목차

[1. Preface 6](#_Toc513301946)

[1.1. Objective 6](#_Toc513301947)

[1.2. Readership 6](#_Toc513301948)

[1.2.1. User Requirement Readership 6](#_Toc513301949)

[1.2.2. System Requirement Readership 6](#_Toc513301950)

[1.3. Document Structure 6](#_Toc513301951)

[1.3.1. Preface 6](#_Toc513301952)

[1.3.2. Introduction 6](#_Toc513301953)

[1.3.3. Glossary 6](#_Toc513301954)

[1.3.4. User Requirement Definition 6](#_Toc513301955)

[1.3.5. System Architecture 7](#_Toc513301956)

[1.3.6. System Requirement Specification 7](#_Toc513301957)

[1.3.7. System Models 7](#_Toc513301958)

[1.3.8. System Evolution 7](#_Toc513301959)

[1.3.9. Appendices 7](#_Toc513301960)

[1.3.10. Index 7](#_Toc513301961)

[1.4. Version of the Document 7](#_Toc513301962)

[1.4.1. Version Format 7](#_Toc513301963)

[1.4.2. Version Management Policy 7](#_Toc513301964)

[1.4.3. Version Update History 7](#_Toc513301965)

[2. Introduction 8](#_Toc513301966)

[2.1. Objective 8](#_Toc513301967)

[2.2. Needs 8](#_Toc513301968)

[2.3. Our Game 8](#_Toc513301969)

[2.4. Expected Effect of the Service 10](#_Toc513301970)

[3. Glossary 10](#_Toc513301971)

[3.1. Objective 10](#_Toc513301972)

[3.2. Term Definitions, Acronyms and Abbreviations 10](#_Toc513301973)

[3.2.1. Term Definition 10](#_Toc513301974)

[A. User & Our game 관련 용어 11](#_Toc513301975)

[B. Development & Method 관련 용어 11](#_Toc513301976)

[C. 기타 관련 용어 11](#_Toc513301977)

[3.2.2. Acronym and Abbreviation 11](#_Toc513301978)

[4. User Requirement Definition 11](#_Toc513301979)

[4.1. Objective 11](#_Toc513301980)

[4.2. Functional Requirements 12](#_Toc513301981)

[4.2.1. 게임의 사회자 12](#_Toc513301982)

[A. Start / End Game 12](#_Toc513301983)

[B. Q & A Management 12](#_Toc513301984)

[C. Elimination 12](#_Toc513301985)

[D. Help / Log 12](#_Toc513301986)

[E. 전적 기록 13](#_Toc513301987)

[4.3. Non-Functional Requirements 13](#_Toc513301988)

[4.3.1. Product Requirement 13](#_Toc513301989)

[A. Performance Requirement 13](#_Toc513301990)

[B. Security Requirement 13](#_Toc513301991)

[C. Efficiency Requirement 13](#_Toc513301992)

[D. Dependability Requirement 13](#_Toc513301993)

[E. Usability Requirement 14](#_Toc513301994)

[4.3.2. Organizational Requirement 14](#_Toc513301995)

[A. Environment Requirement 14](#_Toc513301996)

[B. Operational Requirement 14](#_Toc513301997)

[5. System Architecture 14](#_Toc513301998)

[5.1. Objective 14](#_Toc513301999)

[5.2. Start System 14](#_Toc513302000)

[5.3. End System 15](#_Toc513302001)

[5.4. Q & A System 15](#_Toc513302002)

[5.5. Elimination System 16](#_Toc513302003)

[5.6. Help / log System 17](#_Toc513302004)

[5.7. DB management System 17](#_Toc513302005)

[6. System Requirement Specification 17](#_Toc513302006)

[6.1. Objective 17](#_Toc513302007)

[6.1.1. 게임의 사회자 17](#_Toc513302008)

[A. Start / End Game 18](#_Toc513302009)

[B. Q & A Management 18](#_Toc513302010)

[C. Elimination 20](#_Toc513302011)

[D. Help / Log 20](#_Toc513302012)

[E. 전적 기록 20](#_Toc513302013)

[6.2. Non-Functional Requirements 20](#_Toc513302014)

[6.2.1. Product Requirement 20](#_Toc513302015)

[A. Performance Requiement 20](#_Toc513302016)

[B. Security Requirement 21](#_Toc513302017)

[C. Efficiency Requirement 21](#_Toc513302018)

[D. Dependability Requirement 21](#_Toc513302019)

[E. Usability Requirement 21](#_Toc513302020)

[6.3. Scenario 22](#_Toc513302021)

[6.3.1. Join Scenario 22](#_Toc513302022)

[6.3.2. Question & Answer Scenario 22](#_Toc513302023)

[6.3.3. Elimination Scenario 22](#_Toc513302024)

[6.3.4. Winning Scenario 22](#_Toc513302025)

[6.3.5. Managing User profile Scenario 22](#_Toc513302026)

[7. System Models 22](#_Toc513302027)

[7.1. Objective 23](#_Toc513302028)

[7.2. Context Models 23](#_Toc513302029)

[A. Context Model 23](#_Toc513302030)

[B. Processes Model 23](#_Toc513302031)

[7.3. Interaction Models 23](#_Toc513302032)

[A. Use case model 24](#_Toc513302033)

[B. Tubular description for each Use-Case 24](#_Toc513302034)

[B.1 Quetion 24](#_Toc513302035)

[Sub-system 24](#_Toc513302036)

[Q & A system 24](#_Toc513302037)

[B.2 Answer 24](#_Toc513302038)

[Q & A system 24](#_Toc513302039)

[B.3 Validate the truth value 24](#_Toc513302040)

[Q & A system 25](#_Toc513302041)

[B.4 Request for records 25](#_Toc513302042)

[B.5 Provide records 25](#_Toc513302043)

[B.6 Eliminate player 25](#_Toc513302044)

[B.7 End game 25](#_Toc513302045)

[7.4. Structural Models 25](#_Toc513302046)

[A. ... 26](#_Toc513302047)

[B. ... 26](#_Toc513302048)

[C. .... 26](#_Toc513302049)

[D. .... 26](#_Toc513302050)

[E. .... 26](#_Toc513302051)

[F. ..... 26](#_Toc513302052)

[7.5. Behavioral Models 26](#_Toc513302053)

[A. ... 26](#_Toc513302054)

[B. ... 26](#_Toc513302055)

[C. .... 26](#_Toc513302056)

[D. .... 26](#_Toc513302057)

[E. .... 26](#_Toc513302058)

[F. ..... 26](#_Toc513302059)

[8. System Evolutions 26](#_Toc513302060)

[8.1. Objective 26](#_Toc513302061)

[8.2. Limitations and Assumption 26](#_Toc513302062)

[8.3. Evolutions of User Requirement 26](#_Toc513302063)

[8.3.1. Voice Recognition-Based System 26](#_Toc513302064)

[8.3.2. Rule Modification Sandbox 26](#_Toc513302065)

[8.3.3. New game mods 26](#_Toc513302066)

[9. Appendices 26](#_Toc513302067)

[9.1. Objective 26](#_Toc513302068)

[9.2. Database Requirements 27](#_Toc513302069)

[9.3. User-System Requirements 27](#_Toc513302070)

[9.4. Development Process 27](#_Toc513302071)

[10. Index 27](#_Toc513302072)

[10.1. Table Index 27](#_Toc513302073)

[10.2. Figure Index 27](#_Toc513302074)

[10.3. Diagram Index 27](#_Toc513302075)

[11. Reference 27](#_Toc513302076)

1. Preface
   1. Objective

Preface에서는 본 문서의 예상되는 독자들과 문서의 전반적인 구조, 그리고 각 부분의 역할에 대하여 제시한다. 또한 버전 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 그에 대한 근거들을 서술한다.

