|  |
| --- |
| Result2017\_0001 |

<총알피하기-닷지>

Result Report

V1.7

2017- 12- 06

[Document Info]

|  |  |
| --- | --- |
| Document Num. | Result2017\_0001 |
| Name of Software | 총알피하기-닷지 |
| Type of Software | 게임 |
| Institute | Korea National University of Transportation |
| Researcher | 박휘준, 배현오 |

[Document Configuration]

|  |  |  |
| --- | --- | --- |
| **Version** | **Summary** | **Date** |
| V1.0 | Initiate version | 2017.11.07 |
| V1.1 | 개요 및 운영환경, 스케줄 추가 | 2017.11.13 |
| V.1.2 | 소프트웨어 아키텍처 추가 | 2017.11.18 |
| V.1.3 | Use Case, Class diagram 추가 | 2017.11.21 |
| V.1.4 | Sequence,flow diagram 추가 | 2017.11.23 |
| V.1.5 | 전반적 수정 | 2017.11.27 |
| V.1.6 | 플로우 차트, 시퀀스 다이어그램 수정 | 2017.12.05 |
| V 1.7 | 최종 수정 | 2017.12.06 |
|  |  |  |
|  |  |  |

Contents

[Result Report 1](#_Toc444702051)

[1 Introduction 4](#_Toc444702052)

[1.1 Overview 4](#_Toc444702053)

[1.2 Operational environment 4](#_Toc444702053)

[2 Project Development Methodology 6](#_Toc444702054)

[2.1 Development Process 4](#_Toc444702053)

[2.2 Step-by-step description 4](#_Toc444702053)

[3 Project Scope 7](#_Toc444702055)

[3.1 Physical Scope 10](#_Toc444702056)

[3.2 Logical Scope 11](#_Toc444702057)

[4 Schedule 15](#_Toc444702059)

[4.1 Members 15](#_Toc444702060)

[4.2 Project plans 17](#_Toc444702063)

[5 Software Architecture 19](#_Toc444702054)

[5.1 Subsystem-based architecture 20](#_Toc444702053)

[5.2 Subsystem description 21](#_Toc444702053)

[6 Diagrams 27](#_Toc444702055)

[6.1 Use Case diagram 28](#_Toc444702056)

[6.2 Class diagram 28](#_Toc444702056)

[6.3 Sequence diagram 28](#_Toc444702056)

[6.4 Flow Chart 29](#_Toc444702056)

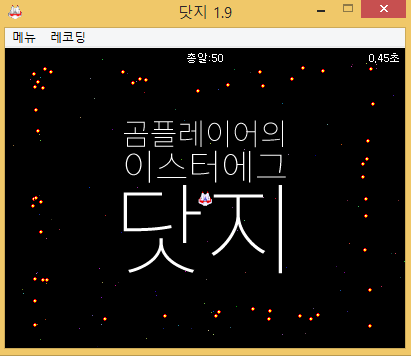
[7 Evaluation 27](#_Toc444702055)

[7.1 Functional Test 28](#_Toc444702056)

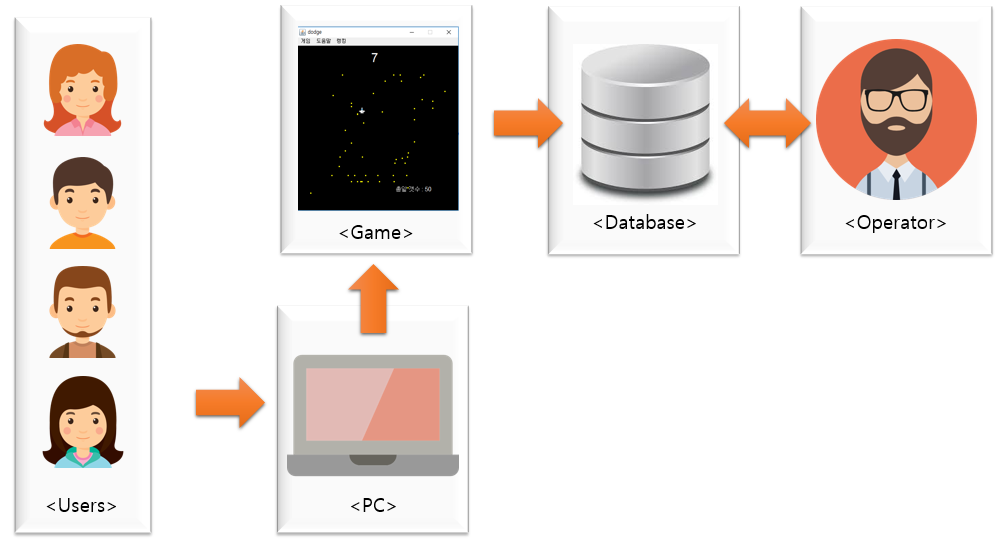
# Introduction

## Overview

일전에 모 프로그램의 이스터 에그로 포함 되어있던 닷지 게임을 하나의 독립적인 게임으로 구현했습니다. 여기에 추가적으로 랭킹 시스템을 도입해 자신의 점수를 서버에 등록하며 다른 유저와의 경쟁을 즐길 수도 있습니다.



