THE USE OF LARGE LANGUAGE MODELS TO AUTOMATICALLY CATEGORISE USER FEEDBACK FOR GAMES

Callum Hemingway
BSc (Hons). Software Engineering (Game Development)

A dissertation submitted for the degree of
Msc. By Research, Computer Science

Supervised by *Professor of Software Engineering, Tracy Hall*

School of Computing and Communications
Lancaster University

October, 2025

Declaration

I declare that the work presented in this dissertation is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. Estimated word count is: **17672**

Name: **Callum Hemingway**
Date: **October, 2025**

THE USE OF LARGE LANGUAGE MODELS TO AUTOMATICALLY CATEGORISE USER FEEDBACK FOR GAMES

Callum Hemingway, BSc (Hons). Software Engineering (Game Development).
School of Computing and Communications, Lancaster University

A dissertation submitted for the degree of *Msc. By Research* in Computer Science.

October, 2025

Abstract

This study investigates the use of large language models (LLMs) to automate the categorisation of unstructured and informal bug reports for video games, aiming to assist developers in organising their feedback more effectively. The study seeks to answer two research questions: "How useful do developers find categories produced by Large Language Models? and "How reliable are different popular Large Language Models at categorising unstructured and informal bug reports for games? "

A dataset of unstructured and informal bug reports was collected from the video game distribution platform Steam, a random sample of posts from this data set was then used to generate 10 primary categories and 10-sub categories based on a selected primary category.

To answer the first research question video game developers were approached through the survey participant tool Prolific and instructed to rate the generated categories based on their perceived usefulness, with the option to include additional qualitative feedback for each category. From this, developers appeared to rate higher level (more vague) categories as being more useful, indicating that to developers usefulness of bug report categories is tied to the frequency in which bug reports will be assigned to the category. However, multiple developers expressed concern in the form of optional qualitative feedback that the vagueness of the categories would reduce the practicality of using them in real world scenarios.

To answer the second research question, three popular LLMs were used to categorise the same sample data set. Cohen's Kappa, a measure of Inter-Rater Reliability, was then used to compare the reliability of each model's categorisations amongst each other and a human reviewer. The findings from this suggest that even older models can perform this task with high reliability, and newer and potentially more expensive and demanding models are not required for this task.

Acknowledgements

I would like to thank my supervisors Tracy Hall and Saad Ezzini for both their professional and personal support during the length of this study. Without their unwavering guidance and understanding I would not have been able to complete this research. I'd also like to thank my examiners Nour Ali and Paul Rayson for providing an interesting dialogue and the advice needed to refine my work. As well as Phil Satchell for providing much needed logistical support.

The gratitude I have for my family's support could not be adequately described in a thousand words and I can only hope that completing this degree serves as a tribute to them and their kindness.

Finally, I would be remiss if I didn't thank my friends. Particularly Richard Ferguson and Alex Dodd. Alex, as both a development partner and friend, has consistently raised the bar for what I want to accomplish and without Richard completing this degree would have been a far more isolating endeavour.

Thank you all.

-Cal

Contents

List of Figures

List of Tables

Glossary

Cohen's Kappa A statistic used to measure Inter-Rater Reliability. vii–ix, 4, 15–20, 26, 27, 32, 33, 40, 52, 56–89

Gameplay Mechanics The rules, elements, and processes that make up a game. I.e. Swinging a sword, Taking Damage, Killing an Enemy. 9, 22

Inter-Rater Reliability The extent to which two or more raters agree.. iv, v, 15, 18–20, 25, 26, 32–36, 40, 52

Chapter 1

Introduction

With the rise of digital distribution platforms and self-publishing, the market for independent (indie) games has undergone an extreme and continuous expansion (See Figure 7.2 in Appendix B.2). Characterised as games produced by single individuals or small development teams, independent (indie) games have quickly become the majority in terms of the development scale of games released on Steam, one of the largest digital distribution platform for PC games (See Figure 7.3 in Appendix B.2) This evolution of game development has allowed for countless creative projects to reach players, offering consumers innovative gameplay mechanics, unique story narratives, and experimental art styles that otherwise would not have reached customers. However, it has also created a highly saturated market with thousands of titles competing for players annually. Because of this saturation, ensuring that games are both performant and meet user expectations of quality is of vitally important to developers.

When a user experiences an issue within a game product, they may report this issue to the developers in the form of a bug report. In cases where there are many bug reports that focus on varying issues, developers may quickly become overwhelmed. Processing these bug reports into groups or categories helps developers identify trends and problem areas, and makes high volumes of bug reports easier to address (Anvik and Murphy, 2011). Processing bug reports is itself a demanding task and developers may not have the time or resources to do so efficiently. Creating a process to automatically complete this bug report processing task reduces this development limitation and allows developers to dedicate more resources into defect correction. Creating an automated approach to bug report processing is the goal of this study.

Steam, as the largest distributor of PC games, not only serves as a store front for games, but also provides tools for community engagement, the most relevant of which for this study being "Discussion Pages" where players can report bugs, suggest improvements, and share experiences. For independent developers that lack the infrastructure to manage their own dedicated support channels, these discussion pages offer a simple method and forum for

collecting user feedback. Whilst the accessibility of these forums ensures that feedback remains abundant for games with a healthy player base, it can quickly become cumbersome. For games that achieve even moderate levels of popularity, the sheer volume of player reported issues and suggestions can become overwhelming for smaller teams with limited human resources. As a result, developers may struggle to prioritise and address the most critical issues in a timely manner, leaving players dissatisfied and potentially hindering the game's overall success.

Triaging bug reports is not exclusive to video game development, with a substantial level of research being conducted in a Software Engineering context. According to Anvik et al, individual developers tasked with triaging bug reports, the triagers, typically have two main goals (Anvik and Murphy, 2011).. The first is to aid developers in improving the product by ensuring the bug report repository has the smallest set of the highest quality and most useful bug reports, allowing developers to focus on the issues that will lead to the most productive improvements. For example, triagers may remove bug reports that focus on the same issue as already existing bug reports, this is known as repository-oriented decision making. The second goal, which is most relevant to this study, is to organise reports to ease their integration into development. This organisation may be done in various ways, such as organising reports based on individual developer strengths and weaknesses, or by organising the reports based on the element of the product that is affected by the bug. This is known as development-oriented decision making.

Attempting to automate this process of bug report/feedback organisation is a long standing topic of research in Software Engineering. Clustering techniques such as X-Means and K-Means clustering have been used to group similar bug reports together. Whilst K-Means is primarily useful for numerical data. Luaphol et al. (Luaphol et al., 2018) completed research utilising various weighting methods to use K-Means to cluster bug reports. This approach requires the prerequisite knowledge of the number of clusters (groups) before clustering can take place X-Means, on the other hand, does not require this information. Limsettho et al. (Limsettho et al., 2014). used X-Means clustering alongside pre-processing techniques to significant success where their unsupervised approach was comparable to supervised approaches that required prior knowledge of the dataset.

In recent years, powerful Large Language Models such as OpenAI's [1] GPT models have demonstrated Natural Language Processing (NLP) capabilities far beyond the abilities of traditional methods (Chen et al., 2025, Hendy et al., 2023, Brown et al., 2020). Generative Pre-Trained Transformer (GPT) models have demonstrated the ability to process, transform, and create context-aware text that is both coherent and usable. More so, this ability is not limited to a single language, with these models being able to perform with many various languages effectively(Hendy et al., 2023). Existing clustering techniques tend to rely on term frequency and the weighting of words to cluster similar reports together (Luaphol et al., 2018, Limsettho et al., 2014,) Large Language Models on the other hand have the ability to

[1] https://openai.com/

understand information based on more than just the frequency of words and to understand the context surrounding information it is given, this means that LLMs have the potential to understand, unstructured, informal bug reports that are lacking in vocabulary and may not be effectively categorised by previous methods.

These abilities propose an exciting opportunity to explore automated bug categorisation without the pre-processing and preparation required by more traditional methods, with what may prove to be stronger results.

This study is motivated by the desire to help game developers make development-oriented triage decisions by automatically organising their unstructured and informal user feedback by the product (game) component affected by the issue reported. This task is typically demanding and diverts effort away from defect correction in the game product to project and bug report management (Anvik and Murphy, 2011). The study aims to achieve this through the use of Large Language Models to generate bug report categories based on the affected game component and then the further use of LLMs to categorise bug reports from various games into these generated categories.

To this end, the study seeks to answer two research questions:

Research Question One: How useful do developers find bug categories produced by Large Language Models for games?

Research Question Two: How reliable are different popular Large Language Models at categorising unstructured and informal bug reports for games?

The first question is motivated by the desire to help developers to organise their bug report feedback effectively regardless of their games genre. This is important as creating a set of bug categories that is overly specific to certain games over others reduces its use cases and the amount of developers it can help. To do this, an accessible and up-to-date LLM was selected to produce categories and sub-categories from a sample dataset of bug reports that was collected from Steam. These categories were then placed into a survey that had participants rate their perceived usefulness. The participants were recruited through the online recruitment tool, "Prolific." The participant pool was made up of people currently working in the video games industry and those likely to have game development experience based on criteria I created through Prolific. A screening survey was used to assure that participants had the required expertise to provide the desired feedback. Standard deviation was calculated to understand how united or divided developers were on their ratings. From this, "Technical Issues & Crashes" was the highest rated primary category, had the lowest standard deviation, and was the most frequently assigned primary category by the LLM models.

The second question is motivated by the desire to provide developers with useful

information regarding the reliability of various LLM models to aid them in their selection of one, that they would otherwise have to research themselves. To answer this question, an experiment was conducted where three publicly accessible LLMs were provided with the same bug report data that was collected from Steam and prompted to categorise the provided data into 10 pre-determined categories pertaining to the nature of the problem reported. A small sub-categorisation test was conducted where the most frequently assigned primary category by the three models was selected for sub-categorisation. The selected primary category was broken down into 10 sub-categories following the same approach used to create primary categorisations. Inter-Rater Reliability was then determined through the use of Cohen's Kappa statistics. Cohen's Kappa takes two pairs of raters (in this case the raters are the models being used to categorise the bug report data) and determines how reliable they are in their categorisations whilst accounting for the possibility of the raters agreeing by chance, this is important to understand whether the models have any level of meaningful disagreement regarding categorisations. The results of this experiment suggested a high level of inter-rater reliability for each of the pairs of models, indicating that older, cheaper models were still capable of completing the task to an acceptable degree alongside the more expensive and newer models, further allowing developers to preserve resources.

My hope is that the findings of this study will help developers in their effort to improve the standards of their video game products by demonstrating a way of efficiently categorising the issues affecting their games. In this dissertation I will explain the methodology used, the experiment process, the results of the experiments followed by a conclusion where I explore possible next steps for research.