* 1. Readership

본 요구사항 명세서는 독자에 따라 크게 User Requirement와 System Requirement의 두 부분으로 구성되어 있다.

* + 1. User Requirement Readership

User Requirement는 사용자의 관점에서 요구사항을 간략히 명세한 것이다. 본 프로젝트에서는 Game Player의 관점에서 서술한다.

* + 1. System Requirement Readership

System Requirement는 본 프로젝트에서 개발해야 하는 시스템의 특정 기능이 어떻게 구현되어야 하는지를 자세하게 기술한 것이다. 본 프로젝트에서는 Software Developer와 Server 및 Database Manager 관점에서 서술한다.

* 1. Document Structure

이 문서는 총 10개의 부분으로 구성되어 있다. Preface, Introduction, Glossary, User Requirements Definition, System Architecture, System Requirements Specification, System Models, System Evolution, Appendices, Index로 구성된다. 각 장의 역할과 전반적인 서술은 다음과 같다.

* + 1. Preface

Preface에서는 본 문서의 예상되는 독자들과 문서의 전반적인 구조, 그리고 각 부분의 역할에 대하여 서술한다. 또한 버전 관리 정책, 버전 변경 기록, 그리고 문서들의 변경사항과 그에 대한 근거들을 서술한다.

* + 1. Introduction

Introduction에서는 시스템의 필요성과 해당 시스템이 어떠한 Needs를 반영하고 있는지 서술한다. 또한 시스템의 기능이 다른 시스템과 어떻게 상호작용하는지 간략하게 서술한다.

* + 1. Glossary

Glossary에서는 본 문서에 등장하는 기술적 용어들과 축약어들을 정의한다.

* + 1. User Requirement Definition

User Requirement Definition에서는 사용자의 요구사항을 User Level에서 서술한다. 크게 Functional Requirement와 Non-Functional Requirement로 나누어 서술한다.

* + 1. System Architecture

System Architecture에서는 각 Component들의 hierarchy에 대하여 Diagram등을 이용해 서술한다.

* + 1. System Requirement Specification

System Requirement Specification에서는 User Level에서 정의되었던 Functional/Non-Functional Requirement들을 Developer Level로 세분화하여 서술한다.

* + 1. System Models

System Models에서는 System Component, System, 그리고 System Environment 사이의 관계를 Figure 및 Diagram을 통해 서술한다.

* + 1. System Evolution

System Evolution에서는 시스템이 세우고 있는 주요한 가정들과 사용자의 Needs 변화, 혹은 추후 Update될 사항들에 대해서 서술한다.

* + 1. Appendices

Appendices에서는 본 프로젝트와 관련한 참고사항들을 서술한다.

* + 1. Index

Index에서는 본 문서에 사용된 Table, Diagram, Fiqure들의 Index를 서술한다.

* 1. Version of the Document
     1. Version Format

Version 번호는 major.minor[.maintenance]로 구성되며, 문서의 버전은 0.1부터 시작한다.

* + 1. Version Management Policy

본 문서를 수정할 때마다 버전을 업데이트 한다. 이미 완성한 파트를 변경할 때는 Minor number를 변경하며, 새로운 부분을 추가하거나 괄목한 만한 변화가 있을 경우 Major number를 변경한다.

* + 1. Version Update History

|  |  |  |
| --- | --- | --- |
| Version | Committer | Explanation |
| 0.1 | 김동현 | 문서 목차 작성 및 Preface 서술. |
| 1.0 | 이용철 | Introduction 부분 서술 및 세부항목으로 Our game 파트 추가. |
| 2.0 | 김장훈 | User Requirement 일부 서술 |
| 3.0 | 팀 전체 | 전체적인 system 구조 틀 재구성 및 분담 구성 |
| 4.0 | 이용철 | Introduction 수정 및 Start / End System 관련 architecture 및 system requirement, modeling 일부 작성. |
| 4.1 | 김장훈 | Q & A System관련의 architecture및 system와functional requirement추가 |
| 4.2 | 이용철 | sequence modeling 부분 작성 |
| 4.3 | 김동현 | Elimination System, Record System의 architecture 및 system requirement 추가 |
| 4.4 | 김장훈 | System functional requirement, 일부 model 내용 추가 |

1. Introduction
   1. Objective

Introduction에서는 시스템의 필요성과 해당 시스템이 어떠한 Needs를 반영하고 있는지 서술한다. 또한 시스템의 기능이 다른 시스템과 어떻게 상호작용하는지 서술한다.

* 1. Needs

사람들이 모여 다같이 즐길 수 있는 놀이로 보드 게임들을 많이 한다. 또한, 최근 몇 년 사이에 보드 보드게임 카페의 수가 증가하고 있고, 카페를 가면 여러 사람들이 모여 다양한 게임을 하는 모습, 심지어 손님이 너무 많아 자리가 없어 다른 카페를 찾아 나가는 사람들을 볼 수 있다. 보드 게임 중에서 마피아류 보드 게임을 하는 것을 심심치 않게 볼 수 있는데, 예를 들어 '뱅!', '스파이폴', '한밤의 늑대인간' 등을 하는 것을 볼 수 있다. 앞의 설명에서 보듯이, 우리는 보드 게임 중에서 마피아류 게임에 집중을 하고 있는데, 그 이유는 이 게임은 특별한 도구 없이 각자의 역할에 충실하여 속임을 통해 몇 명이든 다 같이 참여할 수 있는 게임으로 보드게임 카페뿐만 아니라, MT, 혹은 온라인에서 모여서 하기도 한다.

우리가 이 게임을 하면서 여러 문제를 겪을 수 있는데, 그 중 역할만 주어질 뿐 정보의 획득은 추리를 통해 서로의 심리를 읽어 얻는 것으로 정보의 불균형으로 인해 자칫하면 일방적으로 끝날 수 있다. 또한, 따로 사회자가 없어 게임의 진행을 참여자들이 진행을 간섭함으로써, 방향을 유리하게 만들어 가는 경우도 생긴다.

우리는 이에 따른 대안으로 게임의 사회자로 chatBot을 참여시키는 방법이다. chatBot이 사회자 역할을 함으로써, 게임을 흥미진진하게 진행이 될 수 있도록 상황에 맞추어 한쪽에 유리한 정보의 쏠림 방지하기 위해 약간의 정보의 제공을 해주는 기능, 참여자의 질문에 대해 다른 참여자의 질문에 따른 참 / 거짓의 판별, 투표 등의 행위를 처리 등의 기능을 제공할 계획이다.

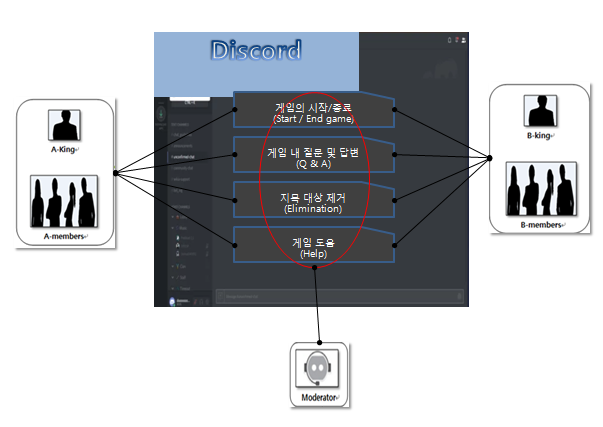
* 1. Our Game

여기서 우리는 기존의 마피아 게임을 수정하여 제안한다. 우리의 게임은 다음의 규칙이 있다.

