project timeline and deliverables.
- Third-Party Components: Dependencies on third-party components or libraries for specific functionalities, such as GUI elements or sound effects, should be documented and managed to ensure compatibility and stability.
- Platform Compatibility: The game's compatibility with PC and Mac platforms is a dependency, as any platform-specific issues or limitations may need to be addressed during development to ensure a consistent user experience across platforms.
- Hardware Resources: The game's performance may depend on the hardware resources available on the user's system, such as CPU, memory, and graphics capabilities. Optimization efforts may be required to ensure smooth gameplay on a wide range of hardware configurations.

3. External Interface Requirements

3.1 User Interfaces Overview

The "Diamond Detonator" game will feature a graphical user interface (GUI) that provides an intuitive and engaging experience for players. The high-level functionality of the system from the user's perspective includes navigation through various screens, gameplay interactions, and feedback mechanisms.

Main Menu Screen:
Functionality: The main menu serves as the central hub for accessing different sections of the game, including starting a new game/ continue game, viewing high scores scrollable pop up, adjusting settings, and help scrollable pop up.
Interaction: Players navigate the main menu using mouse clicks or keyboard inputs to select the desired option.
Feedback: Visual cues such as highlighting or animation indicate the selected option, providing feedback to the player.

Settings Screen:
Functionality: The settings screen allows players to customize game settings such as board size and match options.
Interaction: Players use mouse clicks or keyboard inputs to adjust settings and confirm their selections.
Feedback: Changes to settings are immediately reflected on the screen, providing real-time feedback to the player.

Game Screen:
Functionality: The game screen displays the diamond-filled board and allows players to interact with the game by selecting and matching diamonds.
Interaction: Players use mouse clicks to select diamonds with adjacent matches, forming matches and triggering detonations.
Feedback: Visual cues such as highlighting indicate valid selections, while animations and sound effects provide feedback on successful matches and detonations.

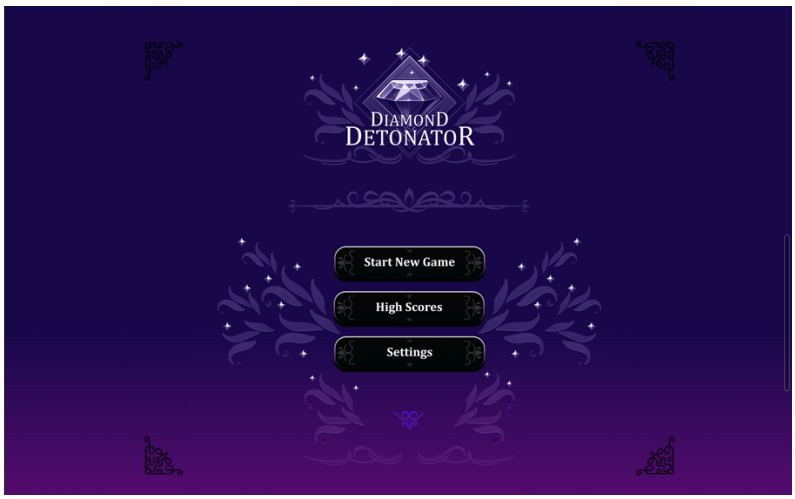
Feedback Information:
Visual Feedback: Visual cues such as highlighting, animation, and color changes provide immediate feedback to the player during gameplay interactions.
Audio Feedback: Sound effects and background music enhance the gaming experience and provide auditory feedback on game events such as matches and detonations.
Score Display: The player's current score is prominently displayed on the game screen, allowing them to track their progress and performance.

Minimum Requirements for the Interface:

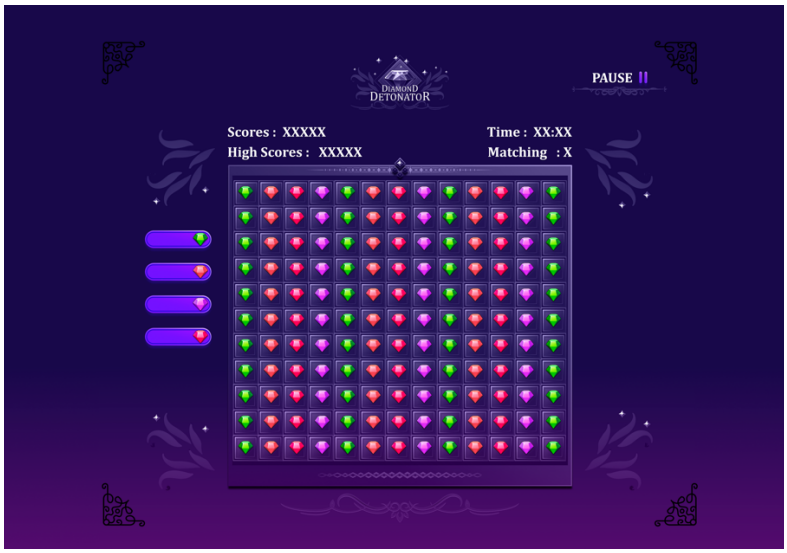
- GUI Standards: The GUI will adhere to standard design principles, including consistency in layout, navigation, and visual elements.
- Screen Layout: Screens will be organized with clear navigation options and prominently displayed content to ensure ease of use.
- Standard Buttons: Common functions such as “Back” or “Main Menu” must be accessible from every screen for consistent user experience.
- Error Messages: Error messages will follow a standardized format, providing clear and actionable information to the player in case of errors or issues.