Chapter 2

Background

In complex software products, the presence of issues or problems in the software is both common and to be expected. These issues are typically referred to as "Bugs" or "Defects, " (Runeson, Alexandersson, and Nyholm, 2007). To document and track these problems, developers and testers create defect reports, also commonly referred to as bug reports. A bug report is record that describes a software issue in detail, typically written in natural language such as standard English or any other language the writer chooses.

Issue trackers are development software applications that help development teams communicate and coordinate the management of bug and defect reports (Johnson and Dubois, 2003, Lee, D. Kim, and Jung, 2019). Additionally, issue tracking systems are used to manage the relationships between software defects (Sandusky, Gasser, and Ripoche, 2004). An example of an issue tracking system is Bugzilla [2], which is used by large projects such as Mozilla [3] and Eclipse [4] (Jalbert and Weimer, 2008).

Software defects are often not isolated and often have various relationships with other reported bugs/defects. Two common relationships between bug reports are duplication and dependency. When a newly reported bug describes an issue that has already been identified and documented, this is known as a duplication relationship. The second significant relationship between bug reports, dependency, is identified when the testing, development or resolution of a report is blocked by another unresolved issue that may have a separate bug report. In this scenario, the dependent issue cannot be addressed until the blocking issue is resolved (Sandusky, Gasser, and Ripoche, 2004).

Timely defect identification and correction are essential practices in software engineering (Jeong, S. Kim, and Zimmermann, 2009). Whilst issue tracking software helps manage bug reports (Johnson and Dubois, 2003,Lee, D. Kim, and Jung, 2019,) manual triage is still necessary. Triaging involves reviewing individual bug reports to determine their validity and

[2]https://www.bugzilla.org/
[3]https://www.mozilla.org/
[4]https://www.eclipse.org/

categorising them appropriately for the development process. However, this process diverts time and effort away from product development and toward project management (Anvik and Murphy, 2011).

The triager, the individual responsible for triaging bug reports, has two primary objectives. The first is to maintain a streamlined bug report repository, ensuring that only the most relevant and high quality reports are included. This helps developers focus on the issues that will have the greatest impact on improving the product. For example, the triager identifies and removes prior mentioned duplicate reports to prevent redundancy (Sandusky, Gasser, and Ripoche, 2004,Anvik and Murphy, 2011). This aspect of triage is known as repository-oriented decision making.

The second goal of the triagers is to organise reports for seamless integration into the development process. This involves categorising reports based on variety of factors, such as the affected product component/element or the developer assigned to resolve the issue. The individual act of assigning developers to reports can itself be a demanding task that is complex and error prone (Jeong, S. Kim, and Zimmermann, 2009). This aspect of triage is known as development-oriented decision making.

Bug and defect reports often contain implicit information that requires additional processing to interpret (Bettenburg et al., 2008, Herzig, Just, and Zeller, 2013). Whilst manual inspection for this purpose is highly accurate, it is time-consuming and impractical in may cases. Bug categorisation has been identified as a valuable technique to address this issue (Limsettho et al., 2014) by aiding in issue/defect management and helping developers to identify the underlying structure of the project.

Word level classification of bug reports is not novel. Various supervised learning approaches have been used to classify bug reports against other types of feedback to varying success. Naive Bayes and Logistic Regression was used to separate bug reports from other submitted feedback by Antoniol et. al in 2008, with their research indicating that their methods can achieve between 77% and 82% accuracy (Antoniol et al., 2008).

Separating bugs from other kinds of data is useful, but more recently unsupervised learning approaches have also be used to group similar bug reports together as well (Luaphol et al., 2018, Limsettho et al., 2014).

For example, Luaphol et al. (2018) introduced an approach to use unsupervised learning to group bug reports via Constraint-based K-means clustering. First, the bug report sample was pre-processed: punctuation, numbers and stop-words were removed. Following this, the words and their respective weighting (importance) were represented in the form of bag of words. Term Frequency (TF), Term Frequency-Inverse Document (TF-IDF) Frequency, and BM25 were the different methods used to weight the words in the bug report sample (Luaphol et al., 2018). Following this pre-processing and preparation, K-Means clustering was used to group similar bug reports together with two additional constraints to the K-Means method. The number of clusters is based on the number of meta-bugs, with each meta-bug serving as the cluster centroid. After testing the precision, recall and F-Measure of the TF, TF-IDF and

BM25 weighting methods, the average scores were 0.37, 0.83 and 0.51 respectively. These results are regarded as quite low.

X-Means, unlike K-Means, does not require prerequisite information such as the number of clusters in order to function (Limsettho et al., 2014.) Limsettho et al. used an X-Means clustering approach combined with extensive pre-processing such as topic modelling as well as the previously described pre-processing steps used for Luaphol et al. K-Means method to create an automatic approach for clustering bug reports that produced results comparable to that of supervised learning approaches.

The prior methods listed above are forms of Natural Language Processing (NLP) (Luaphol et al., 2018, Limsettho et al., 2014, Antoniol et al., 2008). In recent years, NLP capabilities have undergone extensive progression, primarily attributed to Large Language Models (LLMs) (Chen et al., 2025). Popular LLMs rooted in transformer architecture undergo extensive training based on enormous quantities of web-based text. These models have extensively demonstrated the ability to process information and generate responses to user inputs known as prompts, with the latest Generative Pre-Trained Models (GPT) having garnered significant attention due to their ability to generate coherent context-aware text (Hendy et al., 2023, Brown et al., 2020). Some LLMs such as OpenAI's GPT models have demonstrated a strong ability to process, transform and work in multiple languages to gain understanding of content and generate translations (Hendy et al., 2023).

The Natural Language Processing abilities of Large Language Models provides an opportunity to explore clustering and bug report categorisation without the pre-processing and training required for other methods.

Large Language Models have previously been used in Software Engineering for many different tasks. For example, Fruntke and Krinke (Fruntke and Krinke, 2025) investigated the use of LLMs to automate the repair of defects caused by dependency changes in Java projects with promising results. This was done using two approaches, an agentic approach where LLMs were used alongside automated tools and a recursive zero-shot approach where an LLM with no prior training on the specific project was tasked with fixing the issue, with its prompts being improved iteratively. This resulted in repair success rates of up to 23% for the agentic approach and up to 19% for the recursive zero-shot approach.

Hossain et al. (Hossain et al., 2024.)developed a system using LLMs to locate and repair bugs in code. Token-Granulated Bug Localization and Repair -or TOGGLE- integrates three models to locate the bug at a token level, another to address tokenizer inconsistencies, and a final model that is deployed to fix the bug. This approach saw extremely promising performance on the CodeXGLUE benchmark (a collection of datasets and benchmarks used to both test and train Machine-Learning and LLM abilities) as well as other Automated Program Repair datasets.

Alongside repairing bug defects, Plein et al. (Plein et al., 2024) proposed using LLMs (specifically ChatGPT and CodeGPT) to create test cases for bug repair to promising results.

Chapter 3

Methodology

This chapter covers the methodology of my research, the implementation of this methodology and the specific actions taken will be implemented in the following Experiments chapter.

3.1 Data Collection & Sample Preparation

3.1.1 Data Collection Source and Selection Criteria

To begin the study, a dataset of unstructured and informal user feedback was systematically collected from Valve's [5] video game distribution platform and online retailer: Steam [6]. This platform was chosen for this project because of its prominence in the gaming sector and its vast collection of video game titles and software available on its store front, it is also a hub for discussions about games and offers forums to comment about the game, share experiences, and most importantly for this study: offer user feedback to developers to improve their games. At the time of writing, Steam hosts an unparalleled number of games, with more than one hundred thousand titles available in its extensive library. Furthermore, more than 12,000 games were released on the platform in 2023 alone, highlighting its constantly expanding nature (Statista, 2025). This extensive and dynamic ecosystem provided an ideal source for gathering relevant data on bug reports and user feedback from a variety of games.

To collect a sufficient sample of user feedback, a carefully curated subset of no more than 17 games was selected from Steam's vast library of available titles. These games were selected based on specific criteria designed to ensure a high level of quality, relevance, and diversity in the dataset. The primary criteria for each game is that it must have a dedicated discussion category exclusively for bug reports and issues. These dedicated pages are required for the purpose of reducing the likelihood of unrelated and none useful data contaminating the dataset, which ensured that the collected information/data was primarily focused on technical

[5] https://www.valvesoftware.com/en/
[6] https://store.steampowered.com/

issues, issues and user-reported issues. This approach has the advantage of providing highly relevant data whilst minimising noise and streamlining subsequent data analysis once all data has been collected.

Additionally, each selected game must have had at least 50 relevant discussion posts/entries in the games dedicated bug report discussion area. This minimum threshold ensured that the dataset is sufficiently large enough to allow for meaningful analysis and also cover a wide range of issues. **NOTE- This has now been noted as a Threat to Validity.** Using these criteria ensured that the dataset is both manageable and rich in relevant content, providing a solid foundation to work from.

For the purpose of enhancing the variety and comprehensiveness of the dataset produced, the selected games spanned different genres, in this case five genres were selected for collection: Horror, Survival, Shooter, Puzzle, and Sandbox. These genres were carefully chosen to capture a diverse array ofGameplay Mechanics, user interaction experiences, and potential technical challenges. By incorporating games from these vastly genres the study aimed to capture a broad spectrum of technical issues and user feedback, ensuring that the dataset is representative of a wider gaming landscape.

An example and breakdown of the layout and content of a Steam game's Bug Report section of their discussion page can be found in figure 3.1. The "bug section" that has been highlighted shows the dedicated discussion page section for reporting bugs and issues; the individual user posts have also been highlighted.

