Design Specification

about OurBot

7조
김 동 현
김 장 훈
이 용 철
위 함 흔

목차

1 Preface

1.1 Objective

Preface에서는 본 문서의 독자를 정의하고, 구조를 소개한다. 각 구조는 목적을 먼저 서술한다. 또한, 문서의 버전을 서술한다.

1.2 Readership

본 문서의 독자는 다음과 같다. software engineers, architects, 그리고 이와 관련된 팀원 및 외부 사람들을 독자로 정의한다.

1.3 Document Structure

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 구조를 소개한다. 각 구조는 목적을 먼저 서술한다. 또한, 문서의 버전이 업데이트할 때마다 신 버전을 서술한다.

B. Introduction

이 목차에서는 본 문서에서의 System design에 대해 사용된 소프트웨어(tools같은)과 다이어그램들을 서술한다.

C. System Architecture

System Architecture에서는 현재 개발하고자 하는 시스템의 전반에 대해 서술한다. 즉, 전체적인 시스템 구조를 Block Diagram, Package diagram, Deployment diagram을 이용하여 나타내고, 아래의 목차 순으로 설명한다.

D. Start / End System

게임 안의 플레이어 Discord 안에서 봇을 초대하고, 게임을 시작하여 발생하는 과정에 대해서 class diagram, sequence diagram, state diagram을 통해 설명한다.

E. Question & Answer System

Q&A System은 질문자와 답변자 두 종류의 참여자가 Discord상에 입력한 text를 각각 질문과 답변으로 해석하고,답변자가 올바른 답변을 하였는지 분석하여 결과를 내는 시스템이다.참여자와 Bot 사이를 Discord가 중개하고 있으며, Bot은 자연어의 해석 작업을 위해 Dialogflow API를 사용한다.

F. Elimination System

한 플레이어가 특정 플레이어를 상대 팀의 왕으로 지목했을 경우 이를 처리해주는 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagram을 통해 본 시스템의 구조를 설명한다.

G. Help & log System

참여자가 요구할 때, Bot에 대한 도움말과 게임 log를 Discord상에 출력해주는 시스템을 설명한다. Help & Log System의 구조를 표현하기 위해, class diagram, sequence diagram, state diagram을 사용한다.

H. Protocol Design

Protocol Design에서는 서브시스템들이 상호작용하는 프로토콜에 대해 서술한다.

I. Database Design

DB Design은 사용된 DB의 구조 정의를 서술한다.

J. Testing Plan

각 서브 시스템이 예상된 실행을 보여주는지를 테스팅하여, 결함을 찾기 위한 testing plan을 서술한다.

K. Development Environment

시스템을 개발할 때에 필요한 것들을 서술한다. programming language, IDE 등을 서술한다.

L. Develop Plan

개발 계획에 대해 서술한다.

M.Index

Index는 본 문서에서 보여준 Figure, table, diagram들의 순서를 표시하여 각 index의 내용이 본 문서 안에서의 위치를 알 수 있다.

1.4 Version of the Document

A. Version Format

version 번호는 Major와 Minor number(format : {major, minor)의 형태로 표현한다. 본 문서의 버전은 0.1부터 시작한다.

B. Version Management Policy

설계명세서를 수정할 때 마다 버전을 업데이트한다. 다만, 변경 시간 차가 짧을 경우 버전 번호를 업데이트하지 않고, 하나의 버전으로 취급한다. 이미 완성된 부분을 변경할 시, Minor number를 증가시키며, 새로운 부분을 추가 및 문서의 구성 변경 시 Major number를 변경한다.

C. Version Update History

Version	Explanation
0.1	설계 명세서 문서의 목차 생성
1.0	Preface와 Introduction 부분 작성
2.0	Start & End System 작성.
3.0	System Architecture, QnA System, Elimination System, Help&Log system, DB management System 작성.
4.0	Testing Plan, Development Environment 작성.
5.0	Procotol Disign을 제외한 거의 모든 부분 작성
5.1	Develop plan 작성.

Table 1 Version Update History

2 Introduction

2.1 Objectives

Introduction은 본 문서에서 기술될 system Design을 위해 사용된 diagram 및 tool 들에 대해 서술한다.

2.2 Applied Diagram

A. UML

Figure 1 UML Logo

United Modeling Language(UML)은 실용적인 여러 종류의 다이어그램들을 통합하여 만들어진 모델링 언어로 object-oriented software 개발할 때 산출 물을 명세화, 시각화, 문서화할 때 사용한다. 표기 방법을 사용하여 개발하고자 하는 소프트웨어의 구조적 청사진을 시각화하여 의사소통의 도구로 사용된다.

우리의 프로젝트에서 UML을 사용하여 우리의 프로젝트를 시각화하여 우리의 시스템과 솔루션을 명세한다.

A.1 Package Diagram

개발할 소프트웨어을 이해하기 위한 목적으로 abstract한 component들을 그룹 단위로 organization하기 위한 메커니즘이다. 이 매커니즘을 통해 전체 시스템의 abstraction을 이해를 돕기 위해 사용된다. package안 의 component로는 package, class, interface, object 등을 포함할 수 있다. package diagram을 사용할 때에는 구성원 모두가 이해할 수 있고, 사용할 수 있어야 한다.

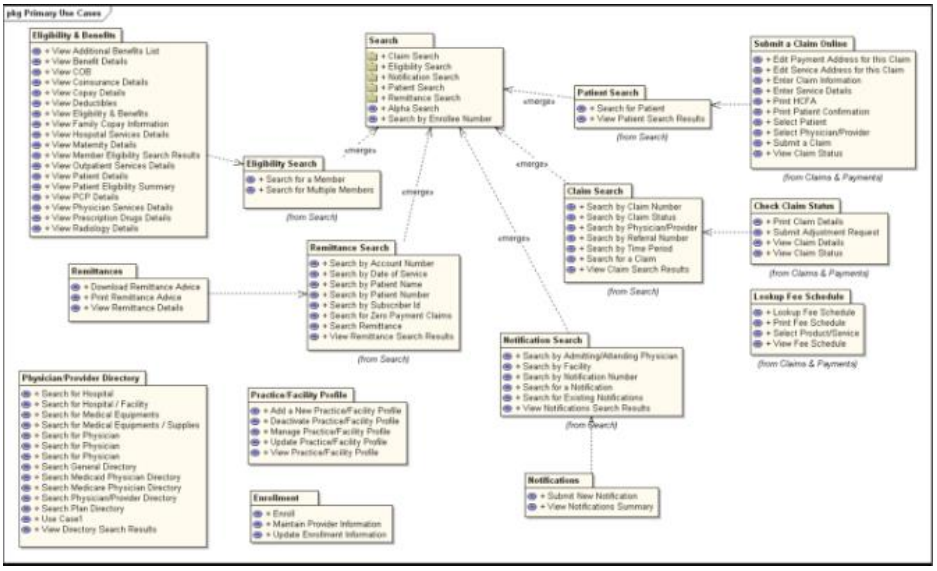

Figure 2 Package Diagram 예시

A.2 Deployment Diagram

system을 구성하는 Hardware resources 간의 relationship를 표현하고, component의 deployment state를 표현하기 위한 다이어그램이다. 즉, 소프트웨어 시스템이 어디에 배치되어 실행되는지 대한 HW들을 정의한다. 예를 들어, 웹 서버, 응용 프로그램 서버 및 데이터베이스 서버와 같은 하드웨어 구성 요소가 어떤 소프트웨어 구성 요소, 응용 프로그램, 데이터베이스 및 다른 부분이 어떻게 연결되어 있는지를 알려준다.

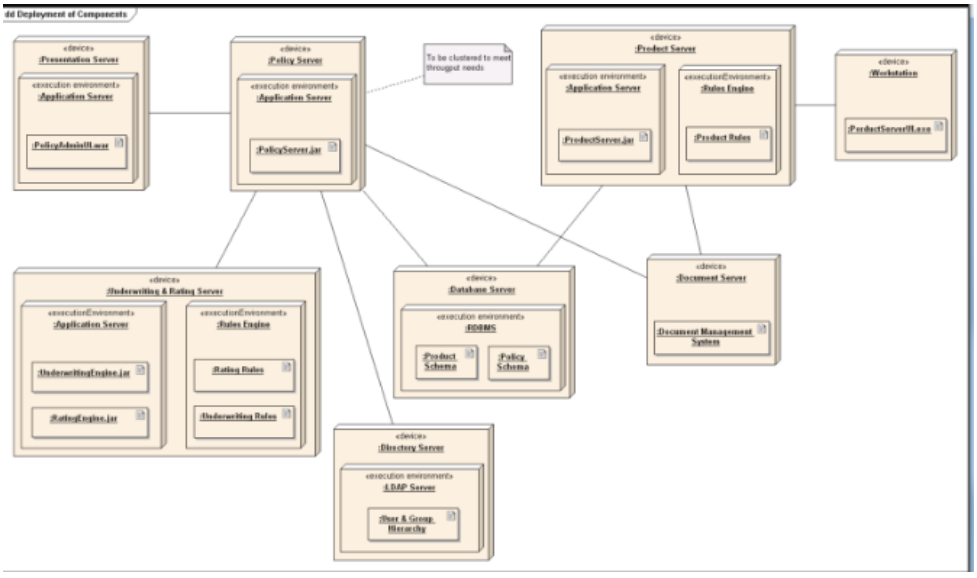

Figure 3 deployment Diagram 예시

A.3 Class Diagram

Class Diagram은 object-oriented design시에 자주 사용되는 다이어그램으

로, 시스템 내부 구조, 즉 클래스들을 나타내는데, 각 클래스는 attribute와 action을 포함하고, 실질적으로 객체의 타입을 정의. 클래스들 간의 관계를 다이어그램으로, 클래스 내부의 내용, 클래스들 사이의 관계(Inheritance, polymorphism)를 표기하는 방법으로 의존관계를 쉽게 보여준다. 이 클래스 다이어그램을 명확히 할 수 있다면, 실질적 코드를 구현함에 있어서 편리해진다.

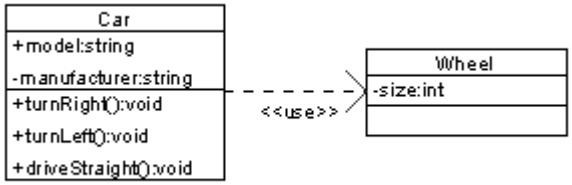

Figure 4 Slass Disgram 예시

A.4 State Diagram

State Diagram은 객체지향 모델에서 클래스의 Event에 의거한 시스템의 전체적인 작동을 상세히 기술하는 다이어그램이다. 즉, 어떤 상태에서 특정 특정 조건을 만족 했을 때, 이에 따른 한 상태에서 다른 상태로의 변화, 즉 상태 변화를 간단히 보여 주는 다이어그램이다. 이것은 interface를 정의하는데 도움을 줄 수 있다.

