

CS2013/CS3013

Development Report

TCD ADAPT - Erwan Moreau

Group 20

Andrew Mc Donald
Conor Mac Amhlaoibh
Divine Mbunga
Ethan Monkhouse
Evan Mc Croary
Luke Mc Grath



Index

| | |
|---|-----------|
| Index | 2 |
| 1 Introduction | 4 |
| 1.1 Background and Problem Statement | 4 |
| 1.2 Technical Approach | 4 |
| 2 Requirements | 5 |
| 2.1 Functional Requirements | 5 |
| 2.2 Non-functional Requirements | 5 |
| 2.3 User Interface Scenarios | 6 |
| 3 Design | 8 |
| 3.1 Simplified Architecture Diagram | 8 |
| 3.2 UML Class and Sequence Diagrams | 9 |
| 4 Implementation | 10 |
| 4.1 Tools, Libraries, Platforms | 10 |
| 4.2 User Interfaces | 12 |
| 4.3 Algorithms | 13 |
| 5 Conclusions | 13 |
| 5.1 Design and Implementation | 13 |
| 5.2 Project Objectives | 13 |
| 5.3 The Team | 13 |
| 5.4 Algorithms | 14 |
| Appendix I: Requirements Document | 15 |
| Appendix II: Software Design Specification | 23 |



1 Introduction

1.1 Background and Problem Statement

The project that our group received was Project Number 44, The Turing Game: game modes, and user interface.

The Turing Game is a mobile app envisioned by our client: Erwan Moreau of TCD ADAPT. It is an educational game in which a player must determine whether they are interacting with a human or an artificial intelligence (AI) after engaging them in a short conversation. The project was split up into three parts: client-side, server-side and AI. Our group handled the client-side aspect of the project. The app was designed last year by a group and this year our client had some improvements in mind that we implemented in our user-interface along with our own ideas and design specifications.

1.2 Technical Approach

We identified Android Studio as the most appropriate software to design the Android app that would act as the interface for the game, as the platform provides all the features required, at an exceptionally high standard when it comes to Android application development. The project consisted of one Android app that contains the interface as a whole. When developing the front-end of the application, we didn't want to rely on external parties and teams to test and debug the application. To combat this we implemented a dummy server to hard-wire data into the app so that we could test different inputs and responses without integrating immediately with the other aspects of the project. This ensured that any bugs and errors that arose were at fault of our development, and we could investigate these issues immediately.. The app was developed using Java, PHP and XML. All of our code was uploaded to GitHub to allow all group members to access, collaborate and modify the code.



2 Requirements

Our requirements are displayed under the headings: Functional requirements, Non-functional requirements and User interface scenarios.

2.1 Functional Requirements

- Login to the app using a Google account.
- Initiate a conversation in a specified game mode.
- Receive a user's guess at the end of the conversation (were they speaking to a bot or human).
- Implement a mechanism that allows the user to flag inappropriate content.
- Send API requests for messages, guesses, etc. to the server
- Parse JSON data received from the server.
- Receive feedback from users in the form of a survey.
- Display user rankings on a leaderboards page.
- Show 'typing bubble' when a conversant is typing.

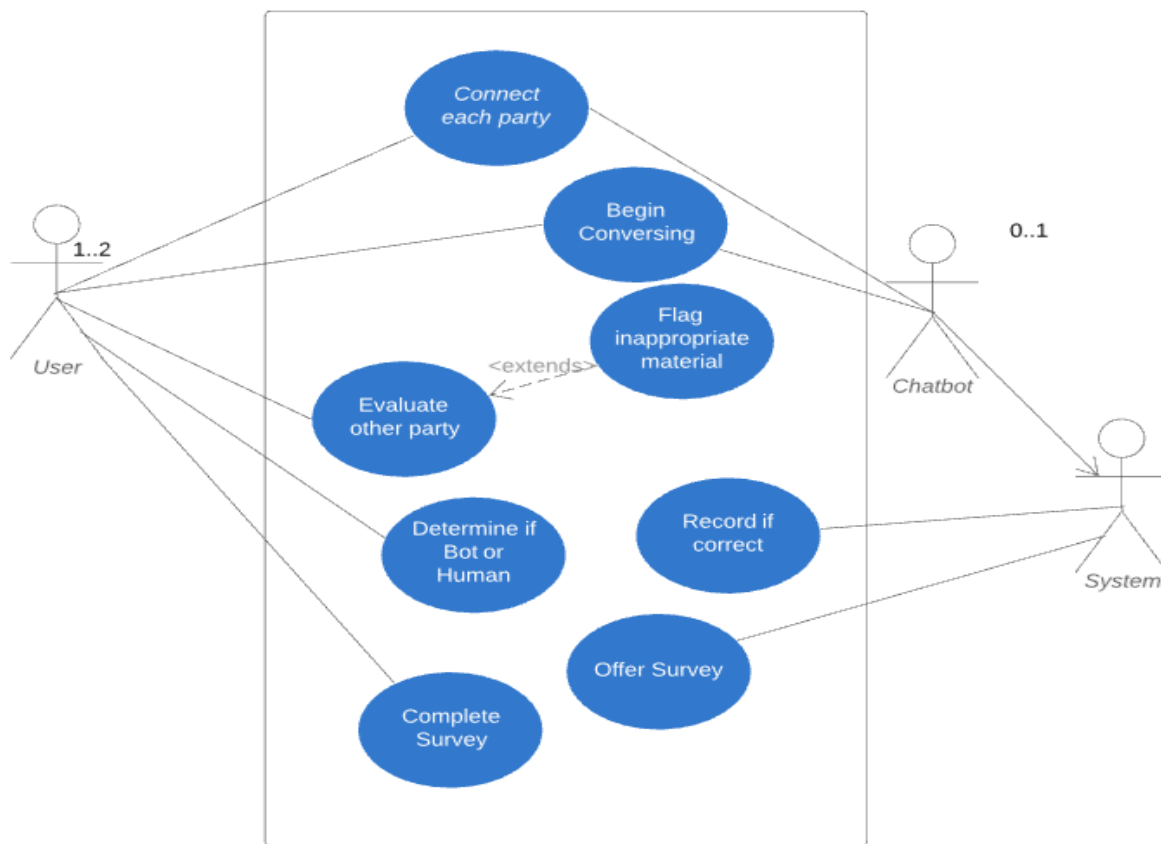
2.2 Non-functional Requirements

- The user interface should be minimal but aesthetic.
- Detailed API documentation should be written.
- API requests should be as efficient as possible.
- User data should be GDPR compliant.