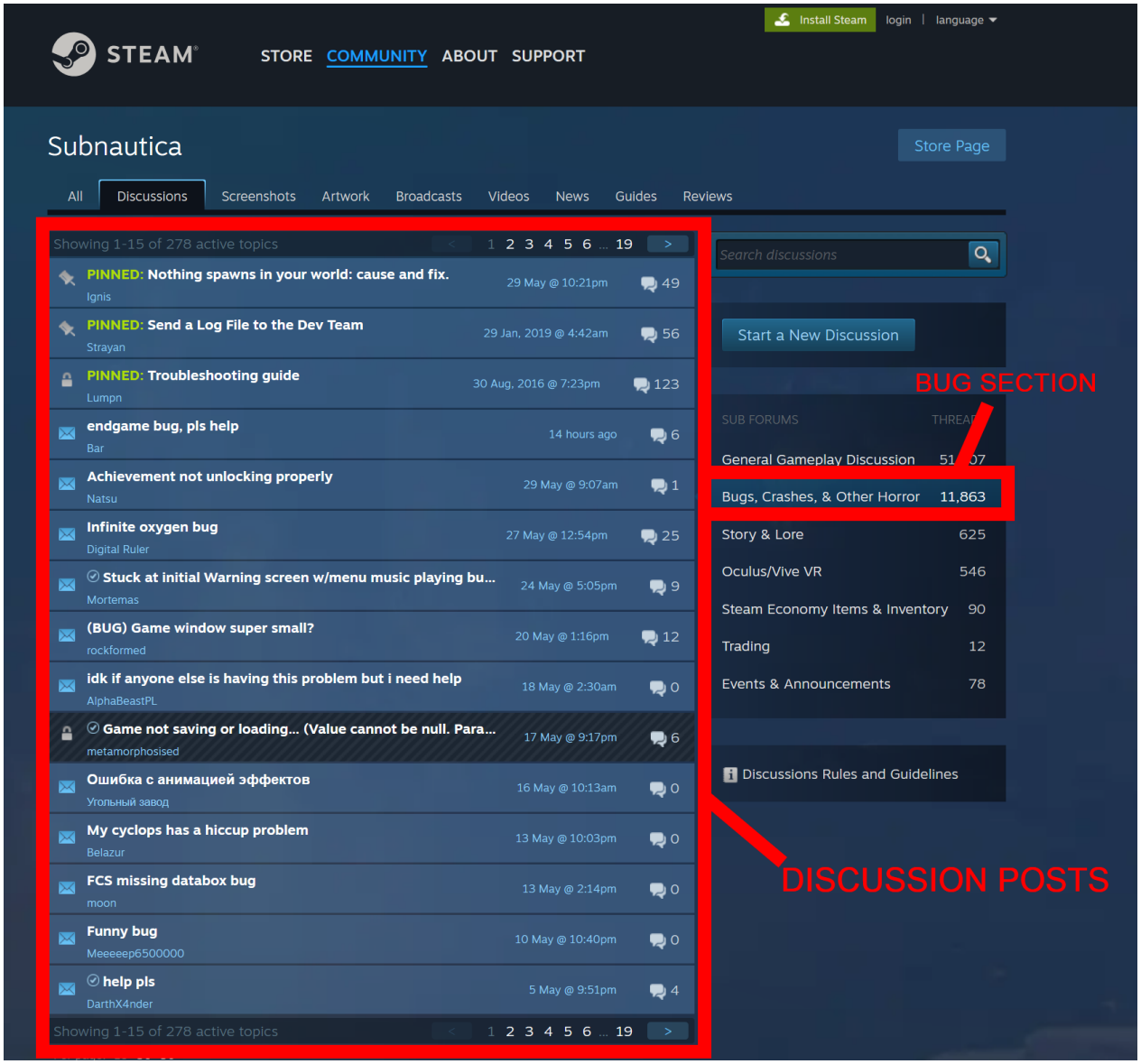


Figure 3.1: Steam: Bug Report section of Subnautica's Discussion Page

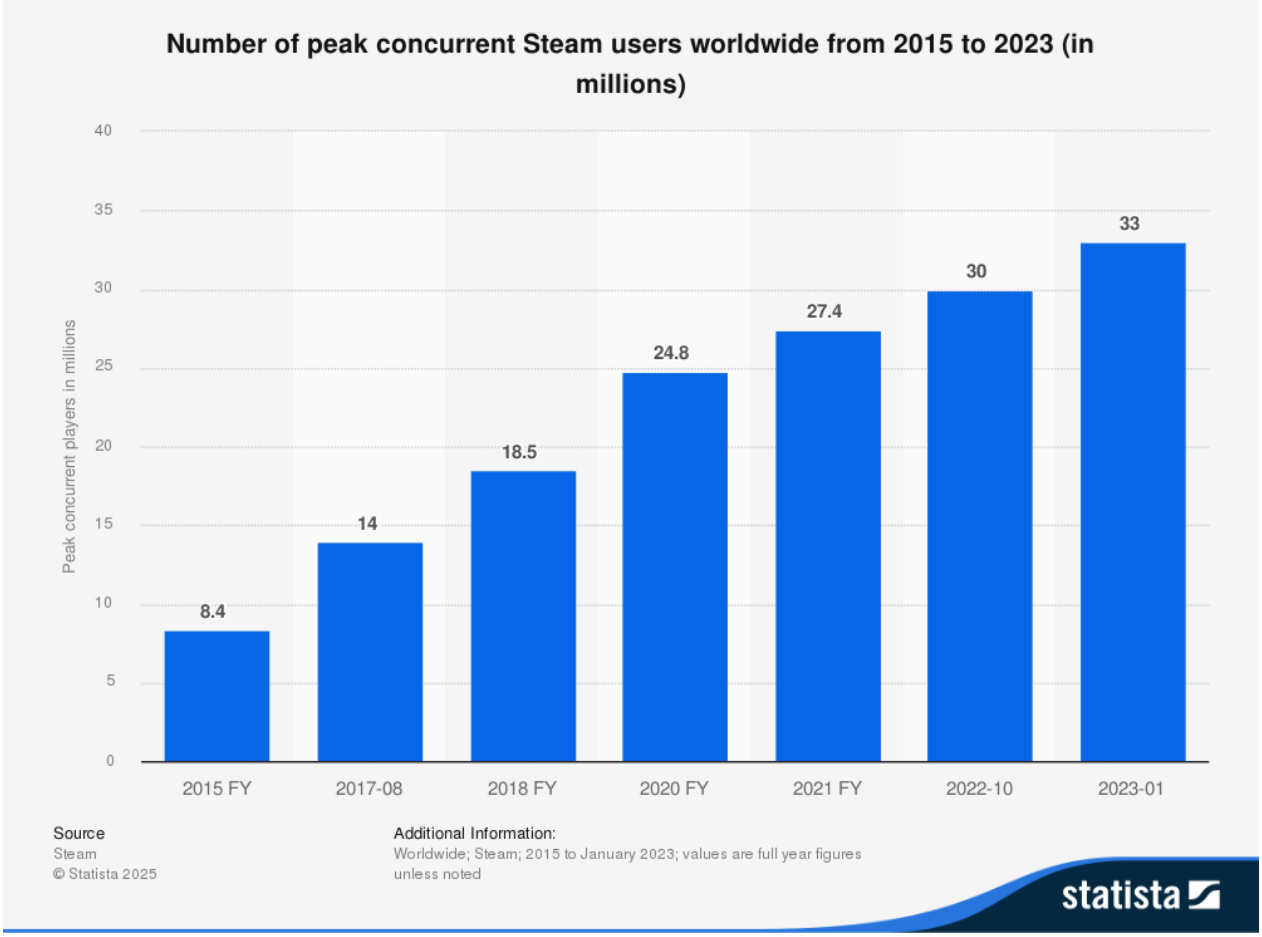


Figure 3.2: Statista: Number of peak concurrent Steam users worldwide from 2015 to 2023

3.1.2 Data Extraction/Collection

Once the games were selected, the next step was to collect the bug reports/user feedback posts from their respective discussion pages on Steam. This presented a challenge as Steam's API does not extend to its discussion pages. In this case, a custom HTML scraper was developed and deployed to efficiently extract the necessary data. The use of a HTML scraper significantly reduced the time and effort required for data collection whilst ensuring consistency in the extracted information. It is important to note that prior to this it was confirmed by Steam's "Robots.txt" page [7] that HTML scraping is permitted on Steam thus ensuring that this approach is both ethical and compliant with the platform's policies/terms of use.

[7] https://store.steampowered.com/robots.txt

The custom HTML scraper had to efficiently navigate the discussion pages of the selected games and extract relevant information regarding each bug report entry. To ensure that the dataset was comprehensive, structured, and well-suited for a wide range of analyses the data collected included the following:

The title of the post. Providing a concise summary of the issue that will often reflect the user's initial understanding of the problem they are facing.

The content of the post. A detailed description of the issue as it is presented and understood by the user plagued by it. This is perhaps the most important piece of data to collect as it should provide useful insights into the nature, severity, and context of the problem occurring.

The date and time of posting. If so desired, this will allow for temporal trends to be analysed, for example: identifying periods of increased bug report activity. Though this is not the kind of analysis this study seeks to conduct.

The date and time of post collection. This will allow for an accurate assessment of the timeliness of the data. The URL/Web address of the post. This facilitated easy access to the original source for the purpose of verification and potential further analysis.

The Steam username of the original poster. In the event that this approach was used in real world scenarios with real developers, it may be beneficial to identify the most frequent bug reporters and to provide insight into user engagement.

3.1.3 Addressing multilingual challenges

Steam's discussion pages are not region-specific, meaning that they attract and allow users from all over the world. As a result, bug reports posted to these discussion pages can be very varied in languages, reflecting the global nature of the platform's user base that initially made it very desirable but now presents a small challenge. Whilst many Large Language Models are capable of understanding and processing multiple languages (Hendy et al., 2023). it is still important that any human reviewer is able to fully comprehend the dataset during any form of analysis, manual annotation or validation. Translation of the collected posts was required. As the GPT models possess the ability to translate posts themselves, these translations were not provided to the models. The purpose of these translations was to allow for a human reviewer to easily understand the content of a post regardless of the language it was written in, not for the purpose of LLM categorisation.

To begin, it was helpful for both analysis and translation purposes that the language that

the post is written in is detected, identified, and labelled. I used Helsinki NLP [8] combined with Python's Lang Detect library to automatically detect foreign languages and create translations where needed. Encoding of this information was done in UTF-8 to allow for special characters.

The translated text was then added to the dataset, with separate fields created for the translated title and the translated content of the post. This approach offered the advantage of preserving both the translated and original versions of the posts, allowing for flexibility when it comes to analysis.

3.2 Categorisation

3.2.1 Selecting LLM models

Once a well-structured and comprehensive dataset had been produced, the next step was selecting the appropriate Large Language Models (LLMs) to analyse and categorise the data. For the study's approach to appeal to wide range of developers, the publicly available GPT models developed by OpenAI were chosen. These models offered several key advantages that make them ideal for the task at hand.

The main reason for selecting OpenAI's GPT models is that they are immensely popular and have experienced widespread adoption. As of the time of writing, GPT models are amongst the most widely used LLMs available across various industries (Hendy et al., 2023). . Because of this, the approach will potentially be more appealing to a larger audience of game developers, regardless of their familiarity with machine learning technologies and techniques.

Moreover, OpenAI's GPT models eliminate the need for significant dedicated computational resources by the developer. Unlike local LLMs, which require high-end and powerful hardware to run, OpenAI's GPT models operate via the cloud, with users being able to use the models via API calls. By removing this barrier, the use of these models makes this study's approach accessible to both small independent (indie) developers with limited resources as well as large studios seeking more scalable solutions.