Sample Screen Images:

Main Menu:



Game Board:



3.2 Hardware Interfaces

For the "Diamond Detonator" game, the hardware interfaces primarily involve the interaction between the software and the input/output devices of the user's system. Here are the logical and physical characteristics of each interface:

- **Mouse Interface:**

Supported Device Types: Standard computer mice or touchpad devices.
Nature of Interaction: The game receives input from the mouse for navigating menus, selecting options, and interacting with the game board.
Communication Protocol: Mouse movements and clicks are translated into corresponding events within the game using the system's input handling mechanisms.

- **Keyboard Interface:**

Supported Device Types: Standard computer keyboards.
Nature of Interaction: Players may use keyboard inputs for navigating menus, entering text (e.g., player names), and triggering certain game actions (e.g., keyboard shortcuts).
Communication Protocol: Keyboard inputs are captured by the game's input handling system and mapped to specific actions or commands within the game.

- **Display Interface:**

Supported Device Types: Computer monitors or displays.
Nature of Interaction: The game renders graphics and visual elements on the screen to provide feedback to the player and display game elements such as the game board, menus, and scores.
Communication Protocol: The game communicates with the display hardware using standard graphical rendering techniques supported by the system's graphics processing unit (GPU) and display drivers.

- **Audio Interface:**

Supported Device Types: Speakers or headphones.
Nature of Interaction: The game produces sound effects and background music to enhance the gaming experience and provide auditory feedback to the player.
Communication Protocol: Audio signals generated by the game are transmitted to the audio output device using the system's audio drivers and protocols (e.g., DirectX for Windows, Core Audio for macOS).

- **Storage Interface:**

Supported Device Types: Hard disk drives (HDD) or solid-state drives (SSD).
Nature of Interaction: The game may read and write data to storage devices for purposes such as saving game progress, storing high scores, and loading game assets.
Communication Protocol: Data transfer between the game and the storage device occurs through the file system interface provided by the operating system, using standard file I/O operations.

These hardware interfaces facilitate the interaction between the "Diamond Detonator" game software and the user's system, enabling players to navigate the game interface, interact with game elements, and receive feedback through various input/output devices.

3.3 Software Interfaces

For the "Diamond Detonator" game, the software interfaces involve connections with various components such as operating systems, development tools, libraries, and potentially databases. Here's a description of the software interfaces:

- **Operating System Interface:**

The game will interface with the underlying operating system (OS) to perform tasks such as window management, input/output handling, and resource allocation.
Data items/messages: Mouse and keyboard inputs, display rendering commands.
Purpose: To interact with system resources and provide a platform for running the game.

- **Python Interpreter:**

The game is developed using Python programming language, and it interfaces with the Python interpreter to execute Python code and access Python libraries.
Data items/messages: Python code files, function calls, variable assignments.
Purpose: To execute game logic, handle events, and interact with system resources.

- **Pygame Library:**

The game utilizes the Pygame library for game development, providing functionalities for graphics rendering, input handling, and sound playback.
Data items/messages: Pygame function calls, events generated by user interactions.
Purpose: To provide a framework for game development and handle low-level interactions with input/output devices.

- **Graphics Libraries:**

The game may interface with graphics libraries or APIs (e.g., OpenGL) to leverage hardware acceleration for rendering graphics on the display.
Data items/messages: Graphics rendering commands, shader programs.
Purpose: To optimize graphics performance and provide visually appealing game visuals.

- **Audio Libraries:**

The game may interface with audio libraries or APIs (e.g., OpenAL) to handle sound effects and music playback.
Data items/messages: Audio playback commands, sound files.
Purpose: To provide immersive audio experiences and enhance the gaming atmosphere.

- **Storage Interface:**

The game may interface with the file system or external databases to read/write game data such as high scores, player settings, and saved game states.
Data items/messages: File I/O operations, database queries.
Purpose: To store and retrieve game data persistently across sessions.

The nature of communications between the "Diamond Detonator" game and these software components involves function calls, event handling, and data exchange. Data sharing across software components may include game state information, player input, and configuration settings. Implementation constraints may include the use of specific APIs or libraries for graphics/audio rendering and adherence to platform-specific guidelines for operating system interactions.

4. System Requirements

4.1 Non-Functional Requirements

Non-functional requirements for the "Diamond Detonator" game may include:

1. **Development Methodology:**

    - Object-Oriented Design (OOD): The game should be developed using OOD principles to ensure code reusability, maintainability, and extensibility. This will allow for easier implementation and integration of new features, game board sizes, and shapes, as well as facilitate the maintenance and scaling of the codebase as the game evolves.

