Policing Bot Software Development
Design Document

Cody Manning
Gabriel Silva
Liam Dumbell

September, 2023

Table of contents

1. Overview.....3
1.1 Scope.....3
1.2 Purpose.....3
1.3 Intended audience.....3
2. Definitions and Acronyms.....4
3. Conceptual Model for software design descriptions.....6
3.1 Software Design in Context.....6
3.2 Software Design Life Cycle.....7
4. UML Diagram for Police Bot Framework.....7
5. Human Interface Design.....9
5.1 Graphical Interface Example.....9

1. Overview

1.1 Scope

The intent of this document is to provide a technical description of the framework we are developing for the Police Bot and Bot detection system. We will be describing the details surrounding the design of the framework, the functionality that it provides and the end deliverables of the framework that is presented to the user.

1.2 Purpose

Millions of people use online social media platforms every day, and most active users of the social media platform Twitter have had interactions with bot accounts at least once. These interactions have the potential to negatively impact the user experience of Twitter as they open up the user base to threats such as scams, predatory advertising, purposely falsified information, etc. Our goal for our project is to create a framework that can reliably detect these malicious bot accounts and recommend actions that should be taken against these accounts to create a safer and more enjoyable Twitter experience for all users.

1.3 Intended audience

This document is intended to be read by the following individuals:

- Dr. Phillip Chan
- Dr. Khaled Slhoub
- Other students that might want to use our data to analyze themselves.
- Other Individuals we allow to access our framework.

2. Definitions and Acronyms

Term	Definition
Bot	Short for robot. In this case we refer to a software or script that performs automated tasks.
Beneficial Bot	A bot that performs tasks that are helpful or productive to individuals interacting with it E.g.: Translators, weather reports, or meme generators
Malicious Bot/MalBot	A bot that performs tasks that are harmful, dangerous or illegal. E.g.: Scams, spam, or fraud.
Twitter/X	A social media platform that allows users to post text or media and allow other people to see it. Can be used personally by individuals or companies to spread information.
Tweet	Posts or replies to other people's posts in Twitter/X that are publicly available if allowed by the user.
Thread	People can tweet on other peoples' tweets and make a sequence of connected tweets. An analogy of threads in computer science terms would be a tree.
Root tweet/Base tweet	The first tweet of a thread. Similar to the root node of a tree.
Retweet	A repost of another accounts post that appears on a user's profile
Trend/Trending topics	Twitter/X has a defined section of the website that enables users to see a list of tweets or hashtags (explanation below) that are being discussed/used by

	numerous accounts in the last 24 hours.
DM/Direct Message	Messages sent privately to users.
@	Symbol used to mention other users on Twitter. This symbol is followed by a username to create a hyperlink to the user's profile and notify them. E.g.: @taylorswift13
#	Called 'Hashtag'. Is a label used to organize and categorize content to make it easier to discover for other users.
Dynamic Analysis	Dynamic analysis is the testing and evaluation of an application during runtime. In the context of this project, Dynamic Analysis refers to the analysis of account data right after it is compiled.
AWS	Amazon Web Service (database hosting service)
MySQL	MySQL is a type of SQL database.
GDPR	General Data Protection Regulation: a set of standards for handling user data.
API	Application Programming Interface (A set of rules and protocol that allows different software applications to communicate to each other.)
Tweepy	Python library used to integrate with the Twitter API

3. Conceptual Model for software design descriptions

This section aims to create descriptions of the decisions made in different contexts of our program and outline the concepts used at each step.

3.1 Software Design in Context

The Police Bot framework is a collection of different processes that fall under three main categories we refer to as the three D's:

1. Detection
2. Distinguishing
3. Deciding

The Detection process of our framework aims to allow users of our framework to reliably detect accounts on Twitter that are run by bot accounts. This will be done using our own bot accounts, named Police Bots, that will scan through Twitter and identify accounts that have the possibility of being bots. The Police Bot will then compile and download all relevant account information on the target and return it to the framework. The framework will then analyze this account data using algorithms modeled off of published bot detection research methods and be able to reliably identify an account as a bot or human.

