# Requirement

The following table shows the functional requirements of milestone1 and the progress of each functional requirement.

|  |  |
| --- | --- |
| 1. The game shall provide a timer that player can check the time since them start the game until the game finish. | TimeController is created and properties and methods are defined |
| 2. The game shall let players know how many bomb left shows on the top of the screen. | The Class: GameBoard has been defined and the basic design of the game page has been completed. At the top of the game board will show how many bombs remain. |
| 3. The game shall provide a button that players are able to restart the game by click a single button when they meet difficulty in the game. | Have designed, development work is ongoing |
| 4. The game shall give the result immediately after players uncover a tile whatever it is a bomb or blank. | Defines the minisquare in the game panel. Defines the method of mouse monitoring. |
| 5. If the tile clicked by the player is not a bomb then a number will shows on the clicked tile to describe how many bombs around this tile | Defined the method, currently developing ongoing |
| 6. The game shall let the game stop immediately when player touch the bomb because they are failed. | Defined the countBomb method to calculate the number of surrounding bombs, ongoing |
| 7. The game shall provide players are able to use a flag to sign on a blank tile as a bomb so that they can know that how many bombs have already found. | Defines the mouseClicked method. |
| 8. The number of total bomb left should reduce after player place a flag on the blank tile. | Will display on the top of the game's interface, in the design phase |
| 9. The game shall allow player to place a question mark on the blank tile to sign it as a undetected tile | Defines the mouseClicked method. |
| 10. The game shall provide a selection after game over whether players want to play again or exit the game. | Defines the Clicked method. |

# Product Use Cases

In this game, the role of the game is a single user, user has the following types of use case:

1. Start Game

In this use case, the user launches the program and enters the game.

2. Action on Square

“Action on Square” is the user's mouse action. The user left-clicks the mouse to perform the mine-clearing operation.

3. Mark on Square

“Mark on Square” is the user's tagging operation. The user right-clicks or double-clicks the mouse to mark the attribute of the square

4. Restart Game

When the user encounters a difficulty and cannot play the game, he can choose to restart the game.

5. Exit Game

When the user finishes finding out all the bombs, or the user clicks on a bomb, the game ends.

6. Decide play or exit

After game over, the system will ask the user whether to repeat the game or opt out.

Use case implementation progress report:

|  |  |
| --- | --- |
| Use Case | Progress |
| Start Game | Have designed classes and methods, functions development are ongoing |
| Action on Square | Have designed and developed right-click operation |
| Mark on Square | Have done right-click mark on minisquare |
| Restart Game | Have designed the interface on the GameBoard |
| Exit Game | Have designed the game process |
| Decide play or exit | Have designed the game process and method |

# Software Architecture

The programming language used in this software engineering design is Java language, and the development tool used is Intellij IDEA.

Java is an object-oriented programming language with simplicity, object-oriented, distributed, robust, security, platform independence and portability, multi-threading, dynamics, etc. At the same time, simple GUI design is also very easy to develop. The development of small games.

The software architecture used in this software project is mainly the design of MVC (Model-View-Controller) separation. MVC is a software design paradigm that organizes code in a way that separates business logic, data, and interface. In this architecture design, the entire engineering module is divided into three parts:

|  |  |
| --- | --- |
| Model | The Model module mainly defines the entity classes and interfaces needed in the project. |
| View | The View module mainly defines the interface of the game and various operations of the user under the interface. |
| Controller | The class StartGame is defined in this module, and the main function is created to start the entire game. |

# Summary of Design

## Design Goals

Design goals:

Implement the functional and non-functional requirements defined by milestone1. There are some design principles which need to satisfy:

* Greater expressive power
* Solution orientation
* Very high modularity
* Easy to change or extend
* Code readability
* Simplification through abstraction

## Design Patterns

Object-oriented design focuses on "things" in the real world, which are part of the software engineering problem. Engineers need to define these things (we will call them objects) based on attributes and behaviors. In this project, the things correspond to the users of the game. The project needs to be designed and developed according to the various operations that the user can perform in the game and the various results obtained.

The whole design idea mainly adopts the top down design approach. The top-down design approach first identifies the main modules of the system, breaking them down into lower-level modules and iterating until the required level of detail is reached. This process is gradually improved. At the top, an abstract design is required, and then the design is refined to a more specific level in each step until the system reaches a level that no longer needs to be refined, and the design can be implemented directly.

Meanwhile, “Abstraction-Occurrence Pattern” and “The General-Hierarchy Pattern” are also used in this software engineering.