However, OpenAI's GPT models all charge a fee based on the amount of tokens that the user inputs and the model outputs. This "Pay-Per-Token" model introduces the need for developers to carefully consider and evaluate the trade-off between cost and usability when choosing which model they wish to use to categorise their dataset. To achieve this, three models were selected to be systematically compared: GPT 3.5 Turbo, GPT 4 and GPT 4 Turbo (note at the time of the study GPT 4 Turbo was the newest model available). By applying each model to the same dataset and analysing how they agree and differ in their categorisations, the study presented a clear indication of their relevant strengths, weaknesses,

[8] https://huggingface.co/Helsinki-NLP

and overall performance. This should provide developers with practical insights into which model strikes the best balance between cost efficiency and utility for them.

3.2.2 Generating Categories

Once the dataset had been compiled and the Large Language Models had been selected, the next step in the categorisation process is to generate categories for later use. To ensure that the categories generated accurately reflect the larger dataset, it was important to start by selecting a random sample. Random sampling prevents bias in the generation of categories and ensures that the categories generated reflect the diversity of the issues in the original dataset. For example: if the dataset contains posts from multiple different games, focusing only on one game or one type of bug could result in categories that are too narrow or irrelevant to other games in the dataset.

The size of the random sample was also important. If the sample was too small it may fail to capture the variety of issues present in the dataset, leading to poor usability in the categories generated. If the sample was too large it may have become difficult to process efficiently and effectively.

Random selection was implemented using the python "random" module to generate pseudo-random numbers, these numbers were then used to select rows from the data file containing the bug reports. A process was then implemented to prevent duplicate entries in the randomly selected sample.

After selecting the random sample, the next step was to provide it to one of OpenAI's GPT models and instruct the model to generate categories based on the sample data. These instructions were made clear and specific to ensure that the categories are usable and neither too high level nor too low level to be used across the entire dataset. To this end, the amount of categories generated was also important to consider. Finding a practical starting point to aim for in the number of categories is vital to strike a balance between generality and specificity. This will be further explained in the next chapter.

A significant technical challenge when working with OpenAI's GPT models is the limitation imposed by the context windows. Each model has a maximum number of tokens it can process at a time, this window is comprised of both the input tokens (the post and categorisation instructions) and the output tokens (the categorisation). The context windows can vary greatly between different models meaning that some models are capable of processing more data than others (**IBM-Context-Windows.**) If the sample dataset is too large to fit within a model's context window, some of the dataset will need to be excluded. This could result in categories that are biased or incomplete because the model wasn't able to consider the full extent/scope of the sample. To address this concern, the model with largest context window was used for generating the categories, even though it is more costly.

3.2.3 Assigning and Comparing Categorisations

Due to the Pay-Per-Token model used by the OpenAI GPT models, it is essential for users to have a clear understanding of how each model categorises the same dataset to make an informed decision about which model is the most cost efficient whilst meeting their needs.

Comparing the three selected models (GPT 3.5 Turbo, GPT 4, GPT 4 Turbo) was not as straightforward as simply evaluating how many categorisation tasks each model performed "correctly." This is because the task of categorising bug reports/user feedback is inherently subjective, with posts containing different overlapping elements. A single post may fit into multiple categories depending on its wording, context, or the ambiguity/definition of the categories provided. For an example using hypothetical categories: a report about a game freezing during loading screens could be categorised under "Performance Issues" or "Loading Errors", or even both depending on how the categories are defined and how the post is interpreted. Because of this, a more nuanced approach was necessary for determining the effectiveness of the models compared to each other.

To compare these models effectively, a measure of agreement between their individual outputs was required. For this, Inter-Rater Reliability (abbreviated to IRR) is appropriate for this purpose, specifically using Cohen's Kappa scores. Cohen's Kappa was well suited for this task as it not only measures the level of agreement between the two raters/models but also accounts for the likelihood of agreement happening by sheer chance (**cohens-kappa**). This is important for a task like categorisation where some categories may be more likely to appear more than others. Without accounting for chance, the agreement between models could be inflated providing the user with inaccurate data to base their decision on. This decision is supported by Kolesnyk et al. who determined that the use of Cohen's Kappa for the comparison of human experts and ML classification is justified and effective (Kolesnyk and Khairova, 2022). Cohen's Kappa is also very easy to understand, a score of 1 indicates a perfect agreement, a score of 0 indicates no agreement beyond sheer chance, and a score below 0 (a negative score) indicates agreement even worse than chance, the full range of Cohen's Kappa interpretations is displayed in Table 3.1.

Cohen's Kappa	Interpretation
Less than 0	Agreement worse than Chance
0	No Agreement
0.10-0.20	Slight Agreement
0.21-0.40	Fair Agreement
0.41-0.60	Moderate Agreement
0.61-0.80	Substantial Agreement
0.81-0.99	Near Perfect Agreement
1	Perfect Agreement

Table 3.1: Cohen's Kappa Score Interpretation

Cohen's Kappa is calculated as follows.

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}$$

p_o = Observed agreement between raters. p_e = Hypothetical probability of chance agreement.

To begin, a representative sample of posts was drawn from the dataset for the models to categorise. To ensure that the results are unbiased and representative of the larger dataset, the sample was randomly selected. This prevented potential biases introduced by cherry-picking or small/narrow data samples, in practice this should ensure that the study reflects the model's real world performance against a variety of bug reports/user feedback.

Reiterating what was previously stated about OpenAI's GPT models: each model can only process a set amount of input and output tokens at a time, with this value varying greatly between models. Because of these differences in context windows, it was necessary that each model be given one bug report to categorise at a time, with the posts being processed sequentially instead of all at once.

Further more, another key consideration was the stateless nature of the OpenAI GPT models. Unlike ChatGPT, which can maintain a memory of the conversation, these models do not retain any prior inputs nor outputs once a task is completed. As a result, the instructions given for categorisation had to be repeated for every bug report.

To ensure cost efficiency is maintained for the user, minimising the number of tokens used in each interaction/use of the models is crucial. A practical and effective way to achieve this without compromising the usability of the results was to assign each category a unique number/key and instruct the model to only respond with the number corresponding to the assigned category for the post. This approach significantly reduced the output tokens generated by the model therefore lowering the cost of the categorisation process.

Once the categories had been assigned by the selected models, Cohen's Kappa scores were used to measure the level of agreement between each of them. Specifically, comparing the outputs of Open AI's GPT 3.5 Turbo, GPT 4, and GPT 4 Turbo against each other in pairs. This resulted in three sets of comparisons: GPT 3.5 Turbo vs GPT 4, GPT 4 vs GPT 4 Turbo, and GPT 4 Turbo vs GPT 3.5 Turbo. Once the Cohen's Kappa scores had been calculated, they were presented in a simple table for the user to review.

	GPT 3.5 Turbo	GPT 4	GPT 4 Turbo
GPT 3.5 Turbo	N/A	0.5	0.5
GPT 4	0.6	N/A	0.6
GPT 4 Turbo	0.7	0.7	N/A

Table 3.2: Example Cohen's Kappa Score Table (GPT models only)

While Cohen's Kappa scores between models provided extremely valuable insight into the consistency of the models' outputs, they did not directly make any statement about the accuracy of those outputs. Essentially, high agreement between models doesn't necessarily mean that the categorisations are accurate or agree with what a human might consider accurate. In this case, consistency is not the same as accuracy. To address this limitation, a human reviewer or reviewers was introduced into the process to establish a baseline of what a human might consider correct categorisation.

The same sample of posts that was provided to the models was then provided to a human reviewer with the same instructions that were provided to the models. These posts were then manually reviewed and categorised by the reviewer. Whilst this approach did not serve as a measure of what 100 percent accurate categorisation would be, it did serve to show the user how similar the model categorisations are to a standard human categorisation approach.

Since the data included posts written in multiple varying languages it was crucial to ensure that the human reviewer could understand the content of non-English posts. The translations generated during the data collection phase were used here. By providing both the original and translated versions of the posts to the reviewer, the accuracy of manual categorisation is maintained regardless of the original language the post was written in and ensures that the reviewer could interpret the content as intended and categorise it appropriately.

Once the human reviewer had categorised the sample dataset, Cohen's Kappa scores were then calculated between each of OpenAI's GPT models and the reviewer's categorisations, this offered some view into how the models align with human judgement, giving more insight into their "accuracy."

	GPT 3.5 Turbo	GPT 4	GPT 4 Turbo	Reviewer
GPT 3.5 Turbo	N/A	0.5	0.5	0.8
GPT 4	0.6	N/A	0.6	0.8
GPT 4 Turbo	0.7	0.7	N/A	0.8
Reviewer	0.8	0.8	0.8	N/A

Table 3.3: Example Cohen's Kappa Score Table (GPT models and Human Reviewer)

3.3 Sub-Categorisation

3.3.1 Generating Sub-Categories for Enhanced Usability

After the main categories had been generated, assigned to the dataset, and compared across models, the next step to increase the usability of the approach of this study was the creation and implementation of sub-categories. Sub-categories offered users lower level/specific insights and provided expanded context into the issue at hand, which is particularly useful when dealing with diverse and complex problems across the user's dataset. Additionally,

from a research perspective, by dividing one or more of the primary/main categories into smaller more specific classifications, the study can explore whether additional levels of detail improve the utility of the study's approach whilst keeping the process manageable.

However, sub-categorisation needed to be approached carefully. With 10 primary categories in use, dividing all categories into 10 sub-categories each would result in a messy and overly complex system, making analysis extremely complicated and defeating the purpose of simplification. Instead, this process focused on the most frequently assigned category within the dataset sample, as this category represented the largest concentration of data and offered the best opportunity for further analysis.

Once the main category for sub-categorisation had been selected, the next step was creating the new sample dataset. Since the posts for the main categorisations were already randomly selected to ensure unbiased representation of the dataset, the sample for sub-categorisations was directly extracted from the posts that were previously assigned the desired category. This straightforward approach eliminated the need to conduct a new round of random sampling and categorisation, whilst preserving consistency within the the dataset.

With the new sample dataset ready, the process of generating sub-categories began. This process mirrored the approach used to create the main categories but also include additional context to guide the model and prevent overlap with existing primary categorisations. The same GPT model used for generating the primary categories was used for sub-categorisation in order to maintain consistency.

The model needed to be provided with additional context in order to generate meaningful sub-categories. The model was informed of the category that the posts belong to, as well as the other existing primary categories so that overlap was avoided to ensure the highest level of usability in the sub-categories. As with the primary categories, the number of sub-categories to be generated was also important to avoid the categories being too low level (too specific) or too high level (too broad) to use. To begin, 10 sub-categories were generated as this maintained consistency with the primary categorisation process.

3.3.2 Assigning and Comparing Sub-Categorisations

Whilst the same principles of calculating Cohen's Kappa and Inter-Rater Reliability applied to sub-categories as they did to the main categories, there was an extra layer of complexity when dealing with the sub-categories. This complexity was entirely due to the differences in how the models assigned posts to the selected primary category that was divided into the sub-categories. Each of the models (GPT 3.5 Turbo, GPT 4, and GPT 4 Turbo,) had variations in their interpretations of what posts belong to the select primary category. These differences meant that there were essentially three distinct samples of posts assigned to the primary category by GPT 3.5 Turbo, GPT 4, and GPT 4 Turbo. Creating complexity in the evaluation process as the IRR evaluations for sub-categories had to take into account not only the sub-categorisation itself but the differences in the primary categorisation as well.