* 구성 : 사회자(chatBot), TEAM-A(king & members), TEAM-B(king & members)
* 각 참여자에 대해 갖는 제약조건들 :
  + .....
  + .....
  + ....
* 규칙 :
  + 각 참여자는 역할에 충실하면서 자유롭게 추리를 한다. 참여자는 자신이 속한 팀의 구성원이 누구인지 알 수 없다. 그 중에 사회자에게 질의를 할 수 있는 기회를 얻은 참여자가 모두에게 혹은 지목 & 선언을 통해 질의를 할 수 있다.
  + 사회자는 받은 질의를 모두에 공개하고 이에 대상자(들)은 답변을 해야 한다.
  + 사회자는 참여자의 제약조건과 규칙에 비교 후 결과를 공개한다.
  + 일정 시간 후에 사회자가 혹은 참여자 중 누군가 '투표!' 등을 발언 시 투표 시스템을 시작한다.
  + 팀의 왕이 죽으면 게임 종료 & 살아남은 왕의 팀이 승리.

우리가 제안한 게임 시스템은 Discord 기반에서 실행 될 것이며, 다음의 기능은 3가지의 기능 제공할 계획이다.

* Start / End game : User가 게임을 시작하기 위해 [봇] 불러와 시작과 종료를 지정하고, 이 때, 정해진 참여자들에게 역할 배정을 한다.
* Q & A : 참여자는 게임 내에서 자유로운 추리를 하다가 Question & Answer을 질의를 하고 있다.
* 지목 대상자 제거(Elimination) : 참여자는 지금까지 추리한 내용을 토대로 상대팀의 왕으로 생각되는 다른 참여자를 지목하여 제거 한다. 이 때, 잘못된 추리로 인한 실패 시 본인이 제거되고, 성공 시 상대팀의 왕을 완벽한 추리로 제거하여 팀의 승리로 이끌 수 있다.
* 게임 도움(Help) : User 혹은 참여자는 명령어를 통해 게임에 관해서 정보를 획득할 수 있다.



* 1. Expected Effect of the Service
* Player
  + 정보의 불균형으로 인한 게임의 편향성을 최소화함으로써, 진행함에 있어서 흥미진진하게 게임을 즐길 수 있을 것으로 예상된다.
  + 데스크탑 혹은 스마트폰으로 진행할 수 있기 때문에 어디서든 discord가 설치되어 있다면 즐겁게 게임이 가능하고, 오프라인에서 이 게임을 통해서 [봇]이 사회자로 게임을 진행함으로서 오프라인에서 즐길 수 있을 것이다.

1. Glossary
   1. Objective

Glossary에서는 본 문서에 등장하는 기술적 용어들과 축약어들을 정의한다. 해당 문서를 어느 누가 읽어도 이해할 수 있도록 모든 용어에 대해 설명한다.

* 1. Term Definitions, Acronyms and Abbreviations
     1. Term Definition

3 가지의 분류로 Term Definitions을 서술한다.

* + - 1. User & Our game 관련 용어

|  |  |
| --- | --- |
| Terms | Definition |
| 유저(User) | 게임의 참여하기 전에 이 [봇]을 이용할 사람들을 말한다. |
| 참여자(Player) | 게임에 참여하는 사람들을 말한다. |
| 사회자(Moderator) | 게임의 진행을 제어하고 원활하게 진행을 이끄는 자를 말한다. |
| 왕(King) | 게임 내에서 한 팀의 왕으로, 생존을 함으로써 팀을 승리로 이끄는 자를 말한다. |
| 멤버(member) | 게임 내에서 한 팀의 왕을 제외한 구성원으로, 추리를 통해서 왕을 보호하고, 상대 구성원 및 왕을 죽이는 자를 말한다. |

* + - 1. Development & Method 관련 용어

|  |  |
| --- | --- |
| Terms | Definition |
|  |  |
|  |  |
|  |  |
|  |  |

* + - 1. 기타 관련 용어

|  |  |
| --- | --- |
| Terms | Definition |
|  |  |
|  |  |
|  |  |
|  |  |

* + 1. Acronym and Abbreviation

|  |  |
| --- | --- |
| Terms | Definition |
|  |  |
|  |  |
|  |  |
|  |  |

1. User Requirement Definition
   1. Objective

User Requirement Definition에서는 사용자의 요구사항을 User Level에서 서술한다. 크게 Functional Requirement와 Non-Functional Requirement로 나누어 서술한다.

* 1. Functional Requirements

우리의 project에서 제공될 기능들은 다음과 같다.

* + 1. 게임의 사회자

[봇]은 사회자로서 다음의 기능들을 관리한다.

* + - 1. Start / End Game

Discord를 통해 게임을 시작하는 명령어가 입력될 경우, [봇]은 게임을 시작하기 위한 준비 과정에 들어간다.이 과정에서 게임에 참가하는 참여자,진행할 게임의 종류,세부 규칙 등의 설정을 할 수 있다.모든 준비가 끝났다면 준비완료 명령어를 입력하여 게임 시작단계로 들어간다.

게임이 시작되면 [봇]은 모든 참여자에게 팀과 역할 그리고 제약 조건을 부여한다.부여된 정보들은 참여자 본인만 확인할 수 있도록 개인 메시지를 통해 전달한다.

게임 도중 제외된 참여자의 역할이 왕이었다면,게임의 승패를 가릴 수 있게 된다. [봇]은 게임 종료를 선언하고, 결과를 발표한 뒤, 참여자들의 전적을 갱신한다.

* + - 1. Q & A Management

통상적인 게임 단계에서 [봇]은 참여자들의 문답을 관리해주는 상태가 된다.한 참여자가 Discord에 질문으로 여겨지는 text를 입력할 경우,[봇]은 질문을 해석한 뒤 질문 대상자에게 대답을 할 것을 요구한다.대답은 긍정 또는 부정 두 가지 형태가 존재할 수 있으며,그 외의 대답과 대상자가 아닌 참여자가 입력한 text들은 혼선을 피하기 위해 무시된다.질문 대상자가 대답 text를 입력하면, [봇]은 질문과 대답 그리고 대상자의 제약 조건을 비교하여 대상자가 올바른 대답을 하였는지 검증해준다.만약 질문 대상자가 부적절한 대답을 했을 경우 [봇]은 대상자에게 패널티를 부과한다.이 과정이 정상적으로 이루어진 뒤 [봇]은 다시 질문을 받아들이는 상태가 된다. 질문과 대답 text는 별도의 명령어 없이 자연어 형태로 입력할 수 있다.

* + - 1. Elimination

참여자가 다른 참여자를 지목하여 상대팀의 왕으로 지목하여 맞으면 승리로 이끌고, 틀리면, 본인의 퇴장.

* + - 1. Help / Log

Discord에 도움말 명령어를 입력하는 것으로 [봇]을 사용하기 위해 필요한 명령어나 게임의 룰을 볼 수 있다.또한,게임 도중에 로그 명령어를 입력하는 것으로 지금까지 이루어진 문답 내역을 볼 수 있다.

