
Software Requirements Document for BotDetect

TEAM: 8

AUTHORS: Carlos Acuna, Corbin Graham, Jose Medina Mani, Adam Riffel, Nhan Tran

1. REWRITE SECTIONS IN THIS DOCUMENT WITH YOUR OWN DESCRIPTIONS
2. OMIT SECTIONS MARKED OMIT

Version	Date	Author	Change
0.1		SM	
0.2	1/25/24	ALL	

Table of Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	3
1.3	Definitions, acronymns, abbreviations	3
1.4	References	4
1.5	Overview	4
2	Overall Description	5
2.1	Product Perspective	5
2.2	Product functions	8
2.3	User characteristics	19
2.4	Constraints	19
2.5	Assumptions and Dependencies	19
3	Specific Requirements.....	20
3.1	External Interface Requirements	20
3.2	FEATURES	20
3.3	Performance requirements	20
3.4	Design Constraints.....	20
3.5	Software System Attributes	20
3.6	Other Requirements	21
	Appendix.....	22

1 Introduction

1.1 PURPOSE

A user (either a player or game developer) can use the software for automated data gathering that is then used by a machine learning algorithm to determine the probability of whether current player is a bot or a real person.

1.2 SCOPE

This software is not intended to be used as anti-cheat software, it provides a probability that a player is a bot and afterwards the user may use this information in whatever way they deem appropriate.

Use Cases:

Cheat Detection

- Admin uses program to extract information

Training Bot Detection Model

- End User Trains the Model

Game service changes its data. Program extracts data.

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Term	Description
Bot	Autonomously controlled agent.
Player	A human playing a game without third-party assistance.
CNN	Convolutional Neural Network; A deep learning technique that compresses (groups) pixels for faster and more memory-efficient predictions
DL	Deep Learning; A well connected graph of nodes and activation functions that result in a prediction
ATT	Automated Turing Test; A test to determine if someone is a robot or human
ML	Machine Learning; Using data and neural networks to create predictions
NN	Neural Network; A series of nodes and activation functions modeled after human brains

1.4 REFERENCES

- A Behavior Analysis-Based Game Bot Detection Approach Considering Various Play Styles, [MMORPG, Algorithms], <https://arxiv.org/pdf/1509.02458.pdf>
- Multimodal game bot detection using user behavioral characteristics, [Pipeline, Results], <https://springerplus.springeropen.com/articles/10.1186/s40064-016-2122-8#:~:text=Server%2Dside%20detection%20methods%20are,method%20for%20detecting%20game%20bots.>
- Bot Detection in Online Games, [Server-side, Client-side], <https://umm-csci.github.io/senior-seminar/seminars/fall2013/Lee.pdf>
- How to Compare Machine Learning Models and Algorithms, [Machine Learning, Compare Model], <https://neptune.ai/blog/how-to-compare-machine-learning-models-and-algorithms>
- CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images [Classification, Machine Learning], <https://doi.org/10.1109/ACCESS.2024.3356122>
 - o (Our problem is conceptually similar to that of another well-researched area of differentiating real versus AI-generated content. Perhaps we could learn from their techniques and approaches to the issue.)
- FakeSpotter: A Simple yet Robust Baseline for Spotting AI-Synthesized Fake Faces [Pattern Recognition, Machine Learning], <https://doi.org/10.48550/arXiv.1909.06122>
- Deep learning and multivariate time series for cheat detection in video games [CNN, Multivariate Time Series, Aimbot] <https://link.springer.com/article/10.1007/s10994-021-06055-x>
- Mario Reinforcement Learning Implementation, [MARIO, DRL], <https://www.kaggle.com/code/ratthachat/aic502-mario-rl>
- Classification of Humans and Bots in Two Typical Two-player Computer Games, [Detection, Turing Test], <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8463277>
- Turing Test Framework for Cooperative Games, [Turing Test, Framework, Github], <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9893684>, <https://github.com/bic4907/Multiplayer-TuringTest>

1.5 OVERVIEW

[OMIT]

2 Overall Description

In video games players may download third-party software that hands control to a computer to play the game for them. This often creates problems in being able to determine whether a real person is playing the game or not. Our purpose is to create software that can distinguish between bots and players with a high degree of certainty.

