Technical Report

EI-40387

Informatics Project

BotBlocker: A Community-Driven Approach to Identifying and Filtering AI Profiles on Social Media

Authors:
Angela Ribeiro - 109061
João Santos - 110555
João Viegas - 113144
Rita Silva - 114220
Hugo Castro - 113889

Advisors:
João Almeida
Vicente Barros

June 17, 2025

Abstract

With the rise of AI, bots have become a prominent presence in every social media app. Even though these can be benign, and sometimes even helpful, they can also be used to infiltrate political discourse, manipulate the stock market, steal personal information, and spread misinformation. The detection of social bots is therefore an important research endeavor.

Without effort to identify these bots, human beings can find themselves easily influenced, while their peers' voices are drowned out by AI. The best possibility to avoid this is a deeply researched topic, yet the detection of bots remains an uncertain challenge.

Contents

Chapter 1

Introduction

The proliferation of AI-driven bots on social media platforms has emerged as a significant challenge to the integrity of online discourse. These automated accounts are often indistinguishable from real users in both appearance and behavior [1], yet they can disseminate misinformation and manipulate public opinion at a massive scale [2]. From algorithmically generated tweets to coordinated bot networks that simulate authentic engagement, the presence of such inauthentic actors erodes trust in digital platforms. In response to these threats, this project introduces a novel, community-driven system for evaluating and verifying the authenticity of social media accounts. By leveraging collective user input alongside digital tools, the system aims to differentiate human-operated profiles from AI-driven or deceptive ones, ultimately promoting greater transparency and trust in online environments.

1.1 Background and Motivation

Understanding the motivations behind this project requires contextualizing the rise of social bots within broader technological and societal trends. This section explores the advances in AI that have enabled sophisticated bot behavior, the resulting sociopolitical implications, and a platform-specific case that underscores the urgency of effective bot detection.

1.1.1 Technological Trends

Advancements in artificial intelligence have significantly increased the capabilities of social media bots. Powered by generative language models and sophisticated algorithms, modern bots can closely mimic human language and interaction patterns [2]. Studies show that many users struggle to distinguish these bots from real accounts [1]. During global events, it has been estimated that bots generate up to one-fifth of social media posts [3], highlighting the scale of automated content creation.

To counter this, researchers have developed a variety of bot detection methods using machine learning classifiers that analyze features like user behavior, network activity, and linguistic patterns [3]. These range from basic algorithms such as logistic regression to more advanced models like deep learning and large language models [3]. Despite these efforts, bot detection remains an ongoing challenge. Detection tools must continually evolve to match increasingly evasive bot behavior, and even trained humans often fail to accurately identify inauthentic accounts [3]. These limitations suggest the need for complementary approaches, such as leveraging community input in bot detection.

1.1.2 Sociopolitical Concerns

Beyond technical difficulties, bots pose substantial sociopolitical risks. They have been used to manipulate political discourse, inflate follower counts, and drown out dissenting opinions. For instance, during the Arab Spring, bots were deployed to simulate political support, while in the 2020 U.S. elections, they were used to spread disinformation and polarize public opinion [3]. During the COVID-19 pandemic, bots fueled anti-vaccine narratives and conspiracy theories, often simulating public consensus through coordinated activity [3].

These operations are frequently orchestrated by interest groups or state actors aiming to destabilize public trust [2]. Crises such as natural disasters or health emergencies provide fertile ground for these campaigns, allowing bots to exploit heightened public vulnerability and amplify false or divisive content [2]. Consequently, bot presence is not just a technical concern but a threat to democratic discourse and societal stability.

1.1.3 Platform-Specific Examples: Twitter (X)

Twitter (rebranded as X in 2023) offers a clear example of the bot problem and its consequences. Elon Musk claimed that over 20% of Twitter's users were bots—significantly higher than earlier estimates [3]. Following his acquisition of the platform, a large-scale purge was undertaken to eliminate fake accounts and impose posting fees to deter spam [3]. However, studies conducted in 2023 suggest that bot activity remained widespread and undeterred [4].

The introduction of paid verification under X Premium, replacing the legacy system that confirmed user identities, further complicated authenticity. This change allowed virtually anyone to obtain a verified badge without meaningful identity checks [5]. As a result, impersonation and misinformation spread rapidly. A striking example occurred in November 2022, when a fake account impersonating pharmaceutical company Eli Lilly falsely announced that "insulin is free now" [6]. The tweet went viral, leading to financial consequences and public confusion before being retracted.

These incidents highlight the real-world risks of inadequate verification and emphasize the need for more reliable and community-driven mechanisms to assess account authenticity. A system that empowers users to participate in the identification of bots offers a promising path toward more trustworthy online interactions.

1.2 Limitations of Automated Bot Detection and the Need for Community Involvement

AI-based bot detection techniques have improved markedly over the past decade, yet they still suffer from important limitations. Classifiers driven by machine learning can process vast amounts of social media data and identify suspicious patterns, but no algorithmic solution is foolproof. In practice, the bot detection problem has become a perpetual "cat-and-mouse" game: as soon as detectors learn to recognize certain bot behaviors, bot developers adapt with new tactics to evade them [3]. Even state-of-the-art detection systems that incorporate deep neural networks or large language models can be circumvented by cleverly engineered bots. Moreover, these automated methods often struggle with false assessments—flagging some legitimate users as bots while failing to detect many actual threats—especially as bots grow more sophisticated. Notably, human evaluators are not inherently better at the task: studies have shown that people (including trained analysts) often perform no better than random guessing when asked to distinguish advanced social bots from real users [3]. This unreliability of both purely automated classifiers and individual human judgment underscores the severity of the challenge.

One major reason for this shortfall is that bots have evolved to closely mimic human behavior, making them harder to detect. Early-generation bots could often be identified by telltale signs of automation (such as repetitive posting or obviously generated content), but modern bots are far more subtle. Powered by advanced AI, they can produce fluent and contextually relevant language, engage in back-and-forth conversations, and generally blend into online communities with little to distinguish them from genuine users. For example, bots backed by large language models now generate posts that are almost indistinguishable from human-written text [2], and many fake accounts present realistic personas that imitate the profiles and interaction patterns of real people [1]. Malicious bot operators also employ adversarial techniques—deliberately adjusting a bot's behavior to exploit weaknesses in detection algorithms—which further exacerbates the arms race between bot creators and defenders [3]. The net effect is that social bots today can appear highly authentic. Empirical analyses have found that newer waves of bots are hardly separable from legitimate accounts when inspected in isolation [3]. Even when platforms undertake drastic measures to eliminate bots, the respite is often temporary. For instance, despite high-profile "bot purges" and policy changes aimed at removing fake accounts, Twitter (now X) continued to see widespread bot activity in 2023, with independent reports indicating that inauthentic networks were as active as ever on the platform post-purge [4].

Given these limitations, there is growing consensus that purely algorithmic detection must be augmented with human insight. This project explicitly adopts a community-driven approach that does not rely on AI or automated classifiers to detect bots. Instead, it centers user input as the primary mechanism for identifying suspicious accounts. By leveraging community evaluation—where users can collectively assess profiles, vote on their authenticity, and flag patterns that seem unnatural—the system harnesses human contextual reasoning, which automated systems often lack. While algorithms are effective at pattern recognition on a large scale, they may miss cultural nuances, humor, or sarcasm, which human users intuitively understand. Moreover, human judgment can adapt quickly to novel behaviors, making it well-suited to spotting emerging tactics before automated tools are retrained.

Community-based detection also serves an important democratizing function: it enables end-users to play an active role in maintaining the integrity of the platforms they inhabit. Recent studies have shown that incorporating human input into detection workflows can enhance the overall reliability of classification systems [7]. Although our system does not integrate such hybrid models, the principle remains: distributing the responsibility of detection across the user base introduces redundancy, responsiveness, and resistance to adversarial adaptation. Ultimately, maintaining trust in online spaces requires a socio-technical response. By placing users at the center of the detection process, this project offers a scalable and transparent alternative to opaque, automated moderation systems.

1.3 Purpose and Scope

The overall objective of this project is to enable users to collaboratively evaluate and verify the authenticity of social media accounts. In pursuit of this goal, the system is designed as a community-driven platform that leverages the collective oversight of its user base to identify genuine versus inauthentic profiles, thereby enhancing trust in online interactions. To achieve this objective, the system comprises two key components:

- A browser plugin that surfaces bot-likelihood estimates and community verification status directly on social media pages for any user.
- A web platform where verifiers and administrators can manage voting outcomes, perform account verifications, and execute moderation actions, while authenticated users can check more thoroughly the evaluations made to a profile.

The system is currently implemented for three major social platforms: Twitter (X), Instagram, and Facebook. Notably, the design does not incorporate any AI-based or algorithmic bot detection. Instead, it relies exclusively on human-driven community evaluation to determine account authenticity. This human-centric approach prioritizes transparency and collective judgment, avoiding the uncertainties and potential biases associated with automated classification methods. However, this community-driven paradigm comes with certain limitations that define the scope of the project. The system relies on voluntary participation from users and evaluates only accounts with publicly accessible information. As a result, the coverage and effectiveness of the verification process are contingent on active community engagement and are inherently limited to accounts open to public scrutiny on the supported platforms. Within these constraints, the scope of the project remains clearly delineated, focusing on demonstrating the viability of community-based account verification as an alternative to purely automated approaches.

1.4 Report Structure

This report is organized into several chapters, each addressing a different aspect of the BotBlocker project, from the initial motivation and background to the detailed implementation and results.

- **Chapter 1: Introduction** – This chapter introduces the problem of social media bots, the motivations behind the project, and the overall goals of the BotBlocker system. It also presents the limitations of automated bot detection and justifies the adoption of a community-driven approach.
- **Chapter 2: State of the Art** – This chapter reviews existing bot detection tools and approaches, comparing their methodologies, strengths, and weaknesses. It identifies gaps in current

solutions that BotBlocker aims to address, particularly the lack of community participation and cross-platform capabilities.

- **Chapter 3: Requirements Gathering** – This chapter outlines the functional and non-functional requirements of the BotBlocker system, including user roles, system performance, and security measures. It also describes the main actors and use cases that define the interactions within the system.
- **Chapter 4: Implementation** – This chapter provides a detailed description of the system's architecture and the technologies used to develop the BotBlocker platform. It covers the presentation layer (browser extension and web interface), the business logic layer (backend functionality), and the data layer (storage and database management).
- **Chapter 5: Results** – This chapter discusses the results of usability testing conducted on the BotBlocker system, including user feedback, task completion rates, and improvements made to the platform based on the test findings. It also includes visual material showing the functionality of the system.
- **Chapter 6: Conclusion** – The final chapter summarizes the project's contributions, discusses its limitations, and outlines potential future improvements to the BotBlocker system.

Each chapter builds upon the previous one, starting with the problem statement and background, moving through the design and development of the system, and concluding with an evaluation of its effectiveness and usability.

1.5 Terminology

The following terminology is used throughout this report to describe key concepts and components of the BotBlocker project:

- **Bot** – A software program or automated account that interacts with a social media platform, typically designed to perform specific tasks such as posting content, following users, or generating engagement. Bots can be benign or malicious, depending on their intent.
- **Social Media Bot** – A type of bot specifically designed to interact with social media platforms such as Twitter (X), Instagram, and Facebook. These bots can be used for a variety of purposes, including the spread of misinformation, manipulation of public opinion, and amplification of certain topics or accounts.
- **Community-Driven Approach** – A system where human users are actively involved in identifying and verifying potential bot accounts, instead of relying solely on automated detection algorithms. In the BotBlocker system, the community's collective input is used to assess the likelihood that an account is managed by a bot.
- **Verified User** – An authenticated user within the BotBlocker system who has been granted the ability to vote on the authenticity of accounts and assign or remove verification badges. Verified users are typically granted more privileges within the platform.
- **Bot-Likelihood Score** – A percentage estimate provided by the BotBlocker plugin indicating the likelihood that a particular social media account is a bot. This score is derived from community votes and reflects the consensus of verified users regarding the account's authenticity.
- **Verification Badge** – A label or marker assigned to a social media account by a verified user to confirm that the account has been assessed and determined to either be a bot or human-operated. This badge is visible on the web platform and plug-in and is used to identify accounts that have undergone community scrutiny.
- **Admin Dashboard** – A section of the BotBlocker web platform that allows administrators to manage the system. Admins have the ability to promote users to verifier status, suspend accounts, or ban users who violate platform guidelines.

- **Plugin** – A browser extension developed as part of the BotBlocker system, allowing users to view bot-likelihood scores and verification badges directly on social media platforms. The plugin offers real-time insights into the authenticity of accounts while browsing.
- **Web Platform** – The online interface where authenticated users can interact with the BotBlocker system. It allows verifiers to manage bot evaluations, while admins can control user privileges and moderation actions.
- **Social Media Platforms** – The online platforms on which users can interact, share content, and connect with others. For the purposes of this report, the supported platforms include Twitter (X), Instagram, and Facebook.
- **Human-in-the-Loop** – A system design that incorporates human input to make final decisions in a process that would otherwise be automated. In BotBlocker, the human-in-the-loop process involves verified users making decisions about the authenticity of accounts, in contrast to relying solely on AI-based detection.
- **BB_user** – An authenticated user within the BotBlocker system who can interact with the platform and vote on the authenticity of accounts. While BB_users can participate in voting, they do not have the same privileges as verified users, who are able to assign verification badges to accounts.
- **Evaluation History** – A record of all the evaluations (votes and comments) associated with a particular account in the BotBlocker system. This history includes information about who voted, what their decision was, and any associated reasoning or context for the evaluation. This helps maintain transparency and accountability within the community-driven process.
- **Personal Blocklist** – A list of users that have been manually blocked by an individual user within the BotBlocker system. Users added to the personal blocklist will not appear in the BB_user's feed, though this is the only restriction placed on them. The blocklist allows users to filter out unwanted accounts from their view without completely restricting interactions with the blocked users on the platform.

This terminology section defines the core concepts and components integral to the BotBlocker system and ensures a shared understanding of key terms used throughout the report.

Chapter 2

State of the Art

To understand the current landscape of bot detection systems in social networks and position *Bot Blocker* effectively, a thorough review of existing solutions and their characteristics was conducted. The following summarizes three key tools, highlighting their methodologies, scope, and limitations.

2.1 Pegabot

Pegabot was launched in 2018 by the Institute of Technology and Society of Rio de Janeiro (ITS Rio) and the Technology & Equity Institute, with funding from the European Union. It allows users to analyze Twitter/X accounts and assess the likelihood of automation based on public data. The system calculates a probability index using three main components: profile metadata (such as account age, description, number of followers and tweets), network interaction patterns (e.g., hashtag and mention behavior), and sentiment analysis of the most recent 100 tweets. Bots often display consistent emotional tones and repetitive posting behavior, which the model uses to flag suspicious profiles.

Despite its innovative approach, Pegabot has notable limitations. The heuristic model is prone to false positives, particularly for thematic accounts or niche profiles. Moreover, as of 2025, the platform is no longer operational ("Internal Server Error"), suggesting it is outdated and no longer maintained—making it less effective against modern bot strategies.

2.2 LiveDune

LiveDune is a social media management tool widely used by businesses and influencers, with functionality for post scheduling, audience analytics, and competitor tracking. One of its key features is bot detection on Instagram, designed to identify fake followers and inflated engagement. The system analyzes accounts based on general profile metrics (total followers, engagement rate, average likes/comments), interaction consistency (such as the ratio between likes and comments), and growth patterns (spotting sudden spikes or drops in followers that may indicate artificial inflation).

The analysis is fast—typically under 30 seconds—and provides visual cues and detailed reports that help users make informed decisions. LiveDune is particularly useful for optimizing advertising strategies by filtering out inauthentic profiles. However, its focus is mainly commercial, and it does not target bots used for spreading misinformation. Additionally, the verification feature is behind a paywall ($15.30/month for 5 accounts), which limits broader accessibility.

2.3 Bot Sentinel

Bot Sentinel, developed by Christopher Bouzy in 2018, is a free platform aimed at identifying manipulative behavior, disinformation, and harassment on Twitter/X. Unlike tools that only detect automation, Bot Sentinel also targets troll-like behavior and coordinated toxicity. It uses a machine learning model trained on thousands of accounts and tweets to classify profiles in three steps: behavioral analysis (e.g., post frequency, interactions, language patterns), rule-based classification (based on

Twitter's guidelines), and assignment of a trust score from 0% to 100%. A browser extension further allows users to check profiles directly while browsing.

The platform is widely praised for its 95% accuracy and ease of use, promoting healthier online discourse. However, it only supports Twitter/X and may miss bots that avoid toxic behavior or operate across multiple platforms, which limits its detection capabilities in today's cross-platform disinformation campaigns.

2.4 Comparative Analysis and Identified Gaps

This overview highlights that current bot detection platforms share some common strengths and weaknesses, which can be grouped into functional categories for further analysis.

1. Bot Detection Methodologies

- **Pegabot** applies heuristic rules and network behavior analysis to estimate automation probability.
- **LiveDune** uses statistical models to evaluate engagement patterns and detect fake metrics.
- **Bot Sentinel** relies on supervised machine learning to classify manipulation and harassment patterns.

All three approaches leverage different perspectives on bot behavior. However, none combine user interaction or cross-platform detection.

2. Social Network Scope

- Pegabot and Bot Sentinel are limited to Twitter/X.
- LiveDune supports multiple platforms, such as Instagram, X, TikTok, and Facebook.

This limitation reduces the ability to detect coordinated bot operations across multiple networks—a growing tactic in modern influence campaigns.

3. User Interaction and Transparency

- Bot Sentinel provides a transparency layer through trust scores.
- Pegabot allows public queries but lacks voting or feedback mechanisms.
- LiveDune restricts access behind a paywall and offers limited user control.

User participation in detection and decision-making remains largely unexplored in these tools.

4. Focus on Disinformation and Harmful Content

- Bot Sentinel addresses manipulation and toxicity directly.
- Pegabot and LiveDune are more generalist, without an explicit focus on disinformation.

Given the increasing use of bots in coordinated disinformation campaigns, a more focused approach is necessary.

5. Limitations Identified in Existing Work

From the reviewed tools, three key gaps were identified:

1. Lack of cross-platform capabilities;
2. Minimal or no community participation in the detection process;
3. Absence of customizable blocklists and user-controlled interventions.