2.3 User Interface Scenarios

Use Cases

Diagram



Textual Descriptions

| | | |
|--------------------------------|----------------|--|
| Use Case Name | | Offer Survey |
| Actors | | System and User |
| Precondtions | | User has finished the game and is still on the app |
| Normal Flow | Description | User has finished the game. The system offers the option of a survey to the user who can accept or decline |
| | Postconditions | User fills out survey for information |
| Alternate flows and exceptions | | User exits app after game and survey cannot be offered |

| | | |
|--------------------------------|----------------|--|
| Use Case Name | | Flag inappropriate material |
| Actors | | Human Player |
| Precondtions | | Player sees a message they believe shouldn't be on the platform |
| Normal Flow | Description | Player sees a message they believe to be inappropriate. They report it. The conversation is terminated and the conversation material is passed onto a trusted third party for review |
| | Postconditions | Player can no longer see material but is being acted upon by a separate party |
| Alternate flows and exceptions | | No exceptions |

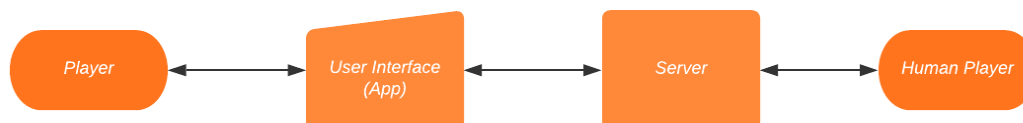
| | | |
|--------------------------------|----------------|---|
| Use Case Name | | Connecting Parties |
| Actors | | Two players or a player and a chatbot |
| Preconditions | | User selects play and is connected to the internet |
| Normal Flow | Description | User connects to server and is matched with either a chatbot or another player |
| | Postconditions | User is ready to play the Turing Game against their partner and guesses if the player is a bot or another human |
| Alternate flows and exceptions | | If the user disconnects or loses connection to the server, the game is stopped and connection failed. |

3 Design

Our project consists of an Android application that implements the client-side aspect of *The Turing Game*. Our application allows a user to converse with other users or an AI for a limited period of time and subsequently guess the nature of their conversant. The application communicates with the server-side part of the project using a shared API.

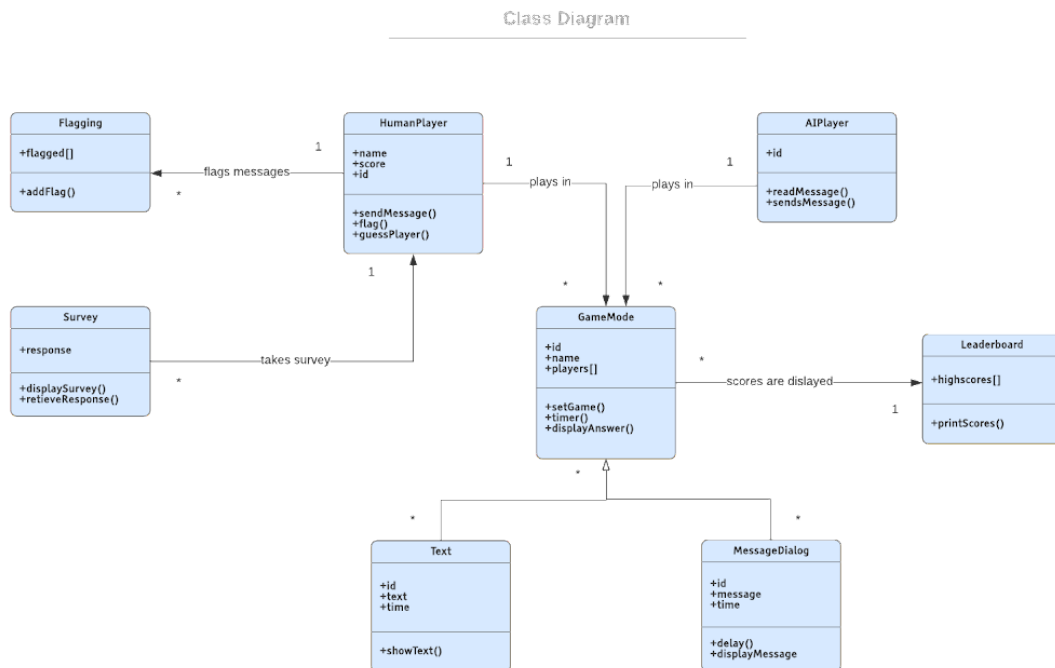
We also had to implement a ‘dummy’ server to take the place of the live serve that the other group would eventually provide. This provided a simple API interface that returned randomised responses to that app’s queries.