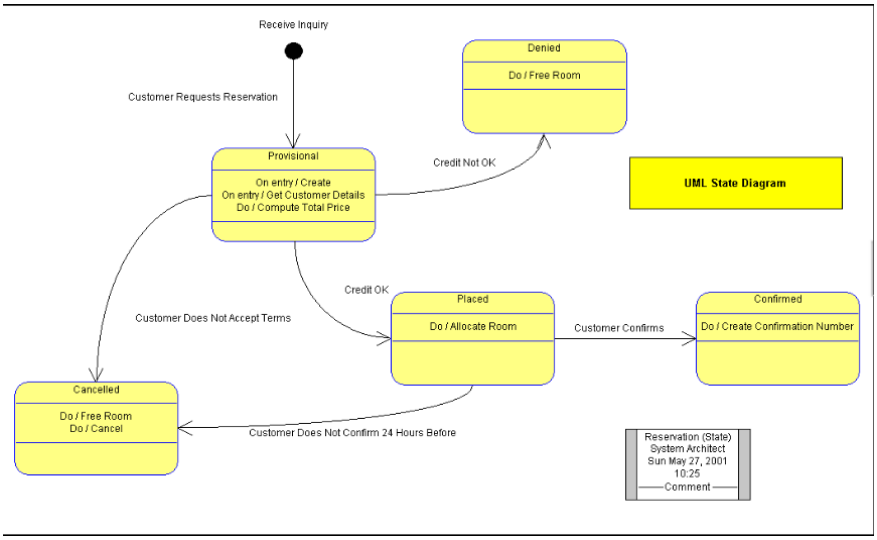

Figure 5 State Diagram 예시

A.5 Sequence Diagram

시스템 내에서의 각 component들간에 주고 받는 메시지의 흐름을 시간이 지남에 따라 표현하는 상호작용 다이어그램이다.주로 use case와 함께 표현되어 자주 사용되는데, use-case에서 간략화된 diagram를 시간의 흐름

으로 명세하여 명확히 볼 수 있게 해주기 때문이다. 이 다이어그램은 event의 발생 순서에 따른 sequence를 보여주기 때문에 동적 측면을 모델링 하기 위한 용도로도 사용된다.

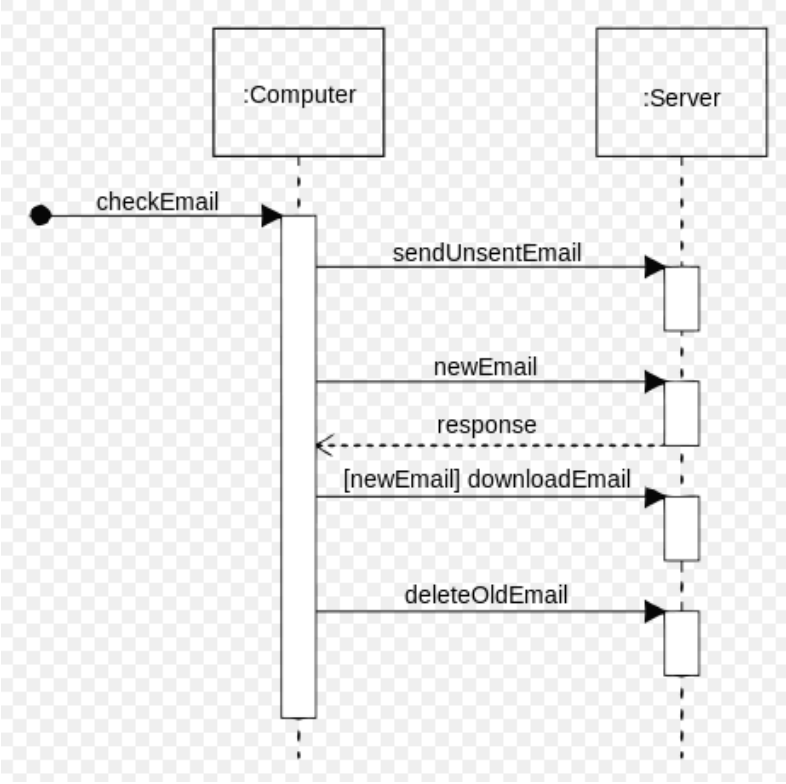

Figure 6 Sequence Diagram 예시

B. ER Diagram

데이터베이스에서 각 개체들(Entities)의 관계를 표현하기 위해 사용하는 다이어그램으로, DB에 저장될 데이터의 형식 정의와 제약조건(Constraints)들의 정의하기 위해서 사용되며, system을 운영함에 있어서 주고 받는 혹은 저장할 필요가 있는 Data Resource들을 어떠한 관계를 갖고 운영되는지를 계획하는데 사용된다.

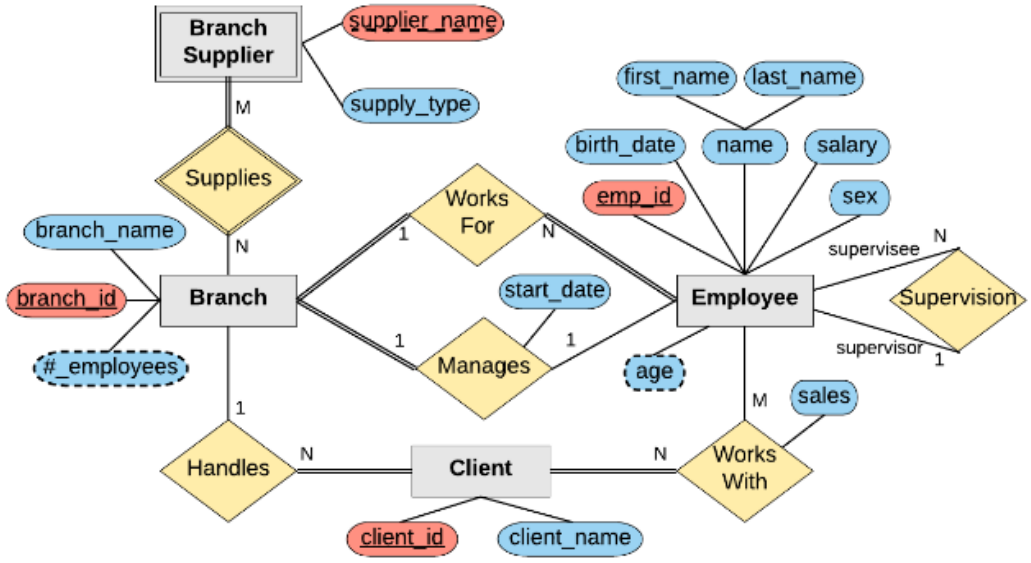

Figure 7 ER Diagram 예시

2.3 Applied Tool

A. Flow Chart Maker & Online Diagram Software

본 문서에서 시스템 설계와 관련된 다이어그램들의 작성은 'Flow Chart Maker & Online Diagram Software' tool을 사용하였는데, 이 tool은 다양한 UML에 말고도 다양한 diagram들을 지원하고, 컴퓨터 내에 설치 회원가입을 하지 않고도 online 상에서 이용할 수 있기 때문에 사용됐다.

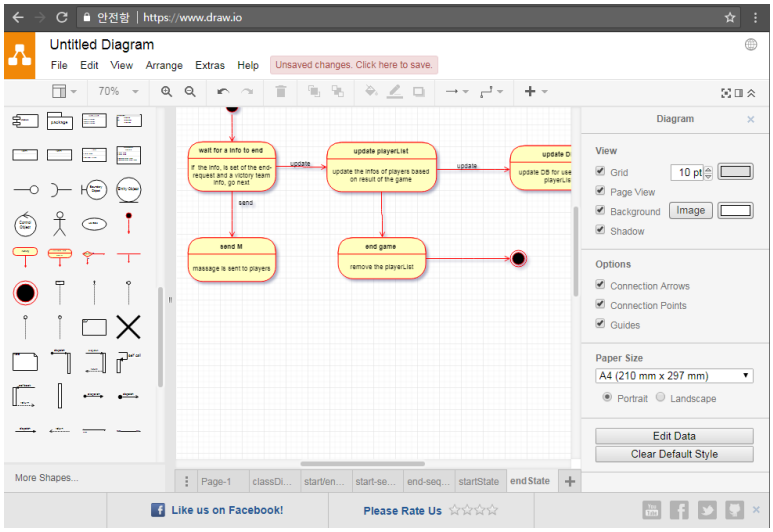

Figure 8 draw.io 화면

B. Amazon Web Services(AWS)

Amazon.com에서 제공하는 Web Service이다. 다른 웹 사이트나 클라이언트, 응용 프로그램에 대해 온라인 서비스를 제공하고 있다. 특정 시간 동안 무료로 사용할 수 있는데, 우리는 여기서 Cloud Server Hosting Service를 이용할 것이다.

Figure 9 Amazon WebServices Logo

C. DialogFlow

DialogFlow는 구글에서 제공하는 코딩 없이 음성 / 텍스트 기반 서비스. 즉, 자동응답봇과 같은 것을 제작할 수 있는 플랫폼을 말한다. 여기서는 여러 가지 연동방법을 제공하고 있다.

Figure 10 Dialog flow Logo

D. Eclipse

본 프로젝트 과정에서 사용될 개발 도구이다. 본 프로젝트는 Node.js 플랫폼을 이용할 것이기 때문에, 자바스크립트 프로그램을 위해 이 도구를 이용한다. 을 이용할 예정이기에 이 도구를 사용하게 되었다.

E. Discord& Discord API

Discord는 게이밍 공동체를 위해 개발된 VoIP 응용 소프트웨어이다. 여러 OS에서도 별 문제 없이 이용되고, 이 소프트웨어는 메신저로의 기능, 정보 공유, 관리 기

능 등을 제공하고 있어 여러 게임 유저들이 이용하고 있다.

Discord API를 이용하여 Discord에 상에서 실행될 봇과 Discord의 상호작용을 위한 API를 제공한다.

3 System Architecture

3.1 Objectives

System Architecture에서는 현재 우리 팀에서 개발하고자 하는 시스템에 대해 전반적인 구조를 서술한다. 이 서술 과정에서 Block Diagram으로 나타내고, 실제 어떻게 사용되는 지를 Package diagram과 deployment diagram으로 사용하여 설명한다.