These limitations informed the design of *Bot Blocker*, a solution that integrates artificial intelligence with community moderation and allows users to create personalized blocklists. It extends the scope beyond a single platform and fosters collective intelligence through transparent voting and justification mechanisms. Through this synthesis, *Bot Blocker* positions itself as a next-generation tool for detecting and mitigating the influence of malicious bots across social media platforms.

Chapter 3

Requirements Gathering

3.1 Functional Requirements

3.1.1 User Authentication and Authorization

- **FR1**: Users must be able to register, log in, and manage their accounts securely.
- **FR2**: The system must define three user roles: regular user, verifier, and administrator.
- **FR3**: Administrators must be able to perform all actions available to verifiers.
- **FR4**: Each role must have distinct permission levels enforced by the system.
- **FR5**: Administrators must be able to promote regular users to the verifier role.
- **FR6**: Only administrators must have the ability to suspend or ban user accounts on the platform.

3.1.2 Profile Evaluation and Scoring

- **FR7**: Users must be able to evaluate social media profiles (from Twitter, Instagram, and Facebook) as either “bot” or “not bot”.
- **FR8**: Only authenticated users must be allowed to evaluate social media profiles.
- **FR9**: Evaluations must allow optional written justifications.
- **FR10**: The system must calculate a bot probability score for each profile, based only on the number of evaluations received.
- **FR11**: The bot probability score must be publicly visible for every evaluated profile.
- **FR12**: Users must be able to view the latest evaluations of a profile, including the @handle of the BB account that submitted each evaluation.

3.1.3 Verification and Badging

- **FR13**: Verifiers must be able to assign verification badges ("Bot" or "Human") to social media profiles.
- **FR14**: Administrators must also be able to assign verification badges to social media profiles.
- **FR15**: Verifiers and administrators must be able to remove verification badges from social media profiles.
- **FR16**: Verification badges must be displayed clearly on both the website and the Chrome extension.

3.1.4 Blocklist and Filter Management

- **FR17**: Users must be able to manually block social media profiles.
- **FR18**: Users must be able to create and manage personal blocklists.
- **FR19**: Users must be able to configure custom filters that automatically block profiles with a bot probability score above a chosen threshold.
- **FR20**: Users must be able to view and manage all blocked profiles through the extension interface.

3.1.5 Browser Extension Functionality

- **FR21**: The Chrome extension must allow users to evaluate social media profiles directly within Twitter, Instagram, and Facebook.
- **FR22**: The extension must display the bot probability score for each profile.
- **FR23**: The extension must display the verification badge assigned to a profile, if any.
- **FR24**: Users must be able to log in and register via the extension interface.
- **FR25**: Users must be able to apply block filters or manually block profiles from the extension.
- **FR26**: The extension must not negatively affect the user experience on supported social media platforms.

3.1.6 Administrative Tools and Monitoring

- **FR27**: Administrators must be able to suspend platform user accounts for a defined time period.
- **FR28**: Administrators must be able to permanently ban platform user accounts.
- **FR29**: Administrators must be able to view a complete log of actions performed by a user account, including evaluations submitted and interactions with profiles.
- **FR30**: Administrators must be able to view abnormal behavior patterns, such as unusually high-frequency voting from a single account.
- **FR31**: The system must automatically alert administrators when suspicious voting activity is detected.
- **FR32**: Verifiers must be able to view flagged profiles that may require further review.
- **FR33**: Verifiers must be able to access full evaluation history of a profile when reviewing it.

3.2 Non-Functional Requirements

3.2.1 Performance

- **NFR1**: The system must respond to profile evaluation and scoring operations in under 2 seconds under normal conditions.
- **NFR2**: The platform must support a minimum of 10,000 concurrent users without significant degradation in performance.
- **NFR3**: The Chrome extension must update and display bot score data with a latency of no more than 1 second.

3.2.2 Availability and Reliability

- **NFR4**: The system must maintain an uptime of at least 99.9% over a 12-month period.
- **NFR5**: The system must recover from server failures or outages within 5 minutes.
- **NFR6**: The platform must have a Mean Time Between Failures (MTBF) of at least 30 days.
- **NFR7**: The platform must have a Mean Time to Recover (MTTR) of no more than 4 hours.

3.2.3 Security and Data Protection

- **NFR8**: User authentication must use secure industry-standard protocols, including Google OAuth 2.0.
- **NFR9**: All data must be encrypted both in transit and at rest, using modern encryption standards.
- **NFR10**: User passwords must be hashed using cryptographically secure algorithms (e.g., bcrypt or Argon2) before storage.
- **NFR11**: Passwords must meet complexity requirements, such as a minimum length and inclusion of special characters.
- **NFR12**: All user input must be sanitized to prevent injection attacks, including XSS, SQL injection, and CSRF vulnerabilities.
- **NFR13**: User data and profile-related data must be logically segregated to prevent data leakage between different accounts or services.
- **NFR14**: Internal identifiers must be used to anonymize user and profile interactions across services when needed.
- **NFR15**: The system must comply with GDPR or equivalent privacy laws, ensuring that no personal data is shared without explicit user consent.

3.2.4 Scalability

- **NFR16**: The platform must be designed to scale horizontally to support increases in user traffic and data volume without requiring major architectural changes.

3.2.5 Usability and Accessibility

- **NFR17**: The user interface (web and extension) must be intuitive and easy to navigate, requiring less than 10 minutes for a new user to become proficient.
- **NFR18**: The platform must comply with WCAG 2.1 accessibility standards to support users with disabilities.
- **NFR19**: Usability testing must be conducted with at least 20 users and must demonstrate a task success rate of at least 68%.

3.2.6 Maintainability and Modularity

- **NFR20**: The platform must follow a modular software architecture (e.g., component-based or microservices) to simplify updates, scaling, and maintenance.
- **NFR21**: The codebase must include clear inline documentation and a user manual to support ongoing development and onboarding.

3.2.7 Compatibility

- **NFR22**: The Chrome extension must be compatible with the latest stable release of Google Chrome.

- **NFR23**: The website must be fully responsive and optimized for desktop, tablet, and mobile devices.

- **NFR24**: The platform must operate consistently across major browsers (e.g., Chrome, Firefox, Edge), with no more than 10% of features showing degraded behavior on any supported browser.

3.2.8 Interoperability

- **NFR25**: The platform must reliably integrate with Twitter, Instagram, and Facebook APIs to retrieve and display social media profile data.

- **NFR26**: When external services are available, the platform must ensure consistent functionality and minimize disruptions caused by external API limitations or downtime.

3.2.9 Extensibility

- **NFR27**: The platform must expose well-documented RESTful APIs to support integration with future services, dashboards, or tools.

- **NFR28**: Adding new social media platforms or external services must be achievable without requiring significant codebase changes.

3.3 Actors

Actor	**Role**
Unauthenticated User	A visitor who can view public profile information, configure temporary filters, and maintain a private local blocklist.
Authenticated User	A registered user who can evaluate profiles, view evaluation histories, and manage persistent filters and blocklists.
Verifier	A trusted user with elevated privileges to assign or remove verification badges from profiles.
Administrator	Responsible for managing the platform, promoting users to verifiers, monitoring voting activity, and enforcing system rules.

3.4 Use Cases

3.4.1 Model

During the requirements gathering phase, different types of users and their associated functionalities were identified. These functionalities were organized into an initial use case model, which is presented in the following image. The use cases are categorized by user type: Administrator, Verifier, Authenticated User, and Non-Authenticated User.

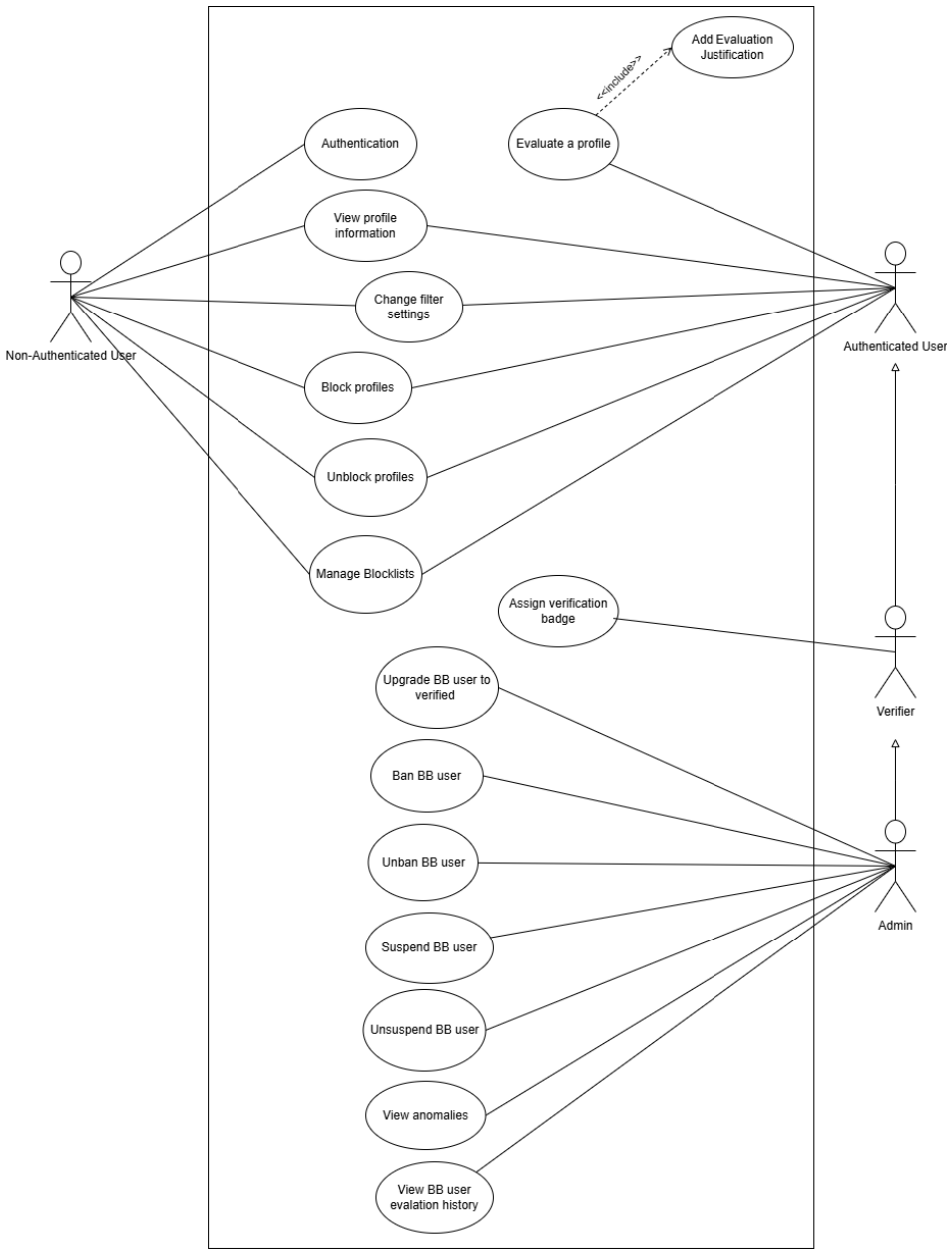


Figure 3.1: Use Cases diagram

3.4.2 Description

Table 3.1: Use cases' description

Use Case	Description
Authentication	Provides access to the system by logging in or registering. Required to perform certain actions such as voting or applying persistent filters.
View profile information	Displays public data of a profile, including its verification status, trust score, and the latest evaluations left on the profile. Available to all users.
Change filter settings	Allows users to configure filters for hiding or showing profiles based on bot probability, verification status, or custom criteria. Also allows to switch between censoring of completely omitting posts from blocked accounts.
Evaluate profile	Enables users to classify a profile as "bot" or "not bot". This evaluation influences the global trust score of that profile. Users may optionally add a justification to support their evaluation.
Assign Verification Badge	Enables verifiers or administrators to label a profile as "Bot" or "Human." This adds credibility and helps users interpret the profile's behavior. The badge can also be reset to "Unknown" if needed.
Upgrade BB User to Verified	Promotes a regular BB user to the verifier role. Verifiers gain additional permissions such as assigning badges.
Ban BB User	Permanently revokes a BB user's access to the platform. Banned users cannot log in or perform any actions unless unbanned by an administrator.
Unban BB User	Restores access to a previously banned BB user account, allowing them to use the platform again.
Suspend BB User	Temporarily restricts a BB user from accessing the platform. Suspended users cannot log in or interact until reviewed and unsuspended.
Unsuspend BB User	Reactivates a previously suspended BB user, restoring access to their account.
View Anomalies	Displays flagged behavioral patterns, such as excessive voting or spam activity, to help administrators detect irregular user behavior.
View BB User Evaluation History	Shows the complete history of evaluations made by a BB user, supporting transparency and moderation decisions.
Block Profiles	Prevents content and interactions from selected profiles from appearing in the user's interface.
Unblock Profiles	Reverses a previously applied block, restoring the ability to see and interact with the unblocked profile.
Manage Blocklists	Allows users to remove entries from their personalized list of blocked profiles.

Chapter 4

Implementation

4.1 Architecture

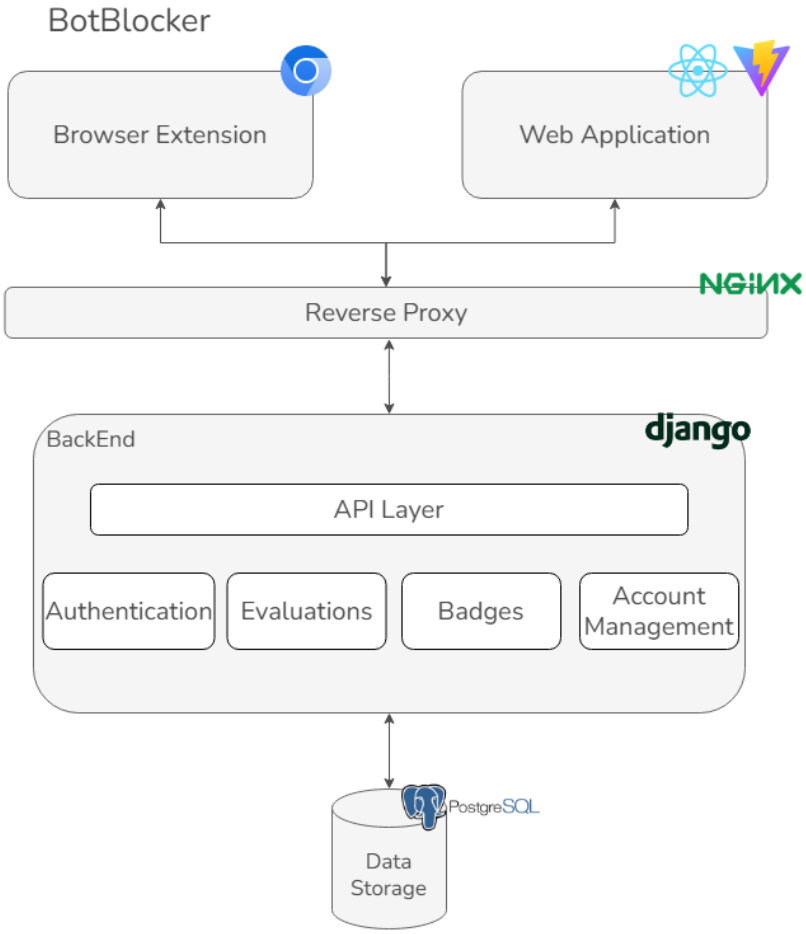


Figure 4.1: Architecture Diagram

4.2 System Architecture

The **BotBlocker** platform follows a modular, layered architecture, designed to promote scalability, maintainability, and separation of concerns. The system is structured into three main tiers:

1. **Presentation Layer:**
   Comprises the user-facing interfaces: a *browser extension* and a *web application*. These clients interact with the backend exclusively via REST API calls and WebSocket events. While the browser extension allows real-time DOM manipulation and profile evaluation/blocking without requiring authentication, the web application provides more advanced functionality such as badge assignment, administrative control, and system monitoring.

2. **Business Logic Layer (Backend):**
   Developed using *Django*, the backend serves as the core of the system, exposing a RESTful API that manages all business rules and workflows. It includes modules for:

   - Authentication and JWT token management.
   - Profile evaluation processing and consistency checks.
   - Badge assignment logic based on role-based access control (RBAC).
   - Anomaly detection through deterministic analysis of vote patterns.
   - User and role management for verifiers and administrators.

   All requests from clients pass through this layer, ensuring strict validation, authorization, and policy enforcement.

3. **Data Layer:**
   The backend interacts with a *PostgreSQL* database that stores persistent data such as:

   - User credentials and roles,
   - Profile metadata and reputation history,
   - Evaluation logs,
   - Settings and blacklist entries.

   This layer is abstracted from clients and interacts solely through the Django ORM.

To ensure efficient request handling and secure routing, the system incorporates a **reverse proxy layer** using *NGINX*. This component acts as a gateway, managing HTTPS termination, static file serving, load balancing, and forwarding of requests to the Django application.

4.2.1 Presentation Layer

The **Presentation Layer** is responsible for all user interaction with the system. It consists of two main components:

- A **browser extension**, which allows users to interact with social media platforms in real-time, offering functionalities such as account evaluation, content filtering, and direct profile blocking from within the timeline or profile view.

- A **web application**, developed using *React* and *Vite*, that provides a full-featured interface for badge management, evaluation history inspection, user settings, and administrative tools.

Communication between this layer and the backend is performed exclusively through **RESTful API calls**, and in specific cases (e.g., admin anomaly alerts), through **WebSocket connections**. This ensures a strict separation of concerns between frontend presentation and backend logic, which simplifies maintenance.

Although the browser extension is fully functional and integrates seamlessly with major platforms (X/Twitter, Instagram, Facebook), we opted not to publish it on the Chrome Web Store. Instead, the extension is freely available for download and manual installation through our open-source repository, allowing users and contributors to access and inspect the code directly.

The choice of **React** for the web application was driven mostly by our familirarity with it from previous projects, its modular component-based architecture, large community support, seamless integration with modern tooling . *Vite* was selected as the build tool for its fast hot-module replacement and minimal configuration.

The extension was designed to operate with minimal permissions and client-side. Its architecture supports *real-time DOM manipulation* through efficient use of `MutationObservers`. This enables dynamic and responsive user interactions without requiring full-page reloads, particularly important when dealing with modern single-page applications like Twitter/X and Instagram.