3.1 Simplified Architecture Diagram

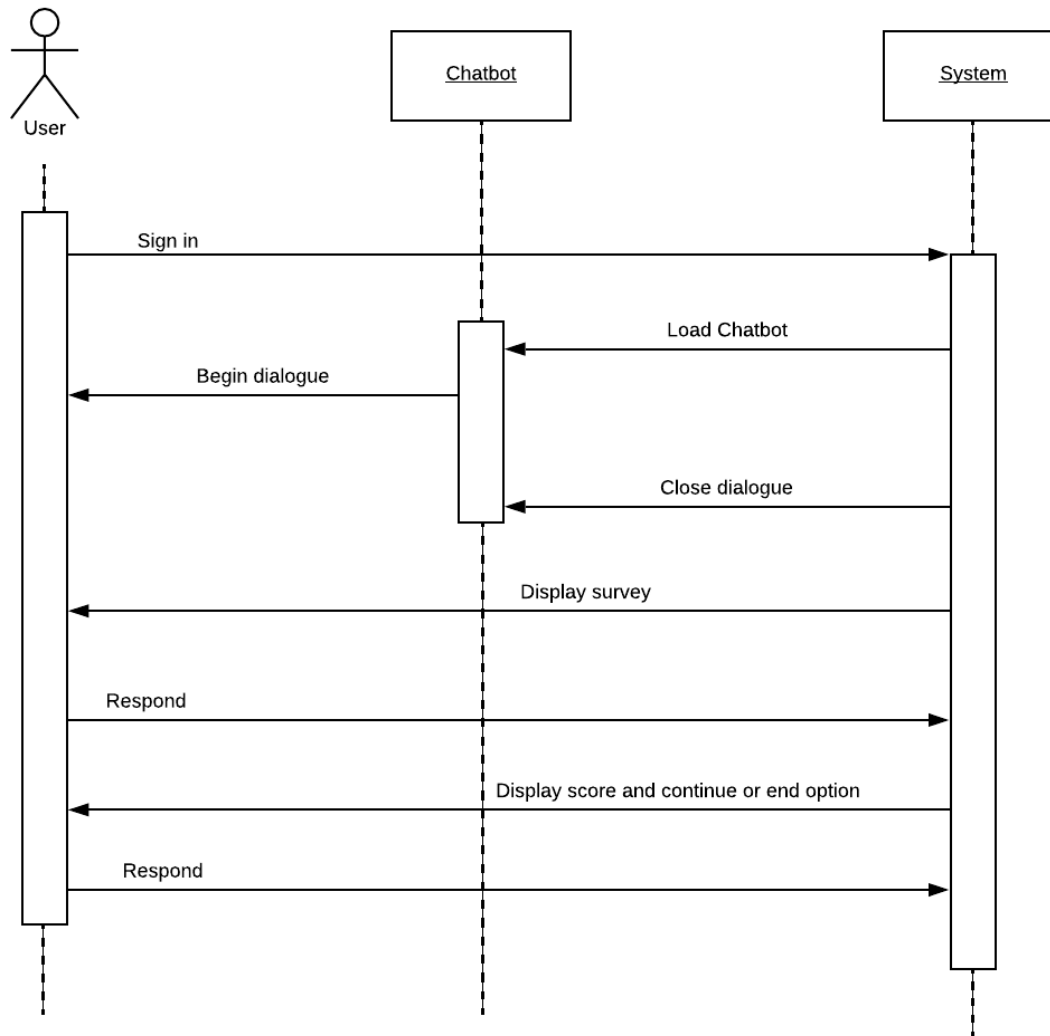


3.2 UML Class and Sequence Diagrams

Class Diagram



Sequence Diagram



4 Implementation

4.1 Tools, Libraries, Platforms



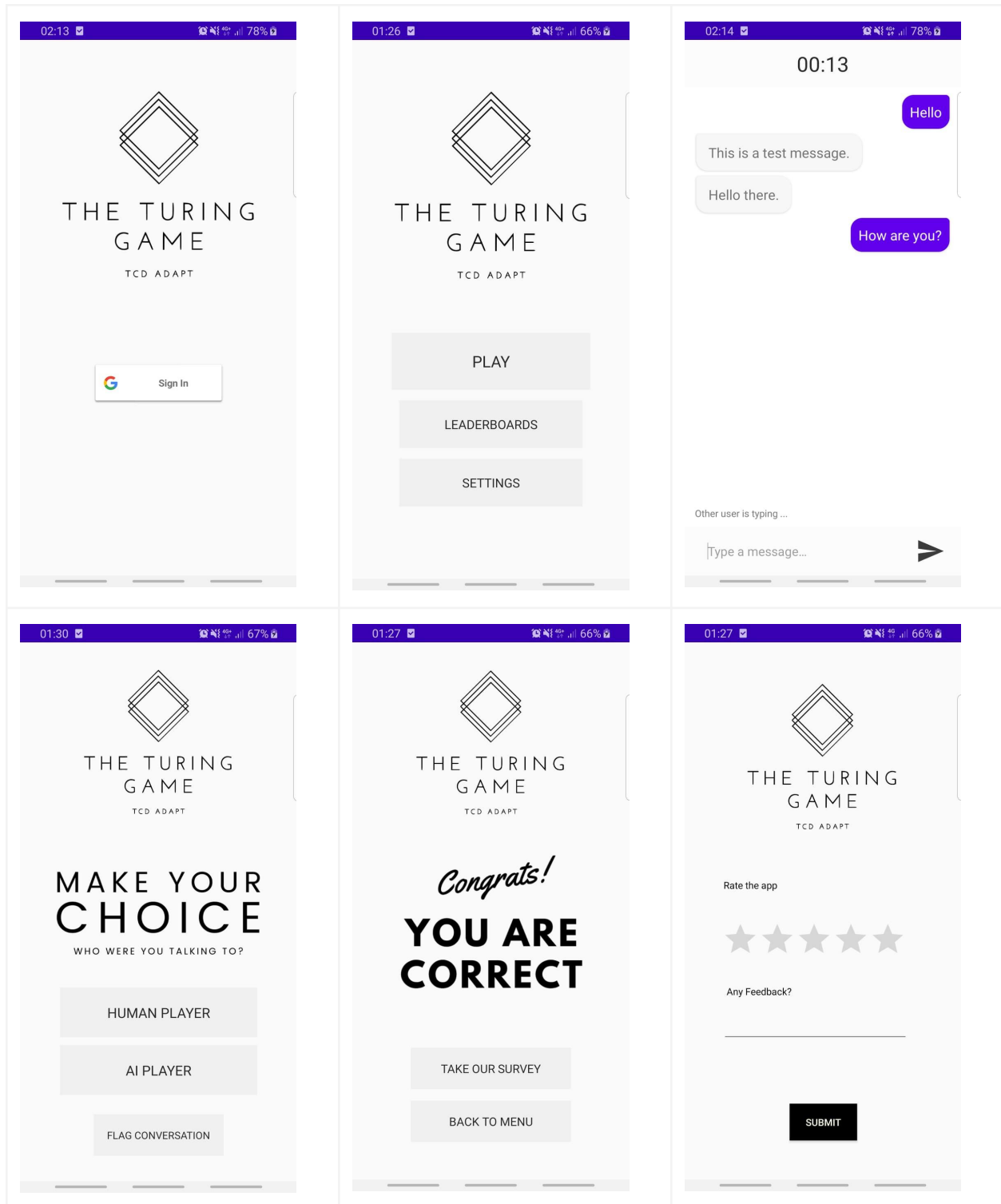
Tools, libraries and platforms used for this project:

- Android Studio was used as the primary development platform. This platform was suggested by the client, as the previous team who developed another version of the project had used the same system. Implementing the design requirements was as expected.
- Firebase - Firebase was used for software development. Firebase took care of the back end (generally!) and allowed us to focus on the front end of our project. As UI and game methods was our job as laid out by the client, this suited perfectly.
- GitHub was used for version control, and the primary code repository. It was necessary that the repository was configured to allow all other teams involved in the various aspects of the project to integrate accordingly. We had to ensure a correct method of version control was adhered to, in order to deliver and communicate the project to the standard agreed with the client, and to ensure a log of all work and changes were recorded in the event of backreferencing.
- Server-side API - The application communicates with the server side of the project using a shared API. We established our API very early so that there would be no question of compatibility even with the Server and AI groups beneath the Turing Game umbrella. As there wasn't any mention of an API at the initial meeting we took the initiative and laid one out. The other groups were asked to give feedback and input but the general consensus was that it suited them perfectly.
- Messenger was used as a method of immediate communication between team members. We had considered using Slack as a primary method of communication, however due to the scale of the project, as well as the size of the team, and from advice from team members, Slack would have been ideal for use in this project. This is primarily due to the immediateness of Messenger compared to Slack. If this were a longer-term project, and a larger, inter-departmental team, our management choice may have been different.
- PyCharm was used to develop the 'dummy' server. It uses the Flask library to implement the features of the API.

Programming Languages Used:

- Java was used with Android Studio to design the methods and functionality of our app.
- PHP was used in combination with XML to design a working and more accessible user experience.
- XML was perfect for outlining every aspect of our application so that we could work better with each other with minimal miscommunications.
- Python was used in the development of the 'dummy' server.

4.2 User Interfaces





4.3 Algorithms

Due to the nature of our project specifications, the algorithm and computation aspect was primarily involving the management of data and issuing requests to the back-end to retrieve the correct, processed data using the correct method. An example of this would be communicating with the server to generate a player, either human or ai. The base algorithm is stored on the server, and the interface will receive the player data through a request. This methodology is continued throughout a large percentage of the interface, as the final deliverable will have the ability to integrate with an external server, and we wanted to ensure this requirement is upheld by introducing a 'dummy' server to handle requests throughout the development of the application.

Furthermore, there were local algorithms implemented to achieve specific design features within the application. These were mostly involving the replication of human activity with the AI player. For example, the 'typing bubble' of a user needs to reflect the length of the text being typed. This was achieved using a simple algorithm to compute the amount of time the 'typing bubble' needs to be displayed, depending on the length of the text.

5 Conclusions

5.1 Design and Implementation


One problem that we had initially was that our group as a whole had little to no experience using the Android Studio software. It was a struggle at first to come to terms with the new software but we made quick progress. Another problem that we encountered was that we had no input for the different outcomes that we had to test in our app so we created a dummy server which we used to hard-wire data into the app and test the different outcomes.

5.2 Project Objectives

Overall we did a great job at reaching our project objectives. We managed to fulfill all of the requirements that we set for ourselves, and also to meet the client's expectations and implement the ideas that he had in mind for us.

5.3 The Team

Working in a group on this project really allowed us to improve both our teamwork and communication skills. It was a great asset to be able to get the opinion and knowledge of other team members to help us each achieve our objectives in completing the project. We worked well as a team and managed to get all of the project components together in time for the end product.



We faced a few problems throughout the design of the project but we managed to overcome them with the help of each other. Circumstances were not always ideal but we managed to maintain a very good level of communication within the group, with our client and also with our demonstrator.

5.4 Algorithms

Overall, the standard of which the communication between elements of the project was implemented at a high level. We experienced a few disruptions in regard to the communication between other teams due to the current situation. Furthermore, a few complications arose in regard to requests to the server to fetch data after a specific method containing an algorithm was called. Again, this was due to cross-platform integrating issues which were resolved as time progressed. Furthermore, the algorithms that were implemented adhered to the efficiency requirements confirmed by the client, and with time we believe we would be able to outperform further expectations in regard to the efficiency and performance of these algorithms within the interface.

Appendix I: Requirements Document

CS2013/CS3013

Requirements Document

Group 20

Andy Mc Donald
Conor Mac Amhlaoibh
Divine Mbunga
Ethan Monkhouse
Evan Mc Croary
Luke Mc Grath



1. Introduction

1.1 Overview - Purpose of the System

In this project three groups will be working on building the Turing Game. The Turing test is popular in the domain of AI. It is a game where a person asks written questions, in the similar to a messaging app, to two entities, player A and player B. These entities will reply through messages. The goal is for the person to guess who between player A and player B is a human and which is the AI. The Turing test game is considered a foundational description of what could be defined as artificial intelligence. If a human cannot recognise the difference between a conversation with a human or with a machine, then this machine shows some form of intelligence.

1.2 Scope


For the Turing Game application, our scope is to design and implement the game modes and user interface of the overall system, focusing on the interface of a sign in component, the dialogue system, a scoring system and a way for the player to flag any inappropriate content from another player or the AI.