For each of the three models a sample of posts was extracted from the original dataset. These samples were comprised of the posts that each model placed into the selected primary category during the primary categorisation stage. Once again, because the models did not completely agree on what belongs in the primary category, these samples had differences in content.

Once the samples had been extracted, each model categorised the sample into the pre-defined sub-categories. This approach was a near complete reprisal of the approach used to categorise the sample into the primary categories. The instructions for sub-categorisation were consistent across models and ensured clarity and minimise ambiguity in the models categorisation process. Posts were processed individually and sequentially to accommodate for the different context windows of the models and to avoid any potential bias caused by variations in token limitations or the contextual understanding that may have changed due to the token limitations. This was repeated three times for each of the samples produced in the previous step: the GPT 3.5 Turbo sample, the GPT 4 sample, and the GPT 4 Turbo sample.

Following the categorisations of the three samples by the three models, once again Inter-Rater Reliability was used to measure the consistency between models. Cohen's Kappa was again used to quantify the degree of agreement between each pair of models/raters (GPT 3.5 Turbo vs GPT 4, GPT 4 vs GPT 4 Turbo, GPT 4 Turbo vs GPT 3.5 Turbo) whilst still accounting for chance agreement. Importantly, Cohen's Kappa between model pairs were calculated three times for all three datasets.

To provide users of this method with a baseline for comparison, a human reviewer once again categorised the same datasets/samples using the same instructions provided to the models. The human reviewer's sub-categorisations offered a reference point and allowed for evaluation of how closely the models align with reasonable human judgement. This did not provide definitive insight into the accuracy of the models, but did increase the usability of the approach by providing additional context to the user.

Whilst repeating the process three times for all three post samples may seem superfluous, it allowed for an analysis of how sub-categorisation agreement/alignment/consistency may differ depending on the composition of the dataset.

3.4 Further Improvements

3.4.1 Determining Causes for Disagreements

Further improvements were made to the categorisation process. Understanding the root causes of a model's underperformance was central to these efforts.

For this process, a model was first selected to be evaluated, to this end, an evaluation of the prior used OpenAI GPT models was required. Users may choose to evaluate whichever model they are currently working with, but for the purposes of this study the decision was data

driven, relying on the previously calculated performance metrics. Specifically, the selection was based on Inter-Rater Reliability and the Cohen's Kappa scores that measure the level of agreement between the model's outputs and those of the human reviewer.

To begin, the average Cohen's Kappa score for each model was calculated across the primary categorisation sample. Once the average scores were calculated, the model with the lowest average Cohen's Kappa score was identified as the target for improvement. By focusing on the model most in need of refinement, this approach ensured that efforts were directed where they will have the greatest impact.

The next step was a detailed comparison of the primary categorisation outputs generated by the selected model and those produced by the human reviewer. For this comparison, posts where there was a disagreement between the two raters was extracted and compiled into a new dataset, referred to as the "Disagreement Sample." Each post in the Disagreement Sample underwent review by a human reviewer, the post and each rater's categorisation of it should were provided and the reviewer categorised each post according to the perceived reason for disagreement between the two raters. In contrast to the previously used categorisation approaches, in this part of the experiment categories were created dynamically, allowing the reviewer to identify and label new reasons for disagreement as they emerged in the sample.

For example, a GPT model and Human Reviewer may have disagreed due to a different understanding of the post, or due to the post being multi-faceted with the different reviewers prioritising different aspects.

3.5 Collecting Feedback from Game Developers

3.5.1 Survey Production & Screening Preparation

To ascertain feedback from real world developers regarding the usefulness of the categories produced, a survey was produced. The survey asked the developers to rate the generated primary and sub-categories by usefulness, optional qualitative data was collected. Qualitative data regarding each category had to be optional to increase participant retention due to the large number of categories being analysed. This has been noted as a potential threat to validity. Other information regarding the developer regarding the number of bug reports they have worked with, their experience shipping games and their general experience in the field was also collected at this stage to allow for potential analysis of trends that arise in this information.

To ensure that participants had the prerequisite knowledge to offer meaningful feedback, a screening survey was used (Danilova et al., 2021).

3.5.2 Survey Distribution & Participant Pools

Prolific was used to distribute the survey to participants as it not only has a large participant pool but also allows researchers to find participants based on their industry experience. Two groups were created, individuals who work within the Games Industry professionally and individuals who are likely to have some form of Game Development experience. This was to ensure that both formally employed and hobbyist developers are included, as the majority of games released on Steam (the place where the bug reports were collected) are designated as "indie games" (Statista, 2024b). This has been noted as a potential threat to validity and is elaborated on later. The screening survey was then distributed to the two groups, those who pass the screening survey were then invited to the primary survey.

3.5.3 Results Analysis

Once the survey was completed, the usefulness ratings of the primary and sub-categories displayed which categories developers found most and least useful. To determine how united developers are regarding these usefulness ratings, standard deviation was then calculated for the ratings of each category. The frequency in which the categories were assigned to the sample datasets were compared to the newly collected usefulness ratings to analyse the correlation between frequency and developer rated usefulness.

Chapter 4

Experiments

This section follows the process of executing the methods detailed previously.

4.1 Preparing to create the dataset

The experiment began with the collection of unstructured user feedback from the digital distribution platform Steam. As described in the Methodology Chapter, to ensure the variety and comprehensiveness of the dataset, five genres were carefully selected to capture a broad and diverse array of Gameplay Mechanics, player experiences and technical challenges. These genres were: Horror, Survival, Shooter, Puzzle, and Sandbox. From these genres, a total of 17 games were identified and selected for inclusion in the study. The selection process ensured that the games represented key aspects of their respective genres and offered sufficient data for meaningful analysis.

The selected games and genres are as follows:

Horror	Survival	Shooter	Sandbox	Puzzle
Hello Neighbor	Subnautica	Warframe	Factorio	The Talos Principle 2
Hello Neighbor 2	Subnautica Below Zero	Day Z		The Witness
The Forest	Stranded Deep	Payday 2		Obduction
Phasmophobia	Rust	Payday 3		Stray

Table 4.1: Games by Genre

The games were chosen based on two criteria. First, each game needed to have a dedicated "bug reports," "technical issues" or otherwise similar dedicated category within its Steam

discussion page. This ensured that the data collected would be directly relevant to the topic of technical problem reporting, minimising the inclusion of irrelevant or off-topic posts. Second, the dedicated bug report category must have at least 50 relevant posts. This threshold was established to guarantee that each game contributed a meaningful volume of data to the dataset whilst avoiding games with limited activity.

4.2 HTML scraping

After selecting the games, a custom HTML scraper was developed to gather posts from the bug report discussion section of each title. The scraper was implemented in Python and used the Requests library to fetch web pages and BeautifulSoup4 to parse and process the HTML data. The scraper was designed to methodically navigate the discussion section of each game and extract specific information from the HTML structure.

Once the location of each important data point had been manually located within the HTML, the scraping process began by accessing the discussion page directories, which lists all posts within the bug report section. From each page in these directories, the following pieces of information were extracted from each post:

The game name, using the "apphub_Appname" class in the HTML.

The links to each individual post, using the "forum_topic_overlay" class.

The post titles, using the "forum_Topic_name" class.

The reply counts, using the "forum_Topic_Reply_count" class.

The username of the Original Poster (OP), using the "forum_Topic_op" class.

To achieve this, the scraper utilised the specific HTML classes mentioned above to locate and extract the relevant data. Each piece of information was processed with BeautifulSoup4 to strip away unnecessary HTML code and other extraneous elements, ensuring that only the desired content was retained. The processed data was then stored in respective arrays, creating a structured collection of information for later use and storage.

Whilst the discussion page directories provided valuable metadata for each post, they did not include the main text of the posts themselves. To address this, the URLs collected for the individual posts were used to navigate to their respective pages and extract their content. Extracting the main text of the posts proved to be slightly more complex than retrieving the metadata due to the way the content was embedded in the HTML.

The scraper identified the main text by locating "script" tags with the attribute "type='text/JavaScript'." Since the HTML contained multiple script tags, the keyword

"InitializeForumTopic" was used to pinpoint the specific tag containing the desired content. After identifying the relevant script tag, the scraper performed additional processing to remove unnecessary JavaScript code and other extraneous syntax.

Some posts contained special characters that triggered Unicode escape sequences, leading to encoding issues during processing. To resolve this, the text content was encoded in UTF-8 and decoded using the Unicode_Escape codec. This step replaced the escape sequences with their corresponding characters, ensuring that the text was both properly formatted and readable. Once the processing was complete, the extracted text was added to the array of post contents, which was then integrated with the metadata collected at the first stage.

This two step process (scraping the discussion directories for metadata and then navigating to individual posts for their main content) was repeated for each game, each discussion directory page in the bug report section, and each post listed on those pages.

During the data collection process, an unexpected challenge arose: the number of posts retrieved by the scraper was lower than the total number of posts listed in the discussion page directories. Initially, this discrepancy was suspected to be a result of user error in the script, However, further investigation and communication with Steam Support confirmed that inactive posts were systematically removed from the discussion page directories over time (See Appendix C/Figure 7.5). This practice was implemented by Steam to reduce server load and improve performance. Attempts to access these old posts through the Internet Archive/Way Back Machine failed.

According to Steam Support, these inactive posts could not be accessed through the discussion pages unless their exact titles were known and individually searched. Since the titles of inactive posts were unavailable, it was not possible to include them in the dataset. As a result, the study was limited to active posts that were still publicly accessible. While this limitation introduced a potential bias toward more recent discussions, it did not significantly impact the study's goals, as the primary objective was to create an approach/process to allow developers to analyse currently active bug reports.

After completing the HTML scraping process, the dataset produced consisted of over 4,500 posts from the bug report discussion sections of the 17 selected games.

4.3 Generating Primary Categories

OpenAI's GPT models were used to generate categories based on comments provided to it. To ensure there was no bias, the sample provided was selected randomly, with no cherry picking. This random sampling helped to ensure that the categories created by the model(s) were based on the entire range of the games selected, instead of just a few. One technical limitation of working with GPT (and other) large language models is their respective "Context Windows." Context Windows refers to the amount of tokens that the models can remember at any single time, a token being the words, character sets or combinations of words and punctuation that LLMs use to process information (Bergmann, 2023). Because of this token limitation, the

sample provided to the GPT models to generate categories had to be small enough to be used by the model with the lowest context window whilst also providing enough data to generate useful categories. The number of posts used for this dataset was 225, as posts can vary in length and therefore token count, this number (225 Posts) is not a strict representation of the maximum number of posts the models can handle, however it was found to be around the average maximum number of posts that GPT 3.5 Turbo could process.