This architecture ensures not only a fluid and responsive experience for end users, but also facilitates the deployment and evolution of each interface independently of the core backend logic.

4.2.2 Business Logic Layer

The **Business Logic Layer** encapsulates the core functionality and domain-specific operations of the system. This layer was developed using the **Django** framework, a high-level Python web framework renowned for its simplicity, scalability, and built-in security features.

At the heart of this layer lies a structured and modular **API layer** that exposes endpoints for the following core functionalities:

- User authentication and session management via JWT tokens.
- Bot evaluation submission and profile reputation handling.
- Badge assignment and moderation workflows.
- User and profile management, including banning, timeouts, and access control.

Django's **Model-View-Template (MVT)** architecture facilitates clean separation of concerns, while the **Django ORM (Object-Relational Mapper)** abstracts database interactions into intuitive Python class definitions. This enable us to perform complex database operations without needing to write raw SQL, reducing both the likelihood of errors and the complexity of database access logic. For example, querying, creating, or filtering data can be done using expressive ORM syntax such as `Model.objects.filter(...)` or `Model.objects.create(...)`.

The framework also provides robust support for user permissions and authentication out of the box, which integrates seamlessly with the system's role-based access control (RBAC) logic used for administrators and verifiers.

This layer also serves as the **central decision-making unit** of the system, responsible for validating requests, applying business rules, coordinating between models, and returning structured responses through REST endpoints or WebSocket events. It ensures the consistency and integrity of operations before data reaches the storage layer.

Overall, Django's ecosystem significantly accelerates development without compromising on maintainability, making it well-suited for the complex backend needs of BotBlocker.

4.2.3 Data Layer

The **Data Layer** is responsible for persistent storage and data consistency across the system. It relies on a **PostgreSQL** relational database to manage structured data, including user accounts, evaluation logs, and metadata related to system interactions. For the anomalies, in the begging, we thought about using InfluxDB (timeseries DB) to store logs from the users but we decided not to use it since we find a way to calculate the anomalies in a deterministic way.

All database access is abstracted through Django's built-in **Object-Relational Mapper (ORM)**, which provides a consistent, high-level, and secure interface between the business logic and the underlying data model. This abstraction promotes maintainability, eliminates the need to write raw SQL for most operations, and ensures compatibility with Django's model validation and migration tools.

PostgreSQL was selected due to its robustness, ACID compliance, and excellent support for complex relational queries, indexing, and constraints, which are essential for ensuring data integrity in

evaluation-heavy systems. Additionally, PostgreSQL offers native support for JSON fields and advanced features like full-text search and concurrency controls, making it a reliable and scalable solution for our application's data needs.

This layer plays a crucial role in ensuring the **durability**, **integrity**, and **scalability** of the system's stored information, regardless of the load or usage pattern.

4.3 Technologies Used

The implementation of the *BotBlocker* system leverages a modular and modern technological stack designed to support performance, scalability, and developer efficiency across all system layers.

- **React & Vite:** Chosen for the web application's frontend due to their component-based structure, strong ecosystem, and efficient development cycle. *Vite* ensures fast builds and smooth hot module replacement during development.

- **Standard Web Technologies:** The browser extension is developed using native web technologies (HTML, CSS, JavaScript), ensuring broad compatibility and direct control over DOM manipulation.

- **Django (Python):** Selected for the backend due to its clean architecture, integrated ORM, and mature ecosystem. Django simplifies development of secure APIs and enforces best practices like modularization, authentication, and role-based access control.

- **PostgreSQL:** A reliable and scalable relational database system offering strong ACID compliance and advanced query capabilities. Its compatibility with Django's ORM ensures safe and efficient access to structured data.

- **NGINX:** Serves as a reverse proxy, handling HTTP routing, static content, and SSL termination. Its role is critical in improving the application's scalability and security.

- **Docker:** Used to containerize the backend, frontend, and database services, ensuring consistent deployment environments. This simplifies testing and production deployment, while isolating services for better maintainability.

The combination of these technologies ensures that the system is robust, modular, and production-ready, while supporting continuous integration and smooth developer workflows. Each technology was chosen to align with the system's real-time requirements and maintain a clear separation between frontend, backend, and data logic.

4.4 Data Domain

4.4.1 Domain Model

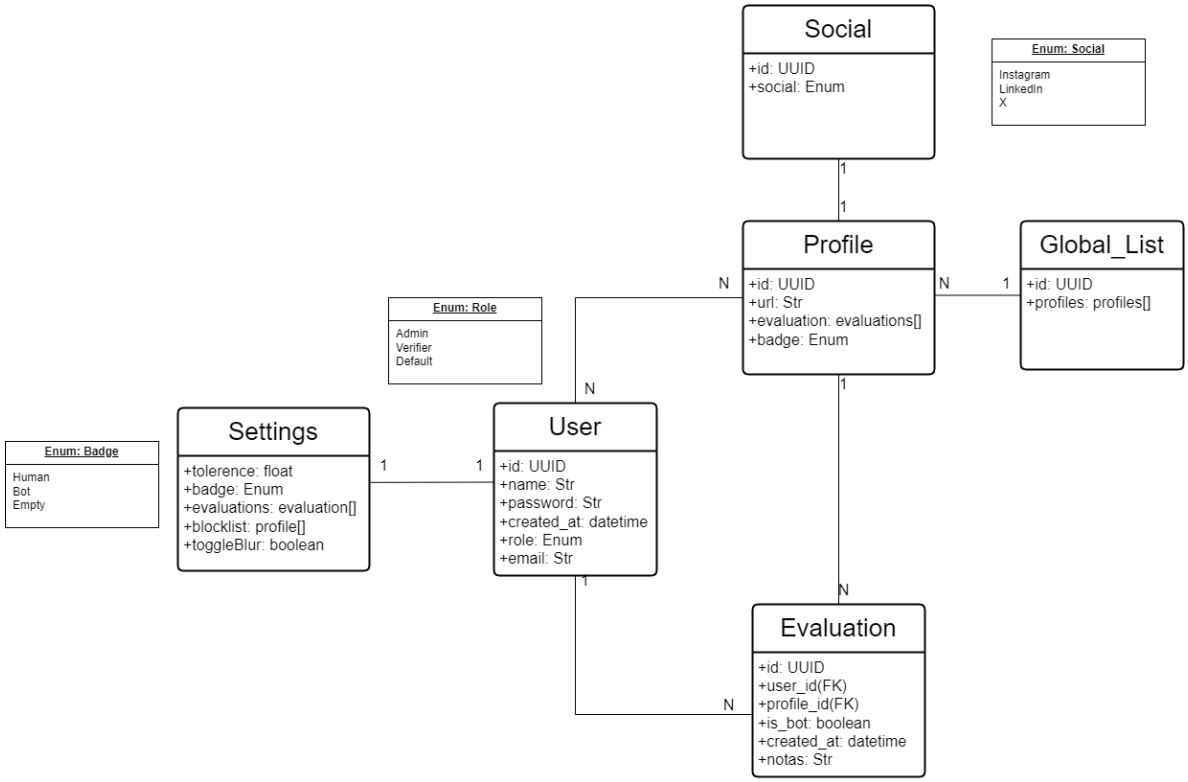


Figure 4.2: Domain Model

The domain model of the *BotBlocker* system is centered around three core entities: `User`, `Profile`, and `Evaluation`. These entities represent the primary functional relationships and data structures within the system.

User

The `User` entity represents an individual who holds an account within the system. Each user has a set of personal data and is associated with a specific `Role`, which can be one of the following enumerated types: `ADMIN`, `VERIFIER`, or `DEFAULT`. These roles determine the user's permissions and level of access within the platform. Additionally, a `User` is associated with a `Settings` object and a list of `Evaluation` instances that the user has submitted.

Profile

The `Profile` entity corresponds to a social media profile, such as Instagram, X (formerly Twitter), or Facebook, as specified by the `Social` enumeration. A `Profile` is a key object of analysis within the system. It may have a `Badge` (enumerated as `HUMAN`, `BOT`, or `EMPTY`) that is assigned by a user with `VERIFIER` privileges, and it can receive multiple `Evaluations` from different users. These badges serve as visible indicators to help users filter or prioritize which profiles to engage with.

Evaluation

The `Evaluation` entity is a weak entity that models the action of a user evaluating a specific profile. Each evaluation contains a Boolean field `is_bot`, indicating whether the evaluator believes the profile

is a bot, and an optional `comment` field for additional notes or justifications. Since each evaluation must be associated with both a `User` and a `Profile`, its existence depends on those entities.

Settings

The `Settings` entity is logically part of the `User` but is modeled separately for modularity and clarity, as all its attributes pertain specifically to user preferences. The attributes include:

- `tolerence`: an integer between 1 and 100 representing the minimum credibility score a profile must have to be visible to the user.
- `badge`: a filter to display only profiles with a specific `Badge` (e.g., only `HUMAN`).
- `evaluations`: a reference to the evaluations performed by the user.
- `blocklist`: a manual, private list of blocked profiles.
- `toggleBlur`: a flag that determines how blocked content is visually presented (e.g., blurred or hidden).

Global_List

The `Global_List` entity represents a centralized repository of all social media profiles that have passed through the system. It serves as a common source that can be filtered based on each user's `Settings`. This design supports scalable, personalized filtering without duplicating profile data across users.

4.5 Internal Lifecycle

1. User Registration and Authentication

Authentication is optional for using the browser extension and basic blocking functionalities. However, logging in enables synchronization of user preferences across devices and access to advanced features such as profile evaluation.

Initially, the system was designed to integrate Google OAuth for authentication. However, due to recurrent integration issues, such as inconsistent token validation and limitations in browser extension contexts, and considering that authentication was not a critical feature for baseline functionality, we decided to replace it with a custom authentication system. This custom approach based on JWT token issuance and Django's native authentication mechanisms offered greater control, better compatibility with our architecture, and simplified deployment.

User registration occurs via a POST request to `/create_user/`, where the `createUserBB` function validates input data, ensures uniqueness of username and email, and creates linked `User` and `User_BB` records. Upon success, the user is authenticated and receives JWT tokens (`access` and `refresh`) for secure session management.

Login is handled through the `/token/` endpoint by the `CustomTokenObtainView`, which authenticates credentials and checks for associated `User_BB` status, bans, or active timeouts. If valid, it returns the tokens along with the user ID and role, allowing the frontend to enforce role-based access control.

2. Profile Evaluation

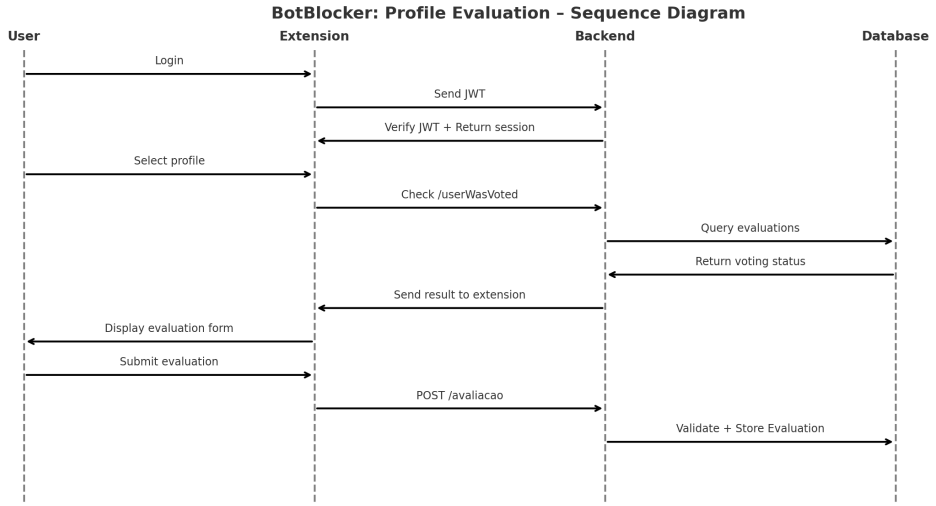


Figure 4.3: Caption

Profile evaluation requires authentication and its made on the extension. Once logged in, the system verifies user identity via JWT tokens. Before allowing a new evaluation, the frontend queries the backend using the `/userWasVoted` endpoint to determine whether the user has already evaluated the profile on a given platform. If not, the evaluation form is displayed.

When an evaluation is submitted, a POST request is sent to the `/avaliacao` endpoint containing the evaluation details (e.g., platform, profile identifier, vote, comment). The backend validates the data and stores a new `Evaluation` entity, linked to both the user and the target profile. This process contributes to building a reputation history for profiles within the system.

3. Badge Assignment and Moderation (Verifier Role)

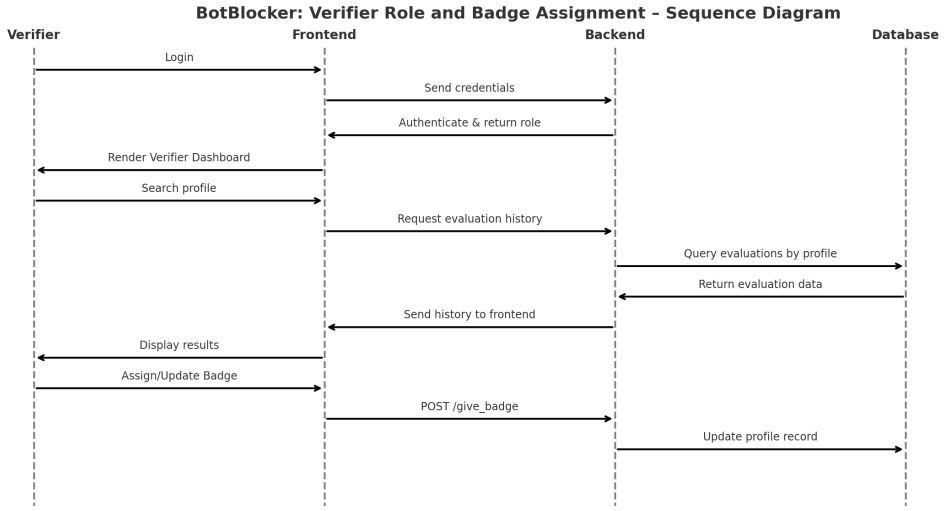


Figure 4.4: Caption

Users with the `VERIFIER` role can access a dedicated interface (the Verifier Dashboard) through the web application. Upon successful login, the system returns the user's role and if the role is `VERIFIER`

or `ADMIN`, the dashboard is made accessible through conditional UI rendering.

In this dashboard, verifiers can search for specific profiles and retrieve their evaluation history through the `/getEvaluationHistory` endpoint. This includes all recorded evaluations for the profile, along with platform metadata and usernames. Based on this evidence, verifiers can assign or update a profile's badge (e.g., `HUMAN`, `BOT`, or `DEFAULT`) by sending a POST request to the `/give_badge` endpoint, which updates the associated `Profile` record in the database.

4. Admin Role and Anomaly Detection

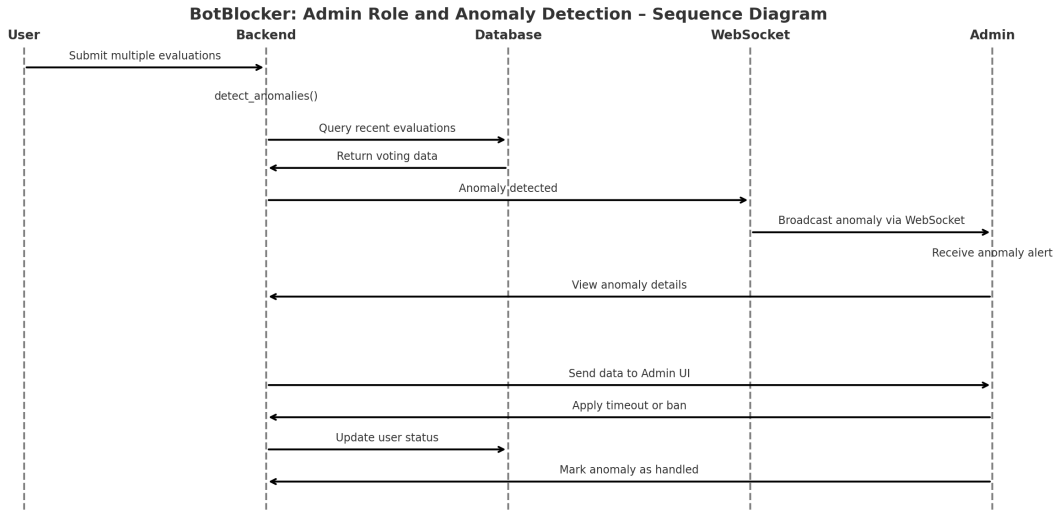


Figure 4.5: Caption

Users with the `ADMIN` role are granted exclusive access to the Admin Dashboard, which includes a real-time anomaly monitoring system. Anomalies refer to atypical patterns detected in user or system behavior, such as:

Vote Attacking: A profile receiving an excessive number of evaluations in a short time span.
Spam Voting: A user submitting evaluations at an unusually high frequency.

These behaviors are detected deterministically by the `detect_anomalies` backend function, which analyzes voting frequency over time intervals. When an anomaly is detected, the backend emits a WebSocket event to all active admin sessions, notifying them of the occurrence. Each anomaly includes an identifier, the associated user or profile, and a description of the issue.

Administrators may respond by applying sanctions to users responsible for the anomalies. These include timeouts or permanent bans, executed through protected endpoints such as `/users/ban` and `/users/timeout`, which modify the affected user's permissions and access. They can also change the status of the anomaly to `solved` or `in_progress` to inform other administrators that the anomaly is being handled.

Additionally, the system implements Role-Based Access Control (RBAC), allowing administrators to manage user roles directly. This includes promoting trusted users to roles such as `VERIFIER`, demoting users, or revoking elevated privileges via secured endpoints like `/promote_user`.