The Distinguishing process of our framework aims to allow users of our framework to reliably categorize a bot account on Twitter as beneficial or malicious. This will be done by analyzing account data of previously identified bot accounts to determine the implicit purpose of the account. Once the purpose of the bot is known, it will then be labeled as malicious or beneficial accordingly.

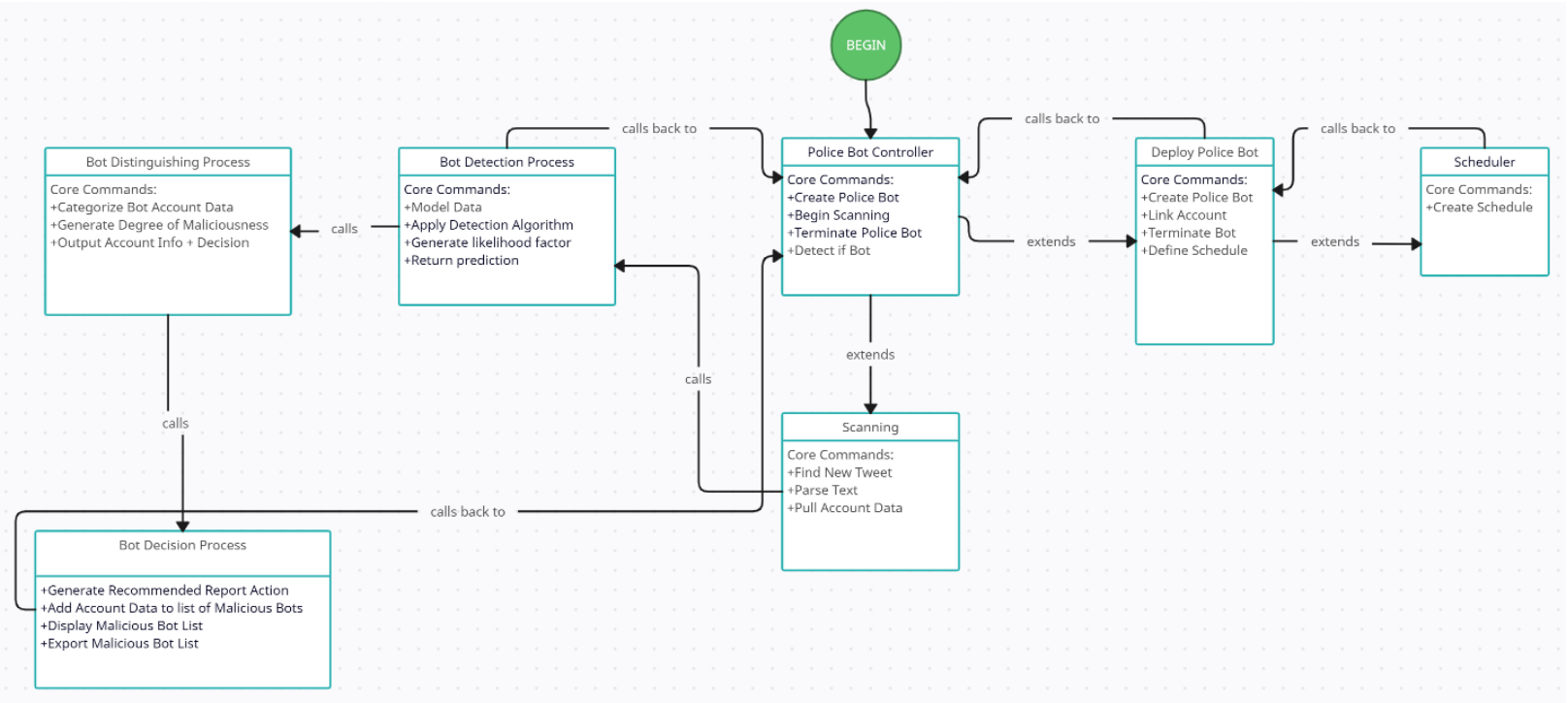
The Deciding process of our framework aims to allow users of our framework to view the findings of the previous two main processes as well as a generated recommended action for a particular bot account. The user can then decide to report the malicious bot accounts or not depending on the level of negative impact the bot account may be causing.

