CS2013/3013 - Group 4 - The Turing Game

Software Design Specification

1. Introduction

1.1. Overview - Purpose of system

The purpose of this system is to enable users to play a mobile app game that tests their ability to differentiate a bot(AI) from a human player. More specifically, the user enters a chat conversation paired randomly with either a bot or a human player. The user is given a specific topic to talk about for a specific time period, after which the user must guess whether they were talking to a human or a bot.

1.2. Scope

For this project we will need to create a small server to connect the human players, we will do so using the cloud messaging/matchmaking service. We will make use of IBM's watson servers and integrate them into the project to generate the bot's dialog to humans. We will be using Android Studio to develop the app using the Java language and the Android SDK.

1.3. Definitions, abbreviations

SDK - A software development kit (SDK) is typically a set of software development tools that allows the creation of applications for a certain software package and system.

AI - Artificial Intelligence is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals.

1.4. References

CCNA Data Center DCICT 640-916 Official Cert Guide; Shamsee, N.Klebenov, D. Fayed, H.;Cisco; p. 934.

https://en.wikipedia.org/wiki/Artificial_intelligence

https://www.ibm.com/watson/about/

https://en.wikipedia.org/wiki/Turing_test

2. System Design

2.1. Design Overview

As mentioned before, the purpose of this project is to test the user's ability to differentiate between an interaction with a bot (AI) and with a human player. We aim to do this by creating a server that will randomly connect users to other human players, and also match users to bots. We will assign topics to limit the conversation and train bots on certain topics. In the future this could be built upon by having many different bots for different topics, but as this is the first version we will be keeping the bot's conversation limited.

2.1.1 High-level overview of how the system is implemented, what tools, frameworks and languages are used etc.

In terms of the network/backend architecture, the app is essentially just a housing for displaying the messages that come in from the bot / the messaging service. In terms of the interaction, the plan was to wait for a certain amount of time to see if a human could be paired and then make a random choice between the bot/human. If no human can be paired, just talk to the bot. For the bot we will use IBM watson, which allows us to essentially just make api requests using a certain key and url that relates to a certain "assistant". This assistant will be trained on a certain topic (i.e. food, cars, etc), as it is not really feasible to implement a bot that can have an open ended conversation. In the future, it may be better to have it randomly select a bot/topic from a database which will store the url & key of the particular assistant. The client has stated that we can assume that if paired with a human, each human will respond promptly (has an interest in playing the game / will not leave an excessive amount of time between responses).

Human to bot:

In terms of this side of the backend, the user will communicate directly with IBM's servers / the bot we choose through api calls and will converse

independently of the matchmaking / chat service. The advantage of this is that the implementation of the bot is pushed off onto the bot providers side, meaning we can focus on how the users see this interaction. The timekeeping and eventual guessing of whether or not the agent is a bot or human will take place locally, within the app, and if required the results can be saved to a database (this may be done using firebase or could be done locally) for score keeping (depending on how the rest of development goes).

Human to Human / Matchmaking: Our current design is based around using Firebase to exchange the messages between players / provide some sort of login system and matchmaking platform