2. **Performance:** The game should run smoothly without lags or delays, even on lower-end devices. Response times for user interactions should be minimal.

3. **Scalability:** The game should be able to handle an increasing number of players and game sessions without significant degradation in performance.

    - Dynamic Grid System: Implement a flexible grid system within the game board that can dynamically adjust to various sizes and shapes, ensuring that new configurations can be integrated seamlessly.

4. **Reliability:** The game should be stable and reliable, with minimal crashes or unexpected shutdowns. It should gracefully handle errors and exceptions.

5. **Usability:** The user interface should be intuitive and easy to navigate, ensuring a seamless and enjoyable gaming experience for players of all skill levels.

6. **Accessibility:** The game should be accessible to players with disabilities, supporting features such as screen readers, keyboard navigation, and color contrast options.

7. **Security:** The game should ensure the security and privacy of player data, protecting against unauthorized access, data breaches, and malicious attacks.

8. **Compatibility:** The game should be fully compatible with Windows and macOS operating systems, ensuring consistent performance and visual fidelity on both platforms. It should be optimized for a variety of screen resolutions commonly used on desktop and laptop computers to accommodate the diverse hardware configurations of PC and Mac users.

9. **Maintainability:** The game should be designed with clean, modular code and well-documented architecture, facilitating easy maintenance, updates, and bug fixes.

4.2 Functional Requirements

Functional requirements for the "Diamond Detonator" game must include:

1. **Game Setup**
   A. The software system shall allow the player to select the desired board size from options.
   B. The software system shall allow the player to choose the desired match size from options including 3, 4, and 5.
   C. The software system shall automatically save any changes made by the user in the settings screen.
2. **Game Mechanics**
   A. The game shall generate a unique configuration for the game board at the start of each new game session.
   B. The game shall ensure that each game board starts with at least one playable move.
   C. The game shall refill diamonds on the game board after matching and detonating with equal probability for each color.
   D. The game shall score points for each matched diamond based on the following criteria:
      - Regular diamonds shall be worth 100 points.
      - Matches involving special diamonds shall double the scored points.
   E. The game shall not allow swapping of diamonds. Matches must be made by selecting adjacent diamonds.
   F. Matching of adjacent diamonds constitute of top, bottom, left, and right. No diagonal matching.
   G. The software system must check if adjacent diamonds match the color of the selected diamonds.
      - If the selected diamond has the match size amount in adjacent diamonds, denotation is triggered for the diamonds that are matched.
   H. The game shall detect, whether after refilling the board after first available move, there are any matches left.
      - If there are no matches detected in the board, the game is automatically over, and a pop up saying “Game Over” will notify the user.
   I. The game shall hint the user about possible matches on the game board.
3. **User Interface**
   A. The software system shall include a splash screen displaying the game logo upon startup.
   B. The software system shall provide a main menu screen offering options to start a new game/continue game, access high scores, adjust settings, and help.
   C. The high scores button will activate a scrollable pop up displaying high scores, or previous high score with the option to delete/reset history.

D. The software system shall include a settings pop up allowing the player to customize board size for square shaped board(6x6,8x8, 10x10), and rectangular shaped board(8x10,9x12,10x15), and board shape (square, rectangle) and match options(3,4,5, +) and option to turn music on/off.

E. The game screen shall display the game board with diamonds, highlighting incorrect selections in orange and providing hints in light blue.

F. The game screen shall have a button that the user can click on so that the system shows the hint.

G. The game screen must include pop up that notifies the user when the game is over.

H. The software system must take the user back to Main Menu after the "Game Over" pop up pops up.

I. The software system must allow the user to undo their last match only one time during the current gameplay.

4. Scoring System

A. The game shall accumulate and store points for the player based on their actions during gameplay.

B. The player's score shall be prominently displayed on the game screen.

C. The game shall focus on encouraging players to surpass their previous high scores.

D. If the player decides to undo a move, the score gained for that match will deduct from the player's score.

5. High Scores

A. The software system shall maintain a record of the highest scores achieved by players in previous game sessions.

B. The high scores shall be accessible to the player from the main menu.

C. The high scores shall be sorted in descending order, with the highest score displayed at the top of the list.

D. The player shall have the option to reset the high scores to zero or delete certain scores from the list.

6. Player Interaction

A. The game shall respond to player interactions with smooth animations and transitions.

B. The player shall be able to make matches by selecting adjacent diamonds using the mouse or keyboard input.

C. The game shall provide visual feedback to indicate successful matches and detonations.

D. The player shall have the option to undo their last move.

7. Game Progression

A. The player shall have the option to pause and resume the game at any time.

B. The game will not have levels, the difficulty will be customizable by the player through the settings pop up [board size options, board shape options, match size options)