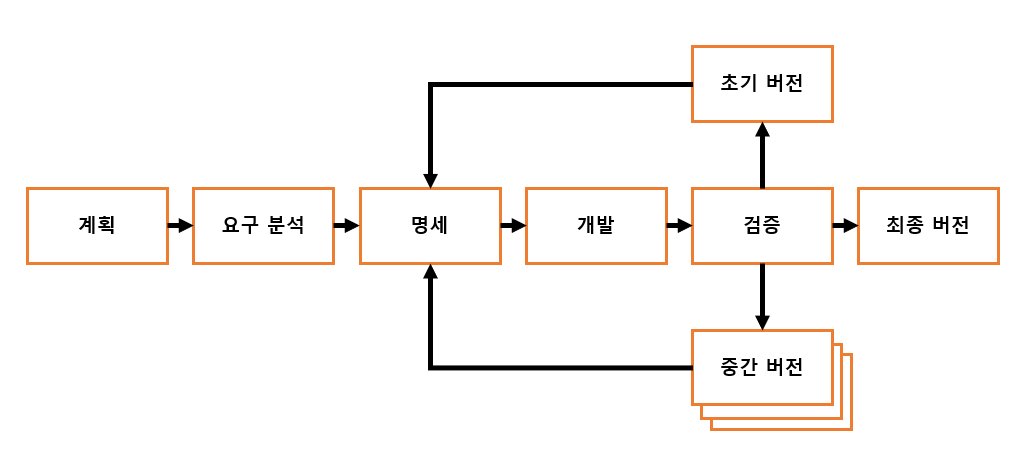
## Operational environment



# Project Development Methodology

## Development Process

< 반복적 진화형 모델 >



## Step-by-step description

1. 계획

* 유사 게임(총알피하기)에 대하여 조사하고 프로젝트 진행 타당성 검토, 프로젝트의 계획 수립 및 계획서를 작성한다.

1. 요구 분석

* 조사한 게임들을 분석하여, 게임을 즐기는 유저들에게 간편하고 쉬운 환경을 제공하여, 가볍게 즐길 수 있도록 제작한다.

1. 명세

* 초안의 설계를 토대로 작성하여 개발하고, 반복적인 버전 업데이트를 통해, 명세를 더욱 더 분명하고 자세하게 작성한다.
* 데이터베이스 : 기능에 필요한 값을 저장하여, 사용자가 보기 편하게 설계한다.
* UI, 기능 모듈 : 사용자가 처음 접하더라도, 한 눈에 이해하여 사용할 수 있는 간편하고 쉬운 디자인과 기능으로 제작한다.

1. 개발

* 앞서 작성한 명세를 토대로 개발을 한다.

1. 검증

* 개발 한 응용 프로그램의 테스트를 진행하여 초기 버전, 중간 버전, 최종 버전의 세가지 중 택한다.

1. 초기 버전

* 첫 번째 테스트 이 후에는 초기 버전으로 설정 후 적합하지 않은 부분에 대하여 명세를 수정한다.

1. 중간 버전

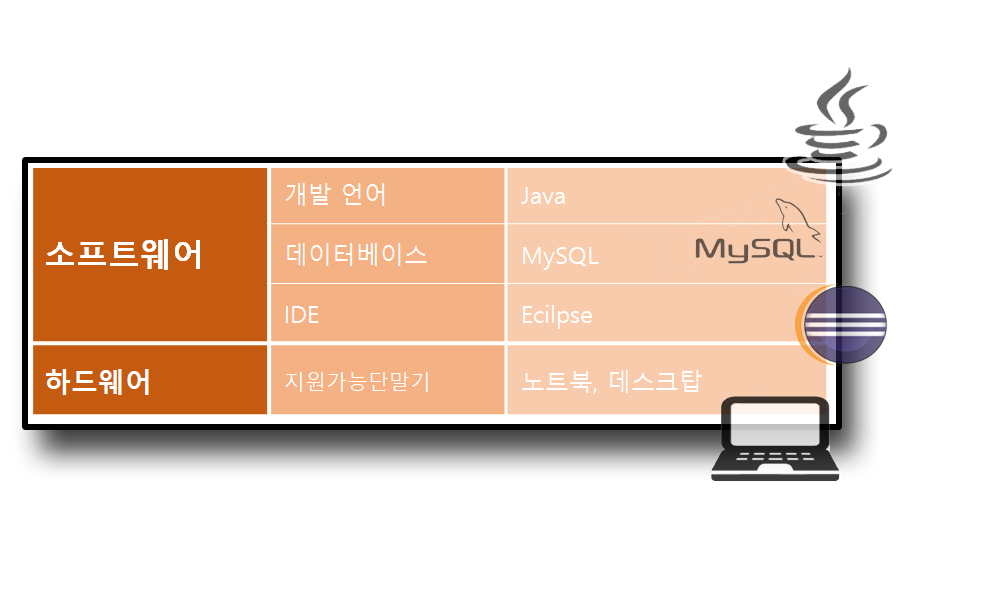
* 초기 버전 이후 여러 번의 테스트를 거쳐 최종 버전이 나오기 전까지 반복적으로 중간 버전을 업데이트하며, 명세를 수정한다.

1. 최종 버전

* 개발 완료된 버전을 출시하여 정식 서비스를 제공한다.

# Project Scope

## Physical Scope



## Logical Scope

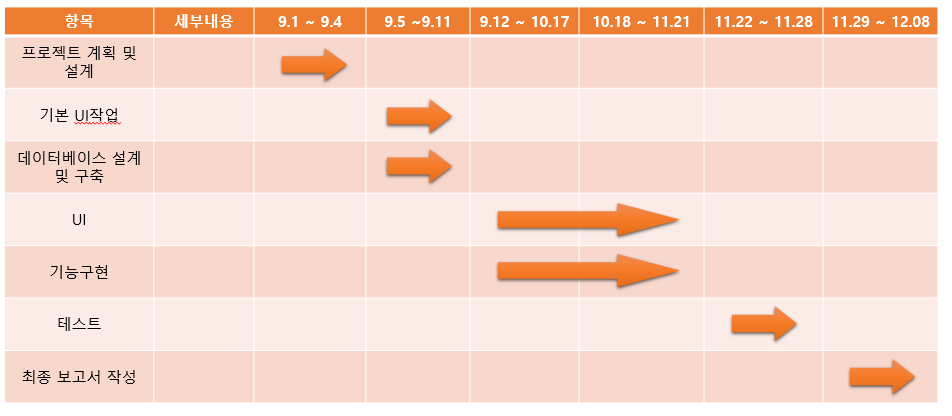


# Schedule

## Members

|  |  |
| --- | --- |
| **Name** | **Role** |
| 박휘준 | 기획, 설계 및 기능 담당을 포함한 프로젝트 총괄 |
| 배현오 | 데이터베이스, 디자인 및 테스팅 담당 |