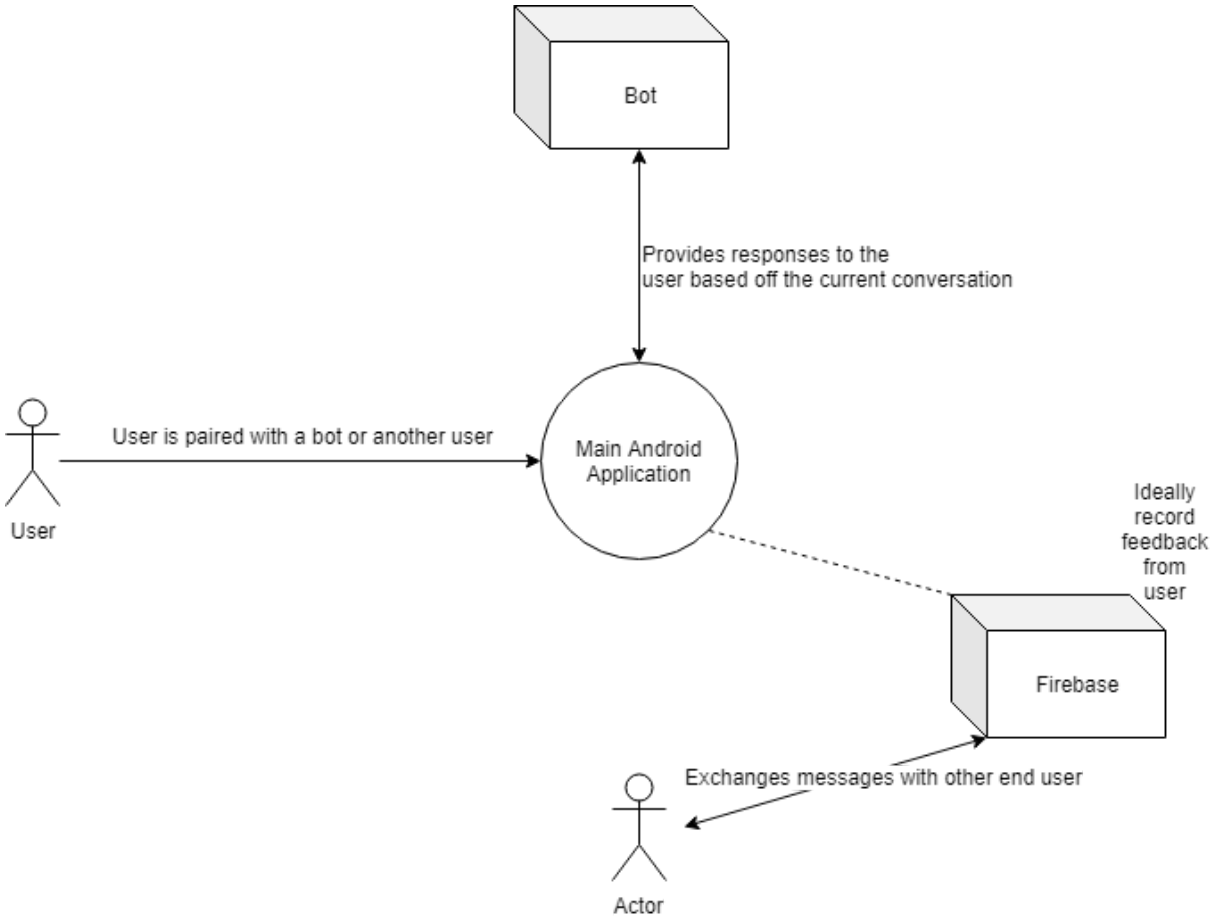
Front-end : All front-end design and implementation will be done through android studio

Tools, frameworks, languages : IBM Watson, Google Firebase, Android Studio, Java.

1.2. System Design Models

1.2.1 System Context

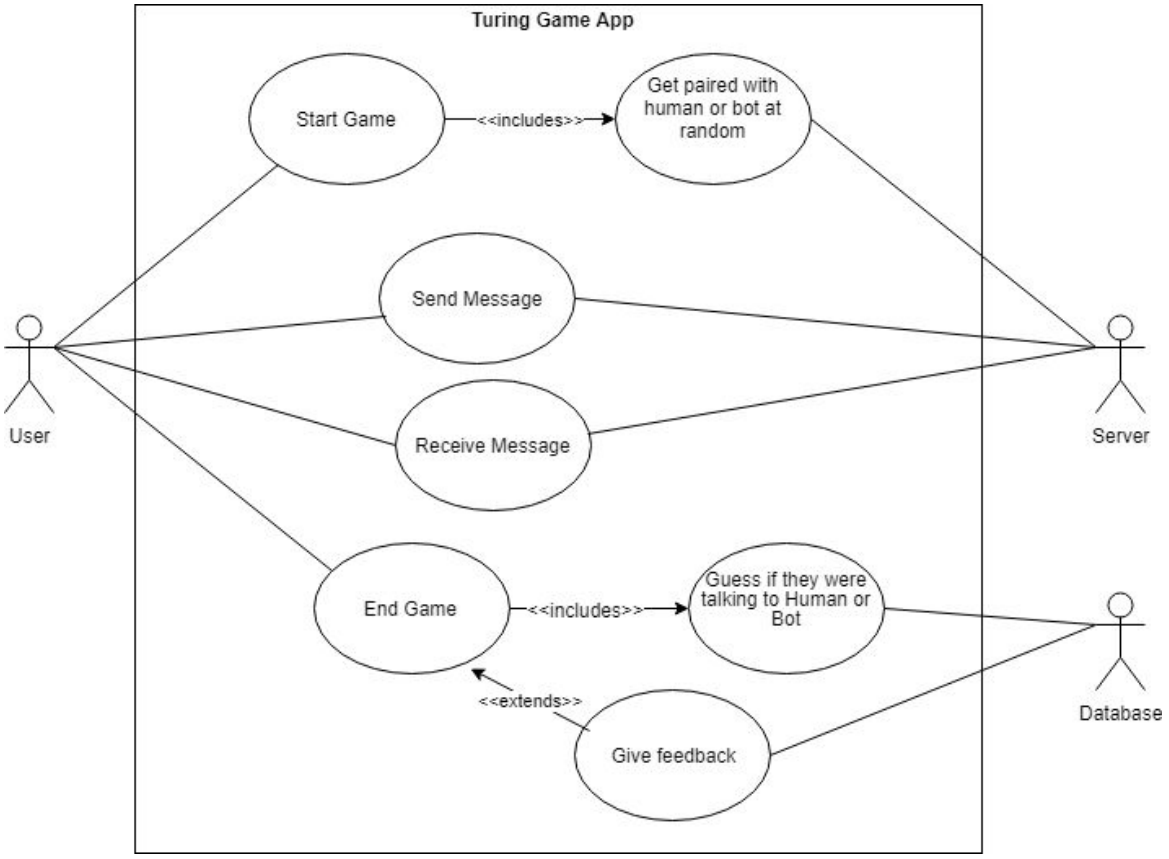
The context of our system will be confined to the domain of AI, specifically AI behaviour and interaction. The user will be paired with either a bot or another person and will have to conclude whether the conversation is with a bot or another human. Feedback from the user will ideally be recorded and stored.