* + - 1. 전적 기록

참여자들의 승패 기록은 매 게임마다 자동적으로 데이터 베이스에 저장된다.프로필 명령어를 입력하는 것으로 자신의 게임 기록, 승률, 순위 등을 확인할 수 있으며, 원한다면 기록을 초기화할 수 있다. 전적 사항의 본인의 요청에 따라 공개 및 비공개로 설정할 수 있다.

* 1. Non-Functional Requirements
     1. Product Requirement
        1. Performance Requirement

Discord에 입력되는 text들을 빠르고 정확하게 처리하여 게임에 혼선을 빚는 일이 없어야 한다. 자연어 text를 해석해야 하는 경우, [봇]이 오역을 하는 일이 없도록 확실한 학습이 필요하다

* + - 1. Security Requirement

[봇]을 사용할 때 발생할 수 있는 개인정보와 보안 문제점들은 대부분 Discord의 시스템에 기인한다. [봇]과 사용자 사이에는 항상 Discord가 존재하기 때문에, Discord 상에 보안 문제가 없다면 [봇]을 사용할 때도 문제가 발생하지 않는다. [봇]이 자연어를 처리하는 과정에서 개인 정보가 포함된 text를 데이터 베이스에 저장하는 일을 방지하기 위해서, 모든 text는 게임에 연관된 정보만을 논리식으로 번역하여 다루도록 한다.

[수정이 필요할 것 같음.]

* + - 1. Efficiency Requirement

[봇]은 서버에 의해서 항시 실행되는 상태를 유지하고 있다. 따라서, 서버에 부하가 걸려 [봇]의 작동 효율이 나빠지는 것을 방지하기 위해 이동하는 데이터의 양을 최소화해야 한다. 복잡한 연산이 필요한 부분은 [봇]내부에서 처리하는 것 보다 외부 API를 적극적으로 활용하여 서버 유지에 필요한 자원을 줄인다..

* + - 1. Dependability Requirement

사람이 사회자를 맡는 경우와 다르게 bot은 게임 도중 발생한 문제를 적절히 해결하고 조율하는 능력이 없다. 따라서, [봇]이 조금이라도 잘못 작동하면 게임의 전체적인 양상이 이상해질 수 있다. 전체적인 게임의 진행을 bot에게 맡기는 이상 [봇]의 dependability가 확실해야 한다.

* + - 1. Usability Requirement

별도의 가이드라인을 제공하지 않아도, [봇]을 사용하는 데 어려움이 없어야 한다. [봇]이 Discord 상에 메시지를 출력할 때마다, 도움말 명령어를 입력하여 자세한 정보를 볼 수 있다는 문구를 첨가하여, 사용자가 도움말을 읽고 [봇]을 사용하는 방법을 익힐 것을 권고한다. 게임 도중 진행 상황을 참여자들이 혼동하지 않도록, 현황 정보를 알려주는 메시지를 눈에 띄게 출력한다.

최대한 명령어의 의존하지 않고, 즉 자연어로서의 명령을 잘 처리하도록 개발해야 한다.

* + 1. Organizational Requirement
       1. Environment Requirement

Discord bot으로서 [봇]의 대부분 기능은 Discord API에 묶여있다.따라서, [봇]은 Discord에서 사용하기에 최적화되어 있어야 한다. Discord는 PC 프로그램과 모바일 어플리케이션 두 종류의 플랫폼을 지원한다. [봇]은 두 플랫폼 모두에서 사용될 것을 고려하여 만들어져야 한다. 즉, 가독성 초점을 두고 모바일 환경과 데스트탑 환경에서의 가독성을 동등한 수준으로 유지해야 한다.

* + - 1. Operational Requirement

[봇]은 자연어의 해석을 위해 Dialogflow API 버전의 업데이트와 자연스러운 자연어 해석을 위해 지속적인 학습을 시킬 수 있도록 설계한다.

1. System Architecture
   1. Objective

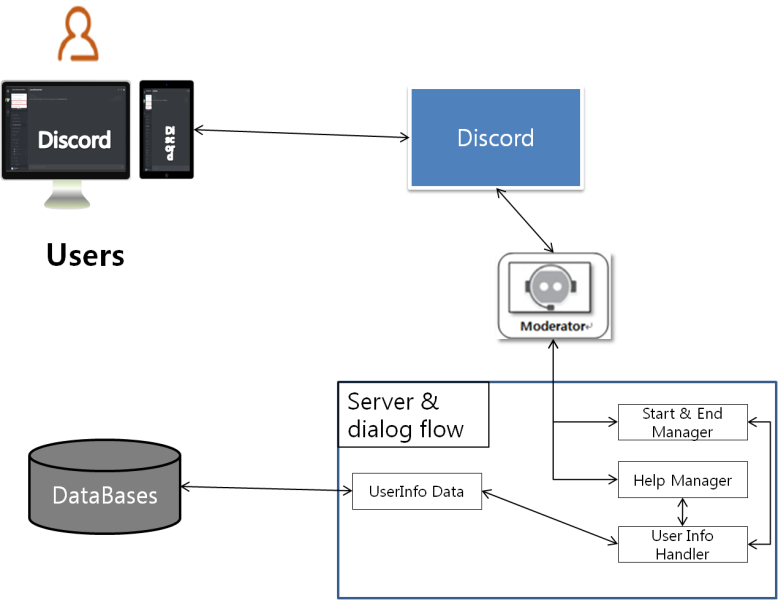
System Architecture에서는 각 Component들의 hierarchy에 대하여 Diagram등을 이용해 서술한다.

* 1. Start System

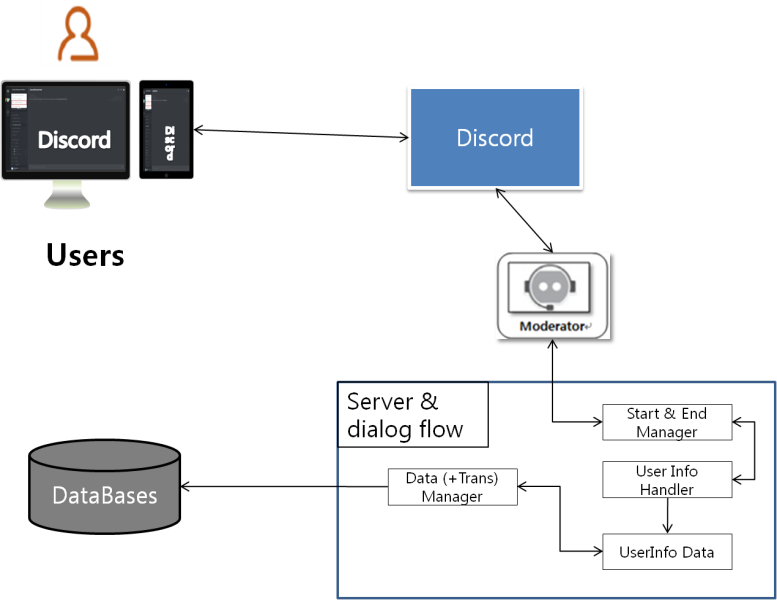
Start System은 한 유저가 게임을 시작을 알리면 [봇]은 게임에 필요한 system들을 초기화 하고, 참가할 참여자를 모집하고 참가할 유저들이 결정되면 참여자들로 구성되어 리스트 형식으로 고유 식별 정보를 받고, 이 정보를 토대로 DB에서 각 참여자의 전적 정보를 받아 구성하고 새로운 참여자는 데이터를 구성한다.