3.2 System Organization

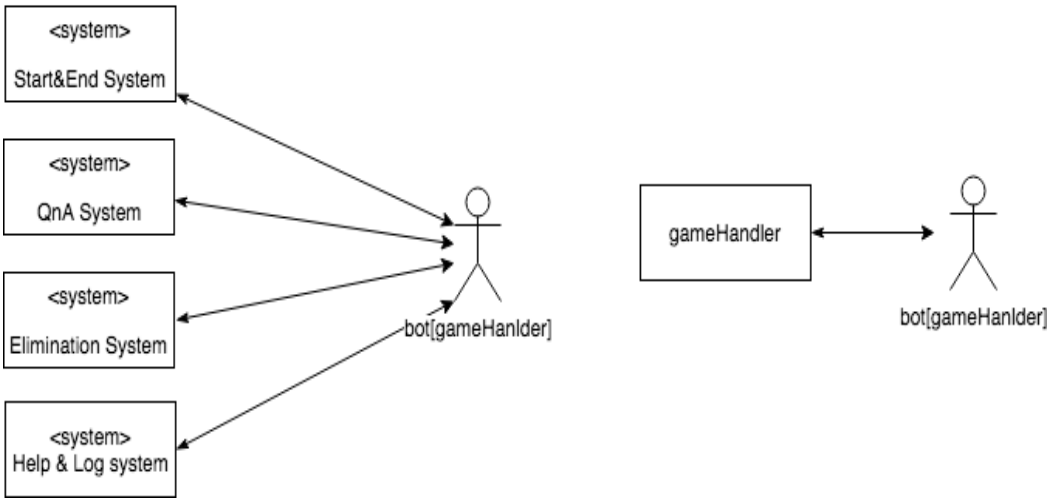

Diagram 1 Our Bot System Block Diagram

우리의 봇 시스템은 Client-server Model을 사용하여 구현된다. 우리가 구현할 게임 봇 시스템에서는 게임 중에 플레이어가 필요한 요구가 있을 때 명령어를 통하여 하위 시스템의 서비스를 제공을 한다.

A. Start & End System

Start & End System의 경우, 간단히 말하면, Discord 상에서 봇을 초대한 후에 게임의 시작을 위한 초기화 기능, 게임의 end 조건을 충족하여, 그 결과에 대해 알리

고 게임 과정에서 생성된 data information 정보를 DB에 저장하거나 갱신 요청하고 게임에 사용된 임시 데이터를 제거하는데 기능을 갖는다. 즉, 클라이언트의 특정 명령을 통하여 각 시스템에서 혹은 게임 Handler에서 제공을 한다.

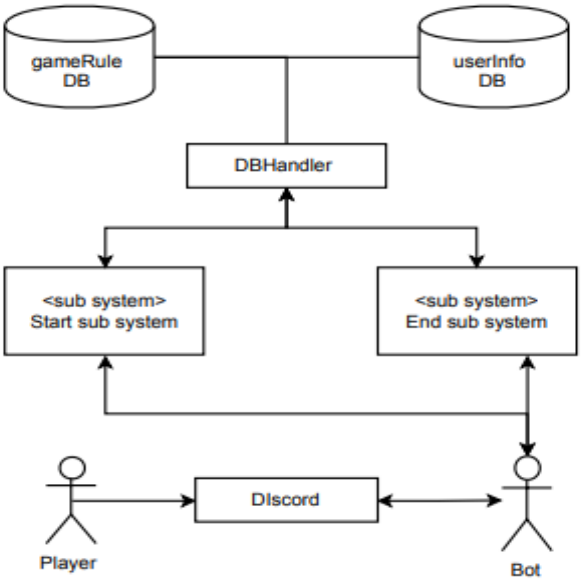

Diagram 2 Start & End system Architecture

B. Question & Answer System

Q&A System은 질문자와 답변자 두 종류의 참여자가 Discord상에 입력한 text를 각각 질문과 답변으로 해석하고,답변자가 올바른 답변을 하였는지 분석하여 결과를 내는 시스템이다.참여자와 Bot 사이를 Discord가 중개하고 있으며, Bot은 자연어의 해석 작업을 위해 Dialogflow API를 사용한다.

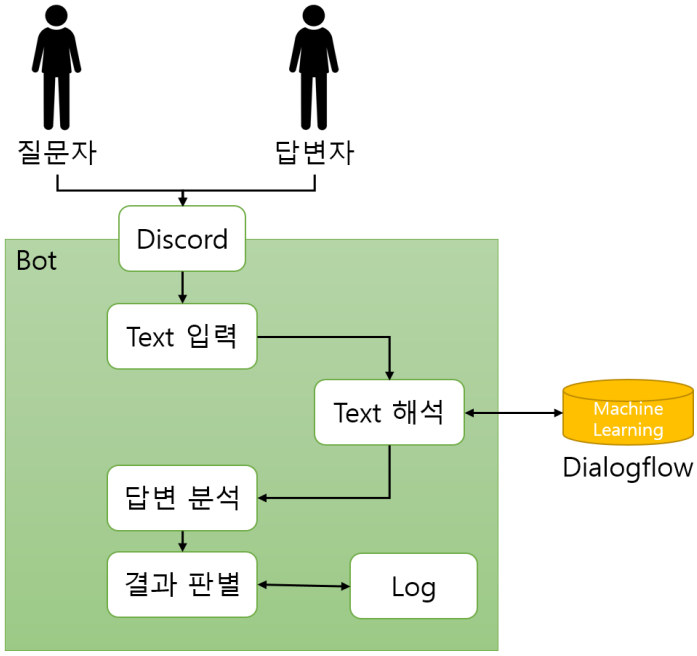

Diagram 3 Question & Answer System Architecture

C. Elimination System

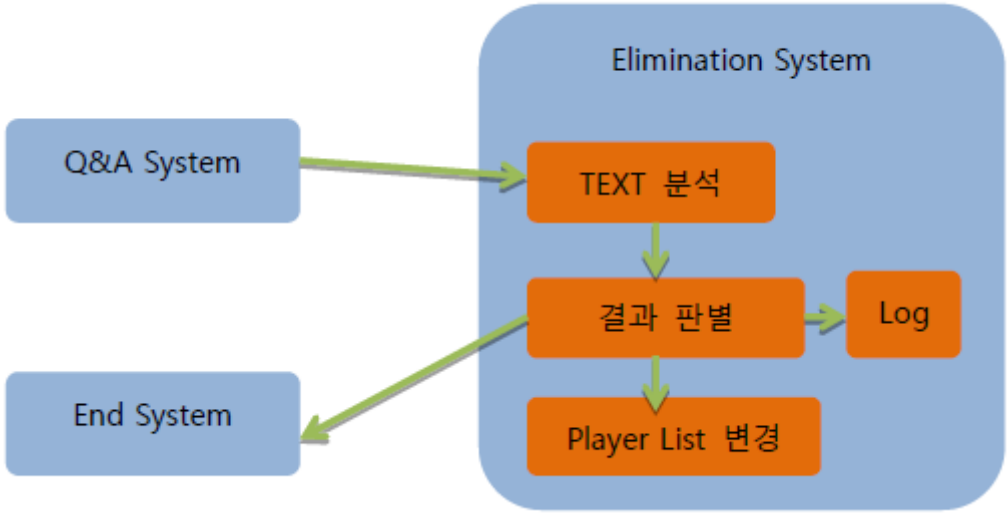

Diagram 4 Elimination System Architecture

D. Help & log System

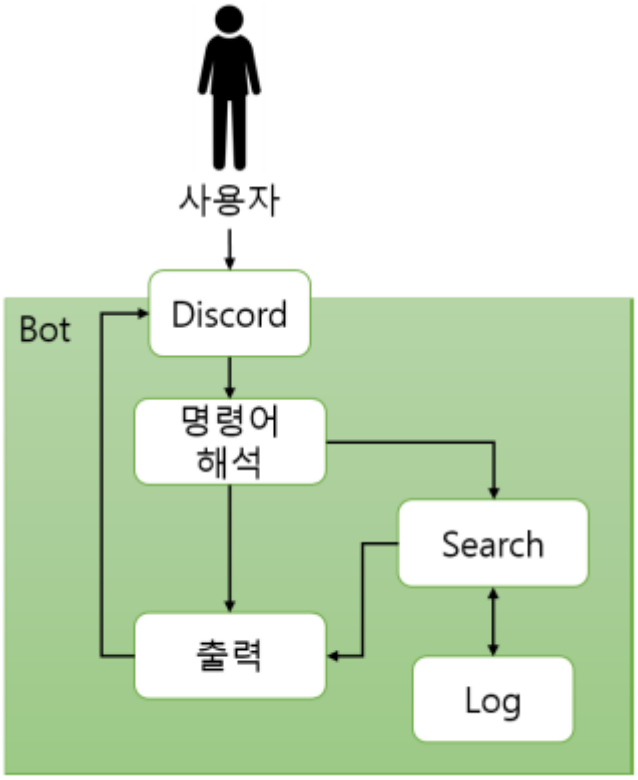

Diagram 5 Help & Log System Architecture

E. DB Management System

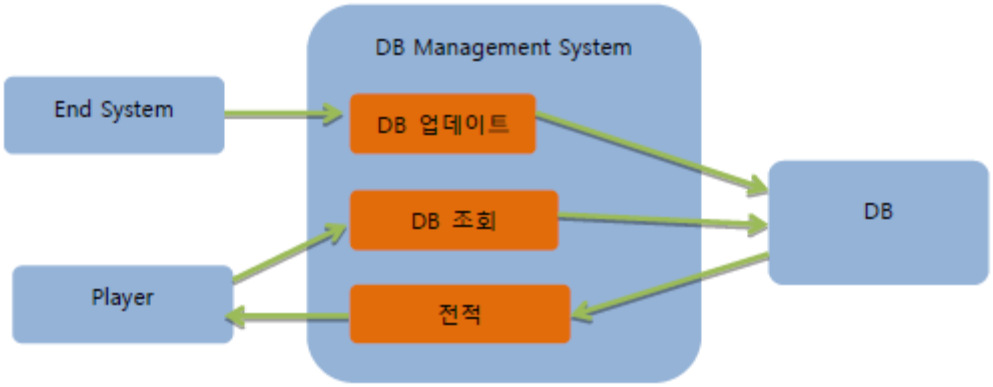

Diagram 6 DB Management System Architecture

4 Start & End System

4.1 Objectives

User가 Discord 상에서 우리의 봇을 초대하고 게임을 하기 위해 봇(game Handler)를 통해 initialization 기능을 통하여 게임을 시작을 진행하고, 게임이 끝나기 위한 조건이 충족하여 받은 정보를 토대로 게임을 종료하기 위한 기능들에 대한 설계를 설명한다. class diagram, sequence diagram, state diagram을 통해 Start & End system의 구조를 표현하고 설명한다.

4.2 Class Diagram

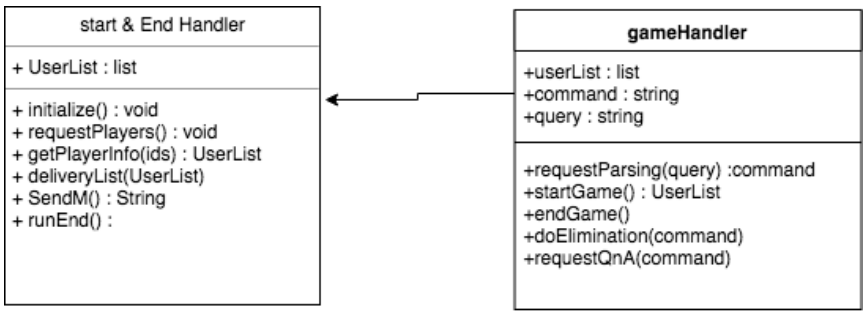

Diagram 7 Start & End Class Diagram

A. gameHandler

A.1 Attributes

+ userList : 게임에 참가할 플레이어들 대한 정보 list

+ command : 게임의 채팅에서 문장을 parsing하여 생성된 명령어 정보

+ query : 게임 내에서 발생된 유저들의 문장들

A.2 Methods

+ requestParsing(query) : query들을 parsing 요청하여 command을 얻는다.