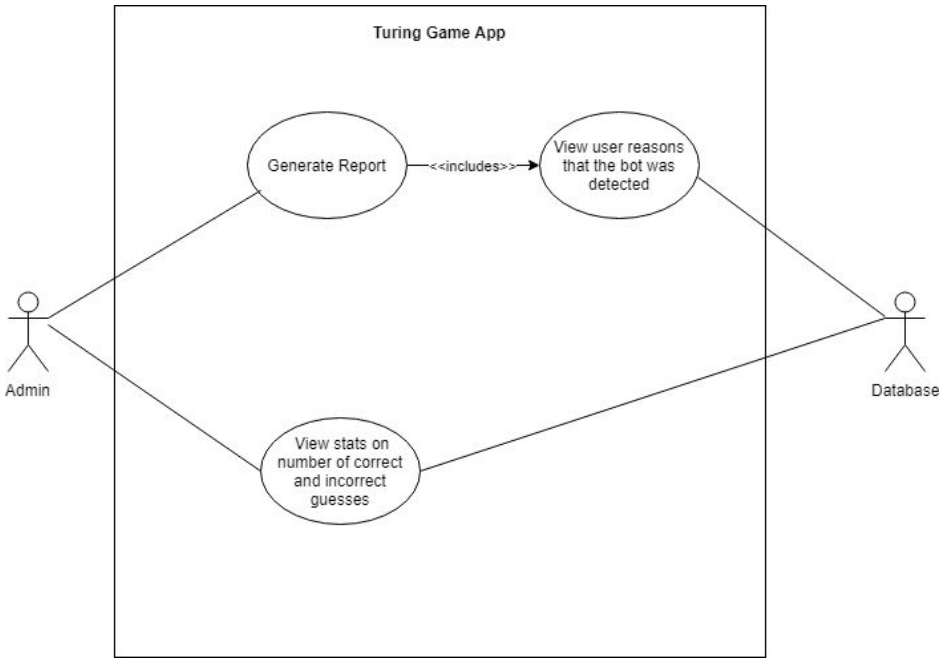
1.2.2 Use Cases

We have at least **2 systems: a server and an android app.**

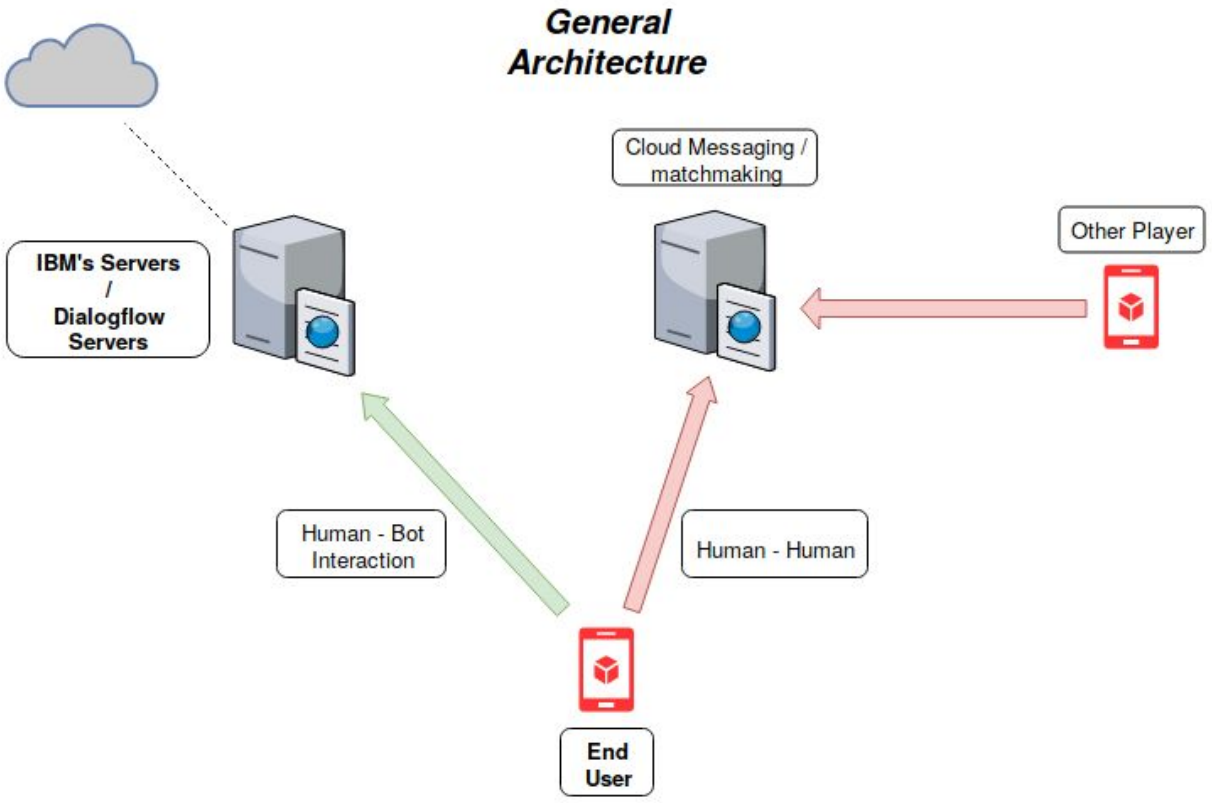
1. For **user** use to play a game