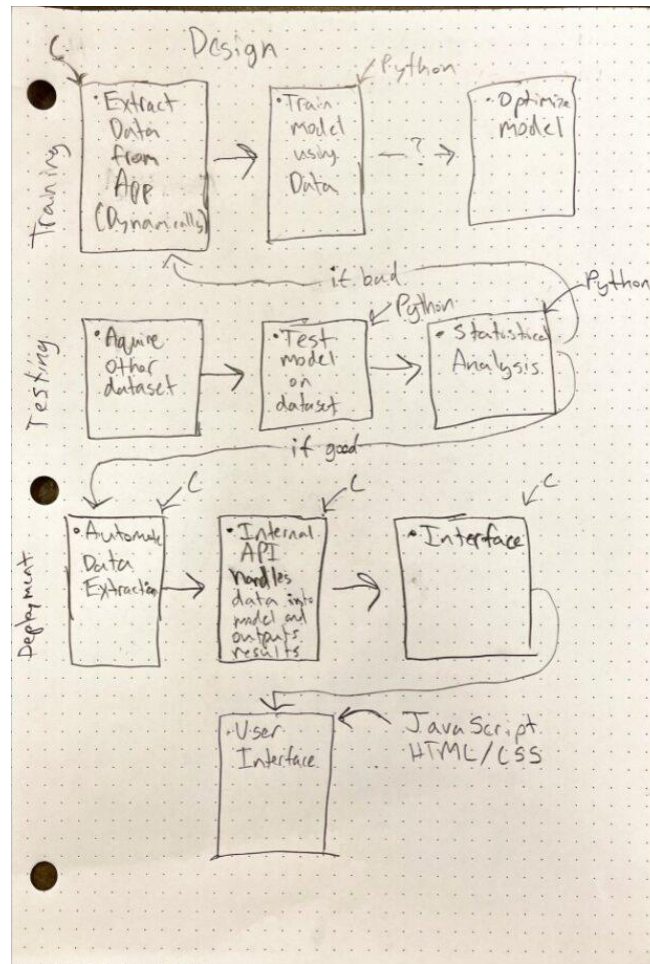
2.1 PRODUCT PERSPECTIVE

“reCaptcha”, a tool made by Google to determine whether a website user is a human or a bot.

“Super Mario Bros”, a 2D platforming game that was released in the 1980’s.

2.1.1 Concept of Operations

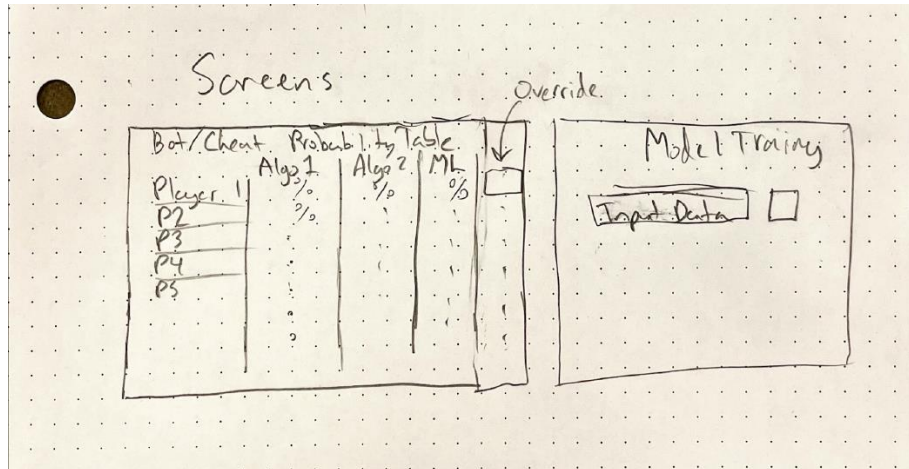
This project will be a web-based application that allows the user to import data for training a model to differentiate bots and players. Game administrators will have access to the model's prediction as the probability that a game agent is a bot. Game administrators can also override the model's prediction, which improves the model by providing an extra data point.



2.1.2 Major User Interfaces

Project is likely to have little to no UI as the intended purpose is to create an API.

2.1.2.1 Example Screenshot and description



2.1.3 Hardware Interfaces

Any device capable of running Super Mario Bros, C, C++, and Python.

2.1.4 Software Interfaces

// example: CGI-URL or function signatures etc (OMIT for now).

2.1.5 Communication Interfaces

// example: modem etc (OMIT for now)

2.1.6 Memory Constraints

// RAM, and other storage constraints (OMIT for now)