각 참여자들에게 게임에 필요함 정보를 도움방법을 제시하고, 적용될 게임 내에서의 역할과 제약조건들을 부여하고 게임의 시작을 알린다.

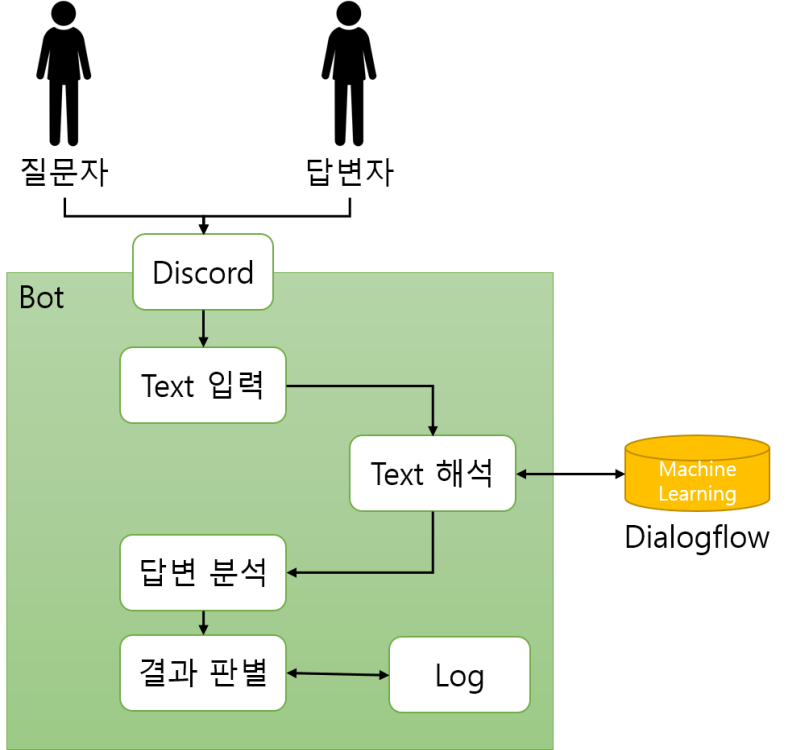
즉, Start & End Manager가 실행되어, Users-List 생성, initialization process를 진행한다.



* 1. End System

 상대팀의 왕을 제거하는데 성공을 했을 때, 승리 팀을 알린다. 그리고 참여자 전적 정보를 DB에 갱신을 하거나 새로이 추가를 하고, 게임을 진행하면서 얻었던 논리식 등을 [봇]의 학습에 이용하기 위해서 dialogFlow에서 사용될 data format(json format)에 맞게 만들어 저장하고 관리한다.

* 1. Q & A System

Q&A System은 질문자의 질문과 답변자의 답변 text를 해석하고,올바른 답변과 실제 답변을 비교하여 결과를 판별하는 시스템이다.질문자가 Discord상에 질문 text를 입력하면, [봇]은 Dialogflow API를 통해서 입력 받은 text를 해석한다.해석된 text는 JSON형태로 번역되며 답변자의 제약조건을 고려하여 올바른 답변이 무엇일지 계산한다.답변자가 Discord상에 답변 text를 입력하면, [봇]은 Dialogflow API를 통해서 입력 받은 text가 긍정의 의미인지 부정의 의미인지 해석한다.해석된 답변이 미리 계산된 올바른 답변과 일치하는지 비교하여,불일치할 경우 답변자에게 패널티를 부과한다.질문과 답변 그리고 결과는 log에 기록된다.

* 1. Elimination System

Elimination System은 참여자가 특정인을 상대 팀의 왕으로 지목했을 때 이를 Dialogflow를 이용해 인지한 후 지목된 참여자가 실제로 상대 팀의 왕인지를 확인해 맞을 경우 지목한 참여자가 속한 팀의 승리를 선언하고 End System으로 넘긴다. 틀렸을 경우 지목한 참여자를 게임에서 제외하고 그 참여자의 죽음을 선언한다.

Q&A System

Elimination System

End System

TEXT 분석

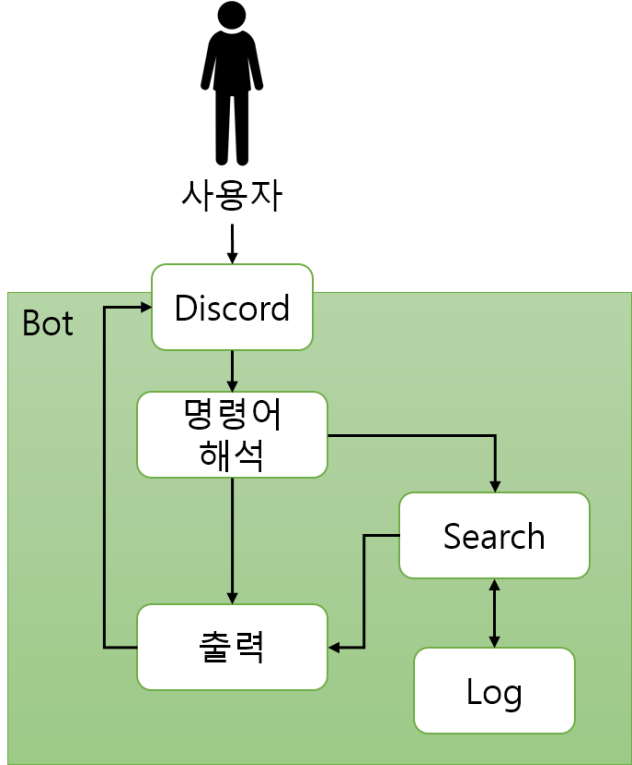
결과 판별

Log

Player List 변경

* 1. Help / log System

사용자가 Discord상에 help 또는 log 명령어를 입력하면, [봇]은 그에 따른 반응을 취한다. Help 명령어가 입력되면 [봇]은 사용할 수 있는 명령어들의 기능을 설명한 도움말을 Discord 상에 출력한다. Log 명령어가 입력되면 [봇]은 저장된 log 중 필요한 내용을 검색하여 Discord 상에 출력한다



* 1. DB management System

End System에서 DB 갱신 요청이 들어오면 참여자들의 소속을 조회하여 승리 및 패배, 생존 여부 등을 DB에 저장한다. 또한 게임 중 언제라도 참여자가 명령어를 통해 자신의 전적 조회를 요청할 경우, DB에서 전적을 조회해 이를 DM(Direct Message)를 통해 참여자에게 제공한다.

End System

DB Management System

Player

DB 업데이트

DB 조회

전적

DB

1. System Requirement Specification
   1. Objective

System Requirement Specification에서는 User Level에서 정의되었던 Functional/Non-Functional Requirement들을 Developer Level로 세분화하여 서술한다.

* + 1. 게임의 사회자

[봇]은 사회자로서 다음의 기능들을 관리한다.