+ startGame() : 게임을 시작을 하기하고 플레이어들의 정보를 얻는다.

+ endGame() : 게임을 종료하고 DB를 갱신한다.

+ doElimination (command) : Elimination을 실행한다.

+ requestQnA(command) : 질의해석을 실행한다.

B. Start & Ed Handler

B.1 Attributes

+userList : 게임에 참가할 플레이어들 대한 정보 list

B.2 Methods

+ initialize() : 게임 규칙 등을 가져오는 등 작업을 한다.

+ requestPlayers() : 참여할 플레이어 정보를 get하기 위함

+ getPlayerInfo(ids) : DB에 저장된 플레이어정보를 얻는다.

+ deliveryList(userList) : list를 gameHandler에게 전달.

+ SendM() : 게임의 시작 및 종료 할 때 발생할 메시지를 전달한다.

+ runEnd() : 게임이 끝난 후 얻어진 정보들을 DB에 저장하고 게임 종료한다.

4.3 Sequence Diagram

여기서 Bot actor는 handler이다.

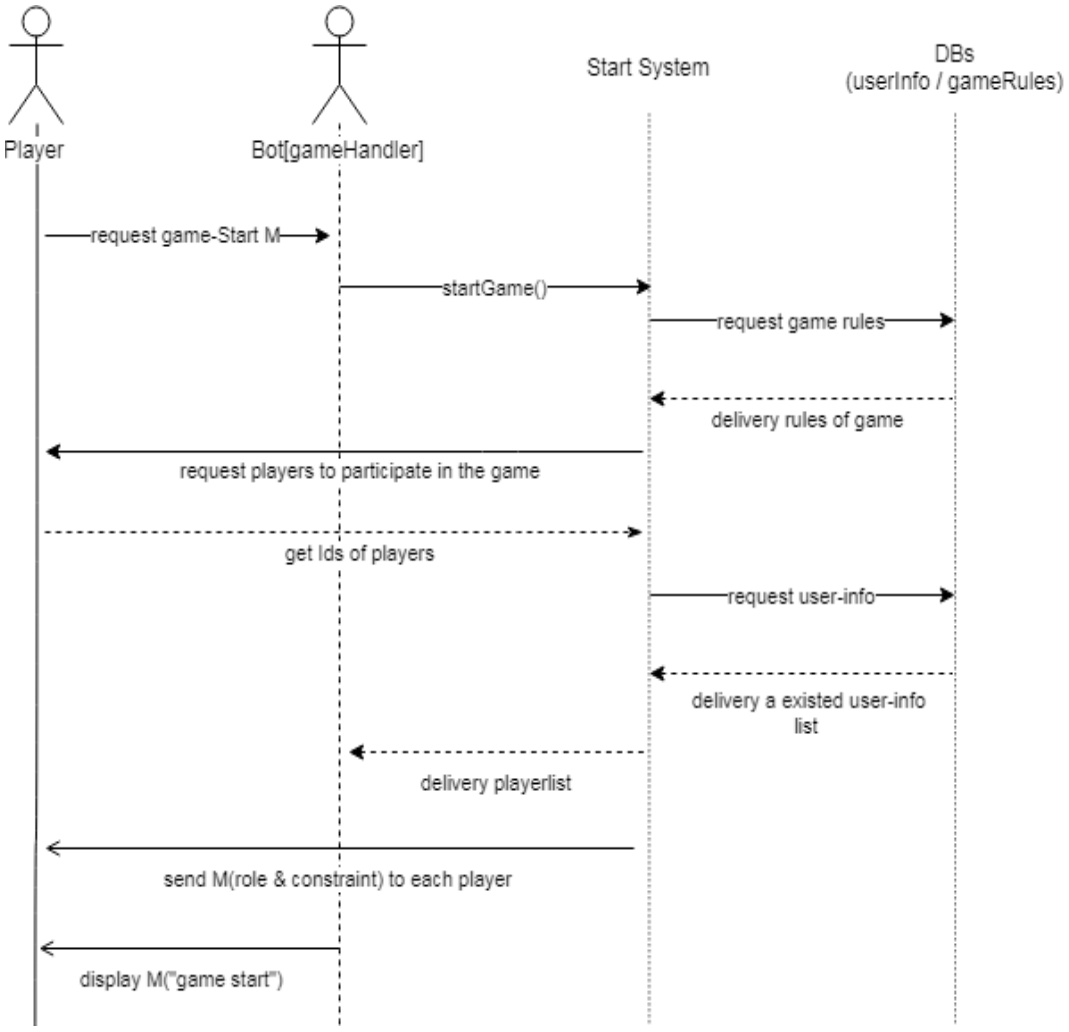

Diagram 8 Start Sub-System sequence diagram

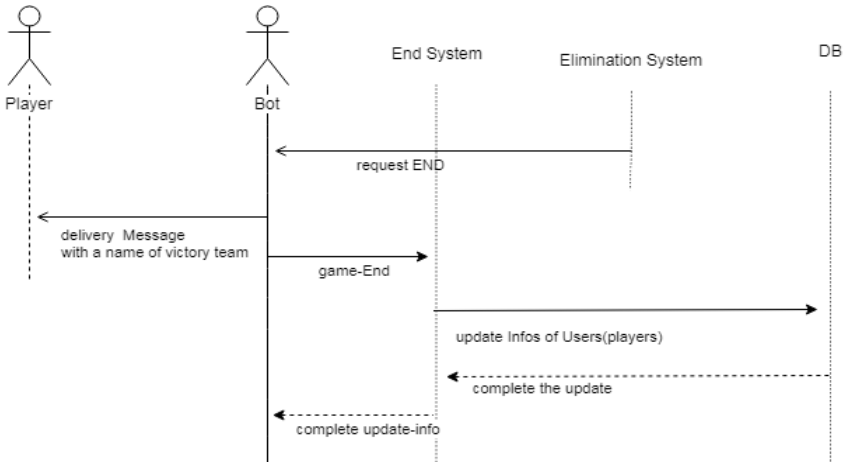

Diagram 9 End sub-System sequence diagram

4.4 State Diagram

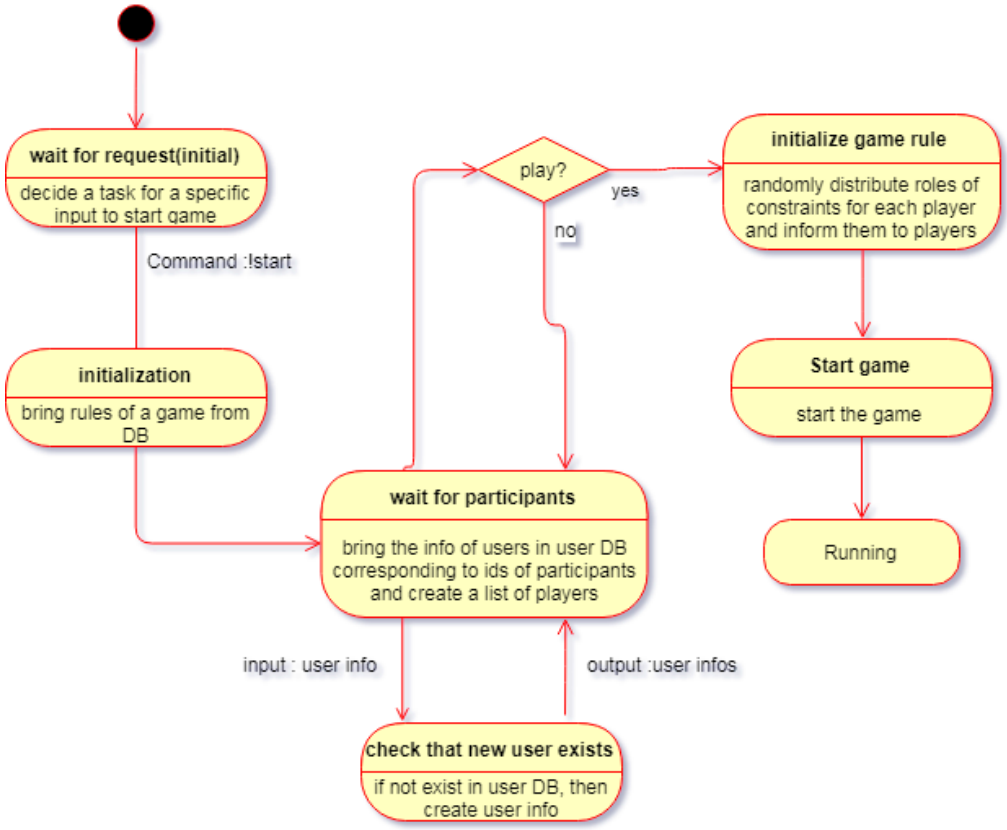

Diagram 10 Start Sub-system State Diagram

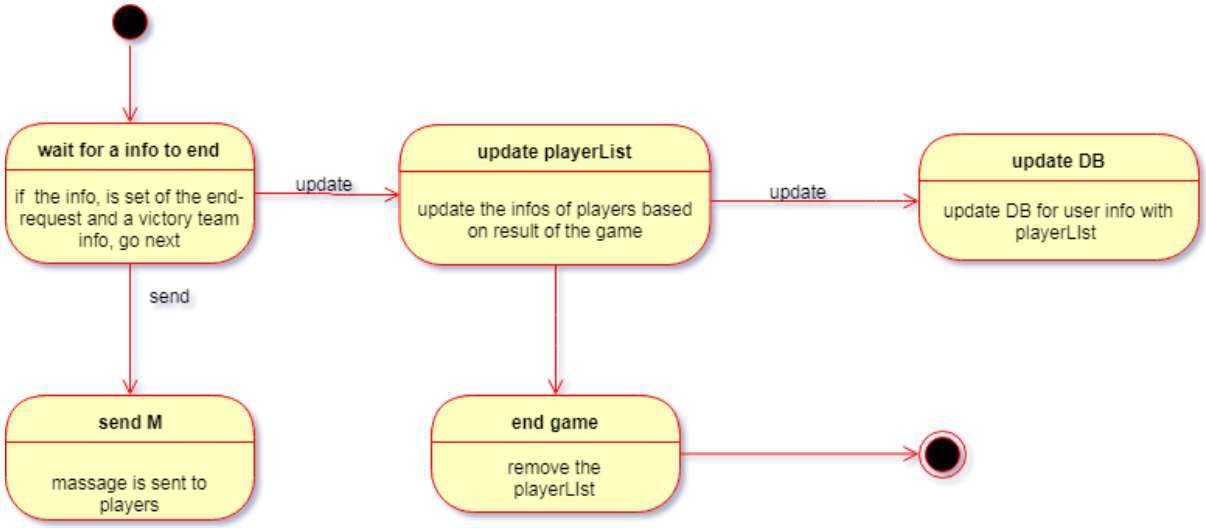

Diagram 11 End Sub-system State Diagram

5 Question & Answer System

5.1 Objectives

참여자가 Discord 상에 입력한 질문과 답변들을 해석하고 처리해주는 시스템을 설명한다. Q&A System의 구조를 표현하기 위해 Class Diagram, Sequence Diagram, State Diagram을 사용한다.