2. For **app developer/admin** use to generate reports and statistics



1.2.3 System Architecture



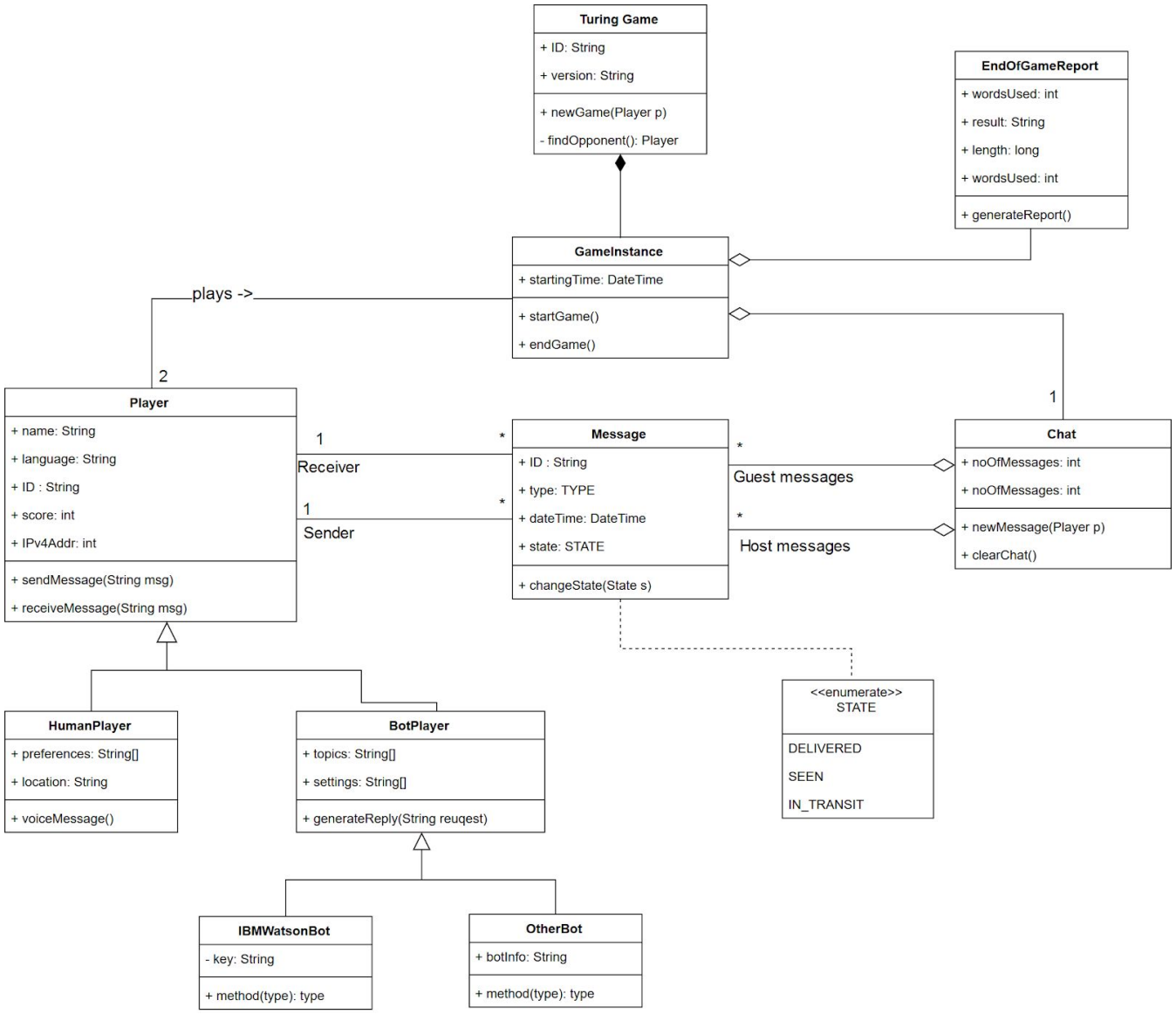
https://firebase.google.com/docs/cloud-messaging/