1.3 Objective and Success Criteria

The main objective of this project following completion is to have built upon the previous years project, and implemented the features which have been agreed with the client, to a standard that all developers, managers, the client and all other parties involved agree is of a satisfactory level. Our specific objectives *to have achieved by the the completion of this project are the implementation and testing of the Turing Game's user interface, game modes, and extra features agreed with the client previously.*

Below outlines the key metrics of which the team completing the following project will follow in order to assess the progress and quality of the project, as well as the overall success, throughout the duration of the project.

1. User Involvement - **Agreed by team**
2. Support from Managers - **Agreed by team**
3. Clear Requirements Documentation - **Client briefed**
4. Satisfactory Planning - **Assessed by team**
5. Realistic Expectations - **Client briefed**
6. Project Milestones - **Confirmed with client**
7. Competent Team Members - **Assessed by team**

- 
8. Ownership of Work Completed - ***Agreed by team***
 9. Clear Vision of Objectives - ***Agreed by team***
 10. Hard-working, Focused Team Members - ***Agreed by team***

2. Current System

The current system was designed by last year's group using Android Studio. The system features an interface which features three buttons. The user can initiate a game and begin conversation by pressing the start button. There are also buttons which bring the use to a contact section or to settings. After starting a game the user can message back and forth with the bot/human and proceed to make a guess. The system then displays the result.

3. Proposed System

3.1 Overview

The proposed system will be an improved version of last year's system. The system will essentially be an interface for the Turing App that includes a sign-in feature, dialogue with a human/bot, a scoring system and a system that allows the user to flag inappropriate content. Building on last year's project, the system will also include requirements discussed with the client such as game modes and an option to receive feedback from the user, in the form of a survey, for example.

3.2 Functional Requirements

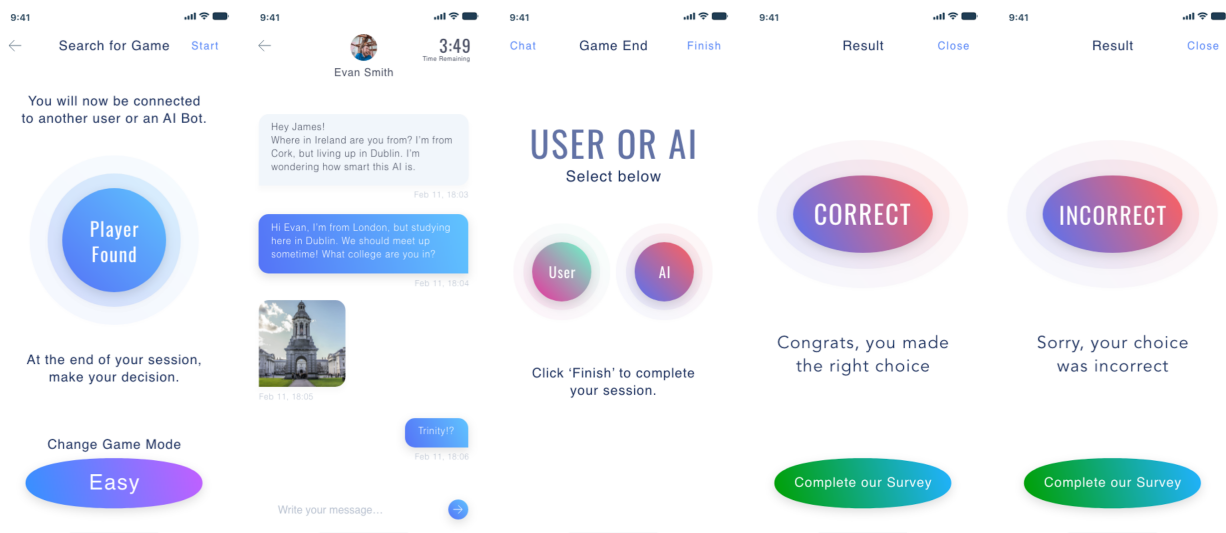
- Login to the app using a Google account
- Start a conversation in a particular game mode
- At the end of a conversation receive a user's guess (were they speaking to a bot or a human)
- Flagging mechanism within a conversation
- Send API requests for messages, guesses, etc. to the server
- Parse JSON data received from the server
- Receive feedback from users in the form of a survey
- Display user rankings in a leaderboard
- Show 'typing bubble' when a conversant is typing

3.3 Non-functional Requirements

- The user interface should be minimal but aesthetic
- Detailed API documentation should be written
- API requests should be as efficient as possible
- User data should be GDPR compliant