5.2 Class Diagram

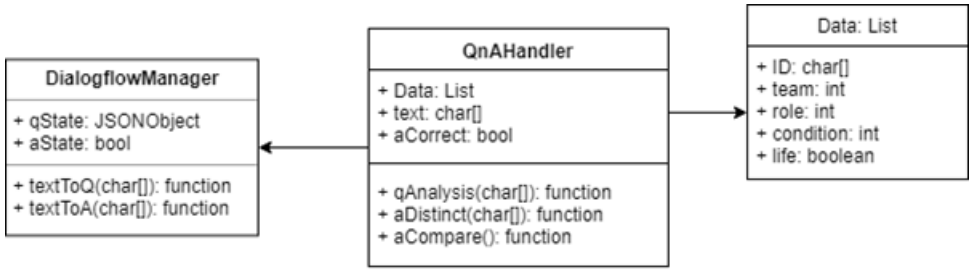

Diagram 12 QnA System Class Diagram.

A. QnAHandler

A.1 Attributes

+Data: 참여자들의 ID, 팀,역할,제약조건 정보

+text: 참여자가 Discord를 통해 입력한 문장

+aCorrect: 답변자가 해야하는 올바른 대답

A.2 Methods

+qAnalysis(char[]): 질문 text를 JSON 형태로 번역한 뒤,답변자의 개인정보를 토대로 올바른 대답을 계산한다.

+aDistinct(char[]): 답변 text를 Boolean형태로 번역한다.

+aCompare(): 올바른 대답과 실제 대답을 비교하고, 결과에 따라서 EliminationHandler에 접근한다.

B. DialogflowManager

B.1 Attributes

+qState: 질문 text를 JSON 형태로 번역한 것

+aState: 답변 text를 Boolean 형태로 번역한 것

B.2 Methods

+textToQ(char[]): 질문 text를 Dialogflow를 통해서 JSON 형태로 번역한다.

+textToA(char[]): 답변 text를 Dialogflow를 통해서 Boolean 형태로 번역한다.

C. Data

C.1 Attributes

+ID : 참여자들을 지칭하는 식별자

+team : 해당 참여자가 소속된 팀

+role : 해당 참여자가 부여 받은 역할

+condition : 해당 참여자가 부여 받은 제약조건

+life : 해당 참여자가 게임 내에서 생존 중인지를 나타낸다.

C.2 Methods

해당 없음

5.3 Sequence Diagram

A. Question Analysis

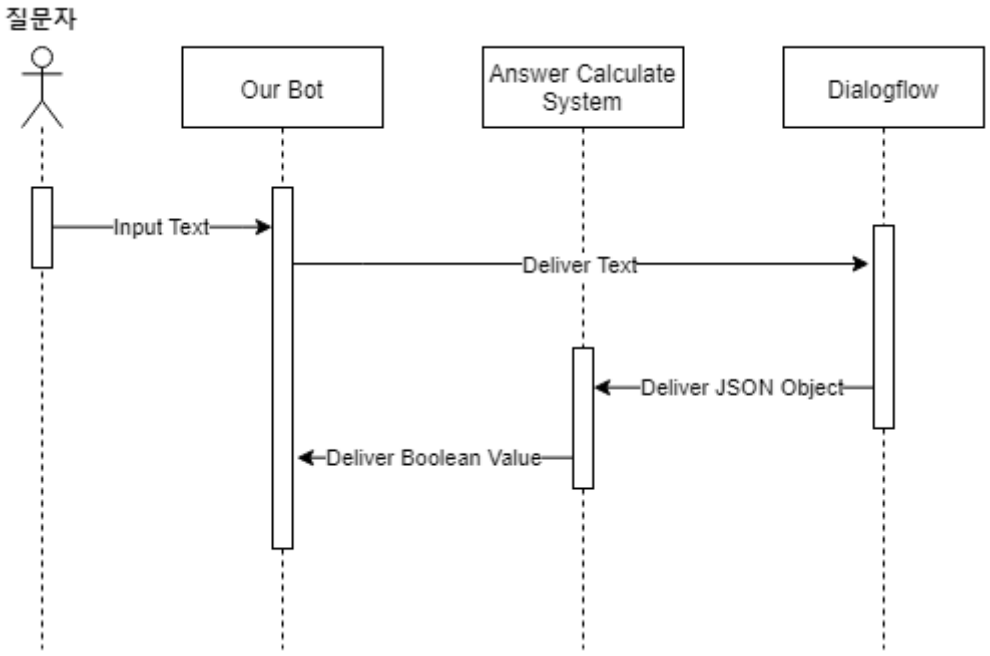

Diagram 13 Question analysis Sequence Diagram

B. Answer Distinguishment

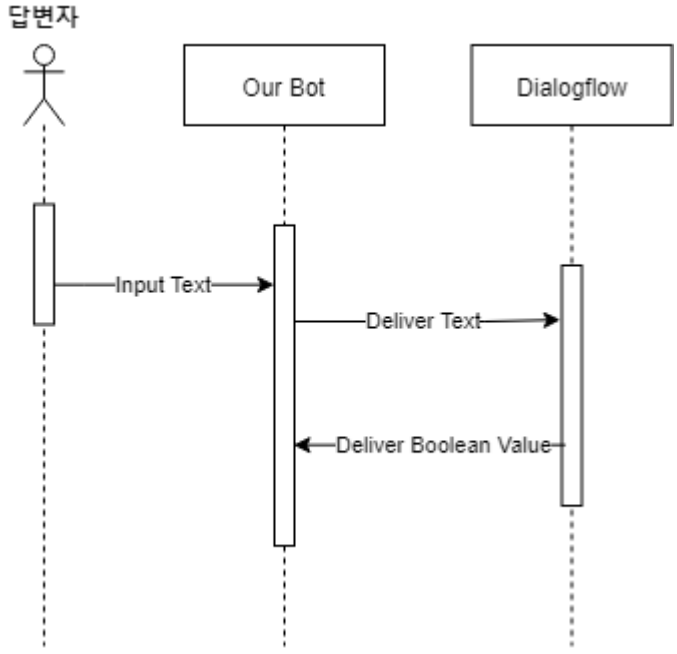

Diagram 14 Answer Distinguishment Sequence Diagram

C. Answer Comparison

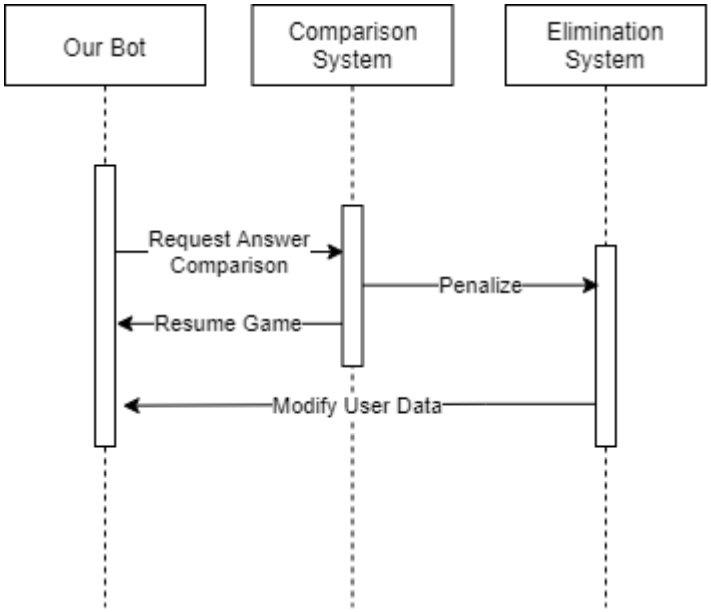

Diagram 15 Answer comparison Sequence Diagram

5.4 State Diagram

A. Question Analysis

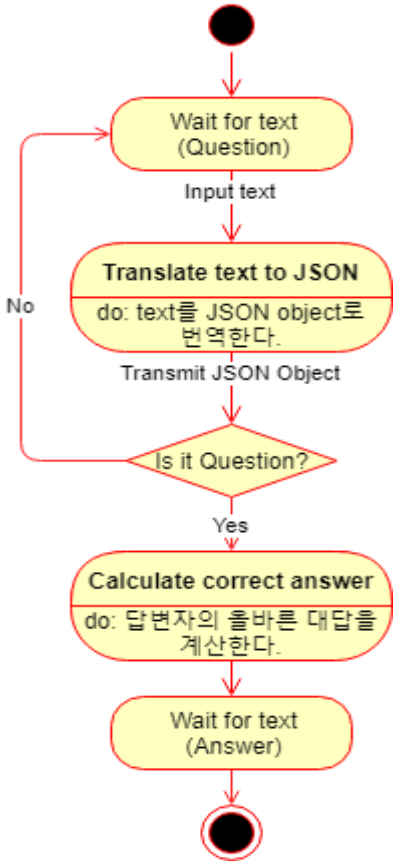

Diagram 16 Question Analysis State Diagram

B. Answer Distinguishment

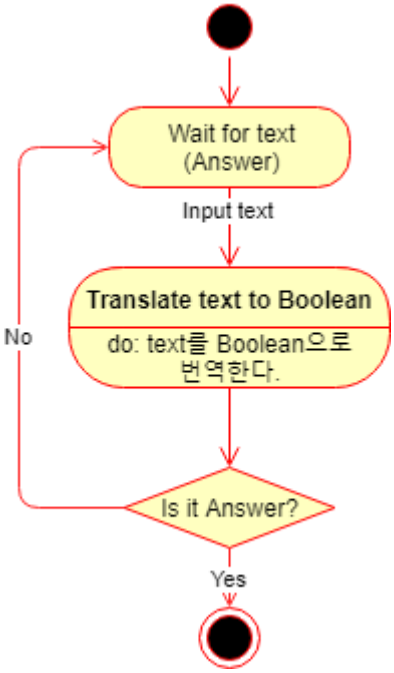

Diagram 17 Answer Distinguishment State Diagram

C. Answer Comparison

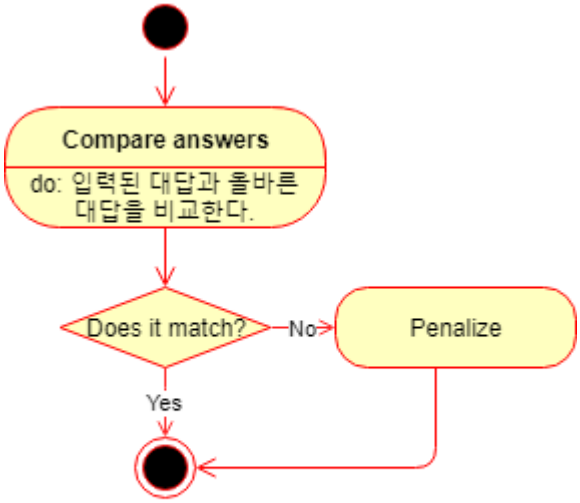

Diagram 18 Answer Comparison State Diagram

6 Elimination System

6.1 Objectives

한 플레이어가 특정 플레이어를 상대 팀의 왕으로 지목했을 경우 이를 처리해주는 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagram을 통해 본 시스템의 구조를 설명한다