8. Audio/Visual Effects

A. The game shall include sound effects and background music to enhance the gaming experience.

B. The game shall provide options for the player to adjust the volume of sound effects and music.

C. The game shall feature visually appealing graphics and animations to engage the player.

5. System Features/Modules

5.1 Use Case 1: Start New Game

Use Case Id:	UC-001	**Use Case Name:**	Start New Game
Actors:	Player		
Description:	This use case describes the process of starting a new game in the "Diamond Detonator" game.		
Trigger:	Player selects the "Start New Game" option from the main menu.		
Pre-conditions:	The game is launched and the player is on the main menu screen.		
Post-conditions:	A new game session is initiated, and the player is taken to the game screen with a fresh game board.		
Normal Flow:	Player selects the "Start New Game" option from the main menu. The system generates a new game board with randomized diamond placements. The player is taken to the game screen, where they can begin playing.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	High		
Special Requirements:	None		
Assumptions:	The player is familiar with the game controls and mechanics.		
Notes and Issues:	None		

5.2 Use Case 2: View High Scores

Use Case Id:	UC-002	**Use Case Name:**	View High Scores
Actors:	Player		
Description:	Displays the high scores achieved by players.		
Trigger:	Player selects the "High Scores" option from the main menu.		
Pre-conditions:	The game is launched and the player is on the main menu screen.		
Post-conditions:	The high scores screen is displayed to the player.		
Normal Flow:	Player selects the "High Scores" option from the main menu.		

	The system retrieves and displays the high scores achieved by players in previous game sessions.
Alternative Flow 1: [Class ID overwritten]	None
Exceptions:	None
Includes:	None
Frequency of Use:	Moderate
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

5.3 Use Case 3: Adjust Settings

Use Case Id:	UC-003	**Use Case Name:**	Adjust Settings
Actors:	Player		
Description:	Allows the player to customize game settings such as board size and match options.		
Trigger:	Player selects the "Settings" option from the main menu.		
Pre-conditions:	The game is launched and the player is on the main menu screen.		
Post-conditions:	The player's settings are updated based on their selections.		
Normal Flow:	Player selects the "Settings" option from the main menu. The system displays the settings screen with options to adjust board size, board shape, and match size options. Player modifies the settings according to their preferences. The system saves the updated settings.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	Moderate		
Special Requirements:	None		
Assumptions:	None		
Notes and Issues:	None		

5.4 Use Case 4: Exit Game

Use Case Id:	UC-004	**Use Case Name:**	Exit Game
Actors:	Player		
Description:	Exits the game and closes the application.		
Trigger:	Player selects the "Exit" option from the main menu.		
Pre-conditions:	The game is launched and the player is on the main menu screen.		

Post-conditions:	The help pop up is displayed, and the player can read about the game and its rules.
Normal Flow:	Player selects the "Help" option from the main menu. The system displays the help screen.
Alternative Flow 1: [Class ID overwritten]	None
Exceptions:	None
Includes:	None
Frequency of Use:	Moderate
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

5.5 Use Case 5: Match Diamonds

Use Case Id:	UC-005	**Use Case Name:**	Match Diamonds
Actors:	Player		
Description:	Allows the player to match diamonds of the same color on the game board.		
Trigger:	Player selects and swaps adjacent diamonds to form matches.		
Pre-conditions:	The game is in progress, and the player is on the game screen.		
Post-conditions:	Matched diamonds are removed from the game board, and the player's score is updated accordingly.		
Normal Flow:	Player selects a diamond. The system verifies if the chosen diamond results in a match of three or more adjacent diamonds of the same color, top, bottom, left, right. If a match is found, the system removes the matched diamonds from the game board. The system updates the player's score based on the number of diamonds matched.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	High		
Special Requirements:	None		
Assumptions:	None		
Notes and Issues:	None		

5.6 Use Case 6: Detonate Matched Diamonds

Use Case Id:	UC-006	**Use Case Name:**	Detonate Matched Diamonds
Actors:	Player		
Description:	Detonates matched diamonds to remove them from the game board and score points.		
Trigger:	Player identifies matches of three or more diamonds of the same color.		
Pre-conditions:	The game is in progress, and the player has identified a match of three or more adjacent diamonds within their selection of diamond.		
Post-conditions:	Matched diamonds are removed from the game board, and the player's score is updated accordingly.		
Normal Flow:	Player identifies a match of three or more adjacent diamonds of the same color from the selected diamond. The system removes the matched diamonds from the game board. The system updates the player's score based on the number of diamonds matched.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	High		
Special Requirements:	None		
Assumptions:	None		
Notes and Issues:	None		

5.7 Use Case 7: Access High Scores

Use Case Id:	UC-007	**Use Case Name:**	Access High Scores
Actors:	Player		
Description:	Allows the player to access the high scores achieved in previous game sessions.		
Trigger:	Player selects the "High Scores" option from the main menu.		
Pre-conditions:	The game is launched, and the player is on the main menu screen.		
Post-conditions:	The high scores scrollable pop up is displayed to the player.		
Normal Flow:	Player selects the "High Scores" option from the main menu. The system retrieves and displays the high scores achieved by players in previous game sessions.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	Moderate		

Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

5.8 Use Case 8: Customize Settings

Use Case Id:	UC-008	**Use Case Name:**	Customize Settings
Actors:	Player		
Description:	Enables the player to customize game settings such as board size, board shape and match options according to their preferences.		
Trigger:	Player selects the "Settings" option from the main menu.		
Pre-conditions:	The game is launched, and the player is on the main menu screen.		
Post-conditions:	The player's settings are updated based on their selections.		
Normal Flow:	Player selects the "Settings" option from the main menu. The system displays the settings screen with options to adjust board size and match options. Player modifies the settings according to their preferences. The system saves the updated settings.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	Moderate		
Special Requirements:	None		
Assumptions:	None		
Notes and Issues:	None		

5.9 Use Case 9: Receive Hint

Use Case Id:	UC-009	**Use Case Name:**	Receive Hint
Actors:	Player		
Description:	Provides the player with a hint to help them find matching diamonds on the game board.		
Trigger:	Player requests a hint during gameplay.		
Pre-conditions:	The game is in progress, and the player is on the game screen.		
Post-conditions:	The system provides a visual cue to highlight a possible matching identified.		
Normal Flow:	Player selects the "Hint" option from the game interface.		

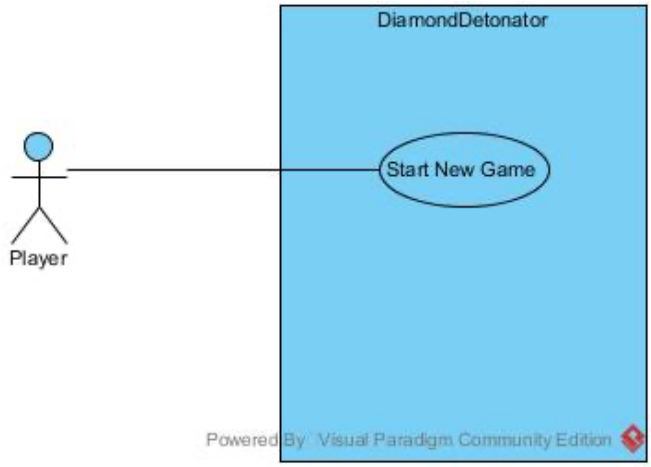
	The system analyzes the game board and identifies possible matching. The system highlights the diamonds involved in the matching to provide the player with a hint.
Alternative Flow 1: [Class ID overwritten]	None
Exceptions:	None
Includes:	None
Frequency of Use:	Low
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

5.10 Use Case 10: Access Previous High Scores

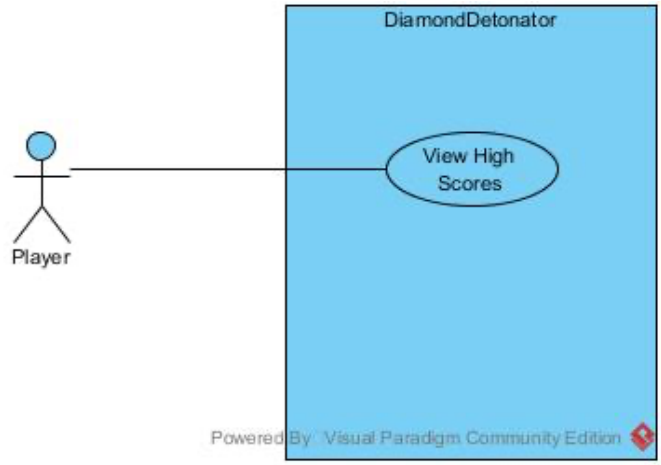
Use Case Id:	UC-010	**Use Case Name:**	Access Previous High Scores
Actors:	Player		
Description:	Allows the player to access their previous high scores achieved in different game sessions.		
Trigger:	Player selects the "High Scores" option from the main menu.		
Pre-conditions:	The game is launched, and the player is on the main menu screen.		
Post-conditions:	The high scores scrollable pop up is displayed to the player.		
Normal Flow:	Player selects the "High Scores" option from the main menu. The system retrieves and displays the high scores achieved by the current player in previous game sessions.		
Alternative Flow 1: [Class ID overwritten]	None		
Exceptions:	None		
Includes:	None		
Frequency of Use:	Low		
Special Requirements:	None		
Assumptions:	None		
Notes and Issues:	None		