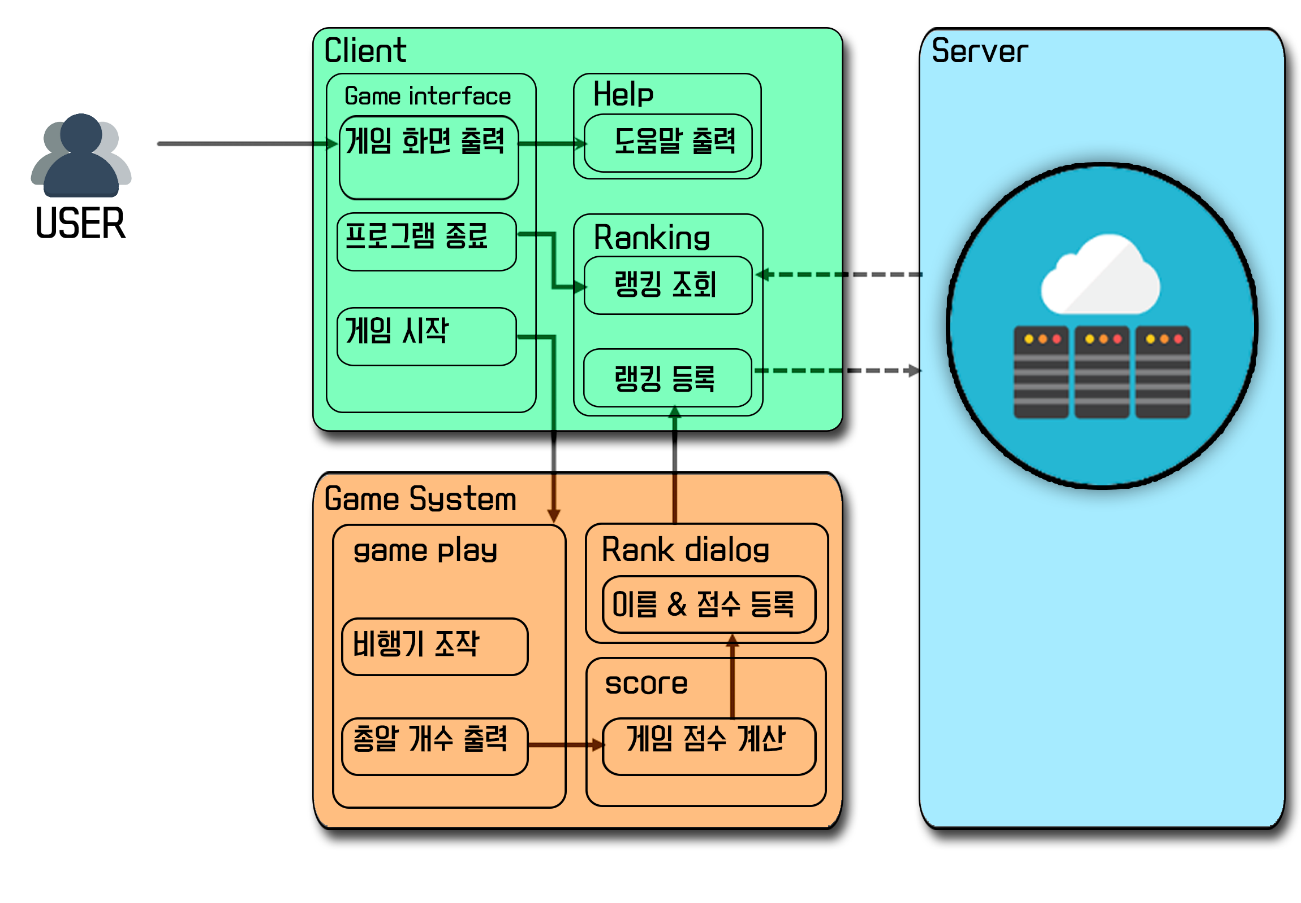
## Project Schedule



1. 계획 : 제작 할 프로그램을 선정하고, 프로젝트에 대해 계획 수립 및 계획서 작성
2. 기본UI작업 : 선정한 프로그램에 대하여 기본적인 UI작업을 통하여 틀을 제작한다.
3. 데이터베이스 설계 및 구축 : 프로그램의 데이터베이스를 설계하고 구축한다.
4. 기능 구현 : 기간을 반으로 나누어 큰 기능을 먼저 구현한 뒤, 부족한 부분을 10.18일 이후부터 구현한다.
5. UI : 전반적 UI를 10/17일 이전에 디자인하고, 구현된 기능 부분에 대한 디자인은 후반부에 작성한다.
6. 테스트 : 일주일의 기간을 두어 주변사람들에게 피드백을 얻거나, 여러 번의 테스트 과정을 통해 프로그램의 부족한 부분을 채운다.
7. 최종 보고서 작성 : 완성된 프로젝트의 결과 보고서를 작성한다.

# Software Architecture

## Subsystem-based architecture



## Subsystem description

1. Client

게임 화면 및 인터페이스들을 출력하여 사용자가 더 직관적으로 주요 서비스 기능들을 제어할 수 있습니다.

이 서비스로 다양한 다이얼로그를 호출하거나 게임을 시작, 혹은 프로그램을 종료할 수 있습니다.

세부 서비스로는 게임 인터페이스와 도움말, 랭킹 기능이 있습니다.

게임 인터페이스는 게임 화면을 출력하거나 메뉴바를 제어할 수 있습니다.

도움말은 게임 진행에 대한 정보가 담긴 다이얼로그를 호출합니다.

랭킹은 서버 database에서 랭킹 데이터를 불러 온 뒤 순차적으로 출력해주는 다이얼로그를 호출합니다.

2. Game System

게임 시스템은 게임을 시작하고 정해진 규칙에 따라 게임을 진행하며 점수를 계산하고 게임오버시 서버에 데이터를 전달합니다.