6.2 Class Diagram

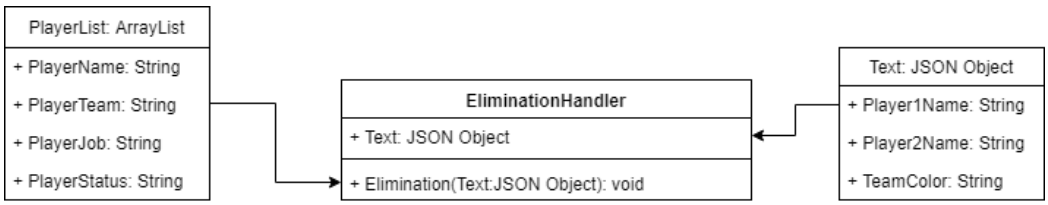

Diagram 19 Elimination system Class Diagram

A. EliminationHandler

A.1 Attributes

Text: 누가 누굴 지목했는지에 대한 정보

A.2 Methods

Elimination(Text): Text에 담긴 정보를 토대로 Elimination 과정을 진행한다.

B. Test : JSON Object

B.1 Attributes

Player1Name : 지목한 플레이어의 이름

Player2Name : 지목당한 플레이어의 이름

TeamColor : 지목한 플레이어의 소속 팀

B.2 Methods

해당없음

C. PlayerList: ArrayList

C.1 Attributes

PlayerName: 플레이어의 이름

PlayerTeam: 플레이어의 소속 팀

PlayerJob: 플레이어의 게임 내 지위

PlayerStatus: 플레이어의 생존 여부

C.2 Methods

해당없음.

6.3 Sequence Diagram

A. "Valid" Elimination: the player accused right person

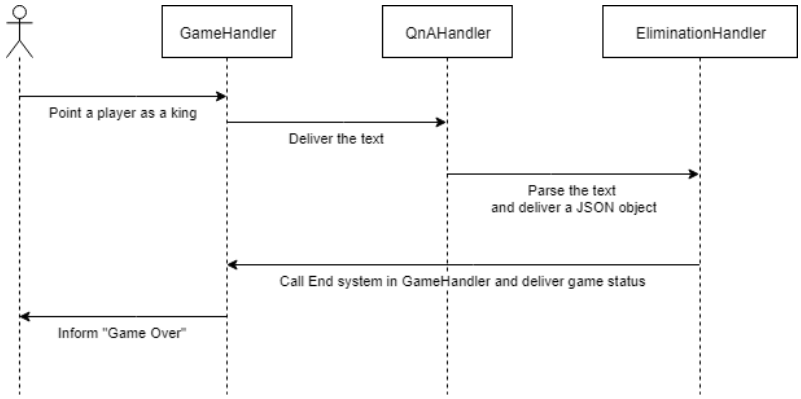

Diagram 20 Elimination System Sequence Diagram -1

B. "Invalid" Elimination: the player accused wrong person.

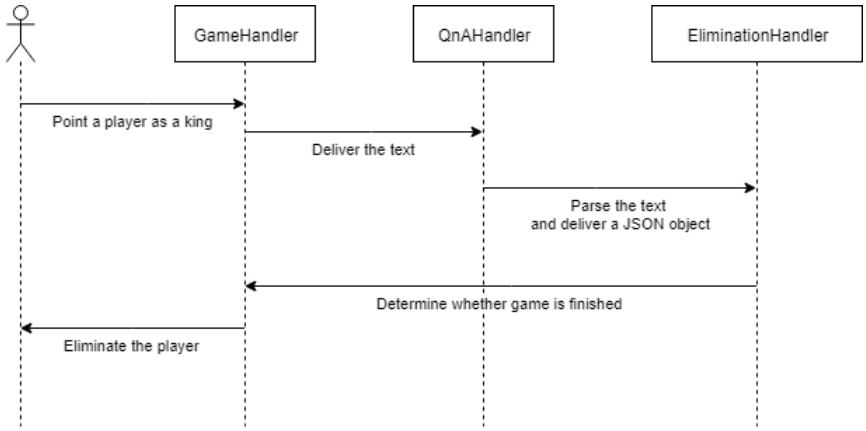

Diagram 21 Elimination System Sequence Diagram - 2

6.4 State Diagram

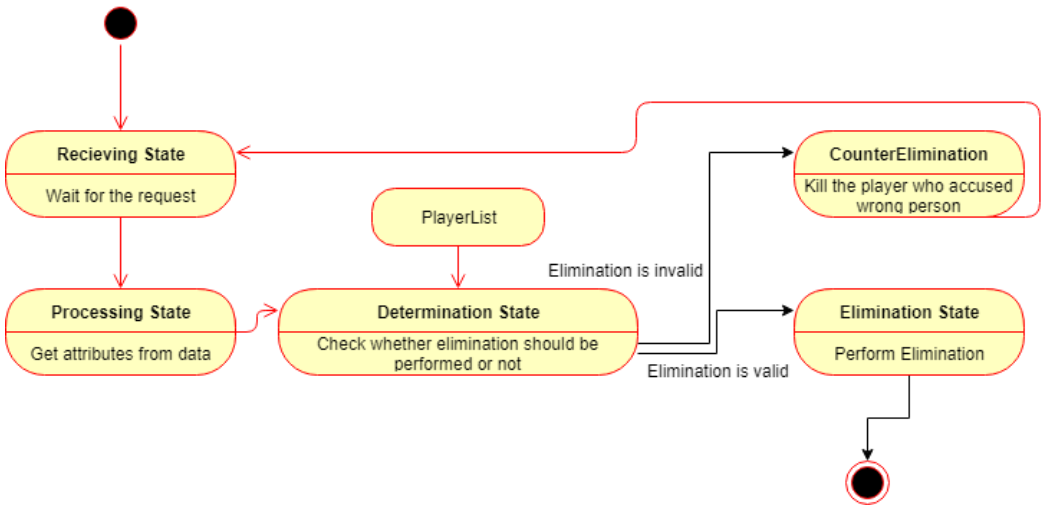

Diagram 22 Elimination System State Diagram

7 Help & log System

7.1 Objectives

참여자가 요구할 때, Bot에 대한 도움말과 게임 log를 Discord상에 출력해주는 시스템을 설명한다. Help & Log System의 구조를 표현하기 위해, class diagram, sequence diagram, state diagram을 사용한다.

7.2 Class Diagram

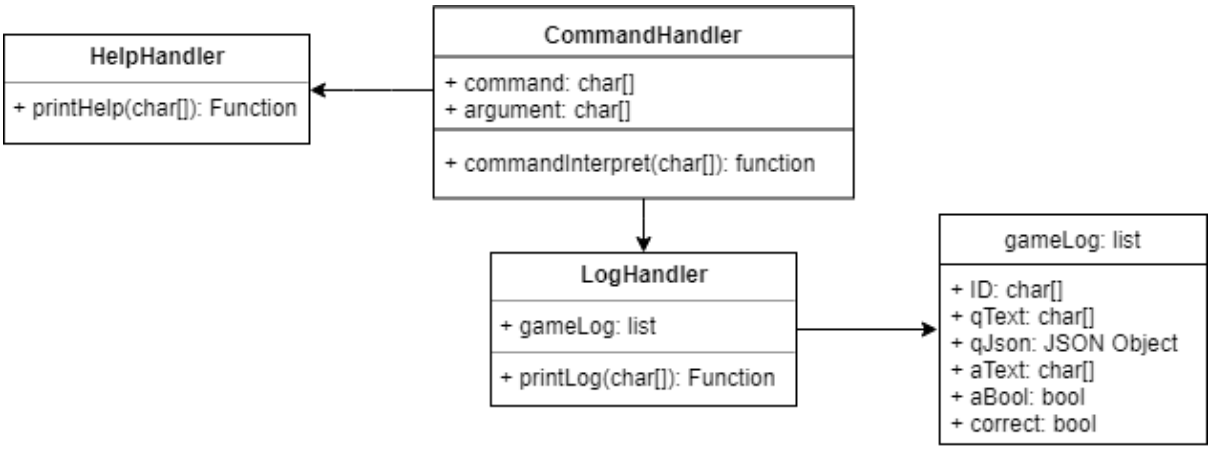

Diagram 23 Help & Log System Class Diagram

A. CommandHandler

A.1 Attributes

+ command: 입력된 명령어

+ argument: 명령어와 함께 입력된 인자

A.2 Methods

+ commandInterpret(char[]): 입력된 명령어를 해석하고 처리하는 함수.

B. HelpHandler

B.1 Attributes

해당 없음

B.2 Methods

+ printHelp(char[]): 인자 값에 맞는 도움말을 Discord상에 출력한다.

C. LogHandler

C.1 Attributes

+ gameLog: 이번 게임에서 이루어진 질문과 답변 내역을 기록한 list.

C.2 Methods

+ printLog: 인자 값에 맞는 게임 기록을 Discord상에 출력한다.

D. gameLog

D.1 Attributes

+ ID: 참여자들을 지칭하는 식별자이다.

+ qText: 대상 참여자가 받은 질문 text.

+ qJson: 대상 참여자가 받은 질문의 논리식 번역.

+ aText: 대상 참여자가 한 대답 text.

+ aBool: 대상 참여자가 한 대답의 Boolean 번역.

+ correct: 대상 참여자가 답변을 올바르게 했는지 여부.