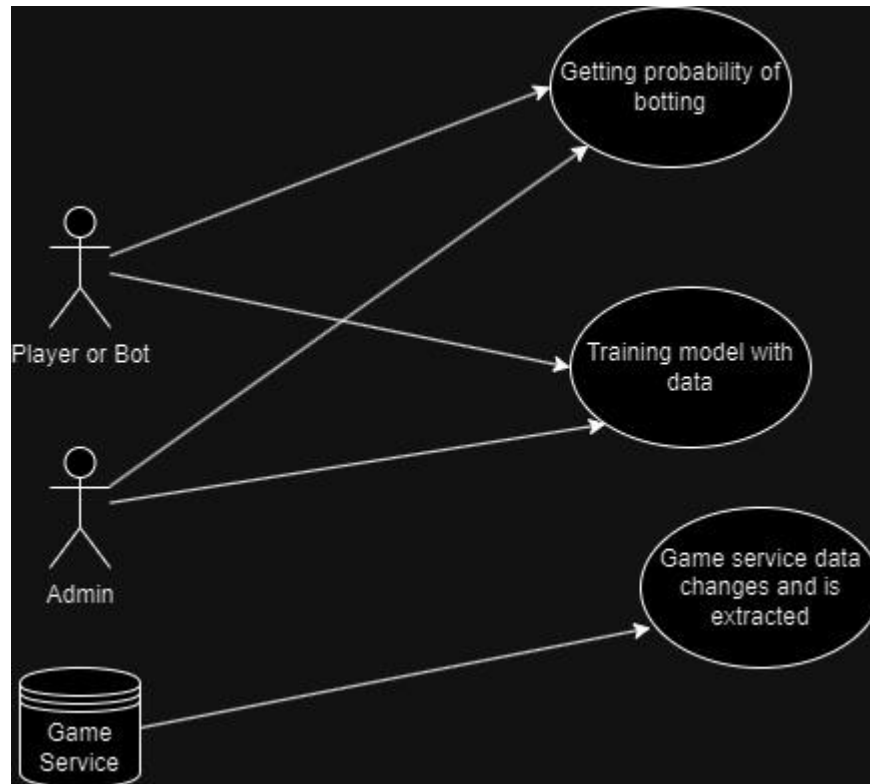
2.1.7 Operations

// special operations (if any) (OMIT for now)

2.1.8 Site Adaptation Requirements

//ex: Japanese language etc (OMIT for now)

2.2 PRODUCT FUNCTIONS



2.2.1 UC-0

“Game service data changes and is extracted” (Game Service)

Main scenario:

- 1) Game service updates variables for the player’s state.
- 2) Software makes note of this change and logs it to be analyzed.
- 3) Data stored is used later to train the model or used to make a prediction.

2.2.2 UC-1



“Training model with data” (Player, bot, or admin)

Main scenario:

1. User indicates to the software a program/source of data to be used for training.
2. Features that are likely predictors are then extracted from the program and split into a test and validation set.
3. Model trains on the test set and prevents overfitting with the validation set.
4. After a series of iterations bias values are linearly fitted to data.

Extensions:

2.2.3 UC-2



"Get probability of botting" (Player, bot, or admin)

Main scenario:

1. Dataset is provided to the software by the user
2. Algorithm is used to analyze the dataset and determine probability of the data coming from a bot.
3. Prediction results are given to the user.

Extensions:

1. Speedrun Validation
 - a. Working under the assumption that the user has selected a model/algorithm that takes in visual/image data, a video could be used as a dataset for the software to analyze.
 - b. A moderator for a website like speedrun.com can use this probability output data to determine whether a speedrun is done by a human or not, and make a decision based on that information.
2. Server-side Bot Detection
 - a. An admin can install this software on the server side of the game where all the player's inputs and actions are processed.
 - b. The admin can then use the information provided to decide whether a player should be disciplined or not.

2.3 USER CHARACTERISTICS

// typical user characteristics, frequency of usage etc

2.4 CONSTRAINTS

// all conditions that may limit design options (INCLUDE NON FUNCTIONAL CONSTRAINTS)

2.5 ASSUMPTIONS AND DEPENDENCIES

// hardware and software assumptions and dependencies

3 Specific Requirements

Specific requirements differ for the number of features available. Generally, these will be features available from reading program memory in production. The other will assume access to the program's screen, then it will be features extracted from on-screen data. Availability of features will vary depending on the state of the program, and where our entry point is.

3.1 EXTERNAL INTERFACE REQUIREMENTS [OMIT THIS SECTION]

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communications Interfaces

3.2 FEATURES

3.2.1 FEATURE 1

The program's main feature is to predict with high accuracy the likelihood of an actor being human or bot.

3.2.2 FEATURE 2

A secondary feature is the differentiation of individual actors. This could be uniquely IDing multiple bots or humans.

3.2.3 FEATURE 3

Allow detection accuracy feedback from the client side to improve the bot detection algorithm.

3.3 PERFORMANCE REQUIREMENTS

THE PROGRAM IS REQUIRED TO PROVIDE REALTIME PREDICTIONS. THIS MEANS THAT THE PROGRAM CANNOT TAKE MULTIPLE CYCLES TO PROVIDE A PREDICTION.

3.4 DESIGN CONSTRAINTS

DESIGN CONSTRAINTS WILL BE LIMITED TO WHAT PROVIDES THE GREATEST PREDICTION ACCURACY.

3.5 SOFTWARE SYSTEM ATTRIBUTES (THESE ARE NON-FUNCTIONAL REQUIREMENTS)

3.5.1 Reliability

The program should produce consistent, accurate results for an undetermined number of features.

3.5.2 Availability

The program should be operable on client or server-side. It should be fully self-contained and not dependent on third party sources. From this, it should be available 100% of the time.

3.5.3 Security

Encryption of data should be used whenever and wherever possible. But there is no expectation that the data produced by the program is secure, since it is relying on non-secure sources.

3.5.4 Maintainability

The program should be fully pre-trained, but available for finetuning to better suit a given task. Since it will be fully pre-trained, there are no maintenance expectations after launch.

3.5.5 Portability

The program should rely only on open-source frameworks and libraries such that they will continue to be supported on any given system for a limited indefinite time.

3.6 OTHER REQUIREMENTS

Requirements will vary depending on the solution platform. Initially, requirements will be limited to those provided above.

// ADD Appendices (if any)

// Regenerate Table of Contents