5. Real-Time Blocking and DOM Manipulation

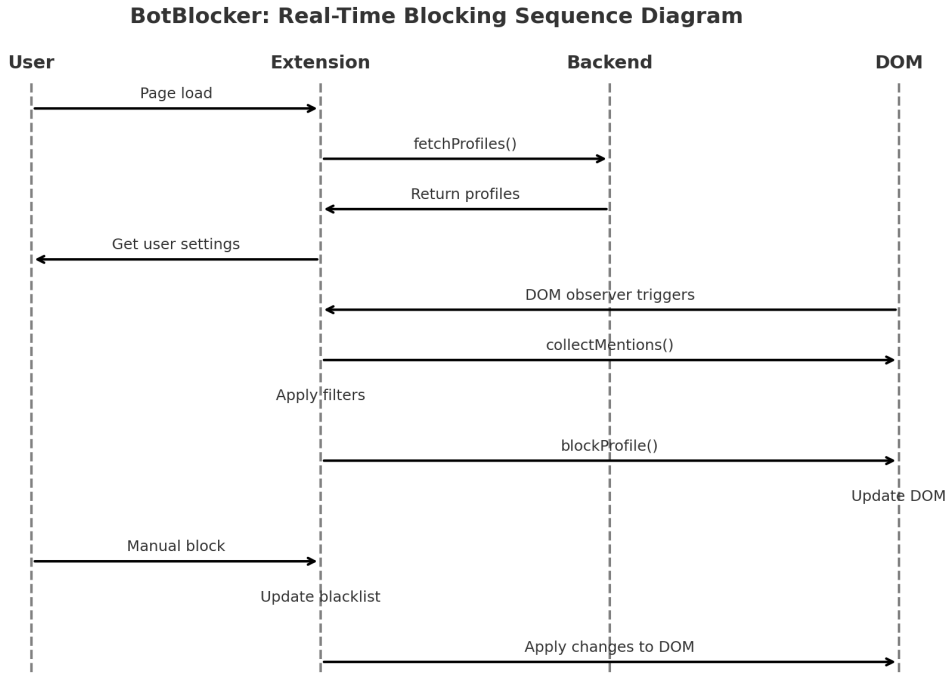


Figure 4.6: Real-Time Blocking Sequence Diagram

The real-time blocking logic is embedded in the browser extension's content script, which operates by continuously monitoring page content and applying filtering rules defined by the user. Upon initialization, the extension fetches all globally evaluated profiles from the backend using the `fetchProfiles` function. It then merges this list with the user's private blacklist to produce a unified set of blocked profiles.

The extension distinguishes between two types of blocking:

- **Manual Blocking:** Performed explicitly by the user via the extension interface, which adds the target profile to the private blocklist.
- **Automatic Blocking:** Triggered when a profile's badge or credibility percentage exceeds the thresholds configured in the user's settings.

The extension employs `MutationObserver` instances to detect DOM changes and SPA navigations, enabling it to:

- Apply blocking logic dynamically as new content loads.
- Monitor transitions between feed views and individual profile pages.

Each supported platform has a tailored DOM manipulation strategy:

- **X (Twitter):** On profiles, the extension intercepts GraphQL post requests and returns empty data to prevent content loading, simulating a "blocked" state. In the feed, specific posts are blurred or removed, with visual indicators added for transparency.
- **Instagram:** On profiles, a blocking overlay is rendered, and content-fetching requests are intercepted. In the feed, blocked posts and stories are either blurred or collapsed to minimal space to preserve layout.
- **Facebook:** Profile content is removed by manipulating primary content containers. In the feed, posts and story previews are targeted based on heuristics such as `aria-label`, visible text, and image `alt` attributes.

All blocking actions are reversible and synchronized with the browser's local storage, which maintains up-to-date lists of blocked accounts and associated metadata (e.g., avatars). The system also accounts for edge cases such as platform-specific loading glitches and route-based state changes, ensuring consistent behavior across navigation events.

Chapter 5

Results

5.1 Usability Tests

5.1.1 Introduction, Consent and Sample

To validate the usability of the *BotBlocker* system across different user profiles, structured usability tests were conducted, focusing on both the browser extension and the web application. The primary objective was to identify usability barriers, assess interface performance, and collect both quantitative and qualitative feedback to guide improvements before the final release.

All participants were informed about the purpose of the study, their voluntary participation, and the anonymous nature of their data. Informed consent was obtained from every participant prior to the tests.

The sample consisted of 11 participants, divided into three target groups:

- **Group A – Regular Users (7 participants):** University students aged between 20 and 23, with moderate to high digital literacy but no specific technical background. This group represented typical social media users.
- **Group B – Technical Users (2 participants):** Computer engineering students with experience in Human-Computer Interaction (HCI). This group provided critical feedback from a technical perspective, with a focus on logic, efficiency, and structural consistency.
- **Group C – Low Digital Literacy Users (2 participants):** Adults around 50 years old, occasional users of digital tools with limited familiarity with browser extensions or advanced interfaces. This group was essential for evaluating accessibility and intuitiveness for less tech-savvy users.

Participants were asked to complete a set of predefined tasks that reflected real usage scenarios, such as evaluating profiles, blocking suspicious accounts, or configuring personal settings. After completing the tasks, each participant filled out a System Usability Scale (SUS) questionnaire and provided open-ended feedback through a dedicated form.

Observers used a structured observer script to record behaviors, navigation difficulties, and spontaneous remarks or errors. These qualitative insights complemented the structured feedback.

5.1.2 Method

The usability evaluation followed a task-based testing protocol. Each participant was asked to complete six practical tasks, split between the browser extension and the web platform. Each task reflected common use cases.

Tasks Overview

1. (Extension) Block a suspicious profile directly on Twitter.
2. (Extension) Evaluate a profile as a bot.

3. (Extension) Adjust the filtering threshold to hide likely bot profiles.

4. (Website) View the evaluation history of a profile.

5. (Website) As an Admin, apply a timeout to a user.

6. (Website) As a Verifier, assign a "Human" badge to a profile.

Participants received a brief system introduction for context, but no step-by-step instructions, in order to simulate a first-time user experience.

Observers recorded:

- Whether tasks were completed and done correctly;
- Time taken to complete each task;
- Navigation issues or confusion;
- Need for assistance;
- Spontaneous user remarks or errors.

Participants also filled out a usability test form after each task, rating difficulty (1–5) and answering short qualitative questions.

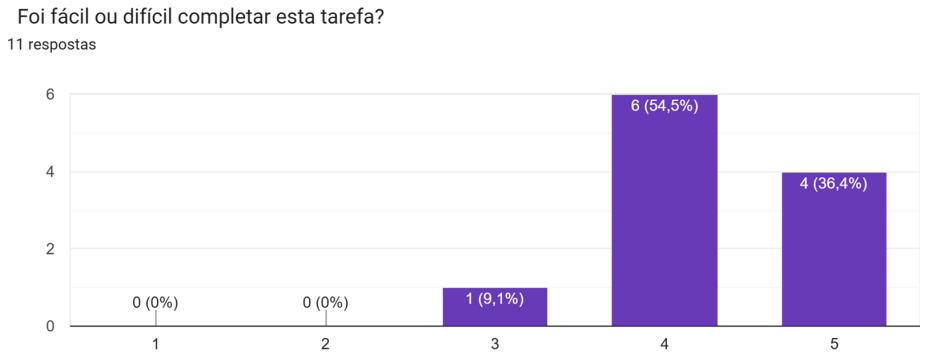


Figure 5.1: Example of task answer analysis

Finally, participants completed the System Usability Scale (SUS), a 10-question validated instrument for quantifying usability perception.

5.1.3 Results, Discussion and Changes

Quantitative Results

A total of 11 participants completed all six tasks and the SUS questionnaire.

- **Average Task Success Rate:** 98%
- **Average Task Completion Time:** 55 seconds per task

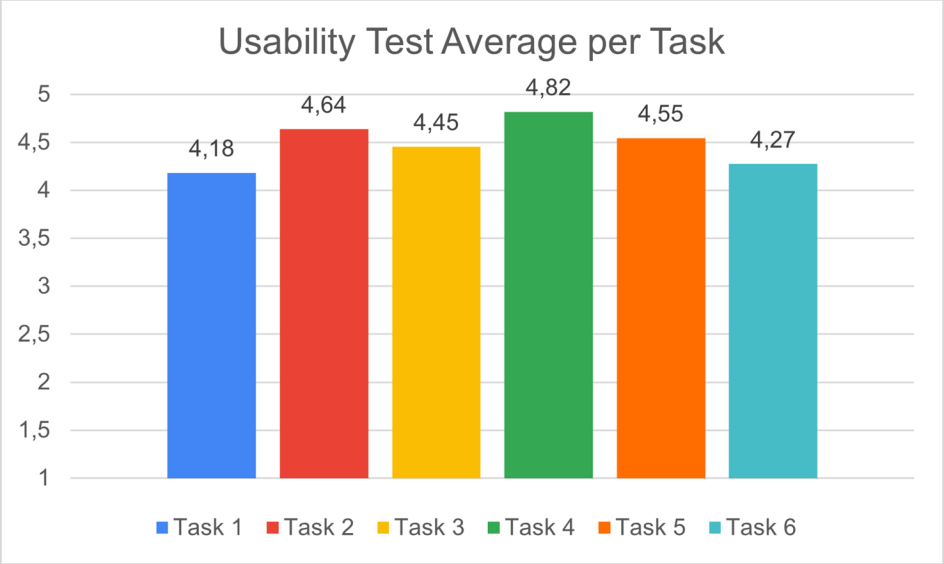


Figure 5.2: Usability test average difficulty per task

Tasks	Did the user complete the task?	Correctly? (Y/N) (correct answer)	Max Time Observed time (mm:ss)	Number of errors?	Was lost?	Asked for help	Observed Easiness/difficulty 1 – very difficult 5 – very easy
1	no | | yes | X |	y	1min 00:45	0	no | X | slightly |_| a lot |_|	no | X | yes |_| which?	1 2 3 4 X
2	no | | yes | X |	y	1min 00:30	0	no | X | slightly |_| a lot |_|	no | X | yes |_| which?	1 2 3 4 X
3	no | | yes | X |	y	1min 01:00	1	no | X | slightly |_| a lot |_|	no | X | yes |_| which?	1 2 X 4 5
4	no | | yes | X |	y	1min 00:55	0	no | X | slightly |_| a lot |_|	no | X | yes |_| which?	1 2 3 X 5
5	no | | yes | X |	y	3min 01:35	1	no |_| slightly | X | a lot |_|	no |_| yes | X | which? ("É aqui que aplico o timeout?")	1 2 X 4 5
6	no | | yes | X |	y	2min 01:00	0	no | X | slightly |_| a lot |_|	no | X | yes |_| which?	1 2 3 X 5

Figure 5.3: Example observer script for a regular user

Performance by Group

Group A (Regular Users) performed consistently across all tasks. Most rated task difficulty between 4 and 5. Some minor confusion was observed in tasks requiring admin functions.

Group B (Technical Users) showed excellent performance with no observed confusion. Tasks were completed quickly and correctly.

Group C (Low Digital Literacy Users) needed more time and occasional assistance. Admin-related tasks were more difficult, but participants still appreciated the system and its potential.

Common Observations

Positive Highlights

- All users completed profile blocking and evaluation (Tasks 1–2) successfully.
- Filtering settings (Task 3) were well understood.
- Profile evaluation history (Task 4) was easy to access.

Pain Points Identified

- Admin and verifier panels lacked explanatory labels or guides.
- Some users were unfamiliar with terms like *timeout* or *badge*.

System Usability Scale (SUS)

	A	B	C	D	E	F	G	H	I	J	K	L
1		User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	User 11
2	Question 1	4	5	3	4	5	4	5	4	5	3	4
3	Question 2	2	1	1	2	1	1	1	1	1	1	1
4	Question 3	4	4	5	4	5	5	5	5	5	4	4
5	Question 4	2	2	1	1	1	1	1	1	1	3	2
6	Question 5	5	5	5	5	5	5	5	5	5	5	5
7	Question 6	2	1	2	1	1	1	1	1	1	1	1
8	Question 7	5	5	5	5	5	5	5	5	5	5	5
9	Question 8	3	2	1	1	2	2	1	1	1	2	2
10	Question 9	4	4	4	5	4	4	5	5	5	3	4
11	Question 10	1	1	1	1	2	1	1	1	1	1	2
12												

Figure 5.4: System Usability Scale (SUS) responses

Group	**Average SUS Score**
Regular Users	91.07
Technical Users	98.75
Low Literacy Users	82.50
Overall Average	**90.91**

Table 5.1: SUS Score Averages by User Group

These results confirm excellent perceived usability across all participant types, exceeding the 68-point SUS benchmark. Technical users gave the highest ratings, while low-literacy users still rated the system positively.

Resulting Improvements

Based on participant and observer feedback, the following improvements were implemented:

- Tooltips were added to explain functions in admin and verifier interfaces (e.g., what a timeout does).
- Confirmation modals were introduced before assigning badges or applying account punishments.
- A new “Understand Bots” landing page was created to educate users:
  - Definitions of social bots;
  - Visual examples of bot and human profiles;
  - Bot detection strategies;
  - Real-world impact case studies.

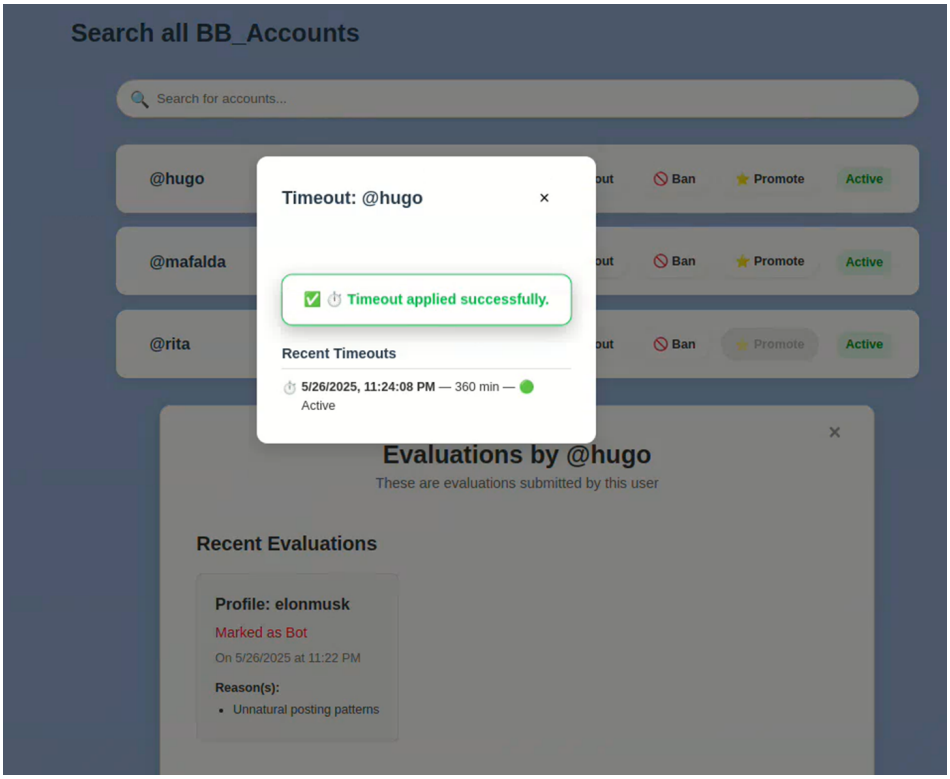


Figure 5.5: Confirmation modal after timeout action

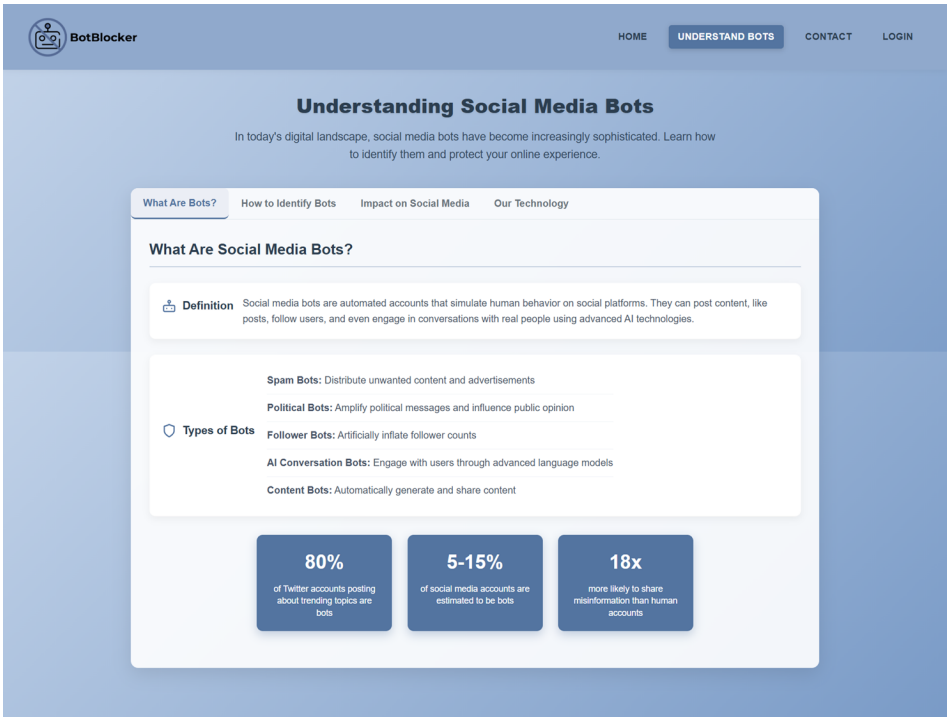


Figure 5.6: "Understand Bots" page – What are bots?