D.2 Methods

해당 없음

7.3 Sequence Diagram

A. Print Help

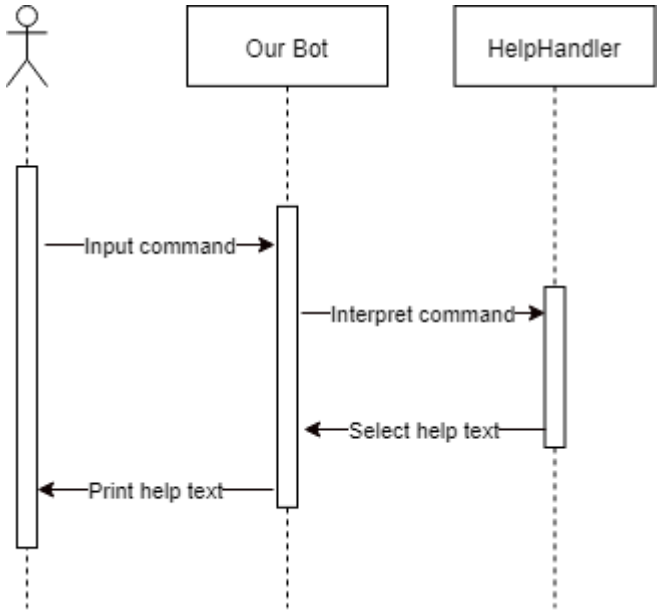

Diagram 24 Help Sequence Diagram

B. Print Log

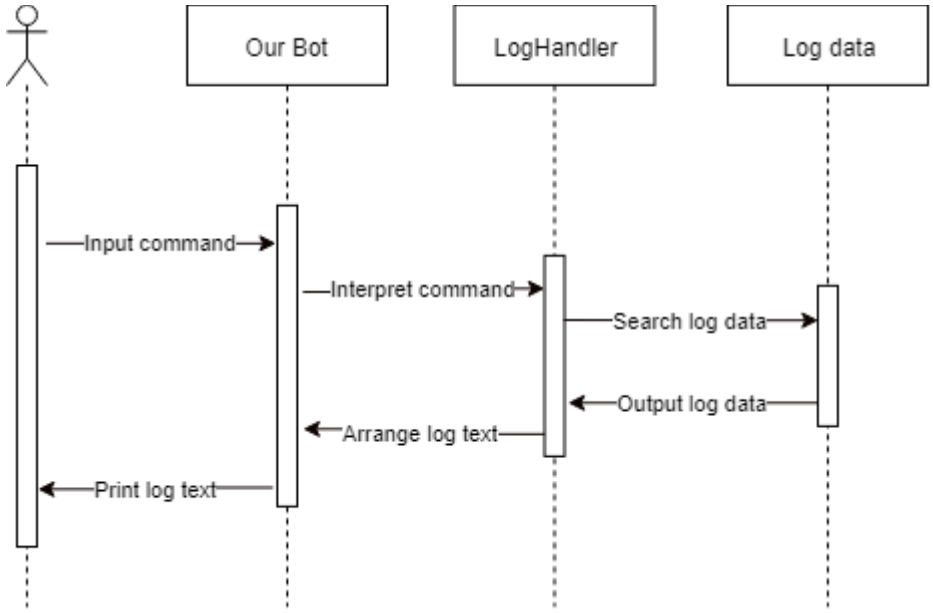

Diagram 25 Log Sequence Diagram

7.4 State Diagram

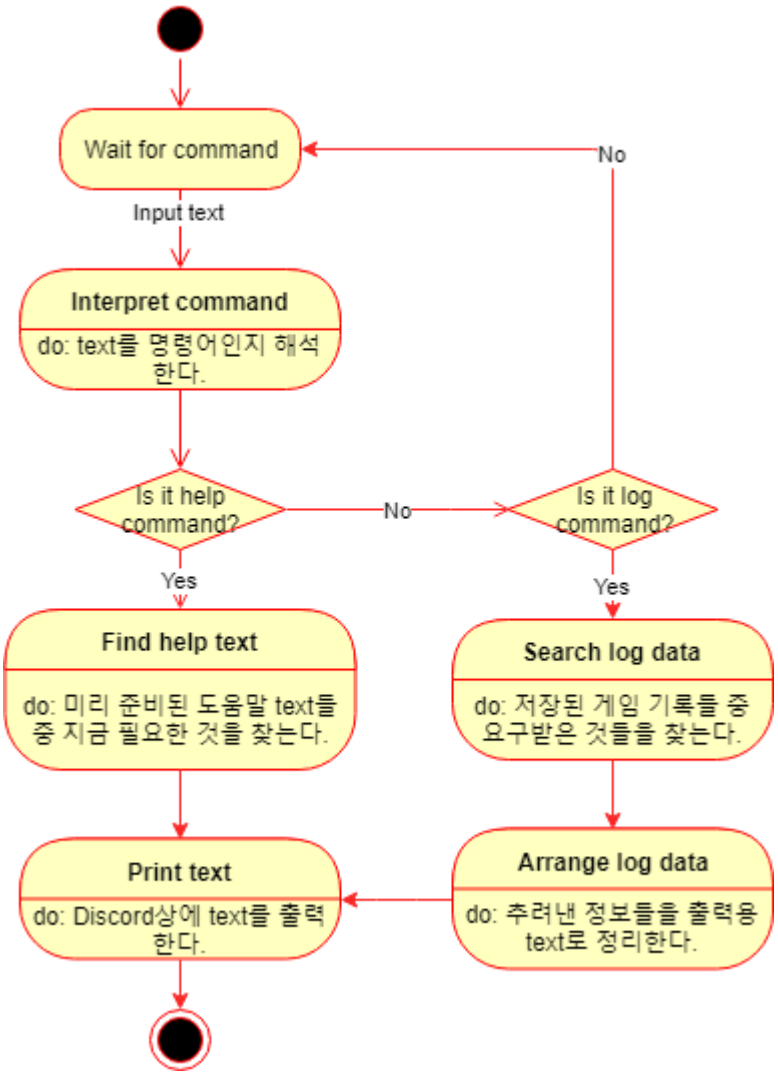

Diagram 26 Help & Log State Diagram

8 DB Management System

8.1 Objectives

본 봇의 Database를 관리해주는 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagram을 통해 본 시스템의 구조를 설명한다