Each of these processes will be made available to the user through a simple Python GUI.

3.2 Software Design Life Cycle

The life cycle of these software design decisions are intended to remain in place for the duration of our time in the senior project course, being the Fall 2023 and Spring 2024 semesters.

4. UML Diagram for Police Bot Framework



The framework begins by running the Police Bot Controller, This controller is responsible for creating any instances of individual Police Bots, linking them to their respective Twitter accounts, setting their schedule and terminating them if needed. The Police Bot Controller can also begin the scanning process by calling the Scanning method that scans through desired Tweets to find accounts of interest, compile the account data and pass it onto the Bot Detection Process. The Bot Detection Process firstly models the account data it receives from the Scanning method into something that is easier to work with. After this is complete, the Bot Detection Process will apply all applicable detection algorithms to the Modeled account data and generate how likely the account is to be controlled by a bot and add its prediction to the existing model of the Twitter account. If the account is determined to be controlled by a bot, the account data is then passed to the Bot Distinguishing Process where it is further analyzed and

categorized in various ways based on account activity. If the account is determined to not be run by a bot, The Scanning method is called again to restart the cycle of execution. The Bot Distinguishing Process will also generate a level of maliciousness associated with the bot account being analyzed and add the level of maliciousness to the model of the bot account data. The account data is then passed to the Bot Decision Process, where a Recommended Report Action is generated for malicious bots and the account data is added to a list of Malicious Bots. The Bot Decision Process will also have the functionality of displaying the current list of malicious bots along with their account information as well as exporting that account information to a file which the user can download. The Bot Decision Process will then call back to the Police Bot Controller once this process is done to complete the cycle of execution.

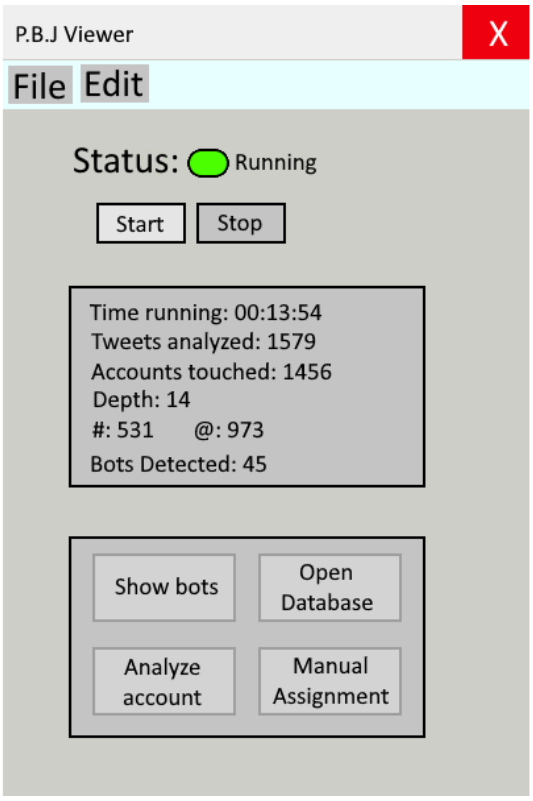
A exemplified pseudo-code for the program would look like this:

```
Global bot_list;

Main() {
   // method to create a bot using the Twitter/X API / Tweepy
   Bot = new setup_bot(account_credentials, schedule, depth);
   While current_time != bot.schedule_time:
      wait()
      // method that finds the new trending topics and returns them
   Trending_topics = find_trending_topics(bot);
   For each root_tweet in trending_topic:
      scan_tweet(root_tweet, bot.depth, 0);
}
scan_tweet(tweet, max_depth, depth) {
   // limiting the depth of the search
   If depth == max_depth:
      return;
   Account = tweet.account;
   Content = tweet.content;
   Responses[] = tweet.responses;
   // analyzing if the bot is suspicious (still to be determined)
   // and adding it to the list of possible bots
   Suspicious = analyze_suspiciousness(account, content);
   If suspicious:
      bot_list.append(account);
   // repeating the process for the responses
   If len(responses) > 0:
      For each t in responses:
         scan_tweet(t, max_depth, depth+1);

   return;
}
```