* + - 1. Start / End Game

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Start function |
| 설명 | User가 게임을 시작하기 위해 요청하는 기능이다. 사용자는 원하는 게임을 진행하기 위해서 [봇]에게 요청한다.  해당 기능을 사용하게 되면, 다음과 같은 단계를 거친다.  게임 시작(준비단계) -> (향후 계획 : 게임 선택) ->참여자 검사 및 리스트 생성 -> 역할 및 제약 조건 배정 -> 게임 시작 알림 |
| 입력 | 1. [봇]에게 시작 명령어  2.(게임 선택) |
| 출력 | 게임 시작 메시지 알림. |
| 처리 | 참여할 참여자를 검사하고, 각 참여자에게 역할 및 게임 내 제약조건 배정 |
| 조건 | 참여자의 정보를 생성하는데, 기존에 게임 경험이 있던 참여자의 정보를 DB에서 가져오고, 처음하는 참여자는 새로 게임 info를 생성 |

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | End function |
| 설명 | 게임이 끝나기 위한 조건을 elimination function이 진행된 후에 검사를 한 후에 조건이 성립하는 경우 게임의 결과에 따른 참여자의 승/패 정보 등의 전적 기록을 갱신 요청, DB에 info 갱신, 원활한 질의 해석을 위해 학습 요청 등을 한 후에, 승리 팀을 알리고 게임을 종료한다. 반대로, 조건이 성립하지 않는다면, 계속 게임을 진행한다. |
| 입력 | Elimination function에서 출력된 승/패 정보 |
| 출력 | 승리 팀을 알리는 메시지 출력한다. |
| 처리 | \* 참여자들에 관련된 DB 갱신  \* 받은 질의 처리로 하여금 봇의 학습 요청  \* 종료 조건 완료 시 위의 설명과 같이 메시지를 출력 |
| 조건 | Elimination function이 진행된 후에 반드시 실행되어야 한다. |

* + - 1. Q & A Management

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Question analysis function | |
| 설명 | 질문자가 입력한 질문 text를 해석하고 올바른 답변을 계산한 뒤, 답변대상자가 답변text를 입력하는 것을 요구하는 기능이다. | |
| 입력 | 질문자가Discord에 질문text를 입력한다. | |
| 출력 | 올바른 답변을 계산하여, 비교를 위해 저장해둔다.  질문text를 보여주며, 답변대상자에게 답변할 것을 요구하는 메시지를 출력한다. | |
| 처리 | \*입력된text를 Dialogflow API를 통해 JSON형태로 번역한다.  \*JSON 논리식과 답변대상자의 제약조건을 통해 올바른 답변을 계산한다.  \*받은 질문 text를 다시 보여주며, 답변대상자에게 답변을 할 것을 요구한다. | |
| 조건 | 입력된text가 Dialogflow를 통해 질문으로 판별된 경우만 처리한다.  답변대상자가 답변을 할 때까지 다른 질문text를 수용하는 기능은 멈춘다. | |

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Answer distinguishment function | |
| 설명 | 답변자가 입력한 답변 text를 해석하고, 긍정의 의미인지 부정의 의미인지 판별하는 기능이다. | |
| 입력 | 답변자가Discord에 답변text를 입력한다. | |
| 출력 | 답변text를 판별하여 저장한다. | |
| 처리 | 입력된text를 Dialogflow API를 통해 긍정의 의미인지 부정의 의미인지 판별한다.  판별된 의미를 비교를 위해 저장한다. | |
| 조건 | 입력된text가 Dialogflow를 통해 답변으로 판별된 경우만 처리한다.  긍정 또는 부정의 의미만 수용한다. | |

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Answer comparison function | |
| 설명 | 답변자가 입력한 답변의 의미와 계산된 올바른 답변이 일치하는지 확인한 뒤,그에 따른 결과를 실행하는 기능이다. | |
| 입력 | 시스템에 의해서 올바른 답변이 계산되고, 답변자의 답변이 판별된다. | |
| 출력 | 결과를log에 저장한다.  입력된 답변과 올바른 답변이 일치하지 않는다면, 답변자에게 패널티를 부과한다. | |
| 처리 | 올바른 답변과 답변자의 답변이 일치하는지 확인한다.  일치할 경우, 결과를 log에 저장하고 Question Analysis Function을 다시 활성화시킨 뒤 게임을 계속 진행한다.  불일치할 경우,답변자에게 패널티를 부과한다. 이후 결과를 log에 저장하고Question Analysis Function을 다시 활성화하여 게임을 계속 진행한다. | |
| 조건 | 부과된 패널티에 따라서 답변자가 게임에서 제외되는 경우, End Game Function을 통해 게임이 종료됐는지 확인한다. | |

* + - 1. Elimination

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Elimination function |
| 설명 | 참여자가 특정 참여자를 상대 팀의 “왕”으로 지목했을 경우, 맞았으면 지목한 참여자의 팀이 승리하고, 틀렸으면 지목한 사람의 죽도록 하는 기능이다. |
| 입력 | Answer distinguishment function에서 출력된 JSON 객체 |
| 출력 | 해석한 바를 바탕으로 정말 Elimination을 계속 진행할 것인지 확인한다. 확인후 지목한 바가 맞을 경우, 지목한 참여자의 팀의 승리를 선언하고 End function에 승/패 정보를 전달한다. 틀렸을 경우, 지목한 참여자를 참여자 리스트에서 지우고 참여자의 죽음을 선언한다. |
| 처리 | \*입력된 JSON 객체를 자연어로 변환한다.  \*변환된 자연어를 다시 보여주며, 정말 Elimination을 계속 진행할 것인지 확인한다.  \*확인을 받은 경우 Elimination을 진행한다.  \* 확인후 지목한 바가 맞을 경우, 지목한 참여자의 팀의 승리를 선언하고 End function에 승/패 정보를 전달한다. 틀렸을 경우, 지목한 참여자를 참여자 리스트에서 지우고 참여자의 죽음을 선언한다. |
| 조건 | Answer distinguishment function을 통해 [지목 단계]로 판별된 경우만 처리한다. |

* + - 1. Help / Log

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Help function | |
| 설명 | 사용자가 도움말 명령어를 입력하면, 사용 가능한 명령어, 게임의 룰 등의 설명이 적힌 메시지를 출력하는 기능이다. | |
| 입력 | 사용자가 Discord에 !help <argument> 명령어를 입력한다. | |
| 출력 | <argument>에 해당되는 설명문 메시지를 Discord 상에 출력한다. | |
| 처리 | \* <argument>에 아무 인자를 넣지 않으면, 기본적인 명령어와 다른 <argument> 종류에 대한 설명문을 출력한다.  \* <argument>에 넣은 인자에 따라서, 게임의 룰 설명, 전적 관련 명령어 등에 대한 설명문을 출력한다. | |
| 조건 | 존재하지 않는 인자가 입력된 경우 경고 메시지를 띄워준다. | |

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | Log function | |
| 설명 | 사용자가 log 명령어를 입력하면, 게임 진행 동안 이루어진 문답 기록을 볼 수 있다. | |
| 입력 | 사용자가 Discord에 !log <player> 명령어를 입력한다. | |
| 출력 | 저장된 게임 기록으로부터 요구 받은 정보들을 찾아서 Discord 상에 출력한다. | |
| 처리 | \* <player>에 아무 인자를 넣지 않으면, 모든 문답 기록을 출력한다.  \* <player>에 특정 참여자의 이름을 넣으면, 그 참여자가 대답한 문답 기록만 출력한다. | |
| 조건 | 게임 진행 도중에만 사용할 수 있다.  존재하지 않는 참여자의 이름이 입력된 경우 경고 메시지를 띄워준다. | |