세부 서비스로는 게임 플레이, 점수, 랭크 다이얼로그가 있습니다.

게임 플레이는 플레이어가 비행기를 조작하거나 화면에 출력된 총알의 개수를 출력하고 그 외 전체적인 게임 진행을 책임집니다.

스코어는 실시간으로 현제 플레이어의 게임 스코어를 계산한 뒤 출력 및 저장합니다.

랭크 다이얼로그는 게임 오버시에만 출력이 되며 사용자의 이름을 입력 받은 뒤 랭킹에 등록시킵니다.

3. Server

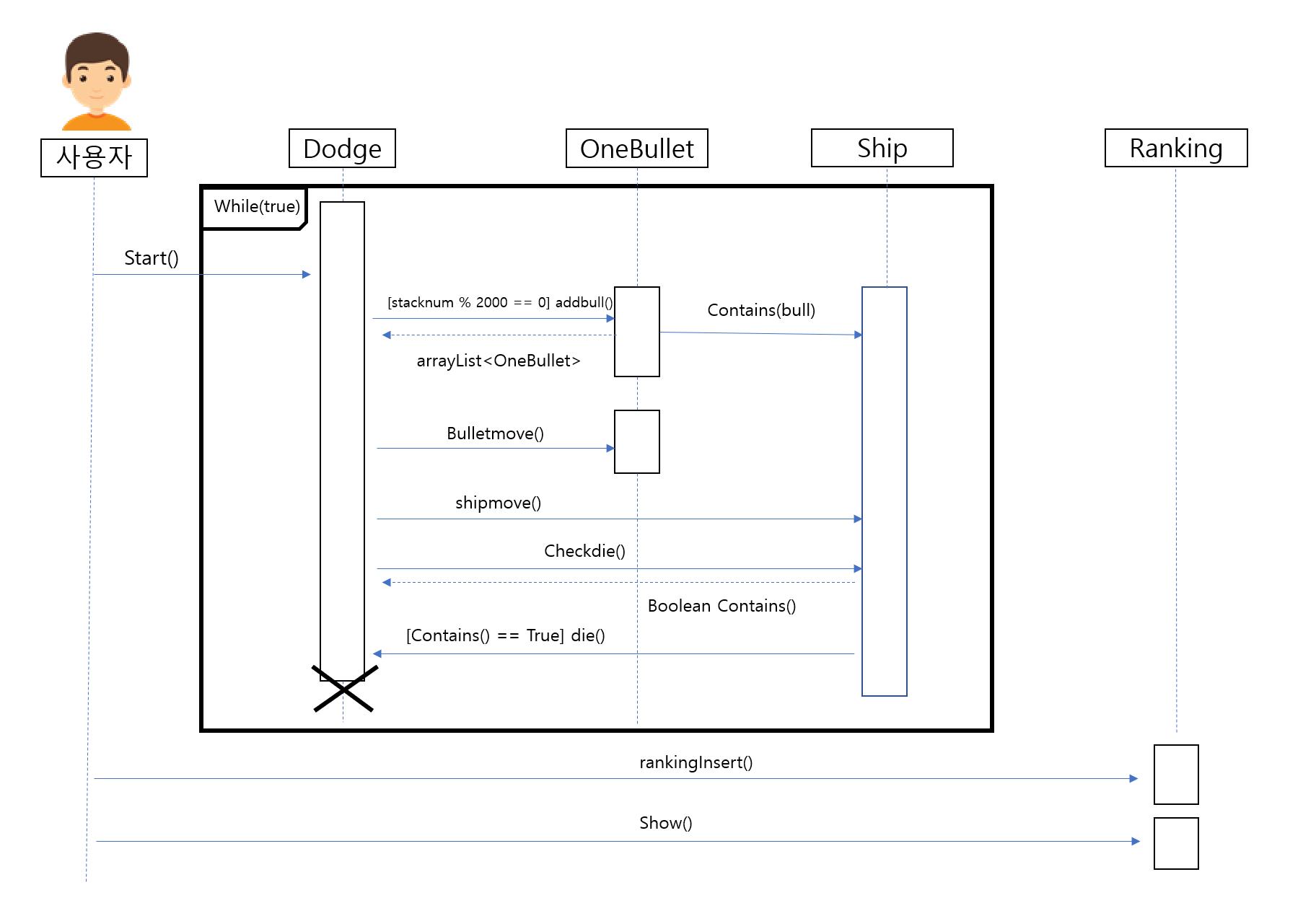
서버는 데이터베이스만을 사용하며 SQL 제어문을 사용하여 자동적으로 내림차순 정렬 후 그 중 10개의 데이터만을 클라이언트에 보내줍니다.