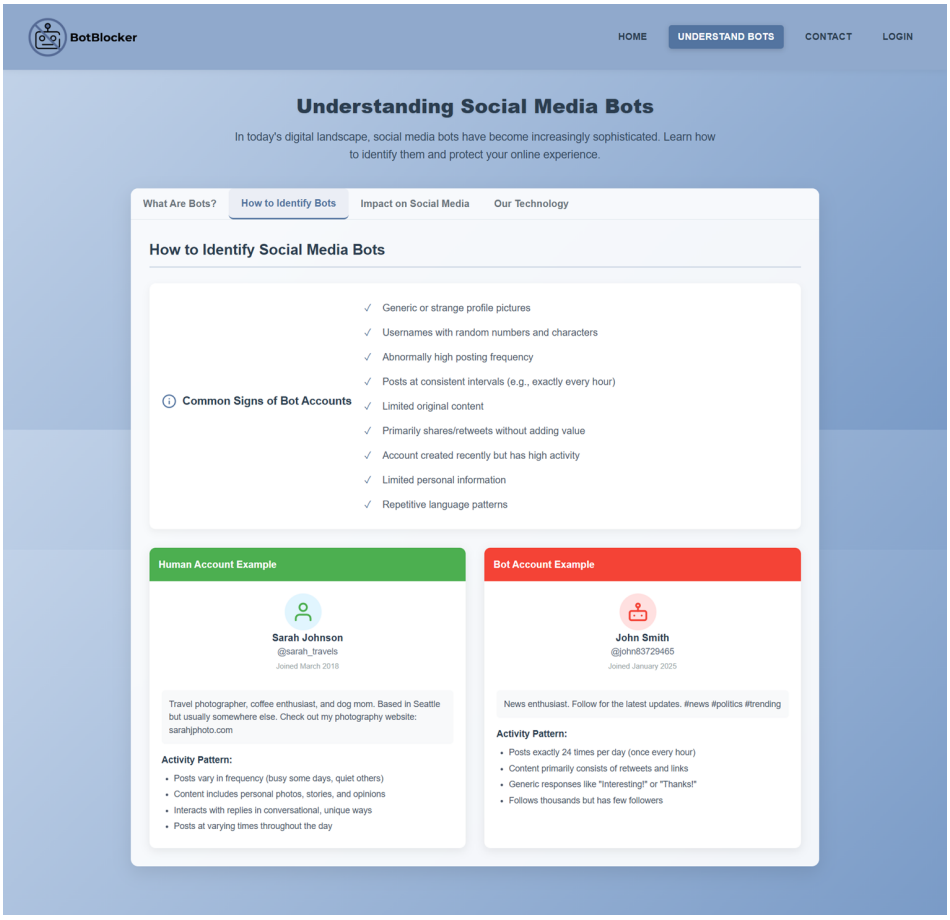


Figure 5.7: "Understand Bots" page – How to identify bots

5.2 Use Cases Materialized

This section demonstrates the implementation of the use cases proposed in Chapter 3, using screenshots of the system as visual examples of functionality. Where relevant, references to visible interface elements are made, helping the reader understand how each use case has been materialized in the system.

We begin with the authentication flow within the browser extension. Figure 5.8 displays the registration form, where the user is required to provide a unique username, a valid email address, and a password with at least 8 characters. Figure 5.9 follows with the login form, which also requires the same minimum password length and a valid username for access.

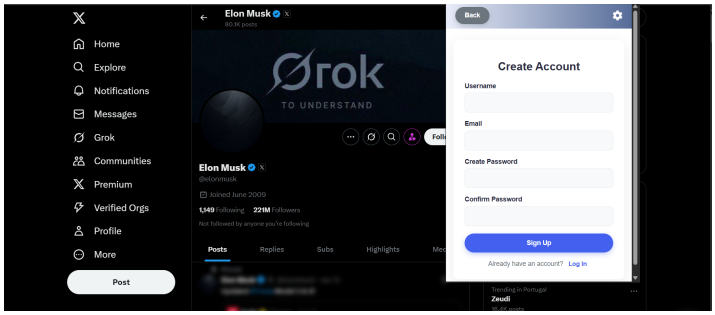


Figure 5.8: Registration form on the extension

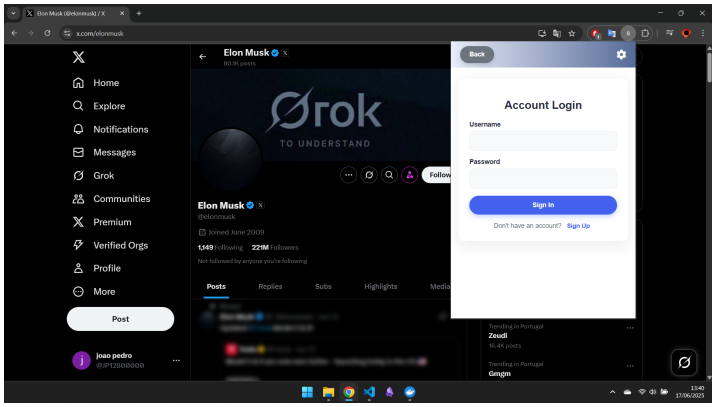


Figure 5.9: Login form on the extension

The same authentication process is available through the website, where users can create and access accounts. Figures 5.10 and 5.11 show the equivalent registration and login forms on the website.

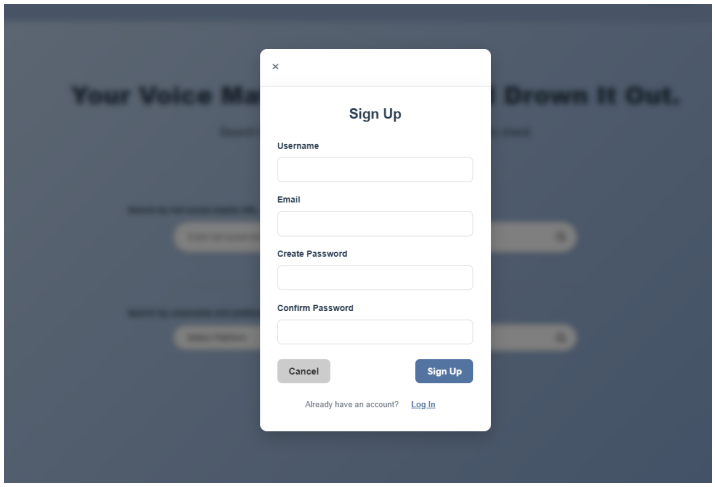


Figure 5.10: Registration form on the website

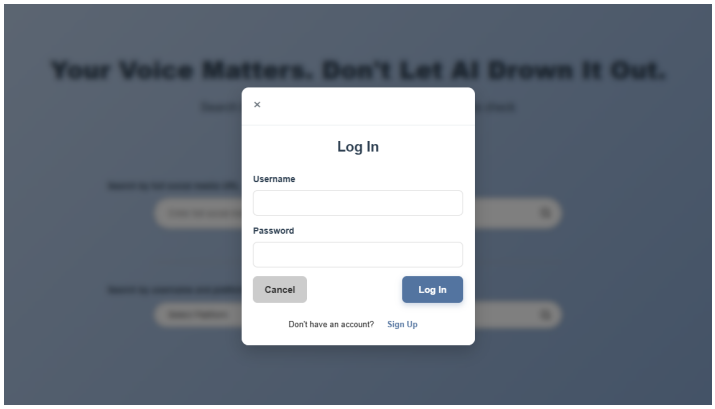


Figure 5.11: Login form on the website

Figure 5.12 illustrates how a logged-in user can evaluate a profile through the extension. The interface allows the user to vote on whether the profile appears to be real or is a Bot/AI. After submitting a vote, a justification can optionally be provided, as shown in Figure 5.13.

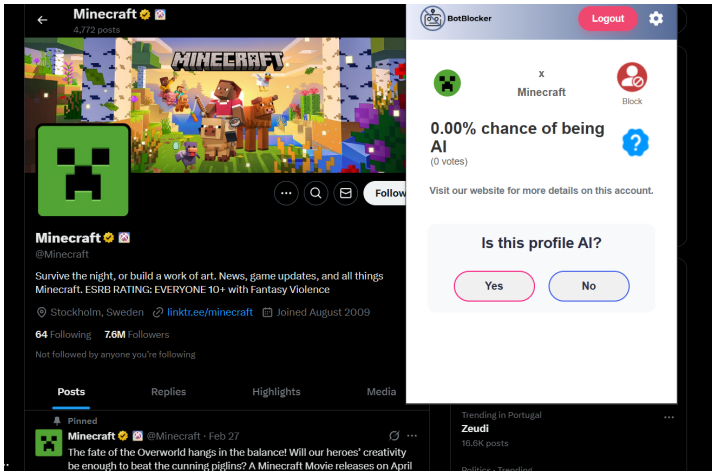


Figure 5.12: Evaluation interface within the extension

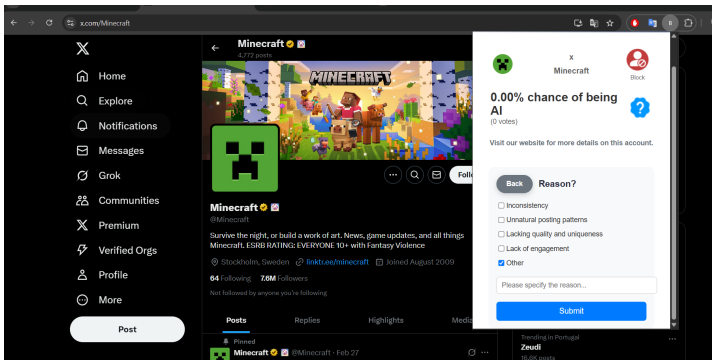


Figure 5.13: Justification screen for profile evaluation

Figure 5.14 illustrates the interface presented to a non-logged user. As shown, the voting options are disabled, and the user is prompted to log in before they can evaluate the profile.

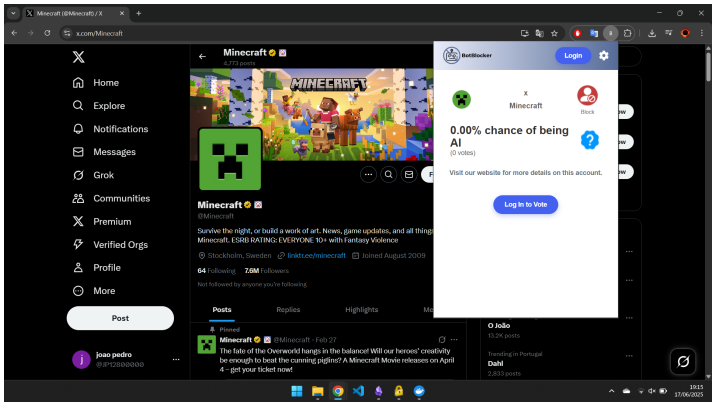


Figure 5.14: Interface for non-logged users, preventing them from voting

Users can configure filtering options through the extension settings menu. Figure 5.15 shows how users can configure filtering thresholds, select which types of accounts should be automatically blocked, choose whether to remove posts entirely or blur them, and manage blocked profiles through a blacklist.

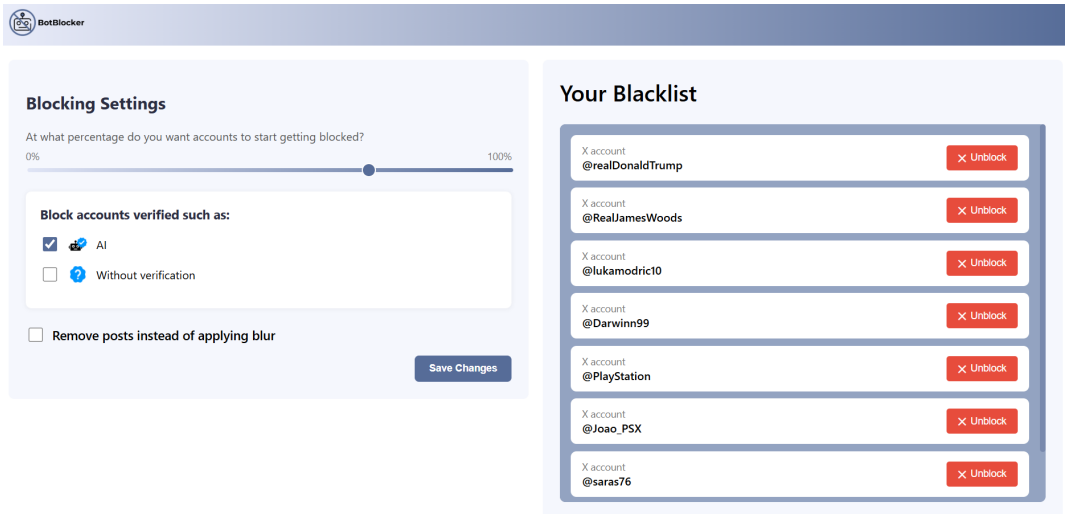


Figure 5.15: Filtering and blocking configuration in the extension

Users can also manually block or unblock profiles. Figure 5.16 demonstrates the block option available while viewing a profile, while Figure 5.17 shows the corresponding interface for unblocking previously restricted accounts.

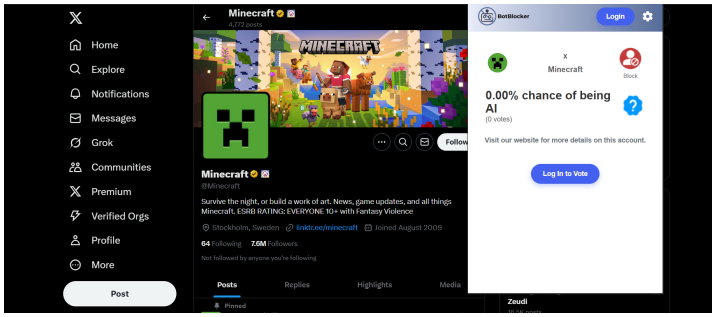


Figure 5.16: Manual blocking of a profile

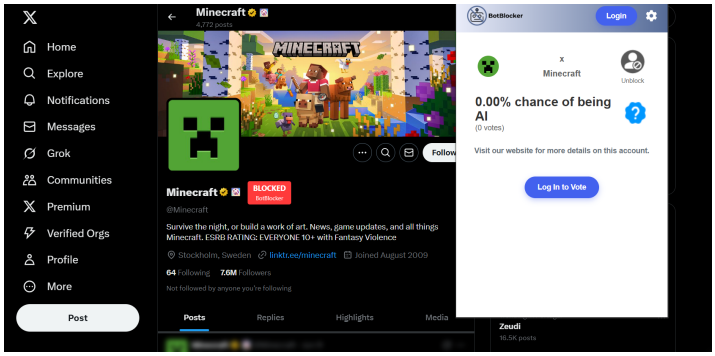


Figure 5.17: Manual unblocking of a profile

On the website, users can search for and view detailed information about a profile, including its bot likelihood, the latest evaluations made on the profile, and the assigned badge. This is shown in Figure 5.18.

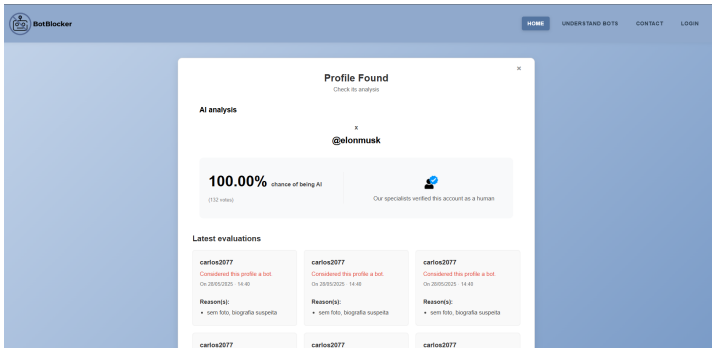


Figure 5.18: Profile information viewer on the website

Verifiers and administrators can flag profiles as a real human or bot. Figure 5.19 presents the interface where these decisions are made and where badges can be applied or removed accordingly.

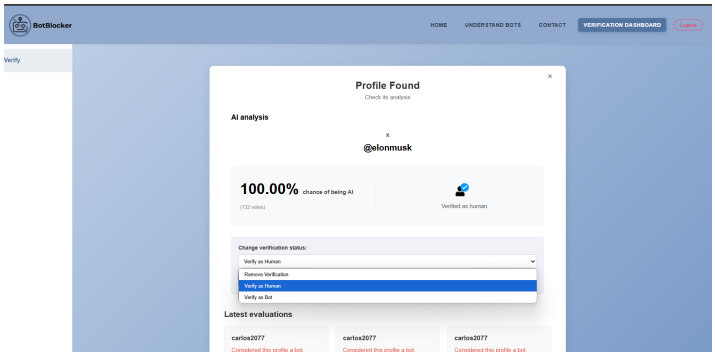


Figure 5.19: Flagging a profile as Bot or Human

The admin dashboard, shown in Figure 5.20, allows administrative users to manage BB accounts, perform moderation actions such as banning and unbanning, suspending or lifting suspensions, promoting users to verifiers, and reviewing individual evaluation histories of BB accounts.

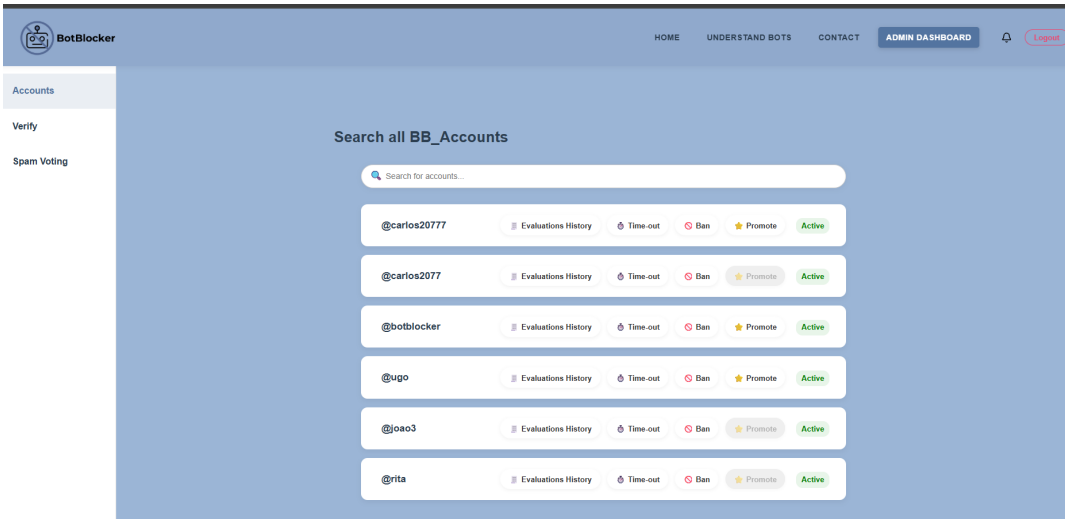


Figure 5.20: Administrative dashboard

Administrators can configure temporary suspensions to restrict user activity. Figure 5.21 shows the interface used to manage these suspensions.