The creation of categories was facilitated by a simple prompt of the instruction to generate 10 categories based on the data provided, followed by the 225 randomly selected posts.

Final Prompt: "This sample contains bug reports from various games across multiple genres. Generate 10 categories for these bug reports and return them to me, make sure the categories are neither too specific nor too broad to be useful for the majority of games..."

The 225 collected posts were then added to the end of this string.

The final categories produced by this prompt are shown in the results section in Table 5.1.

4.4 Assigning Primary Categories and Evaluating Inter-Rater Reliability Results between Models & Human Reviewer for Primary Categorisations

As previously mentioned, OpenAI's GPT models have limited context windows. To accommodate lower context windows, posts were provided to the models sequentially rather than in bulk. To compare the categorisations produced, Inter-Rater Reliability was measured. Inter-Rater Reliability (IRR) is a measure of the consistency and agreement between multiple raters in their assessment of the same information. In the case of this study, raters refers to the GPT models used to produce categorisations as well as an additional human reviewer, the information assessed being the categorisations assigned to the post sample. A single human reviewer categorising the same sample as the GPT models is not indicative of the models' accuracy, but instead offers a comparison point for each of the models. Once again, a random sample of 225 posts was selected. At this stage, prompt engineering, the process of creating a prompt or instruction to achieve the best results from a LLM, was employed Chen et al., 2025). As instructing a model to place a comment into a category is very simple, this small process of prompt engineering was designed to limit the number of tokens required by the LLM to complete the task. Specifically, the inclusion of an instruction to limit responses to a single character was a cost saving measure to decrease the number of output tokens required by the model.

Example Final Categorisation Prompt:

To begin the prompt the LLM was provided with the instructions for categorisation:

"Place this comment in the most fitting category : 1. Gameplay Bugs & Glitches. 2. Technical Issues & Crashes. 3. Game Progression Blocks. 4. Graphics & Rendering

Problems. 5. Controls & Input Recognition. 6. Multiplayer & Connectivity Issues. 7. User Interface & Experience Problems. 8. Audio Issues. 9. Installation & Update Problems. 10. Achievements & Rewards Issues. Limit response to single character, the number of the category, no text other than that value: "

This was then followed by a user post, for example:

"The game crashes with my internet on and my laptop is an Acer Nitro 5. The game just crashes whenever I launch it with internet on so I can't play mods and with internet off it is fine."

For this study,Inter-Rater Reliability was measured using a statistical measure called Cohen's Kappa. Cohen's Kappa is a value that represents the level of agreement between raters whilst accounting for the possibility of them agreeing by chance. Cohen's Kappa was calculated for each pair of raters (including the human reviewer).

4.5 Generating Sub-Categories

The most frequent primary category assigned to the posts was selected for sub-categorisation tests. The primary category selected for sub-categorisation was "Technical Issues & Crashes." The following prompt was produced to generate 10 Sub-Categories based on this primary category,

Sub-Categorisation Final Prompt: "These comments are from the category Technical Issues and Crash Reports, from these comments, generate 10 sub categories and return them to me, make sure there is no overlap with the following categories: 1. Gameplay Bugs & Glitches. 3. Game Progression Blocks. 4. Graphics & Rendering Problems. 5. Controls & Input Recognition. 6. Multiplayer & Connectivity Issues. 7. User Interface & Experience Problems. 8. Audio Issues. 9. Installation & Update Problems. 10. Achievements & Rewards Issues.."

The collected posts from the selected primary category were then added to the end of this string.

The inclusion of primary categories was necessitated when test sub-categories were producing categories that overlapped with existing primary categories. the result of this prompt are shown in the results section in Table 5.9.

4.6 Assigning Sub-Categories and Evaluating Inter-Rater Reliability Results between Models & Human Reviewer for Sub-Categorisations

As each of the models assigned a different amount of posts to this primary category, multiple sample datasets of posts were be extracted from each of the models primary categorisations.

For each of these datasets, the posts within them were categorised in the same manner as the production of primary categorisations, instead substituted with the generated sub-categories and with the additional context provided to the models regarding the primary category they pertain to.

Example Final Sub-Categorisation Prompt: To begin the prompt the LLM was provided with the instructions for sub-categorisation:

"Place this comment in the most fitting category: 1. Specific Mission/Levels Crashes 2. Device-Specific Issues 3. Performance Issues (FPS Drops/Stutters) 4. Crash on Launch/Startup 5. In-Game Freeze/Crash 6. Asset Loading Errors 7. Memory Management and Leaks 8. Game Integrity and Corruption 9. Hardware Compatability Issues 10. Post-Update Problems Limit response to single character, the number of the category, no text other than that value: "

This was then followed by a user post, for example:

"Frequent crashes since updatetitle says it all."

For each of the datasets, Cohen's Kappa values were calculated for all pairs of raters. The mean of these scores were then calculated.

4.7 Creating Survey & Screening Survey

To gain quantitative and qualitative data regarding developer reception of the categories produced, a survey was created. Recruiting developers with the desired skill set can be difficult. Prolific, an online survey participant pool, made this simpler by giving surveyors access to a wide array of participants based on different criteria such as profession, hobbies, location and more. However, it was still vital to personally ensure that these participants recruited from Prolific had the necessary skills to provide usable data. This was done through the use of screening surveys to test participant knowledge, allowing those who succeed to continue to the main survey and those who don't to be excluded, (Danilova et al., 2021).

Developers were split into two groups based of Prolific user data: Games Industry participants, who actively work in the games industry, and None-Games Industry participants, who whilst not reporting to work in the games industry, are likely to have knowledge regarding the video game development process as well as the importance of certain issues as they pertain to video games. This was done to allow for hobbyist/none-professional developers to be included in the study. Both groups had to pass the screening survey to continue.

The screening survey asked a variety of questions beginning with general skill information:

1. **Which of these game engines do you have experience working with (Select ALL engines that apply).** -This question had 2 fake answers to attempt to catch those who answer questions falsely.

2. **What roles have you held in game development projects?**

3. **Which of these programming languages have you used to develop a game? (The game itself, not things related to the game such as websites or launchers).**

Following this, questions to test respondents knowledge were taken from a paper by Danilova et al., 2021.

Following the screening survey , developers were presented with the primary study survey where they rated how useful they thought each primary and sub-category would be, as well as optionally providing qualitative responses regarding extra thoughts and opinions they had over the categories.

These surveys were both piloted twice, the result of which was fixing formatting issues with the survey where certain questions that should have been multiple choice weren't, as well as removing ambiguity from the phrasing of certain questions to make them clearer.

The screening and primary survey in their complete form can be found in Appendix E.1 (Figure 7.6) and E.2 (Figure 7.7) respectively.

Chapter 5

Results & Discussion

There were multiple categories generated before a final set of primary categories was selected, the selected categories were chosen as they were the clearest set of category names that encompass the largest amount of issues affecting the majority of games without being too specific towards one genre or one type of game. The results of this can be seen in Table 5.1.

Initial Category Test	Second Category Test	Selected Final Categories
1. Bugs & Glitches.	1. Game Crashes and Freezes.	1. Gameplay Bugs & Glitches.
2. Technical Issues.	2. Progression Halting Bugs.	2. Technical Issues & Crashes
3. Server & Multiplayer.	3. Startup and Initialization Errors.	3. Game Progression Blocks
4. UI Problems.	4. Graphics and Rendering Problems.	4. Graphics & Rendering Problems
5. Save Corruption.	5. Input and Control Issues.	5. Controls & Input Recognition
6. Support & Community.	6. Account and Access Restrictions.	6. Multiplayer & Connectivity Issues
7. Localization.	7. Save Data Corruption and Loss.	7. User Interface & Experience Problems
8. Content & Progress.	8. Multiplayer Connectivity and Matchmaking.	8. Audio Issues
9. Feedback & Suggestions.	9. Vehicle and Object Interactions Bugs.	9. Installation & Update Problems
10. Optimisation.	10. Game Optimization and Performance.	10. Achievements & Rewards Issues

Table 5.1: Test Categories Generated and Selected Final Primary Categories Generated

The selected final categories shown in Table 5.1 are applicable to the majority of games released on Steam, with the exception of "6. Multiplayer & Connectivity Issues" which only applies to games with online content. Developers may instead wish to focus their categories on more specific and unique aspects of their games, such as gameplay mechanics or systems. This may prove to offer more useful information on a game-by-game basis, however, developers should take care that their produced categories do not exclude more generic and high level issues that are not unique to their game or their gameplay systems.

5.1 Test Categorisation Results

To determine if the experiment was worth pursuing, an initial test was conducted on the initial set of categories generated. The initial categorisations test shown in Tables 5.2 and

5.3 demonstrated that not only were the GPT model categorisations reliable with each other, but also with the human reviewer as well. These results showed that there was a substantial level of agreement between all the pairs of raters. The lowest scoring pair of the models was GPT 4 Turbo and GPT 3.5 Turbo, the highest scoring pair of the models was GPT 4 and GPT 4 Turbo. Amongst the human reviewer comparisons, GPT 4 Turbo had the lowest level of agreement with the reviewer.

These results were promising enough to continue with the experiment and the production of new categories and further tests.

GPT 3.5 Turbo vs GPT 4	GPT 4 vs GPT 4 Turbo	GPT 4 Turbo vs GPT 3.5 Turbo
0.643	0.750	0.582

Table 5.2: Cohen's Kappa Scores between GPT Models, Selected Primary Categories, Initial Categorisation

GPT 3.5 Turbo vs Reviewer	GPT 4 vs Reviewer	GPT 4 Turbo vs Reviewer
0.585	0.597	0.570

Table 5.3: Cohen's Kappa Scores between GPT Models and Human Reviewer, Selected Primary Categories, Initial Categorisation

5.2 Primary Categorisation Results

Category	Avg. Frequency	Freq: (Original) GPT 3.5 Turbo	Freq: GPT 4	Freq: GPT 4 Turbo
2. Technical Issues & Crashes	70	71	67	72
1. Gameplay Bugs & Glitches.	44	58	42	32
3. Game Progression Blocks	31.667	9	43	43
4. Graphics & Rendering Problems	21.667	21	21	23
6. Multiplayer & Connectivity Issues	19	21	17	19
5. Controls & Input Recognition	10.667	8	14	10
10. Achievements & Rewards Issues	10	9	10	11
7. User Interface & Experience Problems	8.333	15	4	6
9. Installation & Update Problems	5	8	2	5
8. Audio Issues	3.667	4	4	3

Table 5.4: Frequency of Primary Categorisations by Models (Sorted by Average Frequency)

After each model had assigned the posts in the sample to the 10 final primary categorises generated previously, the frequency of each assigned category was counted across the three models as seen in Table5.4. To attempt to answer the second research question for this study, "How reliable are different popular Large Language Models at categorising unstructured and informal bug reports for games?" the frequency of each category in each model's categorisations was compared to see how the models categorise the data similarly, and perhaps more importantly, how they categorise the same data differently.