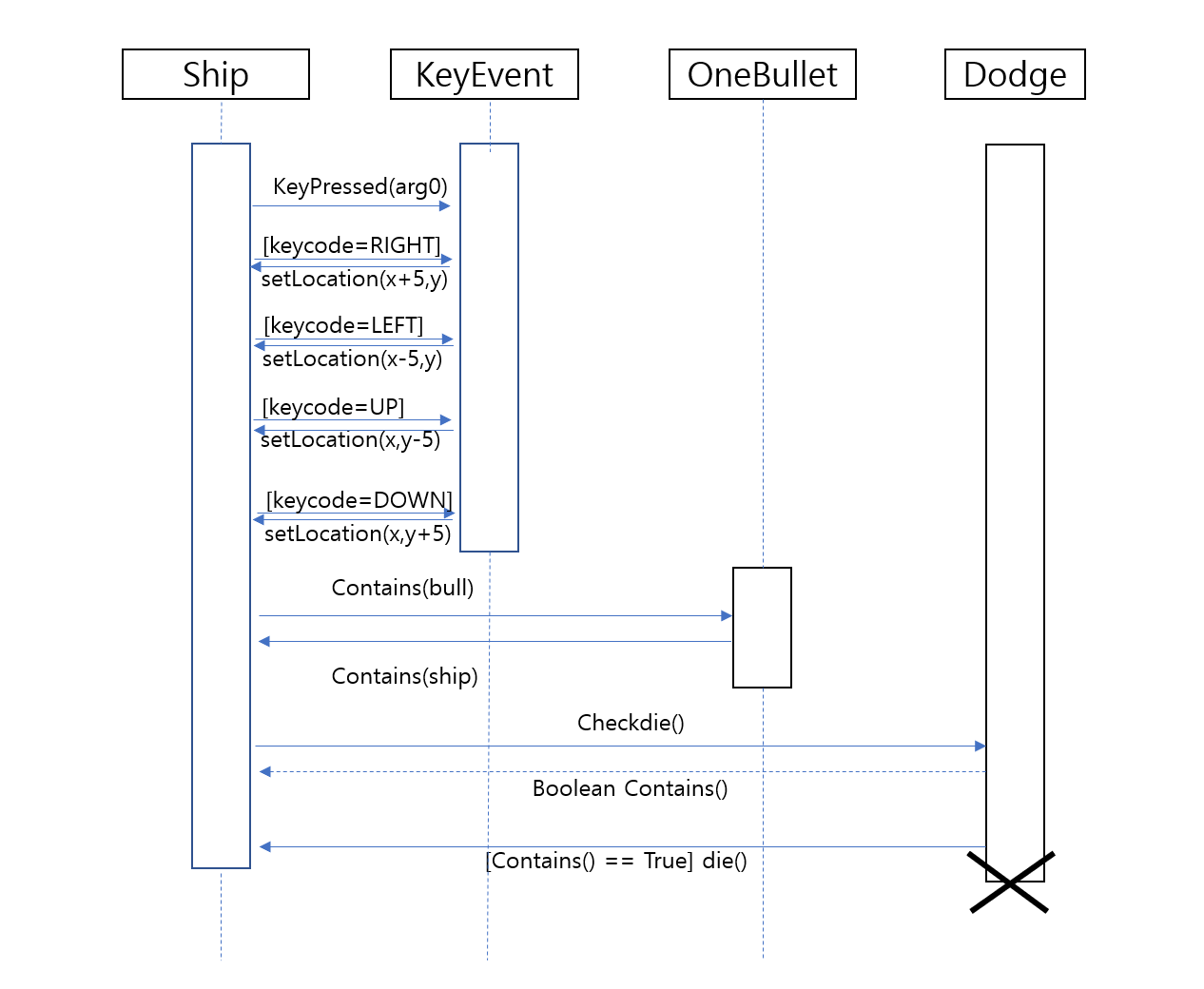
# Diagrams

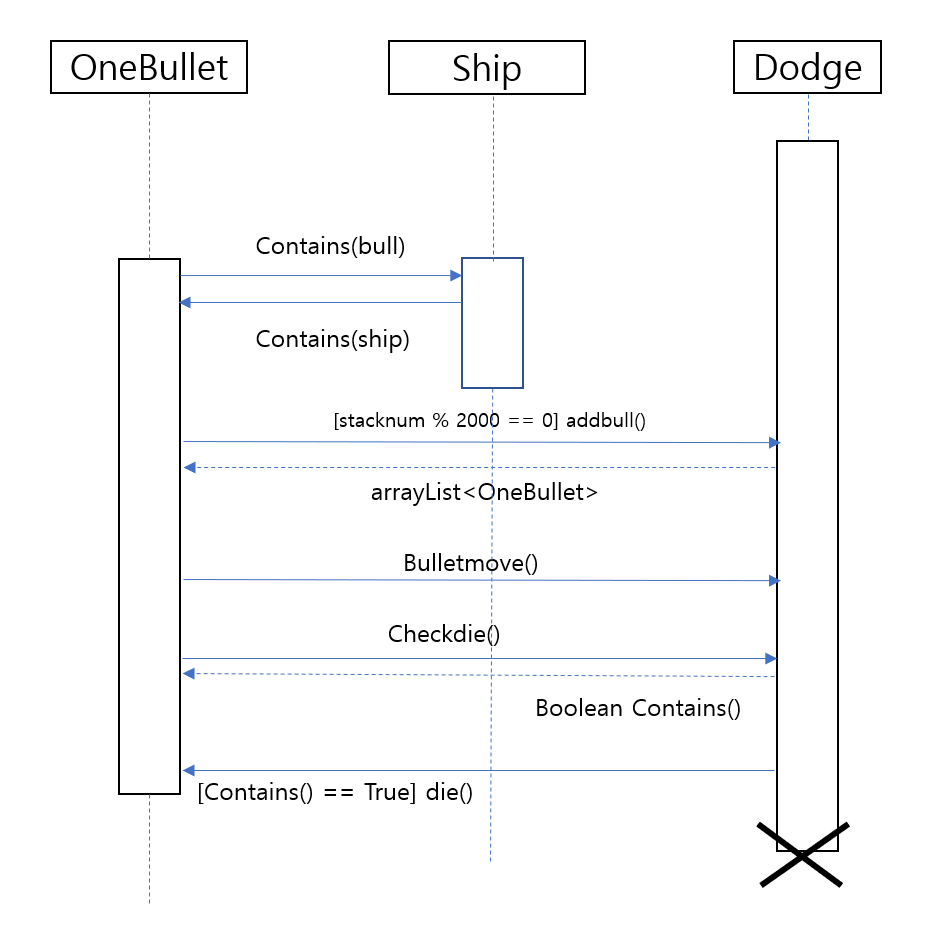
## Use Case diagram