3.4 System Prototype

3.4.2 User Interface Mockups



3.4.2 Use Case Diagram

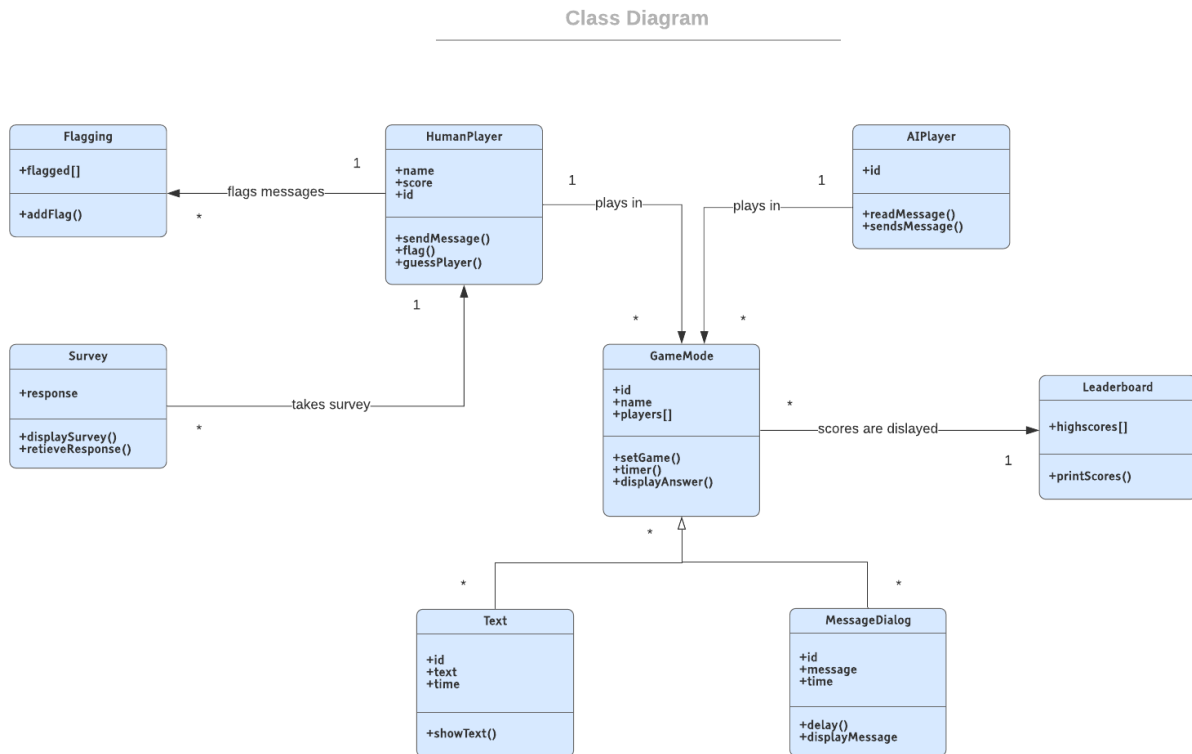


| | | | |
|--------------------------------|----------------|--|--|
| Use Case Name | | Offer Survey | |
| Actors | | System and User | |
| Precondtions | | User has finished the game and is still on the app | |
| Normal Flow | Description | User has finished the game. The system offers the option of a survey to the user who can accept or decline | |
| | Postconditions | User fills out survey for information | |
| Alternate flows and exceptions | | User exits app after game and survey cannot be offered | |

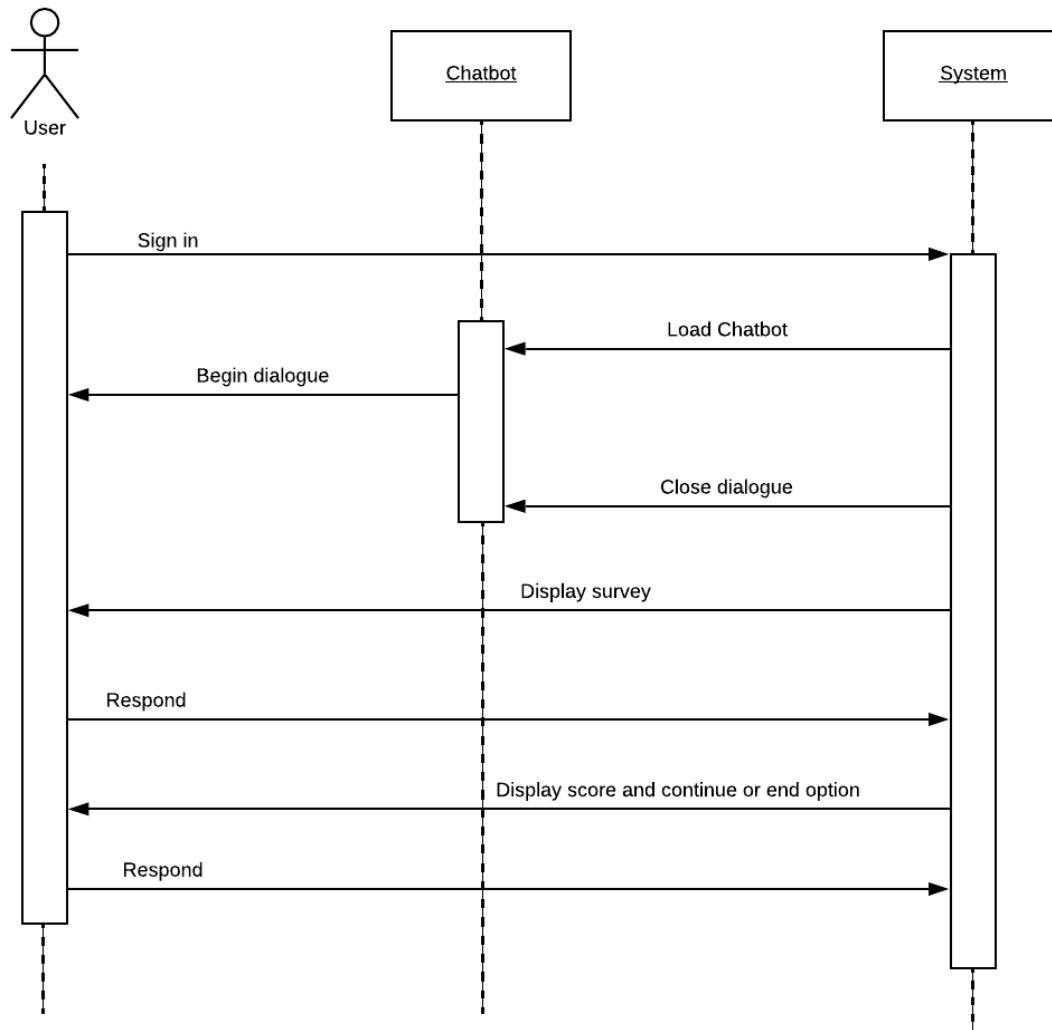
| | | | |
|--------------------------------|----------------|--|--|
| Use Case Name | | Flag inappropriate material | |
| Actors | | Human Player | |
| Precondtions | | Player sees a message they believe shouldn't be on the platform | |
| Normal Flow | Description | Player sees a message they believe to be inappropriate. They report it. The conversation is terminated and the conversation material is passed onto a trusted third party for review | |
| | Postconditions | Player can no longer see material but is being acted upon by a separate party | |
| Alternate flows and exceptions | | No exceptions | |

| | | | |
|--------------------------------|----------------|---|--|
| Use Case Name | | Connecting Parties | |
| Actors | | Two players or a player and a chatbot | |
| Precondtions | | User selects play and is connected to the internet | |
| Normal Flow | Description | User connects to server and is matched with either a chatbot or another player | |
| | Postconditions | User is ready to play the Turing Game against their partner and guesses if the player is a bot or another human | |
| Alternate flows and exceptions | | If the user disconnects or loses connection to the server, the game is stopped and connection failed. | |