The average frequency for each category was then calculated. The average/mean value of the frequency in which the GPT models assigned each category is important and useful as it provides a single value that can later be used to compare against the average usefulness rating calculated from each participant's rating of the category during the developer survey, this will be expanded upon later.

Among the categories, "Technical Issues & Crashes" was the most frequently assigned, with an average occurrence/frequency of 70 within a dataset of 225. The second most frequent category, Gameplay Bugs & Glitches, had much lower mean frequency of 44, 38% lower than Technical Issues & Crashes **(see Table 5.4)**. These two categories are the highest level categories, encompassing the most amount of potential issues without being specific about what the issues entail. This suggests that when the models were not able to assign posts to any of the more specific categories (for example: Audio Issues) they were placed into one of

these two categories.

The uniquely high mean frequency of "Technical Issues & Crashes" invites some interest analysis. The first and most obvious being that it is the most important category and the most common reason for users posting to Steam regarding bug reports. The second relates to the scope of the categories. As the categorisation prompt tells the models to place a comment into the most relevant category, posts that contain multiple issues are likely to be placed into a category that best summarises the post as a whole rather than the individual issues discussed. For instance, a post may talk about graphics and audio problems. Whilst both of these issues have their own dedicated categories - "Graphics & Rendering Problems" and "Audio Issues" respectively- the model may place them into "Technical Issues & Crashes," a category that might encompass both, to avoid prioritising one issue over the other. This has been noted as a threat to validity.

Whilst the two most frequently assigned categories were consistently assigned across all the models, this is not the case for the third most frequent category, "Game Progression Blocks." As see in Table5.4, both GPT 4 and GPT 4 Turbo assigned this category to 43 posts, a relatively high frequency compared to other categories, whereas GPT 3.5 Turbo only assigned it 9 times. This vast difference suggests that GPT 3.5 Turbo assigns less importance to this category compared to its newer counterparts. The only other such case in where GPT 3.5 Turbo appears to disagree with the newer models is in the assignments of the "User Interface & Experience Problems" category, where it assigned noticeably more posts to the category compared to GPT 4 and GPT 4 Turbo. This suggests that the newer GPT 4 and GPT 4 Turbo models categorise certain types of bug reports differently than the older model, partially answering the second research question, however due to the subjectivity inherent in this form of categorisation it cannot be stated whether this represents an improvement or reduction in the accuracy of categorisations produced by these two models (GPT 4, 4 Turbo) compared to the older GPT 3.5 Turbo.

5.3 Inter-Rater Reliability Results between Model & Human Reviewer Primary Categorisations

Following the initial categorisation by both the human reviewer and the selected models, Inter-Rater Reliability/Cohen's Kappa scores for the raters' assignment of the primary categories were calculated. Inter-Rater Reliability offers a direct method of answering the second research question by directly comparing the raters (the models and human reviewer) categorisations against each other and producing a single score for each pair of that represents their Inter-Rater Reliability with each other whilst accounting for the possibility of them agreeing by chance.

GPT 3.5 Turbo vs GPT 4	GPT 4 vs GPT 4 Turbo	GPT 4 Turbo vs GPT 3.5 Turbo
0.574	0.716	0.581

Table 5.5: Cohen's Kappa Scores between GPT Models, Selected Primary Categories, Initial Categorisation

To begin, the models were compared against each other. As seen in Table 5.5 GPT 4 and GPT 4 Turbo demonstrated the highest Inter-Rater Reliability in their categorisations, scoring 0.12 higher than the next most reliable pair. Interpreting the Cohen's Kappa using table 3.1 previously shown in the methodology, GPT 4 and GPT 4 Turbo display substantial agreement with each other whilst only showing moderate agreement with GPT 3.5 Turbo's categorisations. This shows that GPT 4 and GPT 4 Turbo are both more meaningly more reliable (using Inter-Rater Reliability) with each other than either are with GPT 3.5 Turbo.

GPT 3.5 Turbo vs Reviewer	GPT 4 vs Reviewer	GPT 4 Turbo vs Reviewer
0.597	0.628	0.635

Table 5.6: Cohen's Kappa Scores between GPT Models and Human Reviewer, Selected Primary Categories, Initial Categorisation

Following the comparison of the models against each other, each model's categorisations were compared to the human reviewer's categorisations. This was once again done using Cohen's Kappa scores to determine Inter-Rater Reliability. Interestingly, Table 5.6 shows a clear positive correlation between the GPT model recency and alignment with the human reviewer. Once again the Cohen's Kappa scores were interpreted using table 3.1 . GPT 3.5 Turbo and the human reviewer were found to have moderate agreement whilst GPT 4 and the human reviewer and GPT 4 Turbo and the human reviewer were found to have substantial agreement, though this is not due to a large difference in scores (only a 0.38 difference between GPT 3.5 Turbo and GPT 4 Turbo scores with the human reviewer) and instead in the way the thresholds are defined. This still suggests that newer GPT models are more likely to categorise feedback in a manner more similar to human judgement, potentially indicating improvements in their ability to interpret and classify user reported feedback accurately. Though this improvement would be subtle.

5.4 Primary Categories Disagreement Analysis & Improvements

Cause	Frequency
Ambiguous Criteria	15
Subjectivity	26
Multi-Faceted Issue	21
Insufficient Context Provided	4
Different Understanding of Comment	4

Table 5.7: Frequency of Disagreement Causes between GPT 3.5 Turbo Primary Categorisations and the Human Reviewer's Categorisations

Following the primary categorisations, a breakdown of the disagreements between the least reliable model (GPT 3.5 Turbo) and the human reviewer was conducted with the goal of improving the models Inter-Rater Reliability by improving the prompt used to categorise the data. The causes for disagreement fell into 5 different categories, Ambiguous Criteria, Subjectivity, Multi-Faceted Issues, Insufficient Context, and Different Understanding of Comment (See Table 5.7). The most common of these being Subjectivity and the post holding Multi-Faceted issues. These issues are largely entwined, as a model or reviewer will place a post with Multi-faceted issues into the category they believe is most important.

For example, the post: "I have a problemI reinstalled windows and after installing steam. I logged into the game and found that my progress was missing for some reason I didn't understand. Although it seems like there should be cloud saving." was placed into the primary category "Game Progression Blocks" by the GPT model, and into the primary category "Technical Issues & Crashes" by the human reviewer. This disagreement was deemed to be due to the post holding Multi-Faceted Issues due to the problem being a technical issue (saving not working) which was then causing game progression issues.

To attempt to improve GPT 3.5 Turbo's Inter-Rater Reliability with other models and the reviewer, the prompt was modified to include the fully explained category definitions/descriptions instead of just the category name.

Original Prompt:

"Place this comment the most fitting category: 1. Gameplay Bugs & Glitches. 2. Technical Issues & Crashes. 3. Game Progression Blocks. 4. Graphics & Rendering Problems. 5. Controls & Input Recognition. 6. Multiplayer & Connectivity Issues. 7. User Interface & Experience Problems. 8. Audio Issues. 9. Installation & Update Problems. 10. Achievements & Rewards Issues. Limit response to single character, the number of the category, no text other than that value: "

New Prompt:
"Place this comment the most fitting category:
1. Gameplay Bugs & Glitches- This category includes any reports of in-game behaviour not working as intended, such as player or object movement issues, creatures acting strangely, or mechanics not functioning correctly.
2. Technical Issues & Crashes- Comments about the game crashing, freezing, or not launching properly fit here. This includes problems with the games performance on specific hardware and error messages received upon launch or during gameplay.
3. Game Progression Blocks- This category is for issues where players get stuck in a specific part of the game due to a bug or an unclear game mechanic, preventing them from progressing further.
4. Graphics & Rendering Problems- Issues related to the game's visual aspects, such as textures not loading, lighting issues, or any anomalies in the game's graphical presentation, belong here.
5. Controls & Input Recognition- Comments reporting problems with game controls, such as keybindings not working, controller issues, or difficulties with input devices, are categorized here.
6. Multiplayer & Connectivity Issues- This covers any issues related to playing the game in a multiplayer setting, including server connection problems, desync issues, or difficulty joining or hosting games.
7. User Interface & Experience Problems- Issues with the games UI, such as inventory management problems, unclear objectives, or HUD display issues, would be compiled under this category.
8. Audio Issues- This category involves problems with the games sound, including missing or distorted audio, volume issues, or specific sounds not playing correctly.
9. Installation & Update Problems- Comments about difficulties installing the game, updating it, or issues arising from patches and game updates are included here.
10. Achievements & Rewards Issues- Problems related to in-game achievements not being awarded, reward items not appearing, or other issues concerning the games reward system are sorted into this category.
GPT you should Limit response to single character, the number of the category, no text other than that value: "

Cohen's Kappa	**Old GPT 3.5 Turbo**	**Old GPT 4**	**Old GPT 4 Turbo**	**Human**
New GPT 3.5 Turbo	0.574	0.618	0.587	0.584
Old GPT 3.5 Turbo	-	0.643	0.582	0.585

Table 5.8: GPT 3.5 Turbo Recategorisation Test

The sample dataset was re-categorised by GPT 3.5 Turbo using the modified prompt, the results of which are shown in 5.8. There were very subtle improvements in Inter-Rater Reliability between the new GPT 3.5 Turbo categorisations and the GPT 4 Turbo and Human

Reviewer categorisations following recategorisations, though each of these improvements were less than a 0.1 increase. Interestingly, the new GPT 3.5 Turbo recategorisations and the GPT 4 categorisations became less reliable by 0.3, the largest difference between the old GPT 3.5 Turbo pairs and the new GPT 3.5 Turbo pairs. In the context of the second research question, this shows that the prompt used to categorise data holds substantial relevance to the Inter-Rater Reliability of the models in addition to the recency in which they were released, as by removing ambiguity in the meaning of the categories, the model was better able to categorise the data, albeit subtly in this case.

5.5 Sub-Category Generation Results

To further answer the second research question and to understand how reliable the GPT models are with assigning posts to sub-categories, the primary category with the highest average assignment frequency by the three models was selected to undergo sub-categorisation. As previously shown in table5.4 , "Technical Issues & Crashes," with an average frequency of 70, had the highest average frequency across the primary categories. Repeating the process in which the primary categories were generated, sub-categories were produced. Like with the primary category generation, there were some unused categories generated at this stage as well. The unused and selected sub-categories are shown in Table 5.9.