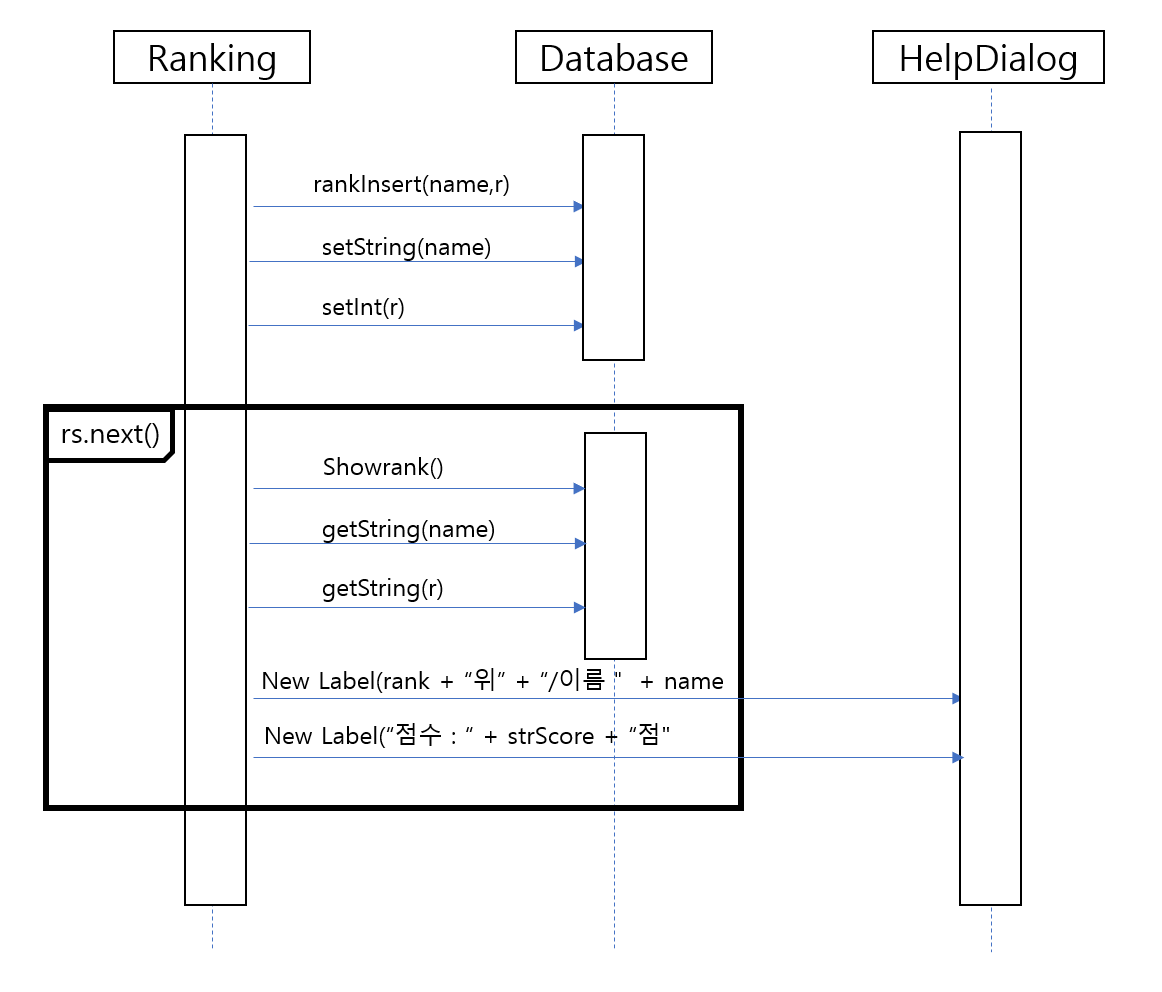
## Class diagram

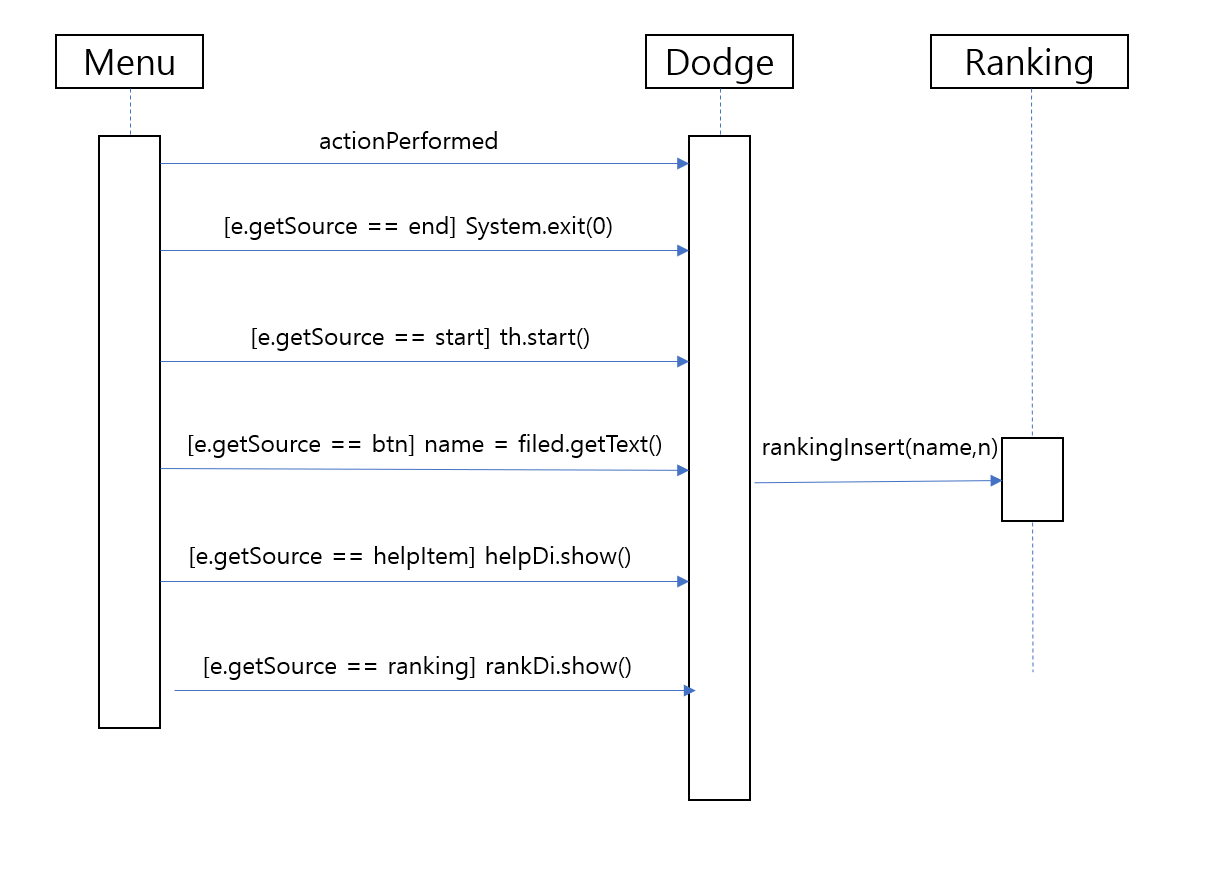
## Sequence diagram

1.사용자

2. 배(조종하는 물체)