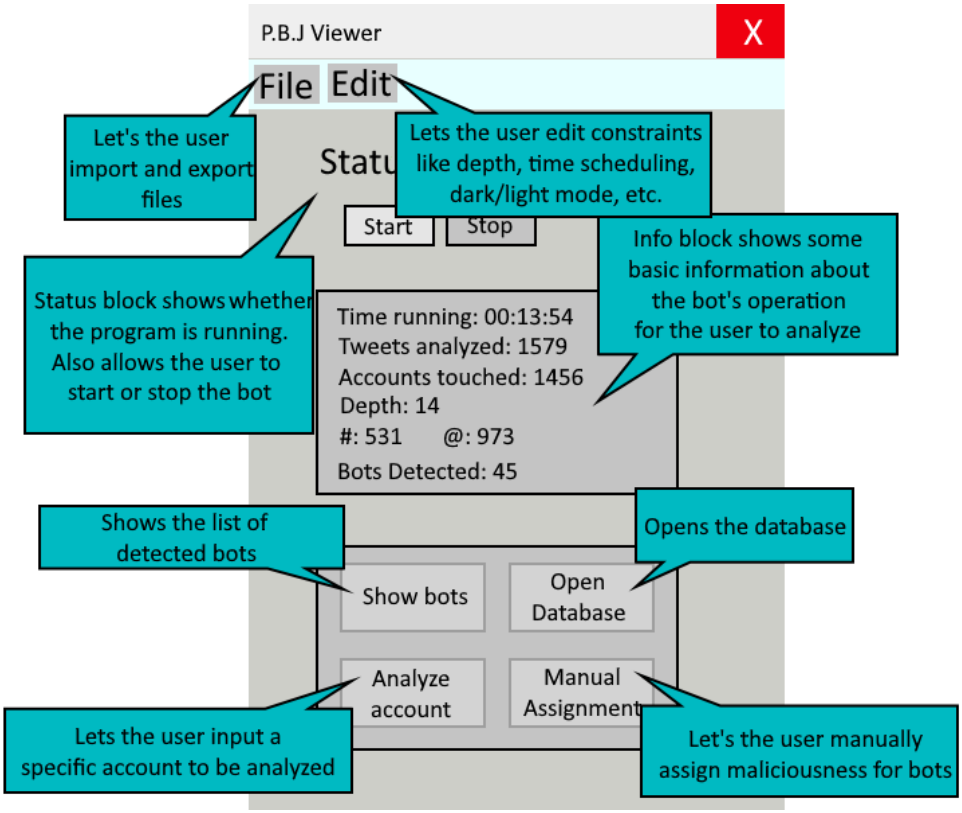
5. Human Interface Design

5.1 Graphical Interface Example

Pictured below is a conceptual mockup of what a potential mockup of our user interface could look like. This is subject to change, but for now this is the design we are keeping in mind for development.



(A base template for the GUI)


P.B.J Viewer
X
File
Edit
Let's the user import and export files
Lets the user edit constraints like depth, time scheduling, dark/light mode, etc.
Statu
Start
Stop
Status block shows whether the program is running. Also allows the user to start or stop the bot
Info block shows some basic information about the bot's operation for the user to analyze
Time running: 00:13:54
Tweets analyzed: 1579
Accounts touched: 1456
Depth: 14
#: 531 @: 973
Bots Detected: 45
Shows the list of detected bots
Opens the database
Show bots
Open Database
Analyze account
Manual Assignment
Lets the user input a specific account to be analyzed
Let's the user manually assign maliciousness for bots