6. Use Case Diagrams

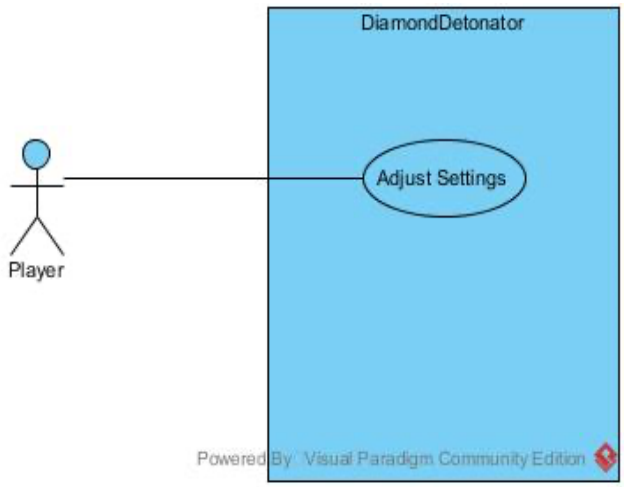
6.1 Use Case 1 Diagram: Start New Game



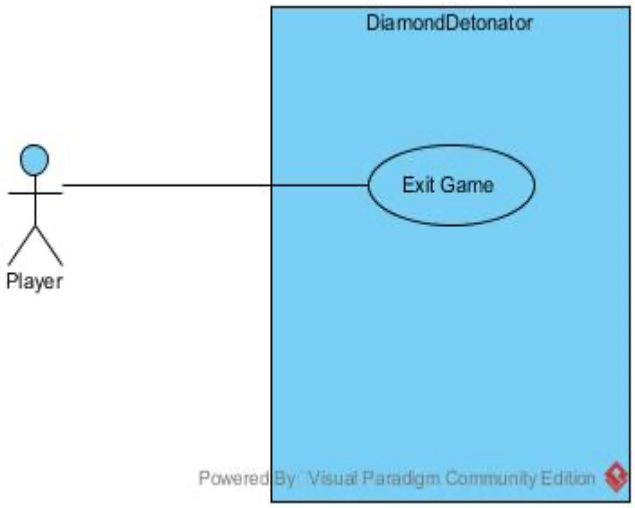
6.2 Use Case 2 Diagram: View High Scores