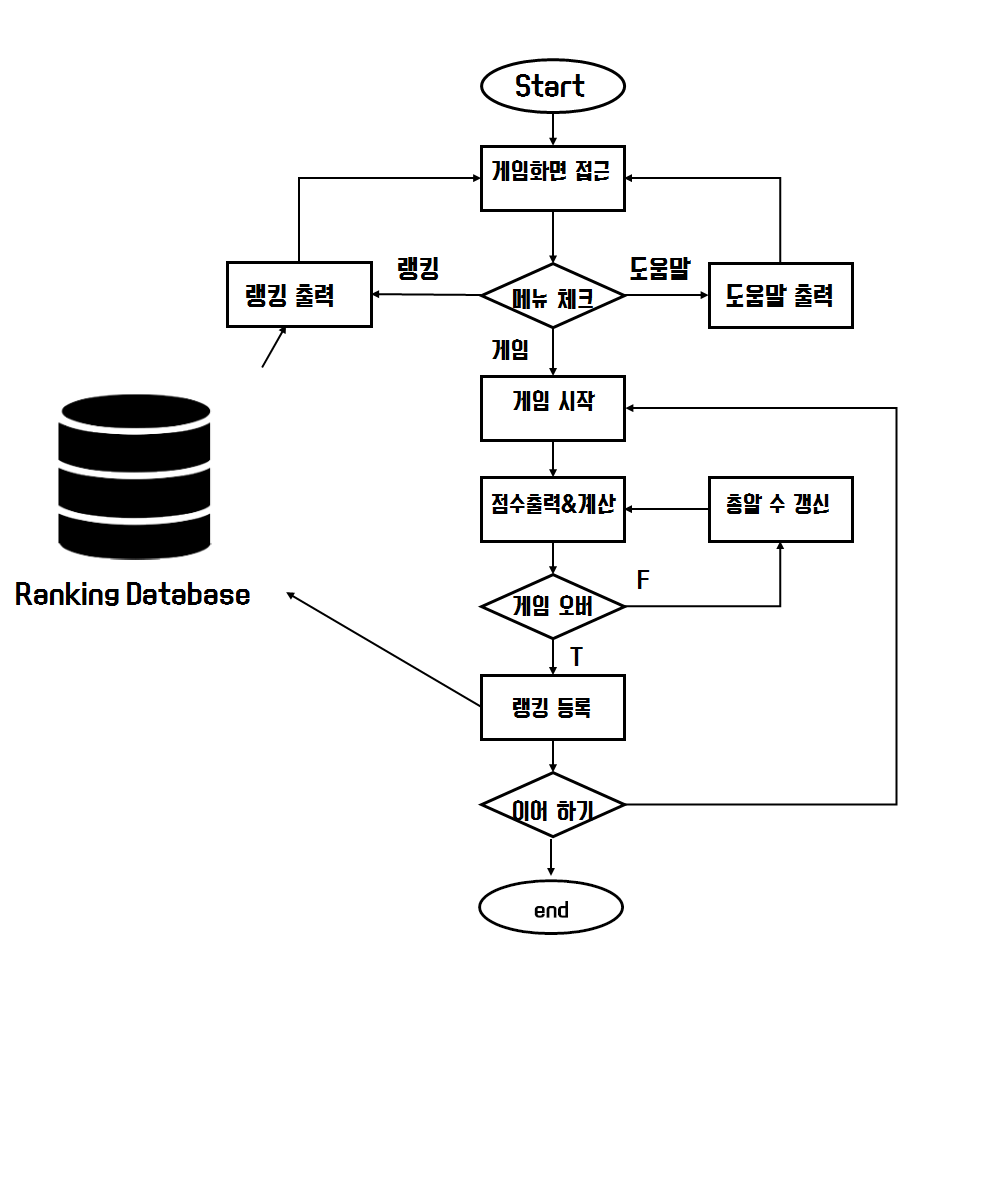
3. 총알

4. 랭킹

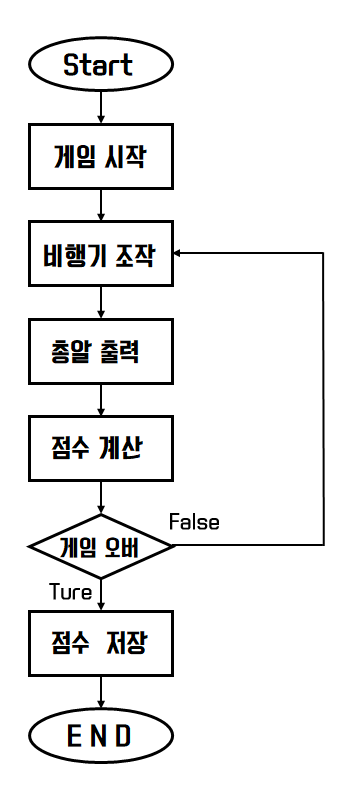
5. 메뉴

## Flow diagram

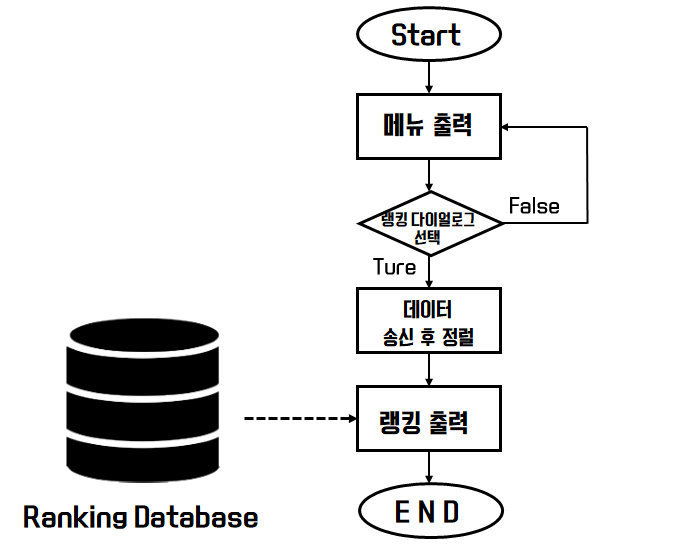
1. 사용자



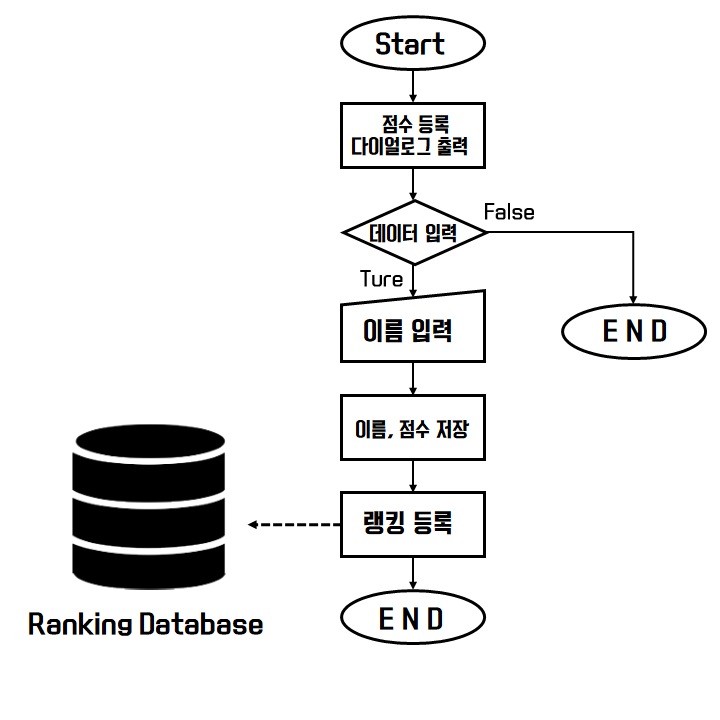
1. 배(비행기)



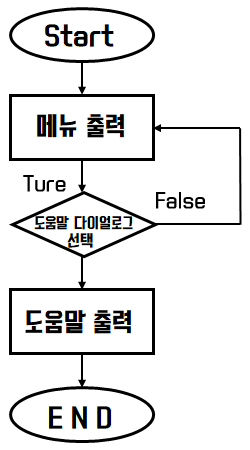
1. 랭킹 다이얼로그



1. 랭킹 입력



1. 도움말



# Evaluation

## Functional Test

|  |  |  |
| --- | --- | --- |
| 항 목 | Y | N |
| 1. 설계가 요구사항 문서에 추적 가능한가? | O |  |
| 1. 설계가 일관되게 기술되었는가? | O |  |
| 1. 설계가 명백하게 기술되었는가? | O |  |
| 1. 인터페이스가 명백하게 정의되었는가? | O |  |
| 1. 모든 에러 메시지가 정의되었는가? | O |  |
| 1. 요구사항의 제약사항이 설계로 통합되었는가? | O |  |
| 1. 모든 의존성이 명백하게 식별되었는가? | O |  |
| 1. 설치 및 마이그레이션 요구사항이 설계로 통합되었는가? | O |  |
| 1. 모든 설계가 최소한 하나 이상의 테스트 활동을 통해서 검증 되었는가? | O |  |
| 1. 모든 설계가 최소한 하나 이상의 테스트 활동을 통해서 검증될 수 있는가? | O |  |