* + - 1. 전적 기록

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | DB Management function |
| 설명 | End function에서 호출되면 승/패 정보와 생존 여부를 각 참여자들 DB에 저장한다. |
| 입력 | 참여자와 승/패, 생존여부가 담긴 리스트 |
| 출력 | 처리한 데이터를 DB에 저장 |
| 처리 | \* 참여자들에 관련된 DB 갱신 |
| 조건 | End function에서 호출되면 반드시 실행되어야 한다. |

|  |  |
| --- | --- |
| Functional requirement | |
| 기능 | DB Quarry function |
| 설명 | 사용자가 명령어를 통해 개인 전적 조회를 요청할 경우 이를 DM을 통해 제공한다. |
| 입력 | 전적 조회 명령어, 사용자 토큰 |
| 출력 | 개인 전적을 DM을 통해 제공 |
| 처리 | \* 전적 조회 명령어를 판별  \* 사용자 토큰을 통해 DB에서 전적 데이터 조회  \* DM을 생성하고 전적 데이터와 데이터에서 계산된 승률, 생존율을 추가하여 전달. |
| 조건 | 전적 조회 명령어가 들어올 경우 실행되어야 한다. |

* 1. Non-Functional Requirements
     1. Product Requirement
        1. Performance Requiement

Discord에 입력되는 text들을 빠르고 정확하게 처리하여 게임에 혼선을 빚는 일이 없어야 한다. 자연어 text를 해석해야 하는 경우, [봇]이 오역을 하는 일이 없도록 확실한 학습이 필요하다

예 ) 1초 이내에 각 프로세싱에 대해 결과를 출력할 수 있도록..

* + - 1. Security Requirement

[봇]을 사용할 때 발생할 수 있는 개인정보와 보안 문제점들은 대부분 Discord의 시스템에 기인한다. [봇]과 사용자 사이에는 항상 Discord가 존재하기 때문에, Discord 상에 보안 문제가 없다면 [봇]을 사용할 때도 문제가 발생하지 않는다. [봇]이 자연어를 처리하는 과정에서 개인 정보가 포함된 text를 데이터 베이스에 저장하는 일을 방지하기 위해서, 모든 text는 게임에 연관된 정보만을 논리식으로 번역하여 다루도록 한다.

[수정이 필요할 것 같음.]

* + - 1. Efficiency Requirement

[봇]은 서버에 의해서 항시 실행되는 상태를 유지하고 있다. 따라서, 서버에 부하가 걸려 [봇]의 작동 효율이 나빠지는 것을 방지하기 위해 이동하는 데이터의 양을 최소화해야 한다. 복잡한 연산이 필요한 부분은 [봇]내부에서 처리하는 것보다 외부 API를 적극적으로 활용하여 서버 유지에 필요한 자원을 줄인다.

communication에 필요한 data structure의 구조의 attr들을 최대 5~6개로 함으로써 json 객체의 크기를 줄여 서버와 클라이언트 간의 통신을 효율적으로 한다.

* + - 1. Dependability Requirement

*// 아래의 문단을 삭제하거나 수정이 필요.*

*[봇]에게 충분한 자연어 해석 학습을 시켜도 오역을 하는 경우가 발생할 수 있다. 이런 경우 봇이 해석한 바를 참여자에게 전달하지 않으면, 사용자의 해석과 시스템의 해석 사이에서 발생한 차이가 메워지지 않은 채로 게임을 혼란스럽게 만들 수 있다. 이를 방지하기 위해서 [봇]은 해석한 바를 참여자가 볼 수 있게 출력한다. 오역이 발생하지 않은 경우 [봇]이 올바르게 작동하고 있음을 시각적으로 확인할 수 있고, 오역이 발생하더라도 참여자가 정정할 기회를 줘서 [봇]에 대한 신뢰도가 높아진다.*

사람이 사회자를 맡는 경우와 다르게 bot은 게임 도중 발생한 문제를 적절히 해결하고 조율하는 능력이 없다. 따라서, [봇]이 조금이라도 잘못 작동하면 게임의 전체적인 양상이 이상해질 수 있다. 전체적인 게임의 진행을 bot에게 맡기는 이상 [봇]의 dependability가 확실해야 한다.

자연어에서 논리식으로 해석 및 번역의 정확도를 90% 이상으로 해야 한다.

* + - 1. Usability Requirement

별도의 가이드라인을 제공하지 않아도, [봇]을 사용하는 데 어려움이 없어야 한다. [봇]이 Discord 상에 메시지를 출력할 때마다, 도움말 명령어를 입력하여 자세한 정보를 볼 수 있다는 문구를 첨가하여, 사용자가 도움말을 읽고 [봇]을 사용하는 방법을 익힐 것을 권고한다. 게임 도중 진행 상황을 참여자들이 혼동하지 않도록, 현황 정보를 알려주는 메시지를 눈에 띄게 출력한다.

명령어의 종류의 사용법, 즉 도움 명령어의 사용법, 전적 확인 및 보기 명령어의 사용법 등을 보여준다. 예) !help , !request

* 1. Scenario
     1. Join Scenario

사용자가 여러 명 모인 챗방에서 [봇]을 초대하면, [봇]이 '무슨 게임을 시작 하시겠습니까' 로 하여금 게임 종류를 받고, 참여 인원 조사를 하고 게임을 시작하는 메시지를 출력하고, 이에 따라 [봇]이 참여자의 역할 배정 등의 초기화 작업을 한다.

* + 1. Question & Answer Scenario

참여자가 한 명이 전체에게 질문을 하던지 혹은 한 명을 지목하여 질문을 할 수 있다. 이에 지목 당한 사람(들)은 제약 조건에 따라서 답변을 하고, 이 답변(들)을 [봇]이 이 부분에서 대해서 제재를 해야 한다.

* + 1. Elimination Scenario

어느 한 참여자가 다른 참여자 한 명을 지목하여 '너 상대방 팀 왕이지!'라고 하면, 봇이 이것을 확인을 하고 지목 당한 참여자가 왕이 아닐 경우, 지목한 참여자를 제거(제외)시킨다 혹은 왕이 맞을 경우 지목한 참여자가 있는 팀이 승리한다.

* + 1. Winning Scenario

모든 참여자의 전적 사항을 업데이트를 해서 DB 안에 데이터의 갱신이 이뤄져야하고, [봇]은 승리 팀의 승리를 했다는 것을 메시지로 출력한다. 사용자가 원할 경우 새로운 게임을 시작하고 처음 과정으로 되돌아간다.

* + 1. Managing User profile Scenario

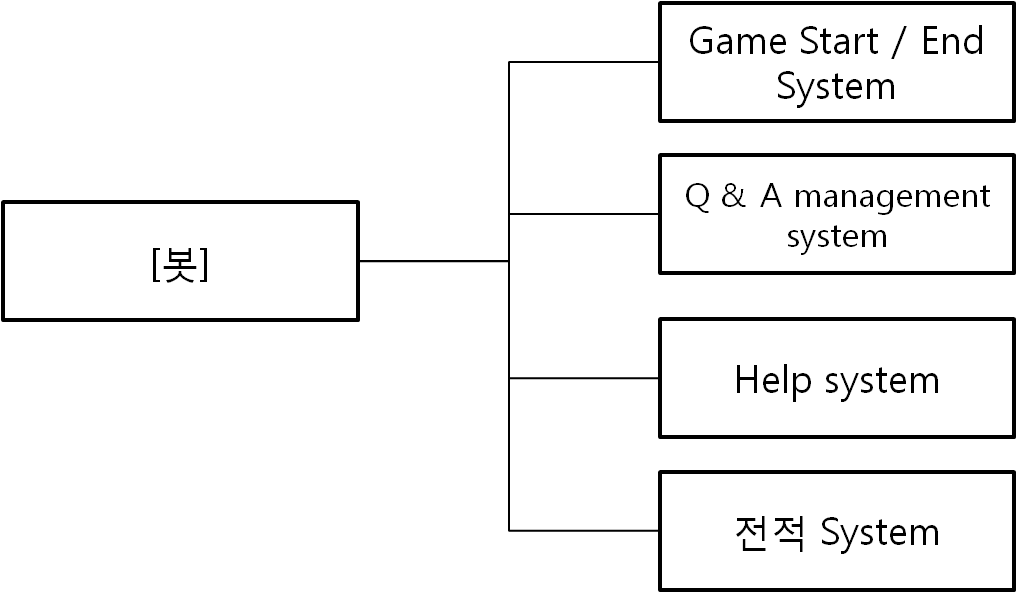
User가 본인의 전적 사항을 요청 시에 DB에 저장된 그 user의 정보를 가져와 출력한다. 초기화를 원할 경우 초기화 해준다.