3.4.3 Object model - Class Diagram



3.4.4 Dynamic model - Sequence Diagram



Appendix II: Software Design Specification


CS2013/CS3013

Software Design Specification

TCD ADAPT - Erwan Moreau

Group 20

Andrew Mc Donald
Conor Mac Amhlaoibh
Divine Mbunga
Ethan Monkhouse
Evan Mc Croary
Luke Mc Grath



| | |
|-----------------------------------|----------|
| 1 Introduction | 3 |
| 1.1 Overview | 3 |
| 1.2 Scope | 3 |
| 1.3 Definitions and Abbreviations | 3 |
| 1.4 References | 3 |
| 2 System Design | 4 |
| 2.1 Design Overview | 4 |
| 2.1.1 High-level Overview | 4 |
| 2.2 System Design Models | 4 |
| 2.2.1 System Context | 4 |
| 2.2.2 Use Cases | 5 |
| 2.2.3 System Architecture | 7 |
| 2.2.5 Sequence Diagrams | 8 |
| 2.2.6 State Diagrams | 8 |
| 2.2.7 Other Relevant Models | 8 |

1 Introduction

1.1 Overview

Our client is Erwan Moreau who is based in the ADAPT Centre in Trinity College Dublin. The purpose of the *The Turing Game* project is to introduce the concepts of AI and machine learning to average people in an interesting yet educational manner. Our client wants the project to be implemented as an application for Android devices. A prototype application was developed by a previous group as part of last year's *Software Engineering* module and one of the primary goals of this project is to improve on that application and to extend the features provided. Our group will be developing the client-side aspect of the project.

1.2 Scope

The scope of our project consists of developing a fully-functioning Android application that implements all of the client's initial requirements for *The Turing Game* as well as numerous additional features. We will be using Android Studio to develop the application as it is the 'gold-standard' when it comes to Android application development. Our application should be fully integrated with the server-side API which will be developed by a separate group - user authentication will be implemented using Google's Firebase tool. Additional care should be taken to ensure that the application is in full compliance with the GDPR.

1.3 Definitions and Abbreviations

| | |
|------|-----------------------------------|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| JSON | JavaScript Object Notation |
| UI | User Interface |
| XML | eXtensible Markup Language |

2 System Design

2.1 Design Overview

Our project consists of an Android application which will implement the client-side aspect of *The Turing Game*. Our application will allow the end-user to converse with other users or an AI for a limited period of time and subsequently guess the nature of their conversant. The application will communicate with the server-side part of the project using a shared API.

2.1.1 High-level Overview

Languages used:

- Java
- PHP
- XML

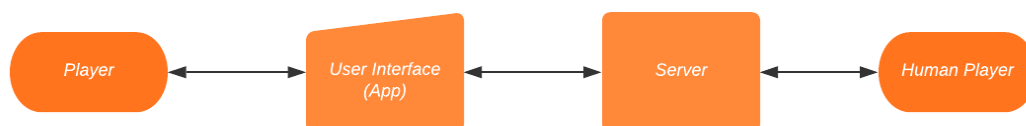
Tools used:

- Android Studio
- Firebase
- GitHub
- Server-side API
- Trello

2.2 System Design Models

Text.

2.2.1 System Context



1. The user requests a new game, this request and details of the game is sent to the server which sources either a human player or AI player.
2. The ID of the human player or AI player, and type is returned.

3. Conversation is exchanged for the set time frame between the users. Any flagged messages are pushed to the server where a log is kept.
4. User decision is recorded and compared to the type passed from the server. If true, the player wins, and this is recorded on the players profile.
5. Survey data is passed to the server alongside the user ID and chat log for further improvement.

2.2.2 Use Cases

Use Case Diagram:



Use Case Textual Descriptions:

1.

| | | | |
|--------------------------------|----------------|--|--|
| Use Case Name | | Offer Survey | |
| Actors | | System and User | |
| Precondtions | | User has finished the game and is still on the app | |
| Normal Flow | Description | User has finished the game. The system offers the option of a survey to the user who can accept or decline | |
| | Postconditions | User fills out survey for information | |
| Alternate flows and exceptions | | User exits app after game and survey cannot be offered | |

2.

| | | | |
|--------------------------------|----------------|--|--|
| Use Case Name | | Flag inappropriate material | |
| Actors | | Human Player | |
| Precondtions | | Player sees a message they believe shouldn't be on the platform | |
| Normal Flow | Description | Player sees a message they believe to be inappropriate. They report it. The conversation is terminated and the conversation material is passed onto a trusted third party for review | |
| | Postconditions | Player can no longer see material but is being acted upon by a separate party | |
| Alternate flows and exceptions | | No exceptions | |