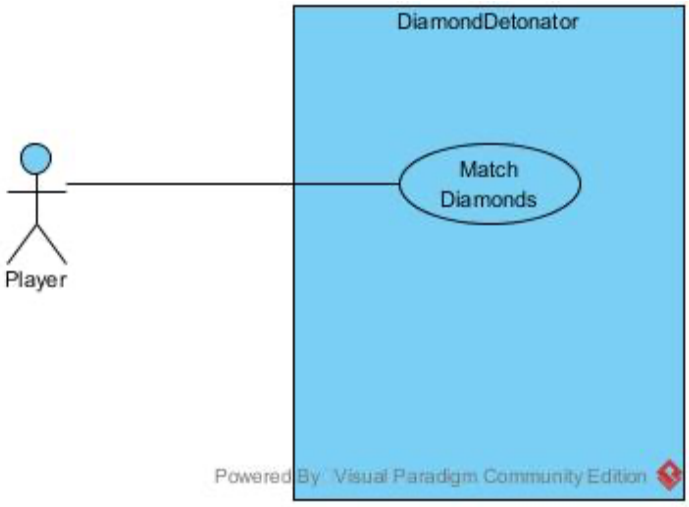
6.3 Use Case 3 Diagram: Adjust Settings



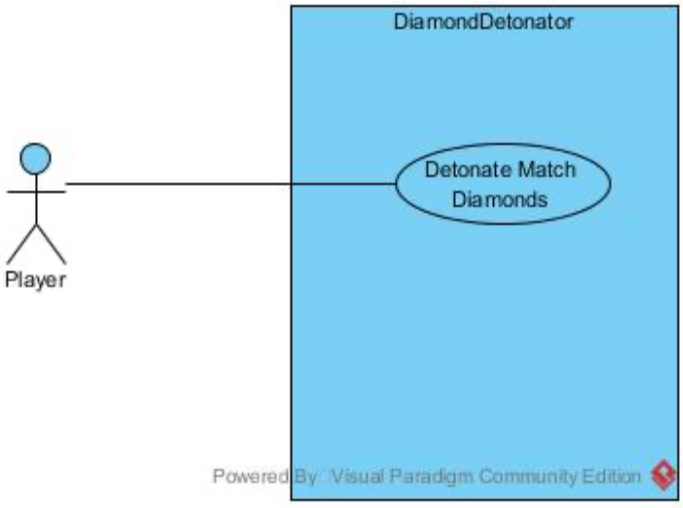
6.4 Use Case 4 Diagram: Exit Game



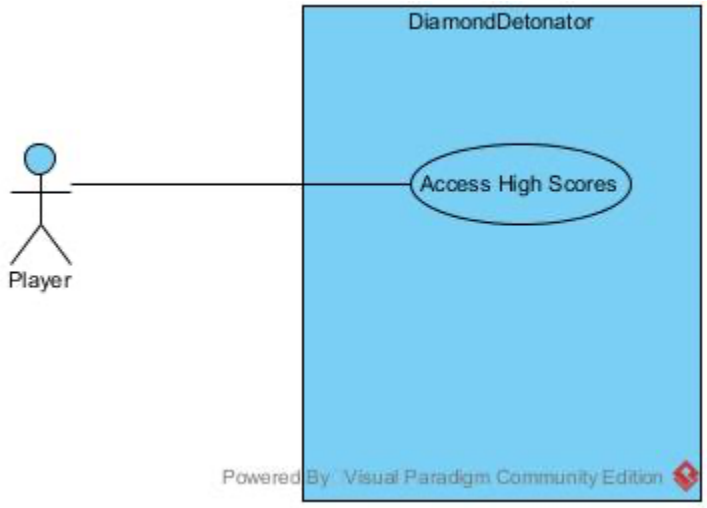
6.5 Use Case 5 Diagram: Match Diamonds



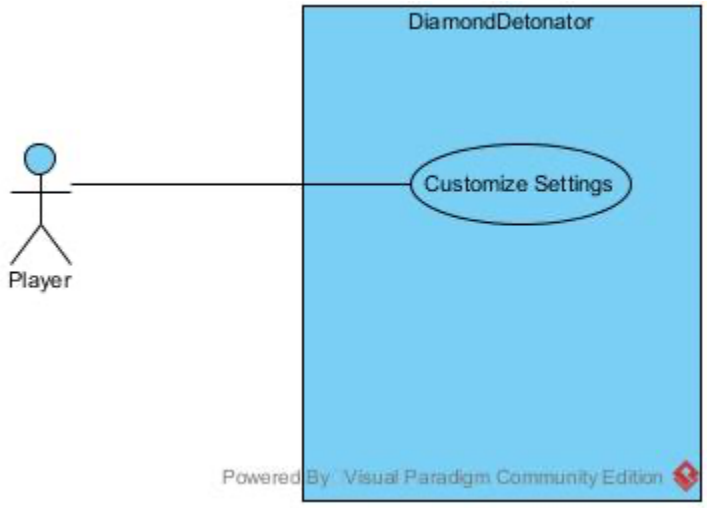
6.6 Use Case 6 Diagram: Detonate Matched Diamonds



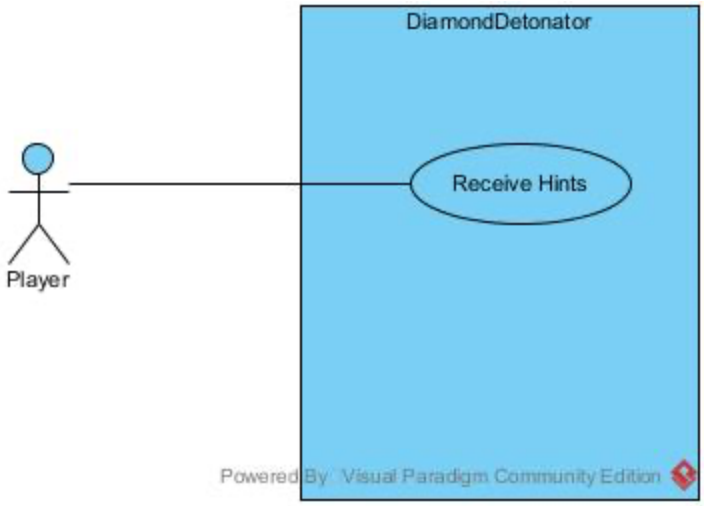
6.7 Use Case 7 Diagram: Access High Scores



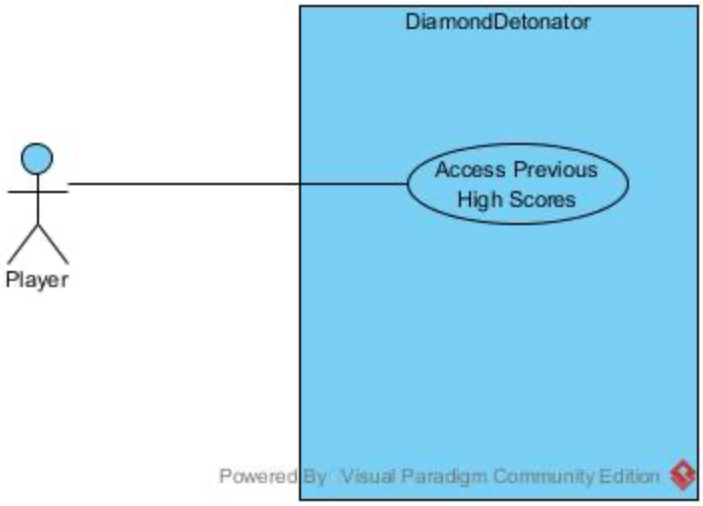
6.8 Use Case 8 Diagram: Customize Settings