*/// 상대의 전적 사항을 보여주는 건 보류*

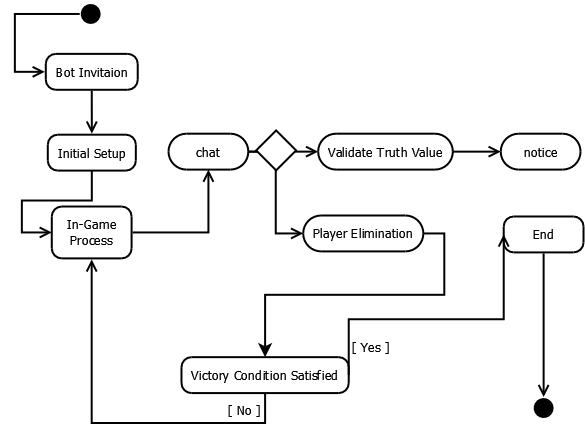
1. System Models
   1. Objective

System Models에서는 System Component, System, 그리고 System Environment 사이의 관계를 Figure 및 Diagram을 통해 서술한다.

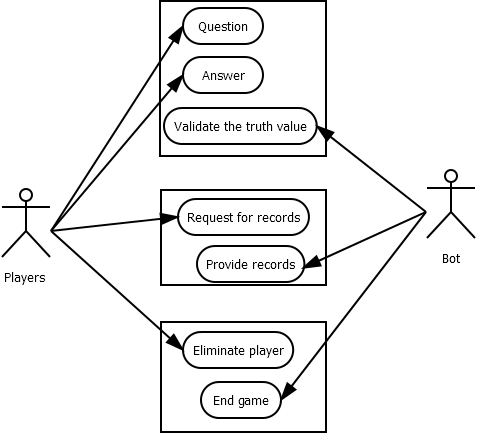
* 1. Context Models
     + 1. Context Model



* + - 1. Processes Model



* 1. Interaction Models
     + 1. Use case model



* + - 1. Tubular description for each Use-Case
         1. Question

|  |  |
| --- | --- |
| Sub-system | Q & A system |
| Use-Case | Question |
| Actor | Player, Bot, Dialogflow |
| Description | Player로부터 입력 받은 text를 Dialogflow를 통해서 JSON 논리식으로 번역한다. |
| Stimulus | Text entered by the player |
| Response | Translate text to JSON |
| Comments |  |

* + - * 1. Calculate correct answer

|  |  |
| --- | --- |
| Sub-system | Q & A system |
| Use-Case | Calculate correct answer |
| Actor | Bot |
| Description | JSON 논리식과 답변 대상자의 제약 조건을 고려하여 올바른 Boolean 답변을 계산한다. |
| Stimulus | JSON 논리식 |
| Response | Calculate Boolean value |
| Comments |  |

* + - * 1. Answer

|  |  |
| --- | --- |
| Sub-system | Q & A system |
| Use-Case | Answer |
| Actor | Player, Bot, Dialogflow |
| Description | Player로부터 입력 받은 text를 Dialogflow를 통해서 Boolean 형태로 바꾼다. |
| Stimulus | Text entered by the player |
| Response | Translate text to Boolean |
| Comments |  |

* + - * 1. Validate the truth value

|  |  |
| --- | --- |
| Sub-system | Q & A system |
| Use-Case | Validate the truth value |
| Actor | Bot |
| Description | 계산된 correct answer와 입력 받은 answer를 비교한다. 불일치 하다면 대상자에게 패널티를 부과한다. |
| Stimulus | Boolean value |
| Response | Penalize the player |
| Comments |  |

* + - * 1. Request for records

|  |  |
| --- | --- |
| Sub-system | Log systems |
| Use-Case | Request for personal records |
| Actor | Player, Bot |
| Description | If a player requests for one’s personal record, bot recognize that command and search DB for the data. |
| Stimulus | Request from a player |
| Response | Search DB |
| Comments |  |

* + - * 1. Provide records

|  |  |
| --- | --- |
| Sub-system | Log systems |
| Use-Case | Provide records |
| Actor | Player, Bot |
| Description | If the record is found from DB, bot creates DM and deliver the data to player. |
| Stimulus | Data found in DB |
| Response | Deliver data to player |
| Comments |  |

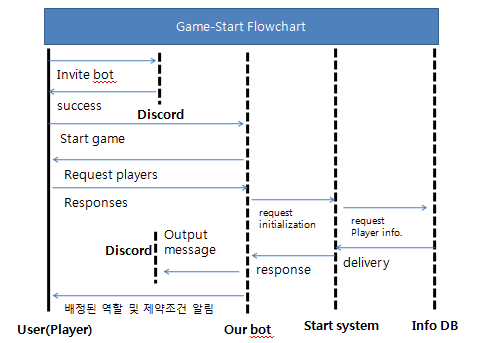
* + - * 1. Eliminate player

|  |  |
| --- | --- |
| Sub-system | Elimination system |
| Use-Case | Elimination |
| Actor | Player, Bot |
| Description | If a player points out someone as a “king” of other team, then bot should check whether that is correct or not. If it is correct, the team of player who pointed out right wins. Otherwise, the player dies. |
| Stimulus | JSON object from Question distinguishment function |
| Response | Call End function/ eliminate player from players list |
| Comments |  |

* + - * 1. End game

|  |  |
| --- | --- |
| Sub-system | Start & End system |
| Use-Case | End game |
| Actor | Bot |
| Description | After a elimination phase occurs, bot should check whether end condition is satisfied. if satisfied, act game-end through start / end system. |
| Stimulus |  |
| Response | If the condition is satisfied, send a message to all. |
| Comments |  |

* + - 1. Sequence Models



//시작 flowchart

* 1. Structural Models
     + 1. ... // 예시로 하나만 작성하죠.
       2. ...
       3. ....
       4. ....
       5. ....
       6. .....
  2. Behavioral Models
     + 1. ...
       2. ...
       3. ....
       4. ....
       5. ....
       6. .....

1. System Evolutions
   1. Objective

System Evolution에서는 시스템이 세우고 있는 주요한 가정들과 사용자의 Needs 변화, 혹은 추후 Update될 사항들에 대해서 서술한다.

* 1. Limitations and Assumption
  2. Evolutions of User Requirement
     1. Voice Recognition-Based System
     2. Rule Modification Sandbox
     3. New game mods

1. Appendices
   1. Objective

Appendices에서는 본 프로젝트와 관련한 참고사항들을 서술한다.

* 1. Database Requirements
  2. User-System Requirements
  3. Development Process

1. Index
   1. Table Index
   2. Figure Index
   3. Diagram Index
2. Reference