| 1. 에러 복구 및 백업 요구사항이 설계로 통합되었는가? | O |  |
| 1. 설계에 명시된 모든 알고리즘이 명백하게 기술되었는가? | O |  |
| 1. 고객의 성능 요구사항이 테스트 가능하도록 정량화가 되어 있는가? | O |  |
| 1. 개발해야 할 소프트웨어와 다른 소프트웨어 및 하드웨어 사이의 인터페이스가 요구 명세에 적절하고도 완전하게 기술되었나? | O |  |
| 1. 주요 기능적 요구사항에 대해선 계층적으로 세분화되어 기술되어 있는가? | O |  |
| 1. 모든 요구사항이 모델링 결과물과 연결되어 있는가? | O |  |
| 1. 요구사항에 대한 변경관리가 이루어지는가? | O |  |
| 1. 요구사항이 현재 시스템으로 수용가능한가? | O |  |
| 1. 사용자가 도움말을 보고 한번에 이해 할 수 있는가? | O |  |
| 1. 메뉴 인터페이스가 정상적으로 작동하는가? | O |  |
| 1. 총알이 정상적으로 늘어나는 가? | O |  |
| 1. 비행기를 움직이는데 불편함이 없는가? | O |  |
| 1. 랭킹 다이얼로그가 제대로 삽입되는가? | O |  |
| 1. 랭킹 다이얼로그를 보고 한눈에 이해 할 수 있는가? | O |  |
| 1. 총알 닿을 시 제대로 게임이 종료되는가? | O |  |
| 1. 타이머가 제대로 작동하는가? | O |  |
| 1. 랭킹 조회 시 데이터가 정상적으로 출력 및 업데이트 되는가? | O |  |