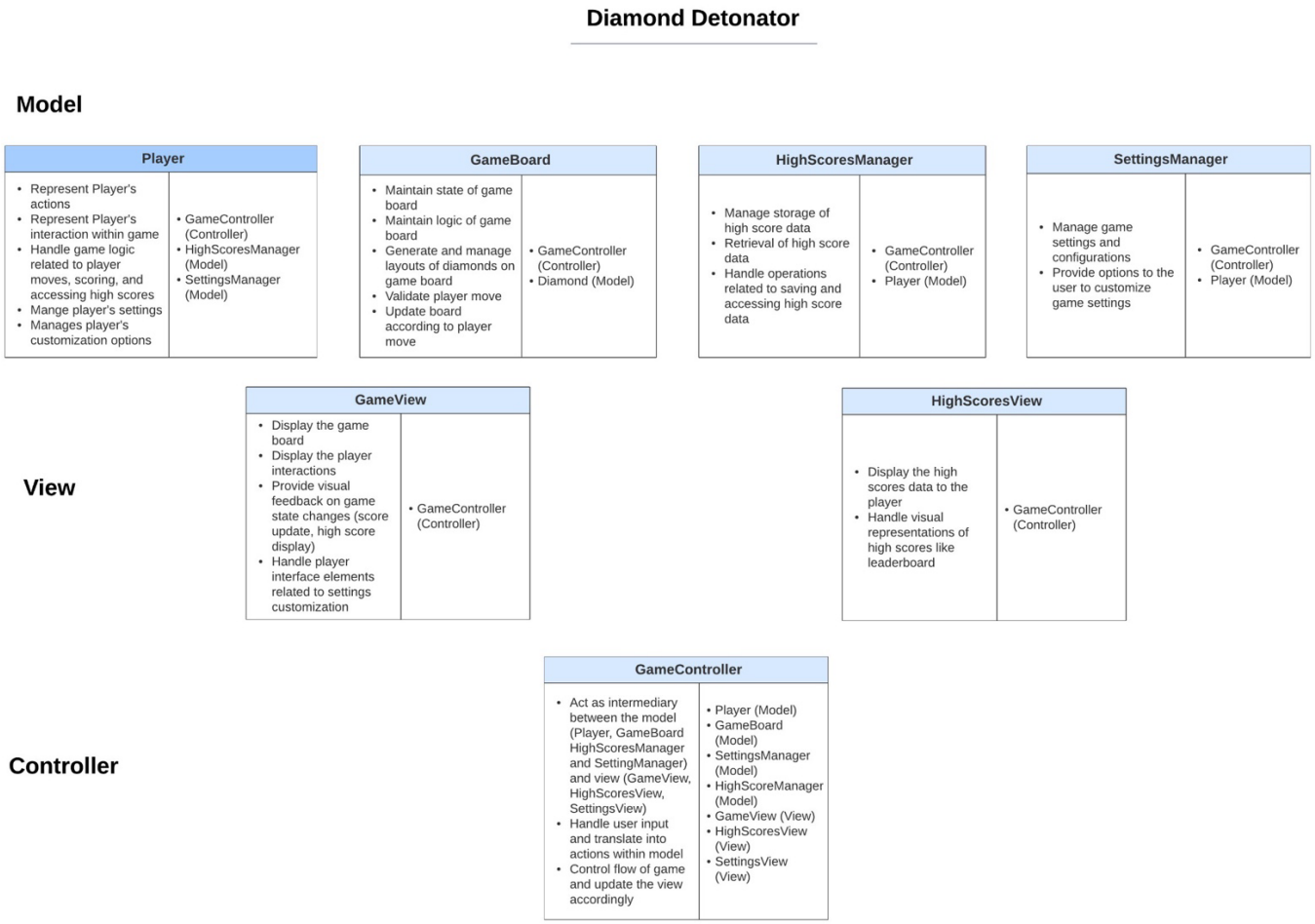
6.9 Use Case 9 Diagram: Receive Hints



6.10 Use Case 10 Diagram: Access Previous Scores



7. Class-Responsibility-Collaboration (CRC) Cards



Explanation:

Model:

Class: Player

Responsibilities:

- Represent the Player's actions and interactions within the game.
- Handle game logic related to player moves, scoring, and accessing high scores.
- Manage player settings and customization options.

Collaborators:

- GameController (Controller)
- HighScoresManager (Model)
- SettingsManager (Model)

Class: GameBoard

Responsibilities:

- Maintain the state and logic of the game board.
- Generate and manage the layout of diamonds on the game board.
- Validate player moves and update the board accordingly.

Collaborators:

- GameController (Controller)
- Diamond (Model)

Class: HighScoresManager

Responsibilities:

- Manage the storage and retrieval of high scores data.
- Handle operations related to saving and accessing high scores.

Collaborators:

- GameController (Controller)
- Player (Model)

Class: SettingsManager

Responsibilities:

- Manage game settings and configurations.
- Provide options for players to customize game settings.

Collaborators:

- GameController (Controller)
- Player (Model)

View:

Class: GameView

Responsibilities:

- Display the game board and player interactions.
- Provide visual feedback on game state changes like score update, high scores display.
- Handle user interace elements related to settings customization.

Collaborators:

- GameController (Controller)

Class: HighScoresView

Responsibilities:

- Display high scores data to the player.
- Handle visuals representations of high scores like leaderboard.

Collaborators:

- GameController (Controller)

Controller:

Class: GameController

Responsibilities:

- Act as an intermediary between the model (Player, GameBoard, HighScoresManager, SettingsManager) and the view (GameView, HighScoresView, SettingView)
- Handle user input and translate it into actions within the model.
- Control the flow of the game and update the view accordingly.

Collaborators:

- Player (Model)
- GameBoard (Model)
- HighScoresManager (Model)
- SettingsManager (Model)
- GameView (View)
- HighScoresView (View)
- SettingsView (View)