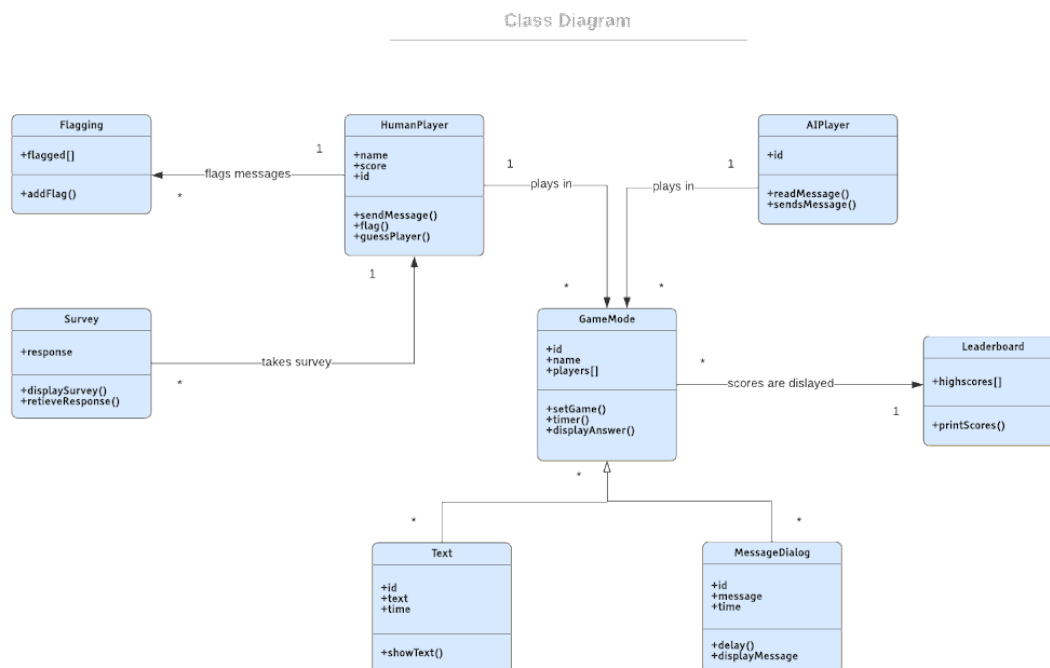
3.

| | | | |
|--------------------------------|----------------|---|--|
| Use Case Name | | Connecting Parties | |
| Actors | | Two players or a player and a chatbot | |
| Precondtions | | User selects play and is connected to the internet | |
| Normal Flow | Description | User connects to server and is matched with either a chatbot or another player | |
| | Postconditions | User is ready to play the Turing Game against their partner and guesses if the player is a bot or another human | |
| Alternate flows and exceptions | | If the user disconnects or loses connection to the server, the game is stopped and connection failed. | |

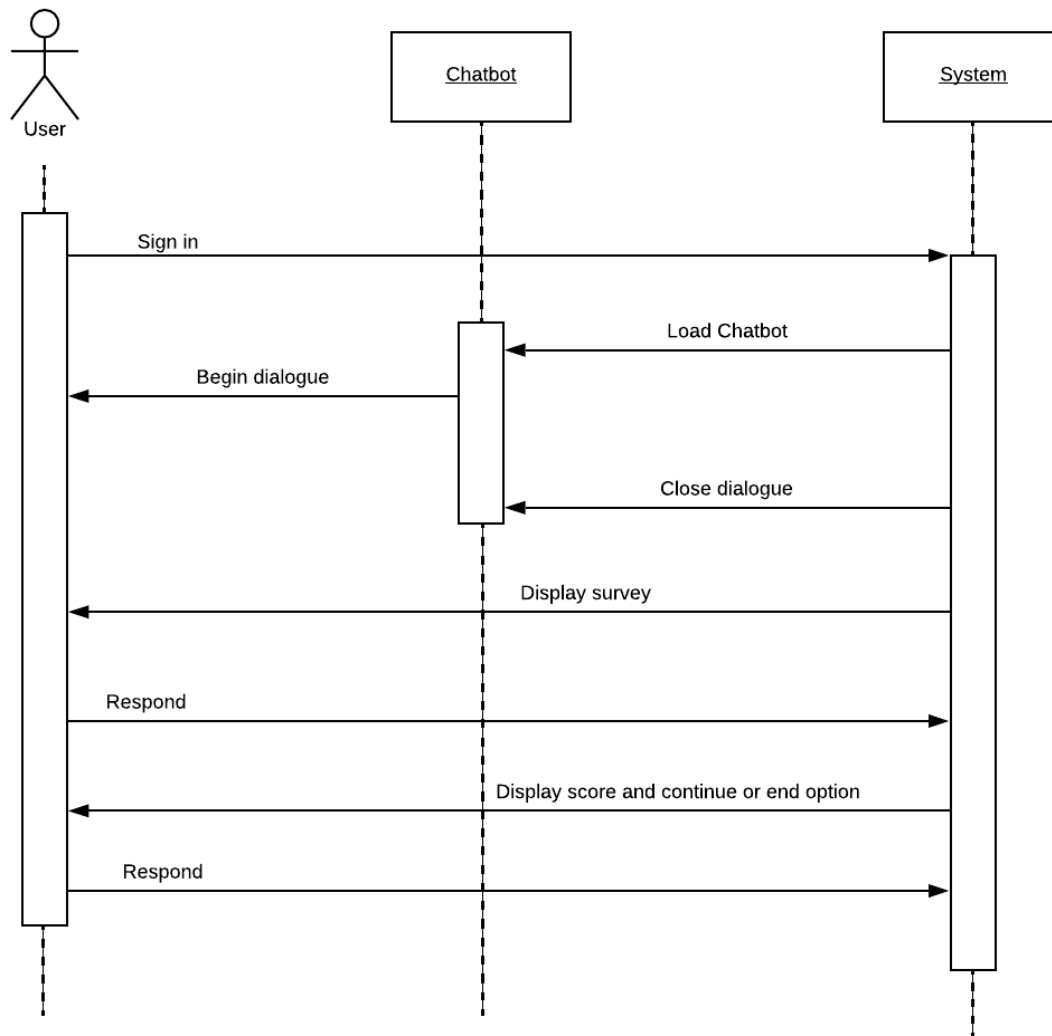
2.2.3 System Architecture

The system will allow a user to initially sign in using Google's Firebase tool. Upon authentication the user can choose to initiate a game, view the leaderboards or view the settings screen. During a game the user will converse with either the chatbot (which is designed by another group) or another human. Dialogue will be displayed on screen, which is all designed using Android Studio. The user can also view the leaderboards screen or change settings in relation to the application by navigating to the settings screen.. Both of these screens are also designed using Android Studio.

2.2.4 Class Diagrams



2.2.5 Sequence Diagrams



2.2.6 State Diagrams