8.2 Class Diagram

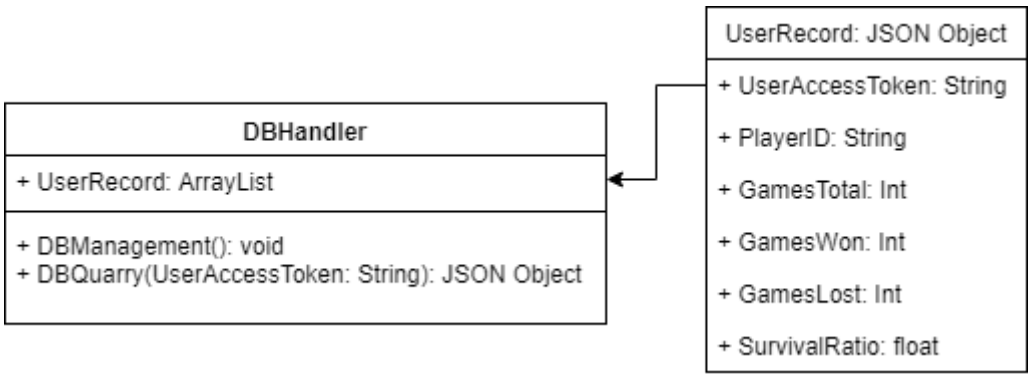

Diagram 27 DB Management System Class Diagram

A. DBHandler

A.1 Attributes

UserAccessToken: 유저에게 부여된 고유한 식별번호

PlayerID: 플레이어의 ID

GamesTotal: 총 플레이한 게임 수

GamesWon: 이긴 게임의 수

GamesLost: 진 게임의 수

SurvivalRatio: 게임 내 생존율

B. UserRecord : JSON Object

해당없음

8.3 Sequence Diagram

A. DB Quarry Sequence : 플레이어가 자신의 전적을 요청했을 때

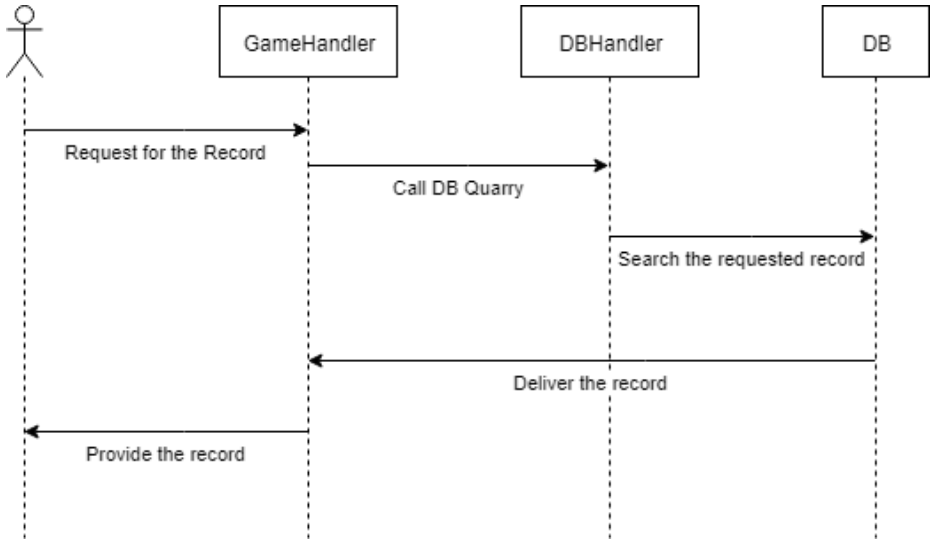

Diagram 28 DB management System Sequence Diagram - 1

B. DB Save Sequence : 게임 종료 후 플레이어들의 전적을 갱신했을 때

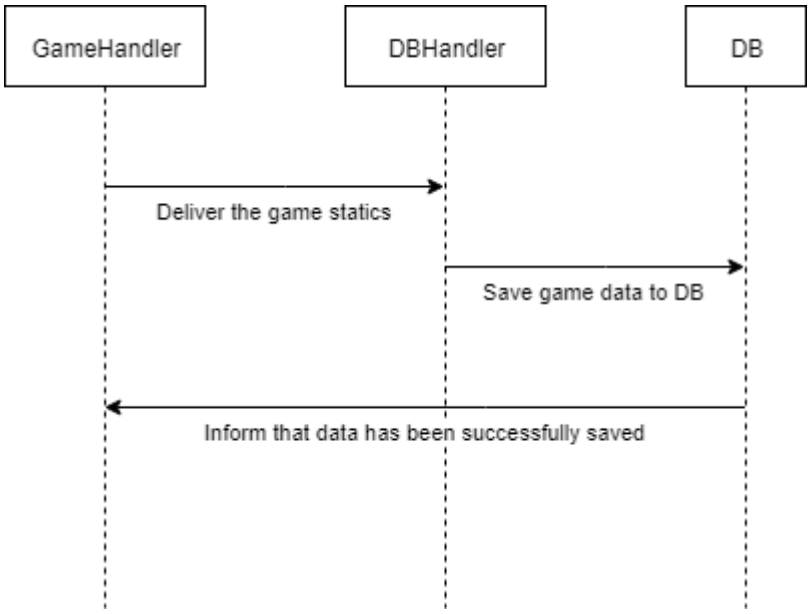

Diagram 29 DB management System Sequence Diagram - 2

8.4 State Diagram

A. DB Quarry State

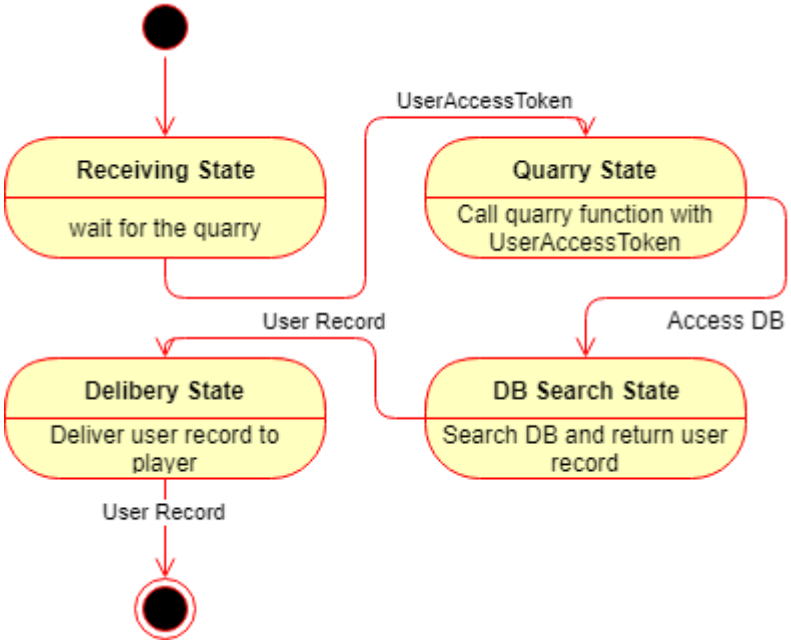

Diagram 30 DB management System State Diagram - 1

B. DB Save State

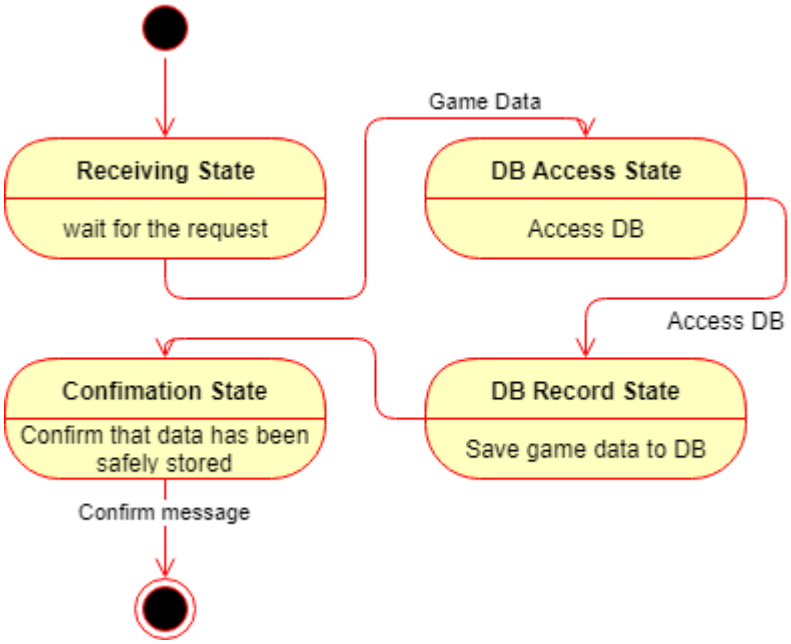

Diagram 31 DB management System State Diagram - 2

9 Testing Plan

9.1 Objectives

시스템이 의도한 방향대로 실행되고 시스템 내부의 결함을 찾기 위해 Testing을 실시한다.본 문단에서는 설계 단계에서 미리 기획한 Testing Policy와 Test Case에 대해 서술한다.

9.2 Testing Policy

본 시스템의 Testing은 Developing Testing, Release Testing, User Testing의 총 3 단계로 나누어 진행한다.

A. Developing Testing

개발과정에서 수행되는 testing으로 Component Testing, Integrating Testing, System Testing, Acceptance Testing으로 나뉜다.

B. Release Testing

사용자에게 출시하기 전에 최종 시스템을 testing하는 단계이다. 요구사항 명세서에 명시된 요구사항이 잘 충족되었는지 확인하는 것을 중점으로 진행한다.

C. User Testing

사용자가 사용자의 환경에서 시스템을 testing하는 것이다.

9.3 Test Case

A. Start & End System

A.1 Start

1) User : 봇을 초대한 후, 한 플레이어가 게임을 시작을 알리는 내용을 Discord상에 입력한다.

2) System 동작 : gameHandler가 chat의 내용을 파싱하기 위해 Dialogflow를 통해 parsing한 결과 Start로 판명되면 Start를 호출한다.

B. Question & Answer System

1) User: 특정 플레이어에게 질문을 하는 내용의 chat를 Discord상에 입력한다.질문을 받은 플레이어는 긍정 또는 부정의 의미를 담은 대답의 chat를 Discord상에 입력한다.
2) System동작: GameHandler가 QnAHandler에게 chat 내용들을 전달하고 Dialogflow를 통해 parsing한 결과 Question 과정으로 판명되었을 경우 qAnalysis 함수를 호출하여 질문을 처리한다. Answer 과정으로 판명되었을 경우 aDistinct, aCompare 함수를 호출하여 판별하고 결과에 따라서 Elimination을 호출한다.

C. Elimination System

1) User: 특정 플레이어를 상대 팀의 왕으로 지목하는 내용의 chat을 Discord상에 입력한다.
2) System동작: GameHandler가 QnAHandler에게 chat의 내용을 전달하고 Dialogflow를 통해 parsing한 결과 Elimination 과정으로 판명되었을 경우 Elimination을 호출한다.
3)

(1) System동작: 플레이어가 상대 팀 왕을 맞게 지목했을 경우 End System을 호출하고 팀 승리 정보를 전달한다.

(2) System 동작: 플레이어가 상대 팀 왕을 틀리게 지목했을 경우 UserList에서 해당 플레이어를 "사망"으로 설정한다.

4) System 알림:해당 플레이어가 사망했음을 공지한다.

D. DB Management System

D.1 User requests personal record

1) User: 명령어를 통해 본인의 전적 조회를 요청한다.
2) System 동작:해당 유저의 Access Token을 DB Handler에 전달한다.
3) System 동작:Access Token을 이용해 DB를 검색해 전적을 불러온다.

System 알림:플레이어의 전적을 DM을 통해 전달한다.

D.2 Update records

1) System 동작:게임이 끝난 후 End System이 DB Handler를 호출하고 인게임 정보를 전달한다.
2) System 동작:각 유저의 Access Token을 이용해 해당 유저들의 기록을 갱신하여 DB에 저장한다.
3) System 알림:성공적으로 전적 갱신에 성공했음을 공지한다.

10 Development Environment

10.1 Objectives

Development Environment에서는 개발 과정에서 사용하는 프로그래밍 언어나 IDE 등의 개발 환경을 설명한다.

10.2 Programming Language & IDE

A. Programming Language

Discord 봇 개발 언어로는 node.js를 사용한다.node.js는 chrome V8 JavaScript 엔진으로 빌드된JavaScript 런타임이다.이벤트 기반,논 블로킹 I/O 모델을 사용하여 가볍고 효율적이다.

Discord 봇 모듈인 discord.js가 오픈소스로서 제공되고 있기 때문에,이를 활용할 수 있도록 해당 프로젝트에서는 node.js를 사용한다.

Figure 11 Node js logo

B. IDE

Figure 12 Eclipse IDE logo

프로젝트의 핵심적인 통합 개발 환경으로는 이클립스를 사용한다.이클립스는 자바를 비롯한 다양한 언어들을 여러 환경에서 사용할 수 있도록 지원해주는 응용 소프트웨어 플랫폼이다.기본적으로 자바를 기반으로 작성되었으며, C/C++, 웹 개발 등 다른 목적으로도 사용 가능하다.

node.js를 사용하기 위한 환경 중에서 가장 접하고 다루기 쉬울 것으로 판단되어 프로젝트 IDE로 결정됐다.

10.3 Coding Rule

프로젝트 진행 도중 혼선을 방지하기 위해서,이와 같은 Coding rule을 세우고 지킬 것을 권고한다.

1) 작업 시간을 나눠서,동시에 동일한 작업을 하거나 서로 충돌하는 작업을 하지 않도록 한다.
2) 다른 팀원의 작업물에 대해서도 구조와 변수 또는 함수의 선언명 등을 숙지해야 한다.서로 직접적으로 연결된 시스템을 제작하는 간에는 특히 중요하다.

작업물은 버전 관리를 위해서 반드시 Github등의 저장소에 우선적으로 업로드 한 뒤 관리해야 한다.

10.4 Version Management Tool

Figure 13 Github logo

Configuration Management는 Github을 통해서 이루어진다.Github는 분산 버전 관리 툴인 Git을 사용하는 프로젝트를 지원하는 웹호스팅 서비스이다.영리적인 서비스와 오픈소스를 위한 무상 서비스를 모두 제공하며, 가장 큰 규모의 Git호스팅 사이트이다.버전 별 수정사항들을 확인하기 쉽고,프로젝트에 도움이 될 오픈소스들이 많이 등록되어 있어 유용하다.

11 Develop Plan

11.1 Objectives

No.	수행 내역	개발 일정(주 단위)(3월 4일 기준)													비고
		3주	4주	5	6	7	8	9	10	11	12	13	14	15	
1	Idea														
2	Feasibility study														
3	Requirement Analysis														제안서 발표
4	Design Study														요구사항 명세서 제출
5	Design														설계 명세서 제출
6	Develop														
7	Integrate Testing														
8	Document work														결과 발표

Table 2 Develop Plan Table (초록색 : 실제 시간, 파란색 : 예상기간)

12 Index

12.1 Diagram Index

12.2 Figure Index

12.3 Table Index

13 Reference

[1] wikipedia UML, 2018.05.17, <https://ko.wikipedia.org/wiki/통합_모델링_언어>

[2] wikpedia Packege diagram, 2018.05.17,
< https://en.wikipedia.org/wiki/Package_diagram>

[3] wikpediaDeploymentDiagram, 2018.05.17,
<https://en.wikipedia.org/wiki/Deployment_diagram>

[4] wikipediaClass Diagram, 2018.05.17, <https://en.wikipedia.org/wiki/Class_diagram>

[5] IBM State Diagram, 2018.05.17,
<https://www.ibm.com/support/knowledgecenter/en/SS6RBX_11.4.2/com.ibm.sa.oomethod.doc/topics/c_UML_State_diag.html>

[6] wikipedia Sequence Diagram, 2018.05.17
<https://en.wikipedia.org/wiki/Sequence_diagram>

[7] Wikipedia ER Diagram, 2018.05.17,
<https://ko.wikipedia.org/wiki/개체-관계_모델>

[8] dialog flow 설명부분, 2018.05.17,
<http://www.kwangsiklee.com/ko/2018/01/구글-ai-플랫폼-dialogflow-눈으로-따라하며-배우기?

[9] Github 가이드 문서, 2018.05.18, <https://guides.github.com/activities/hello-world/#what>

[10] wikipedia Eclipse(software), 2018.05.18, <https://en.wikipedia.org/wiki/Eclipse_(software)>

[11] Node.js-v10.1.0 Documentation, 2018.05.18, <https://nodejs.org/api>