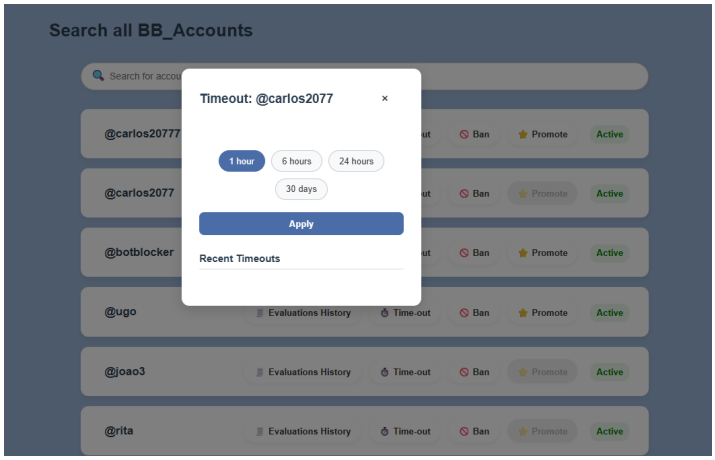


Figure 5.21: Suspension configuration interface

Figure 5.22 shows the evaluation history of a BB user. This view supports moderation and transparency by enabling a full audit of past votes.

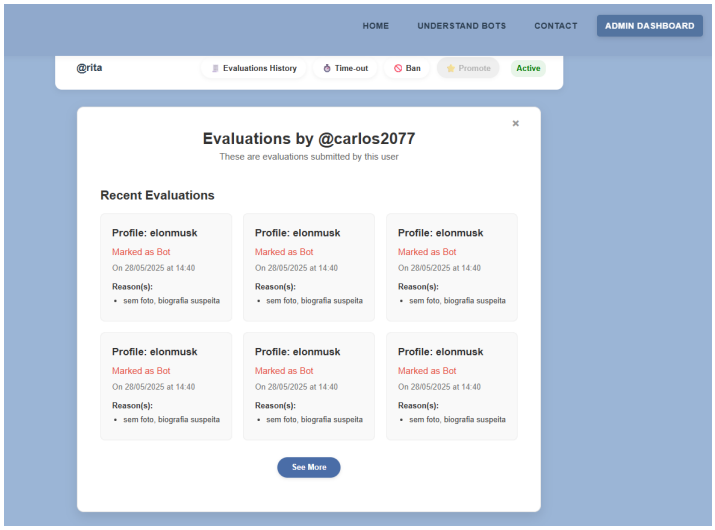


Figure 5.22: Evaluation history viewer for a BB user

Finally, Figure 5.23 presents the anomaly detection interface. This table displays patterns such as abnormal voting behavior or signs of spam activity, supporting the use case related to platform monitoring for misuse.

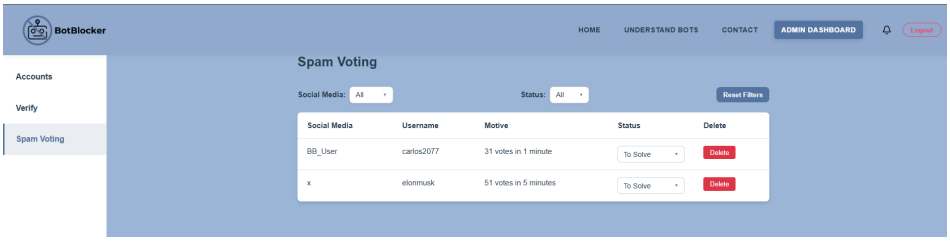


Figure 5.23: Anomaly detection interface

Chapter 6

Conclusion

The BotBlocker project has proven to be an innovative step towards solving the growing issue of AI-driven bots on social media. By focusing on a community-driven approach to evaluating and verifying the authenticity of social media accounts, the project emphasizes the importance of user participation in the battle against bots. While the system demonstrated several strengths, it also faced challenges that provide valuable lessons for future improvements. The combination of a cross-platform design, flexible user interface, and scalable architecture provides a strong foundation for future work, but technical and integration hurdles remain, offering room for enhancement.

6.1 PMI Evaluation

The project demonstrated several positive outcomes, starting with its **cross-platform support**. Designing the system to function across Twitter (X), Instagram, and Facebook involved creating specific strategies for each platform. For instance, while posts on X could be blurred efficiently by accessing API calls directly, Instagram and Facebook required HTML manipulation, resulting in slower performance but more thorough censorship. Despite these performance concerns, the flexibility of the design proved valuable, allowing for a range of content filtering approaches to be implemented across different platforms.

An unexpected positive was the **local-first design**, which ensured that the system could operate with minimal reliance on server-side resources. This design not only improved the user experience by reducing latency but also optimized resource usage. Furthermore, the **admin dashboard** provided a solid foundation for detecting abnormal behaviors, with the potential for future expansion. Although the current anomaly detection system was limited to flagging irregular voting patterns, it holds promise for detecting new types of bot activity as the system evolves.

However, there were several **negative aspects** that need to be addressed. One significant challenge was the failure to implement **Google Authentication**. This integration would have provided a secure and reliable way to verify user identity and ensure that the email addresses associated with BB_users were valid. Without this functionality, we were unable to fully guarantee the authenticity of the user data. Additionally, the lack of a proper authentication system introduced potential security risks and limited the robustness of the user management system.

Another challenge was the inability to implement **AI-based bot detection**. While we initially planned to use AI to detect social media bots, the lack of available, high-quality datasets for training AI models posed a significant barrier. This, combined with the increasing sophistication of bots, highlighted the need for further research and more accessible data sources to make AI detection viable in the future. This setback reinforced the value of our community-driven approach, where human input fills the gap left by automated systems.

The project also faced **scalability** issues. Since the system depends heavily on active user participation to evaluate and validate social media profiles, there is a risk that the effectiveness of the system could be compromised if community involvement is low. As the system grows, finding ways to incentivize and encourage participation will be crucial to maintaining its effectiveness.

Despite these challenges, the project succeeded in creating a **flexible, scalable, and user-friendly** platform for identifying and filtering social media bots. The system's strengths in its design,

adaptability, and potential for future improvement lay the groundwork for more sophisticated tools in combating AI manipulation on social media. With further technical refinements, particularly in user authentication and AI integration, BotBlocker has the potential to play a key role in promoting transparency and authenticity online.

Bibliography

[1] Evan Lerner. Social media bots may appear human, but their similar personalities give them away, November 2021. Accessed: 2025-06-16.

[2] University of Virginia School of Education and Human Development. Q&a: Is that real? bots make it hard to recognize truth, October 2024. Accessed: 2025-06-16.

[3] Lynnette Hui Xian Ng and Kathleen M Carley. A global comparison of social media bot and human characteristics. *Scientific Reports*, 15(1):10973, 2025.

[4] The Guardian. Bots on x worse than ever according to analysis of 1m tweets during first republican primary debate, September 2023. Accessed: 2025-06-16.

[5] Social botnets - eithos - european identity theft observatory system, 11 2023. [Online; accessed 2025-06-16].

[6] Fake twitter blue hits eli lilly falsely proclaiming free insulin, 11 2022. [Online; accessed 2025-06-16].

[7] Benjamin E. Shapiro, Sameer Gheria, and Jacob Eisenstein. Improving bot detection with crowd-sourced judgments. In *Proceedings of the 2023 Conference on Human Factors in Computing Systems (CHI)*, Hamburg, Germany, 2023. ACM.

Appendix A

API Documentation

Part 1 - API Documentation

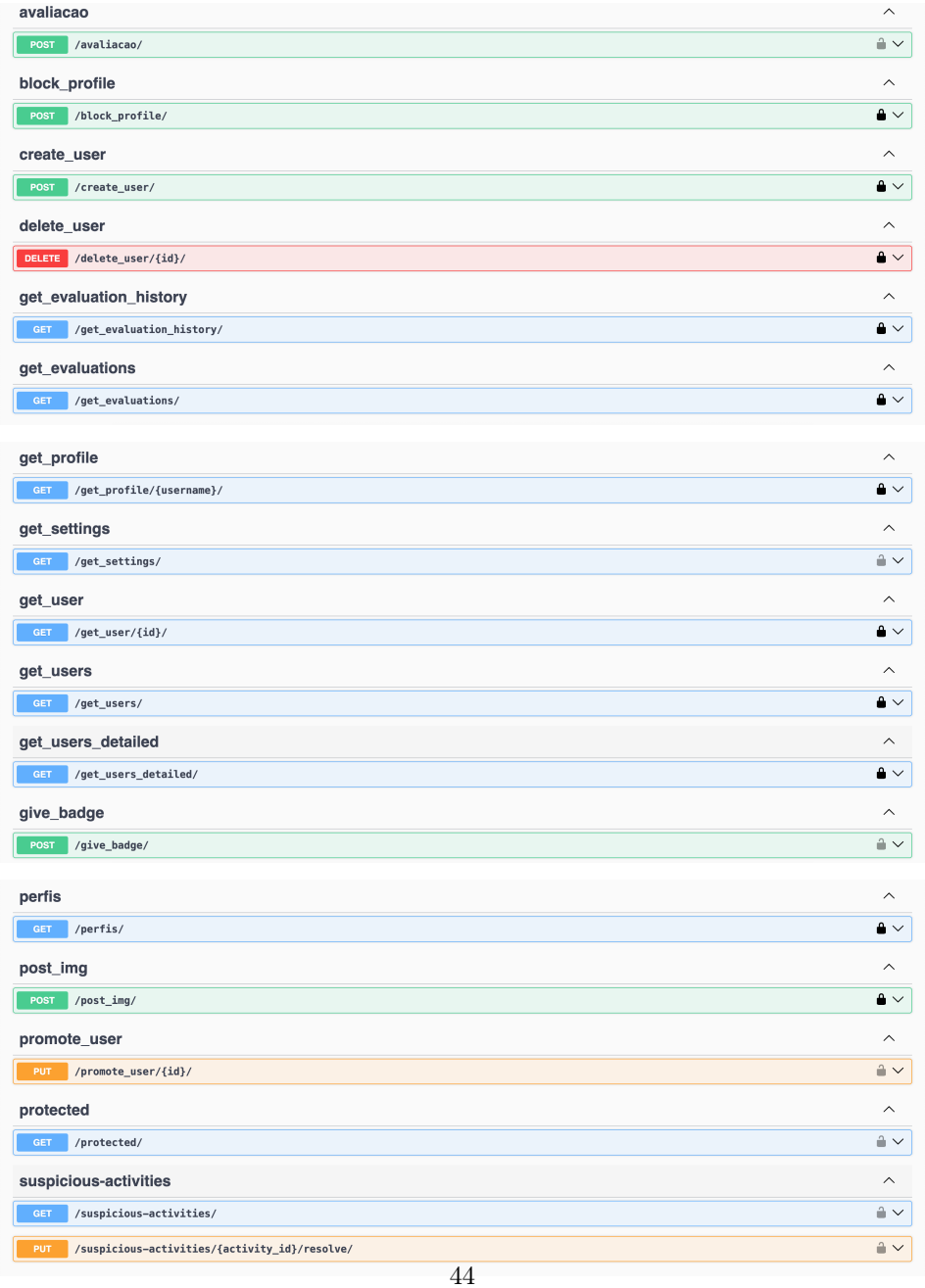


Figure A.1: Part 1 - API Documentation

Part 2 - API Documentation

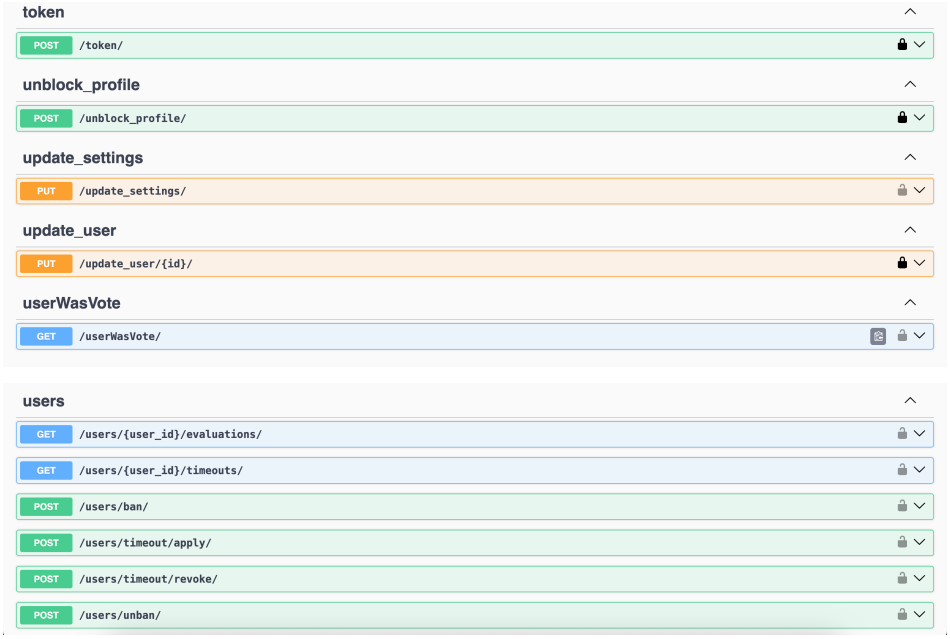
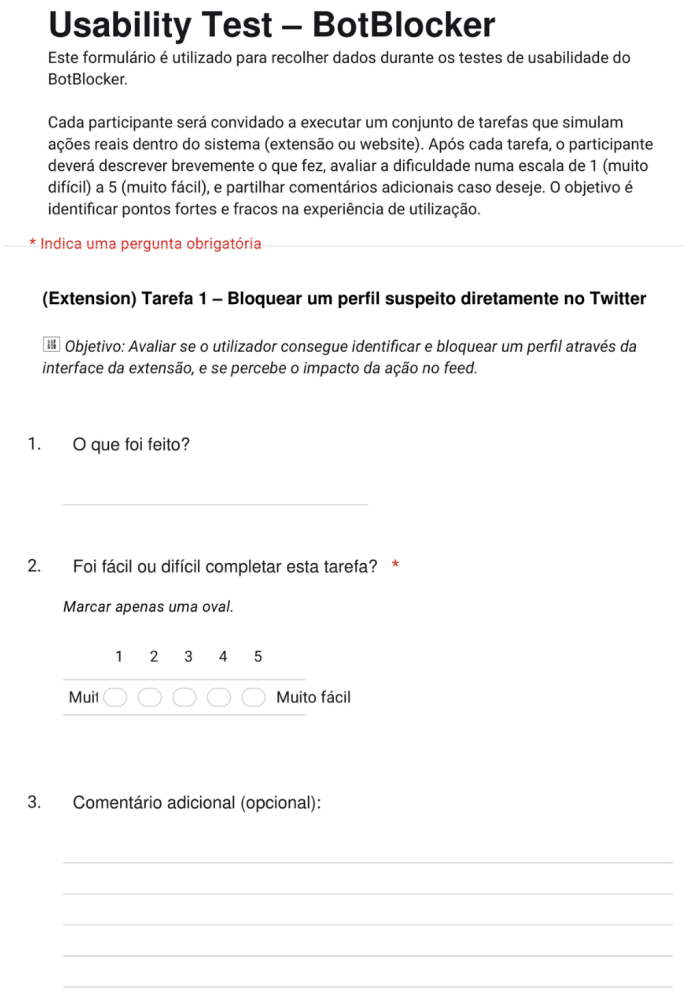


Figure A.2: Part 2 - API Documentation

Appendix B

Usability Test

B.1 Usability Test Questionnaire



Usability Test – BotBlocker

Este formulário é utilizado para recolher dados durante os testes de usabilidade do BotBlocker.

Cada participante será convidado a executar um conjunto de tarefas que simulam ações reais dentro do sistema (extensão ou website). Após cada tarefa, o participante deverá descrever brevemente o que fez, avaliar a dificuldade numa escala de 1 (muito difícil) a 5 (muito fácil), e partilhar comentários adicionais caso deseje. O objetivo é identificar pontos fortes e fracos na experiência de utilização.

* Indica uma pergunta obrigatória

(Extension) Tarefa 1 – Bloquear um perfil suspeito diretamente no Twitter

🔢 *Objetivo: Avaliar se o utilizador consegue identificar e bloquear um perfil através da interface da extensão, e se percebe o impacto da ação no feed.*

1. O que foi feito?

2. Foi fácil ou difícil completar esta tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito fácil

3. Comentário adicional (opcional):

Figure B.1: Usability Test - First set of questions

(Extension) Tarefa 2 – Avaliar um perfil como "Bot"

Objetivo: Verificar a clareza do processo de avaliação de um perfil e a facilidade com que o utilizador encontra e utiliza essa funcionalidade.

4. O que foi feito?

5. Foi fácil ou difícil completar esta tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito fácil

6. Comentário adicional (opcional):

(Extension) Tarefa 3 – Diminuir a percentagem de filtragem para esconder bots suspeitos

Objetivo: Testar a usabilidade das definições de filtragem automática e perceber se os utilizadores compreendem o impacto da alteração de percentagem.

7. O que foi feito?

Figure B.2: Usability Test - Second set of questions

8. Foi fácil ou difícil completar esta tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito fácil

9. Comentário adicional (opcional):

(Website) Tarefa 4 – Consultar o histórico de avaliações deixadas num perfil

Objetivo: Avaliar a navegabilidade do site e a facilidade de acesso à informação detalhada sobre a credibilidade de um perfil específico.

10. O que foi feito?

11. Foi fácil ou difícil completar esta tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito fácil

Figure B.3: Usability Test - Third set of questions

12. Comentário adicional (opcional):

(Website) Tarefa 5 – Aplicar um timeout a um perfil enquanto Admin

Objetivo: Testar a interface de administração, nomeadamente a funcionalidade de penalizar temporariamente perfis abusivos, e perceber se a ação é clara e segura.

13. O que foi feito?

14. Foi fácil ou difícil completar esta tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito fácil

15. Comentário adicional (opcional):

Figure B.4: Usability Test - Fourth set of questions

(Website) Tarefa 6 – Aplicar uma badge de Humano a uma conta enquanto Verifier

Objetivo: Avaliar a fluidez da experiência para utilizadores com funções de verificação e a clareza do processo de atribuição de badges de confiança.