https://www.ibm.com/watson/

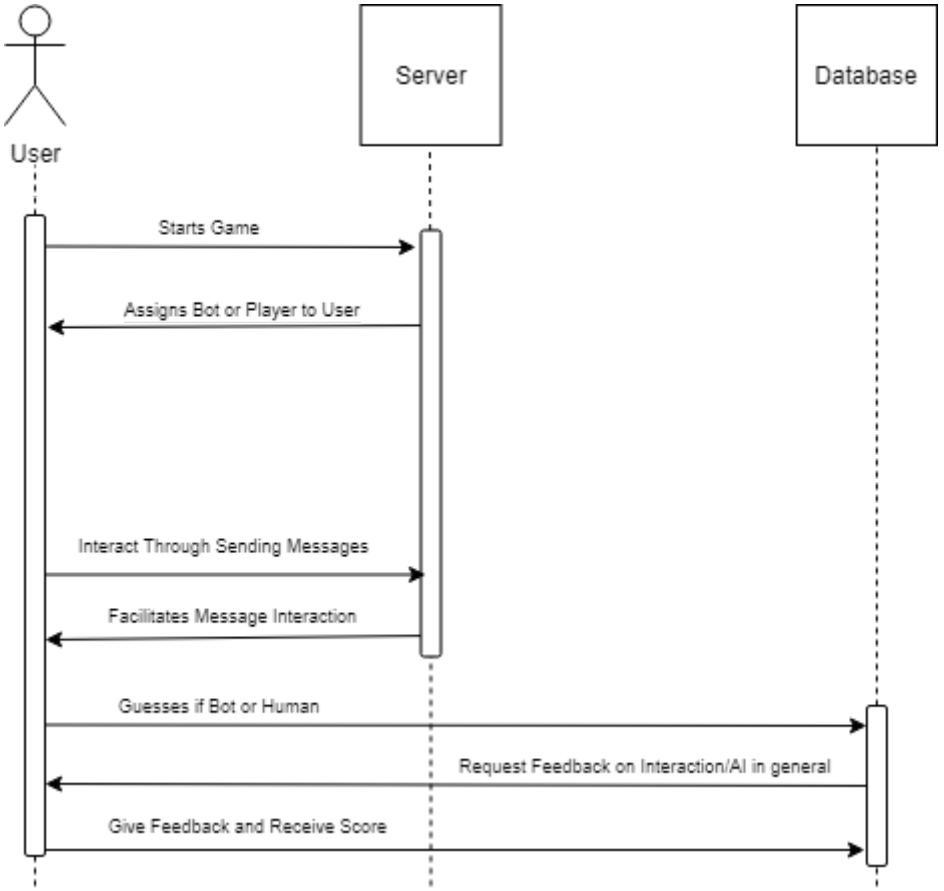
https://firebase.google.com/products/

https://developers.google.com/games/services/

1.2.4 Class Diagrams



1.2.5 Sequence Diagrams



1.2.6 State Diagrams