16. O que foi feito?

17. Foi fácil ou difícil completar esta tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muit	◯	◯	◯	◯	◯	Muito fácil

18. Comentário adicional (opcional):

Figure B.5: Usability Test - Fifth set of questions

B.2 Usability Test Results

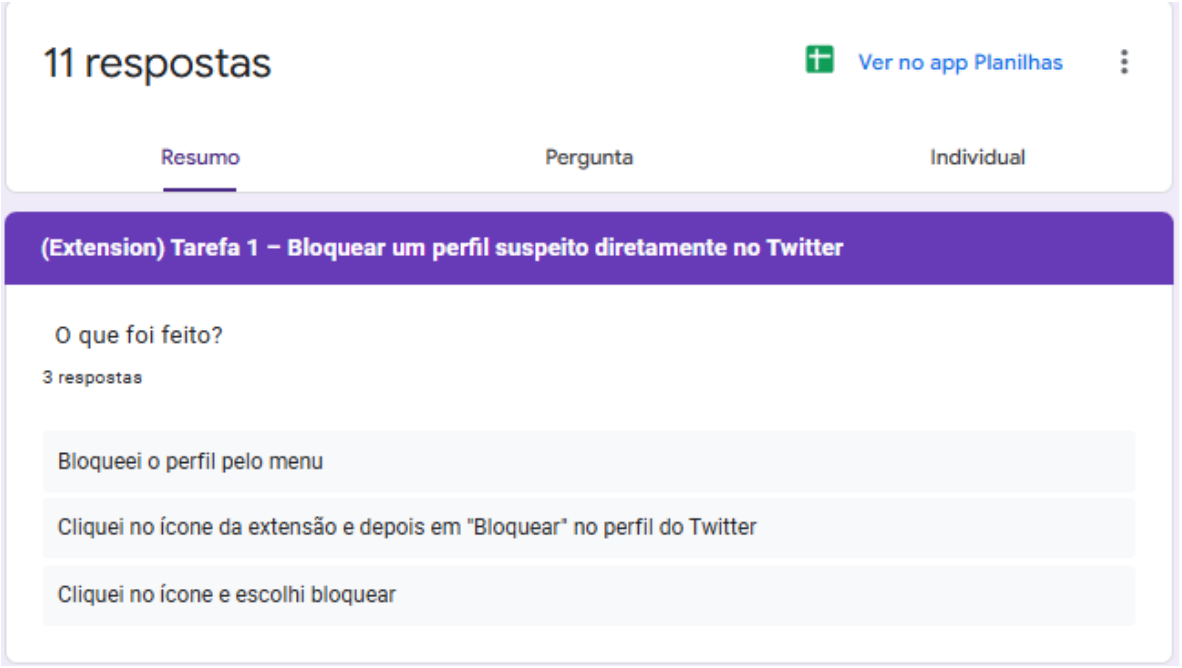


Figure B.6: Usability Test - First set of answers

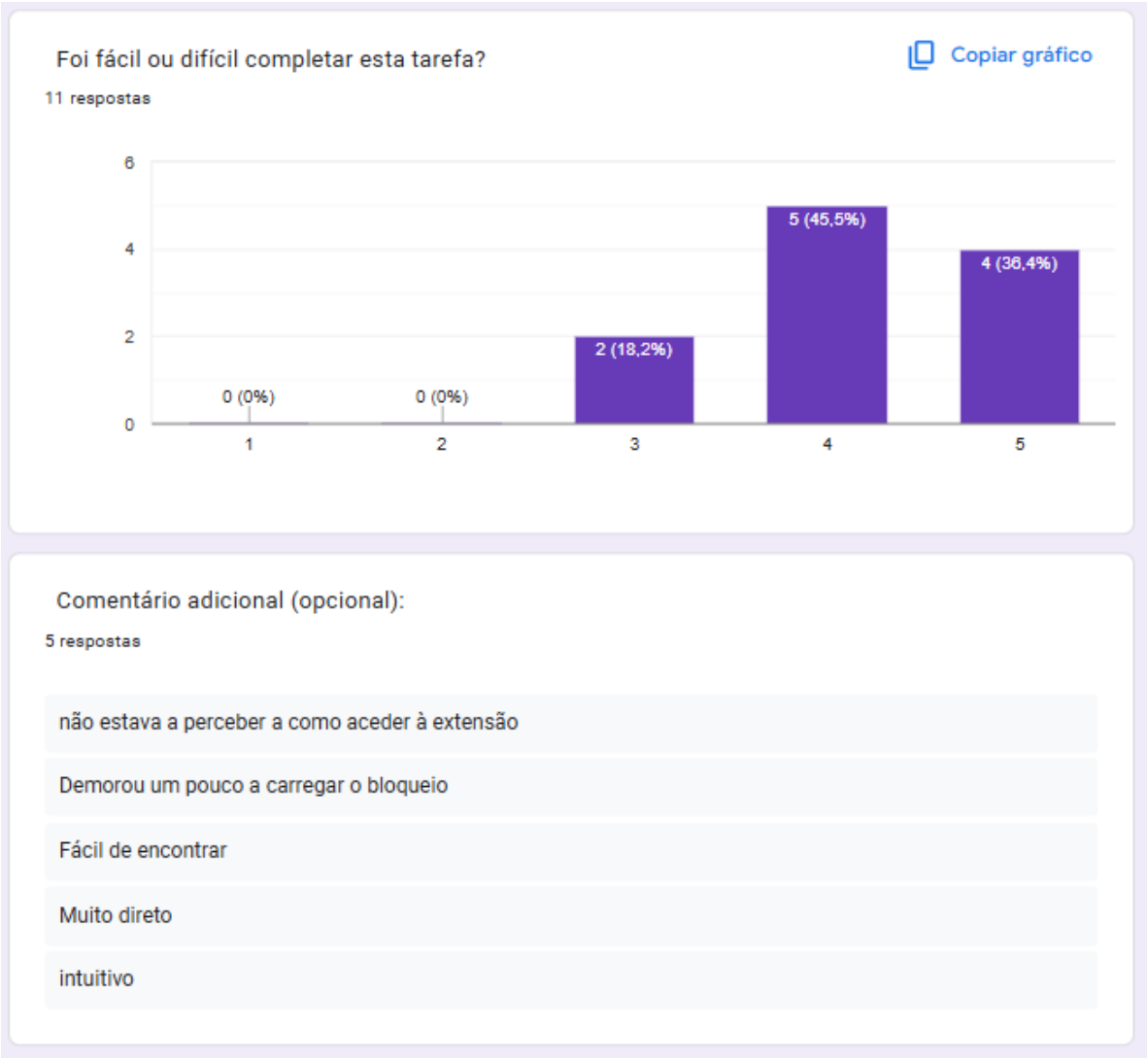


Figure B.7: Usability Test - Second set of answers

(Extension) Tarefa 2 – Avaliar um perfil como "Bot"

O que foi feito?

4 respostas

Carreguei no botão de votar, depois em 'Bot' e depois justifiquei

Avaliei um perfil como "Bot" com o botão que apareceu no pop-up

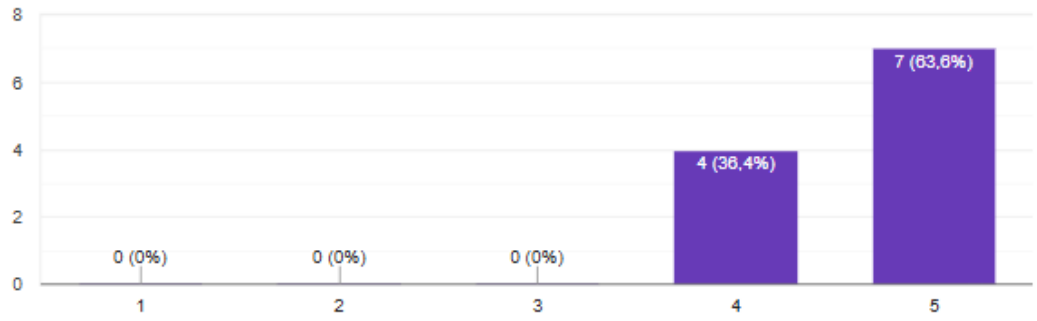
Carreguei no botão 'Bot'

a valiei como bot pela extensão

Foi fácil ou difícil completar esta tarefa?

Copiar gráfico

11 respostas



Comentário adicional (opcional):

2 respostas

Foi muito intuitivo

Sistema bem feito

Figure B.8: Usability Test - Third set of answers

(Extension) Tarefa 3 – Diminuir a percentagem de filtragem para esconder bots suspeitos

O que foi feito?

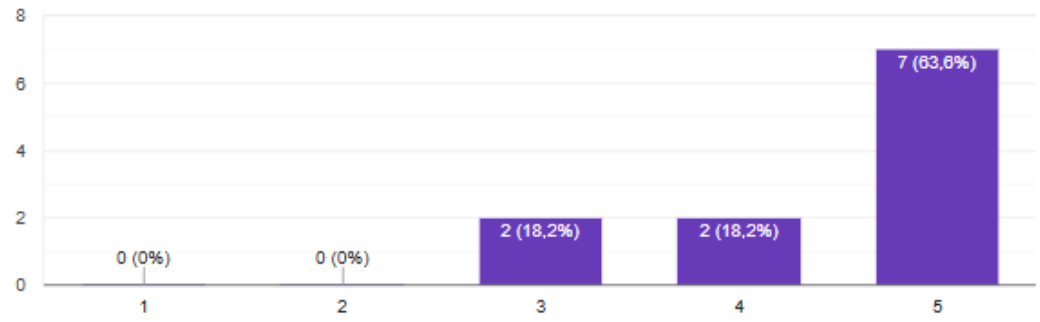
3 respostas

- Baixei o valor nas definições
- Fui às definições e baixei a percentagem de filtragem
- Altereio o filtro nas definições

Foi fácil ou difícil completar esta tarefa?

Copiar gráfico

11 respostas



Comentário adicional (opcional):

4 respostas

- demorei um pouco a perceber que tinha de entrar nas definições para fazer isto
- Não tive dificuldades
- Demorei um pouco a perceber como chegava às definições
- fácil de usar

Figure B.9: Usability Test - Fourth set of answers

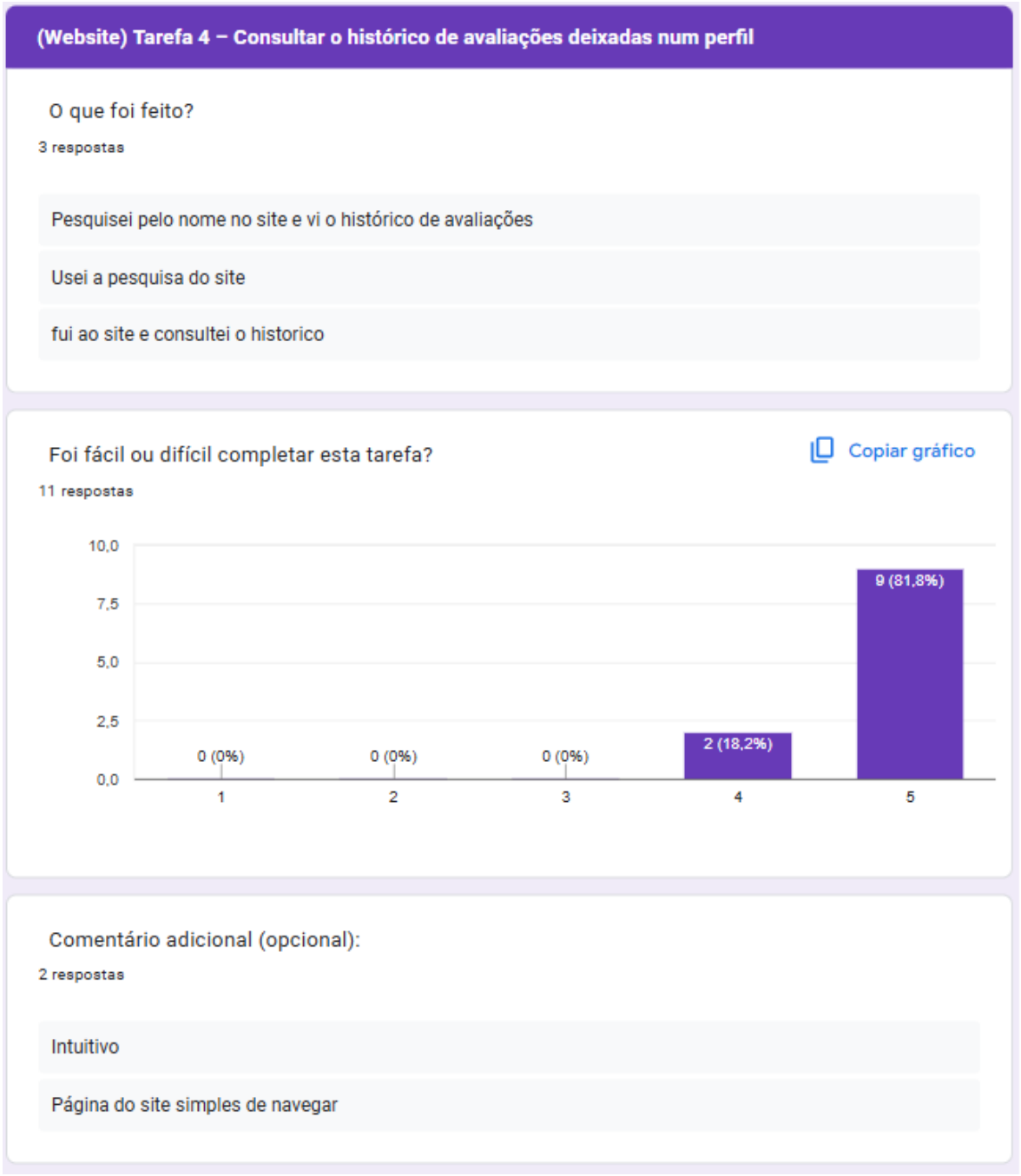


Figure B.10: Usability Test - Fifth set of answers

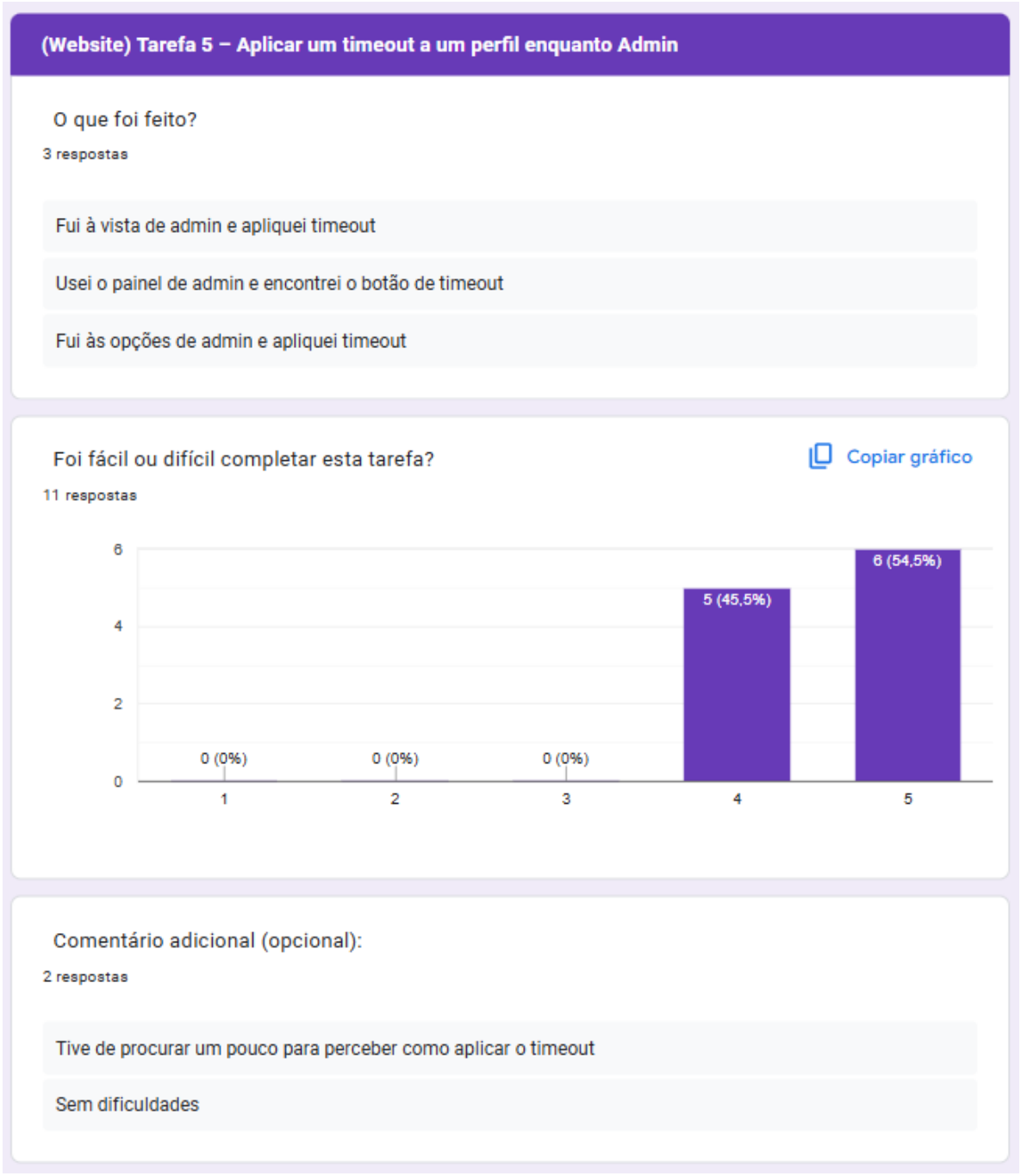


Figure B.11: Usability Test - Sixth set of answers

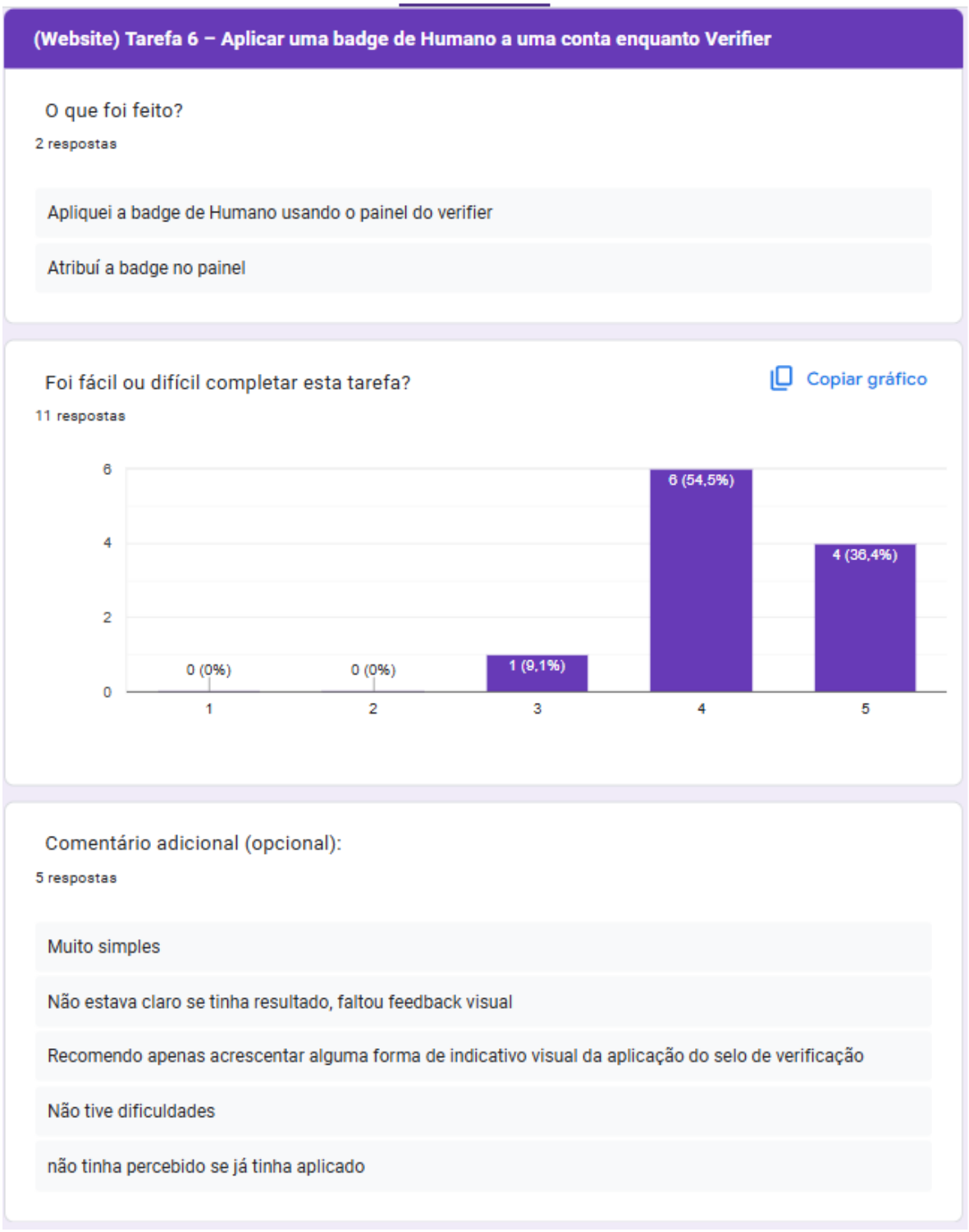
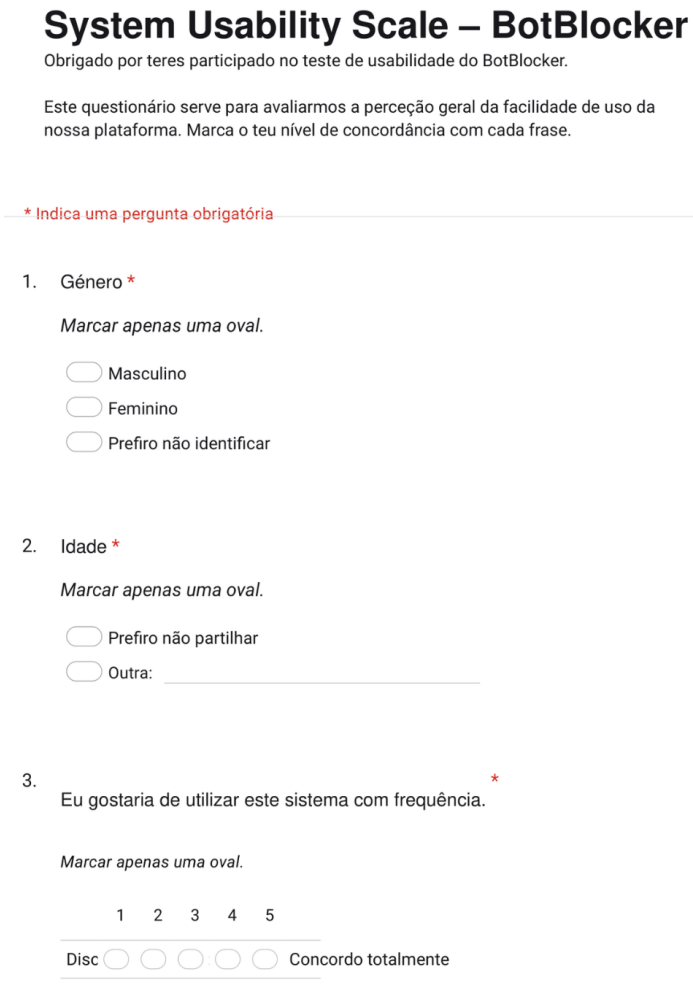


Figure B.12: Usability Test - Seventh set of answers

Appendix C

System Usability Scale (SUS)

C.1 System Usability Scale (SUS) questionnaire



System Usability Scale – BotBlocker

Obrigado por teres participado no teste de usabilidade do BotBlocker.

Este questionário serve para avaliarmos a perceção geral da facilidade de uso da nossa plataforma. Marca o teu nível de concordância com cada frase.

* Indica uma pergunta obrigatória

1. Género *

Marcar apenas uma oval.

- Masculino
- Feminino
- Prefiro não identificar

2. Idade *

Marcar apenas uma oval.

- Prefiro não partilhar
- Outra: _____

3. Eu gostaria de utilizar este sistema com frequência. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc						Concordo totalmente

Figure C.1: SUS Questionnaire - First set of questions

4. Achei este sistema desnecessariamente complexo. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

5. Achei que este sistema foi fácil de utilizar. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

6. Penso que vou precisar de ajuda técnica para conseguir utilizar este sistema. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

7. Achei que as várias funcionalidades do sistema estavam bem integradas. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

Figure C.2: SUS Questionnaire - Second set of questions

8. Achei que havia demasiada inconsistência neste sistema. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

9. Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

10. Achei este sistema muito confuso de utilizar. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

11. Senti-me confiante a utilizar este sistema. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

12. Precisei de aprender muitas coisas antes de conseguir usar este sistema. *

Marcar apenas uma oval.

	1	2	3	4	5	
Disc	◯	◯	◯	◯	◯	Concordo totalmente

Figure C.3: SUS Questionnaire - Third set of questions

C.2 System Usability Scale (SUS) Results

Figure C.4: SUS Questionnaire - First set of answers

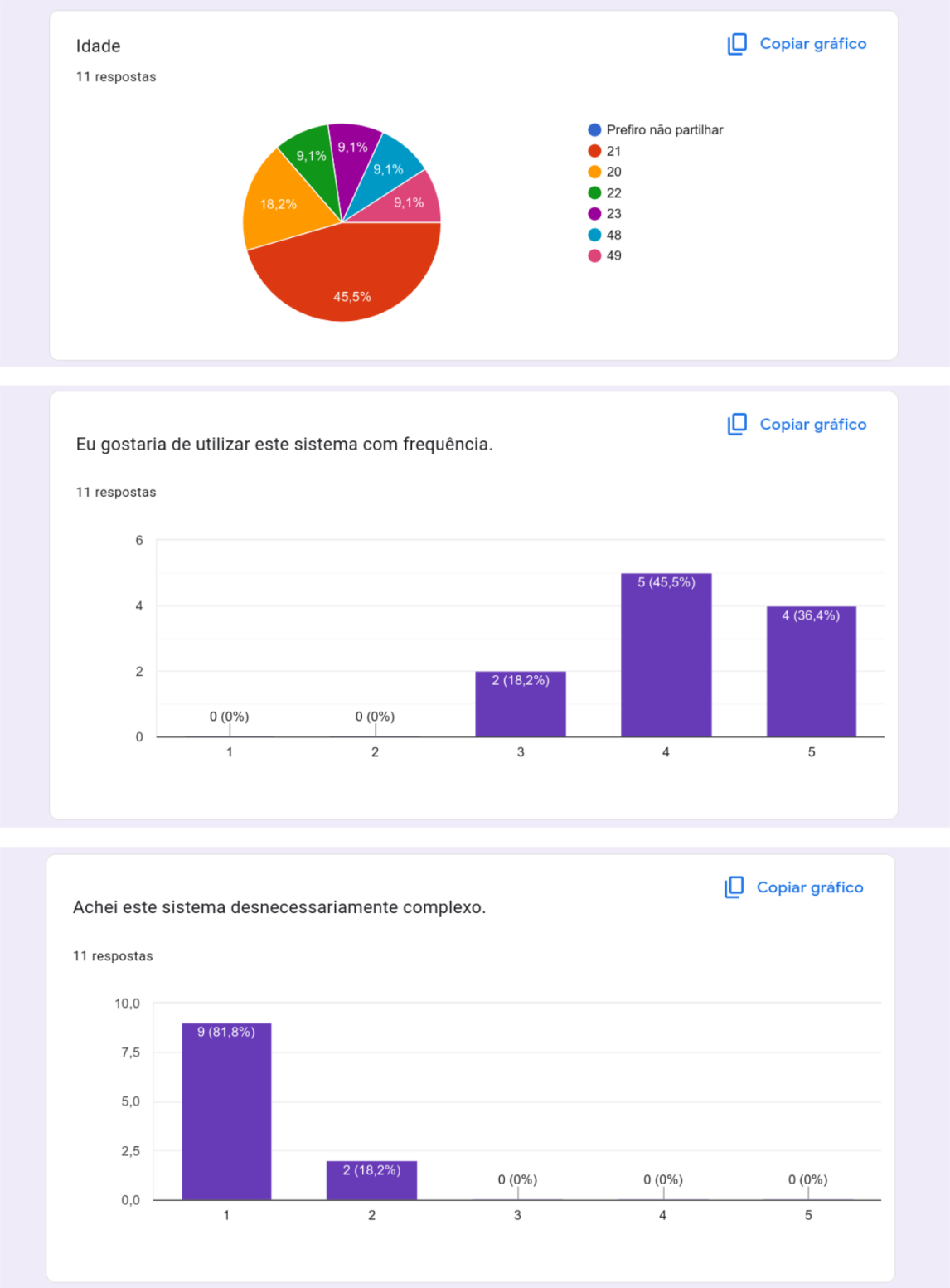


Figure C.5: SUS Questionnaire - Second set of answers

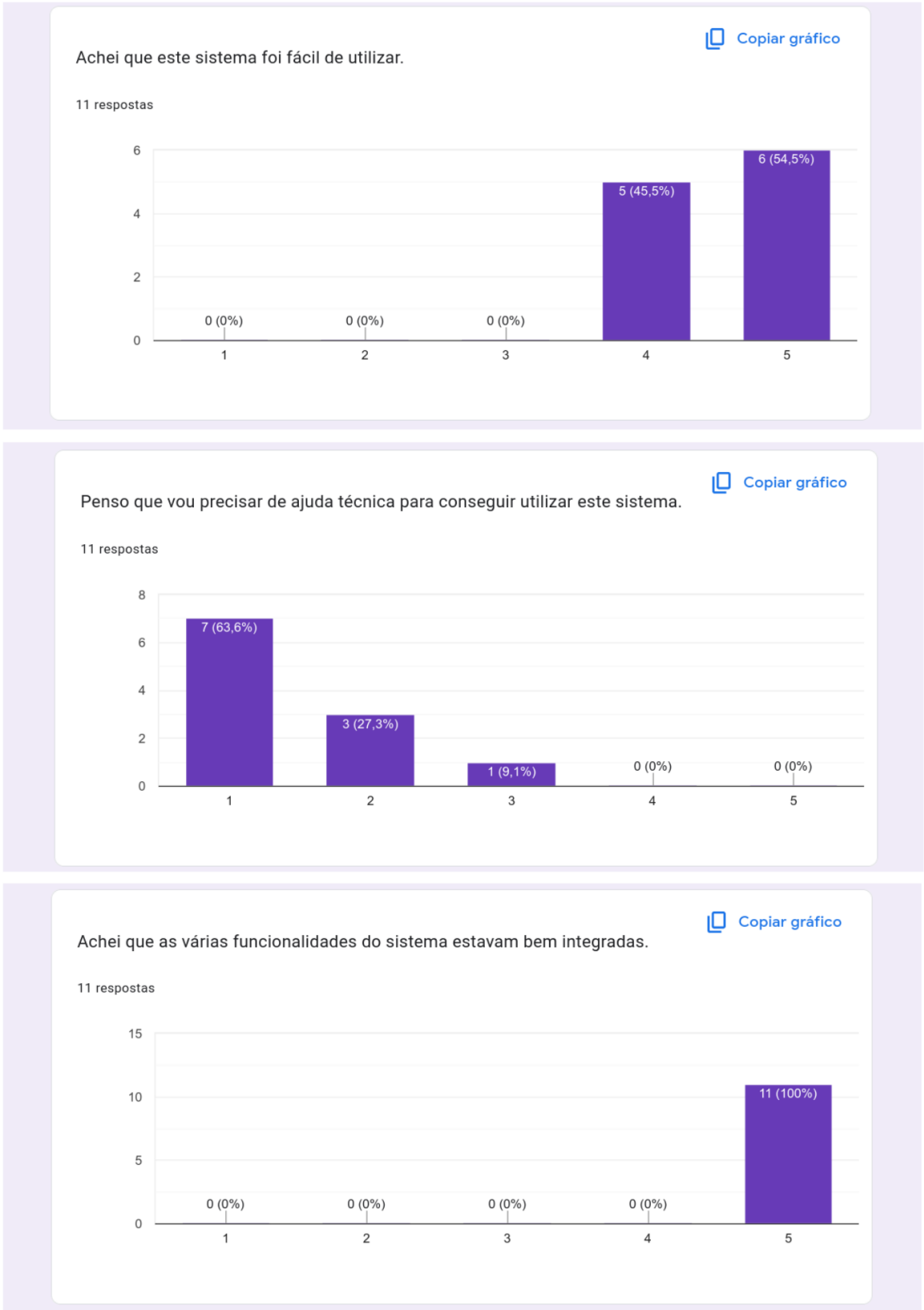


Figure C.6: SUS Questionnaire - Third set of answers

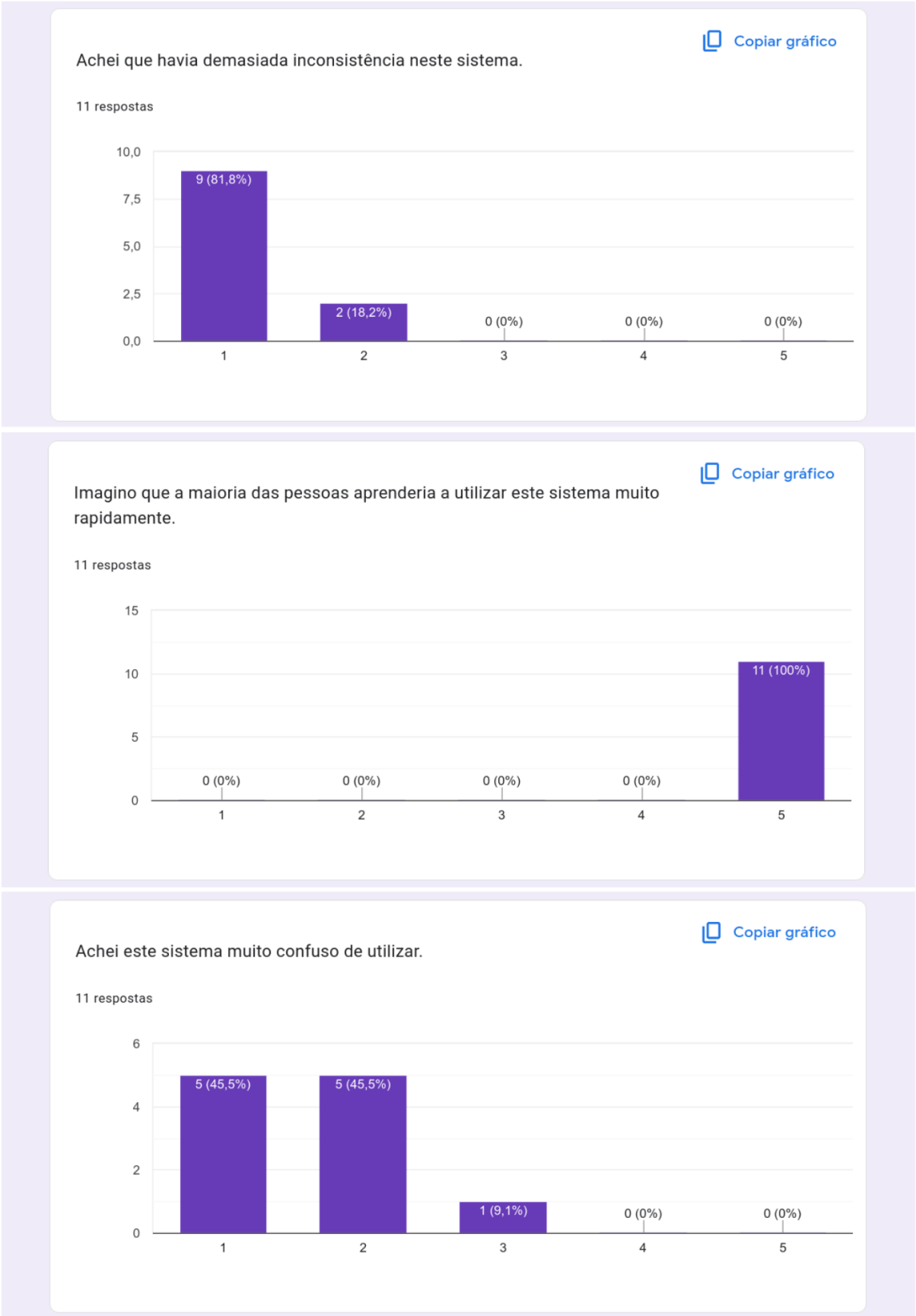


Figure C.7: SUS Questionnaire - Fourth set of answers

Figure C.8: SUS Questionnaire - Fifth set of answers