Initial Category Test	Selected Final Categories
Controls Issues.	1. Specific Mission/Level Crashes
2. Missing Assets.	2. Device-Specific Issues
3. Objective Progression.	3. Performance Issues (FPS Drops/Stutters)
4. Performance.	4. Crash On Launch/Startup Issues
5. Visual/Audio Glitches.	5. In-Game Freeze/Crash
6. Artificial Intelligence Issues.	6. Asset Loading Errors
7. Combat Issues.	7. Memory Management/Leaks
8. Game Stability.	8. Game Integrity and Corruption Problems
9. Interaction.	9. Hardware Compatability Issues
10. Achievement Issues.	10. Post-Update Problems

Table 5.9: Test Categories Generated and Selected Final Sub-Categories Generated

5.6 Sub-Categorisation Results

After "Technical Issues & Crashes" was selected for sub-categorisation, the posts assigned to the category by each model were collected into 3 different sample datasets consisting of the posts assigned to this category by GPT 3.5 Turbo, GPT 4, and GPT 4 Turbo.

The following tables (Table 5.10, Table 5.11and Table 5.12) represent how the GPT models categorised the three prior mentioned new sample datasets. Table 5.10 shows how each model categorised posts deemed to be in the "Technical Issues & Crashes" by GPT 3.5 Turbo, Table5.11 does the same for posts assigned to this primary category by GPT 4, and Table 5.12 shows the same for posts assigned to "Technical Issues & Crashes" by GPT 4 Turbo.

Category	Avg. Frequency	Freq: GPT 3.5 Turbo	Freq: GPT 4	Freq: GPT 4 Turbo
4. Crash On Launch/Startup Issues	25	25	25	25
5. In-Game Freeze/Crash	24.333	29	22	22
2. Device-Specific Issues	5	9	3	3
1. Specific Mission/Level Crashes	4	2	5	5
6. Asset Loading Errors	3.667	3	4	4
7. Memory Management/Leaks	2.667	2	3	3
10. Post-Update Problems	2.667	0	4	4
8. Game Integrity and Corruption Problems	2	0	3	3
3. Performance Issues (FPS Drops/Stutters)	1	1	1	1
9. Hardware Compatability Issues	0.667	0	1	1

Table 5.10: GPT 3.5 Turbo Dataset: Frequency of Primary Categorisations by Models (Sorted by Average Frequency)

Category	Avg. Frequency	Freq: GPT 3.5 Turbo	Freq: GPT 4	Freq: GPT 4 Turbo
4. Crash On Launch/Startup Issues	22.333	23	20	24
5. In-Game Freeze/Crash	17.333	24	10	18
3. Performance Issues (FPS Drops/Stutters)	6.667	6	9	5
9. Hardware Compatability Issues	4.667	3	7	4
10. Post-Update Problems	4.667	2	8	4
1. Specific Mission/Level Crashes	3.333	1	6	3
2. Device-Specific Issues	3.333	6	2	2
7. Memory Management/Leaks	1.333	1	1	2
8. Game Integrity and Corruption Problems	1.333	0	2	2
6. Asset Loading Errors	1	0	1	2

Table 5.11: GPT 4 Dataset: Frequency of Sub-Categorisations by Models (Sorted by Average Frequency)

Category	Avg. Frequency	Freq: GPT 3.5 Turbo	Freq: GPT 4	Freq: GPT 4 Turbo
4. Crash On Launch/Startup Issues	23	22	21	26
5. In-Game Freeze/Crash	19	27	11	19
9. Hardware Compatability Issues	6.333	6	9	4
3. Performance Issues (FPS Drops/Stutters)	5.667	3	9	5
10. Post-Update Problems	4.333	1	8	4
2. Device-Specific Issues	4	7	2	3
1. Specific Mission/Level Crashes	2.667	0	5	3
8. Game Integrity and Corruption Problems	2.667	2	3	3
7. Memory Management/Leaks	2	2	2	2
6. Asset Loading Errors	1.333	1	1	2

Table 5.12: GPT 4 Turbo Dataset: Frequency of Sub-Categorisations by Models (Sorted by Average Frequency)

From these categories, the mean frequency of categorisations by the three models was calculated to create an average dataset from the three sub-categorisation samples (see Table 5.13).

Category	Avg. Frequency	Freq: GPT 3.5 Turbo	Freq: GPT 4	Freq: GPT 4 Turbo
1. Specific Mission/Level Crashes	3.333	1	5.333	3.666
2. Device-Specific Issues	4.111	7.333	2.333	2.666
3. Performance Issues (FPS Drops/Stutters)	4.444	3.333	6.333	3.666
4. Crash On Launch/Startup Issues	23.444	23.333	22	25
5. In-Game Freeze/Crash	20.222	26.666	14.333	19.666
6. Asset Loading Errors	2	1.333	2	2.666
7. Memory Management/Leaks	2	1.666	2	2.333
8. Game Integrity and Corruption Problems	2	0.666	2.666	2.666
9. Hardware Compatability Issues	3.888	3	5.666	3
10. Post-Update Problems	3.888	1	6.666	4

Table 5.13: Combined Dataset: Frequency of Sub-Categorisations by Models (Sorted by Average Frequency)

The two most frequent categories, "Crash on Launch/Startup Issues" and "In-Game Freeze/Crash," were consistent across all 3 models, being the first and second most frequent sub-category respectively. These two categories were vastly more occurrent than any other category across all of the datasets (See Table 5.10, Table 5.11 and Table 5.12). Unlike in the primary categorisations where "Technical Issues & Crashes" and "Gameplay Bugs & Glitches" were likewise far more frequent than other categories (see Table 5.4), these frequent sub-categories do not appear to be any more or less vague than other sub-categories and do not seem to be umbrella categories that may encompass other smaller sub-categories. Therefore, it is likely that these categories are simply more genuinely occurrent in the dataset.

The most frequent category, "Crash on Launch/Startup Issues," had such a high frequency that it's mean frequency across the 3 datasets was larger than the mean frequency of the 7 least occurrent categories combined (see Table 5.13).. This has various interesting implications, the first is that these are simply much more common in the dataset, the second is that players are more likely to report issues that actively prevent them from playing the game (not being able to progress due to crashing) than smaller, less destructive issues such as loading issues. The third, perhaps more useful implication, is that players do not have the technical expertise or experience to identify more complicated issues such as "Memory Management/Leaks" on their own, instead only reporting the symptoms of the issue (crashing, lag, etc). without being able to articulate the cause.

5.7 Inter-Rater Reliability Results between Model & Human Reviewer Sub-Categorisations

With sub-categorisations complete, each model's categorisations were compared using Cohen's Kappa to determine Inter-Rater Reliability in each of the 3 produced datasets. When comparing the model's categorisations against each other, GPT 4 and GPT 4 Turbo was the pair with the highest level of Inter-Rater Reliability in the GPT 3.5 Turbo and GPT 4 datasets, whereas GPT 4 Turbo and GPT 3.5 Turbo had the highest level of Inter-Rater Reliability in the GPT 4 Turbo dataset. Notably, GPT 3.5 Turbo and GPT 4 was consistently the least reliable pair across all datasets.

As seen in Table 5.14 GPT 4 and GPT 4 Turbo was the pair with the highest mean Inter-Rater Reliability, followed by GPT 4 and GPT 3.5 Turbo, and then by GPT 3.5 Turbo and GPT 4. These results shown in Table 5.14 indicate a substantial drop in the Inter-Rater Reliability of the GPT 3.5 Turbo vs GPT 4 Turbo and GPT 4 vs GPT 4 Turbo pairs compared to their Cohen's Kappa from the primary categorisations (see Table 5.5). This shows that the GPT models are less reliable with each other when categorising posts into lower level sub-categories than they were with categorising the higher level primary categories.

	GPT 3.5 Turbo vs GPT 4	GPT 4 vs GPT 4 Turbo	GPT 4 Turbo vs GPT 3.5 Turbo
GPT 3.5 Turbo Dataset	0.359	0.579	0.523
GPT 4 Dataset	0.442	0.685	0.601
GPT 4 Turbo Dataset	0.438	0.588	0.615
Mean	0.413	0.617	0.580

Table 5.14: Cohen's Kappa Scores between GPT Models, Sub-Categories

Similarly to the primary categorisation results seen in Table 5.6, the results shown in Table 5.15 show that there was a positive correlation between the recency of the model and the meanInter-Rater Reliability score between that model and the human reviewer.

Table 5.15 shows that as the recency of the model that produced each dataset increased, the Inter-Rater Reliability score of each model with the human reviewer also increased, with the exception of the GPT 4 dataset where GPT 4 had a decrease in its Inter-Rater Reliability between its categorisations and that of the human reviewer.

	GPT 3.5 Turbo vs Reviewer	GPT 4 vs Reviewer	GPT 4 Turbo vs Reviewer
GPT 3.5 Turbo Dataset	0.427	0.580	0.616
GPT 4 Dataset	0.585	0.536	0.708
GPT 4 Turbo Dataset	0.619	0.603	0.719
Mean	0.544	0.573	0.681

Table 5.15: Cohen's Kappa Scores between GPT Models and Human Reviewer, Sub-Categories

5.8 Survey Quantitive Results

Research Question One asks" How useful do developers find categories produced by Large Language Models?" To attempt to answer this, a survey was conducted using Prolific that asked people employed in the Games Industry and people who aren't currently employed in the industry yet like game development experience to rate the usefulness of the produced categories.

To ensure that developers had the prerequisite knowledge to provide meaningful and usable data, a screening survey was produced that tested basic knowledge. Two screening surveys were conducted with two participant pools based on the previous mentioned criteria, the Games Industry participant pool and the None Games Industry participant pool which had 100 responses each. People who passed this survey were later invited to the primary survey.

The primary survey had 101 total usable responses made up of the Games Industry participant pool and the None Games Industry participant pool which had 38 and 63 usable responses respectively.

For each primary and sub-category, developers were required to rate the category on a scale of 1-4, with the option to provide none-required qualitative feedback.

5.8.1 Usefulness Ratings, Assignment Frequency & Standard Deviation values across Primary Categories.

Using the average usefulness ratings (measured on a scale of 1-4, "Not Useful At all" to "Extremely Useful") collected from the results from the Games Industry Participant Pool and the Non-Games Industry Participant Pool provides a straightforward measure of each category's perceived usability in the eyes of developers.

Standard deviation was used as a metric to assess the level of agreement amongst developers regarding the usefulness of the categories. Whilst the average/mean rating indicates the overall perceived usefulness of a category by developers, standard deviation

reveals the spread and variance in said ratings. A low standard deviation suggests that most developers rated the category similarly, indicating a large level of agreement. In contrast a high standard deviation indicates greater variability/variety in responses, showing a lack of agreement. For example, the primary category "Technical Issues & Crashes" had an average usefulness rating of 3.663 and a standard deviation of 0.605 in those ratings, this indicates that developers both found this to be a largely useful category and were quite united in their opinion regarding this category.

Table 5.16 shows the combined participant pool's (Games Industry pool combined with the None-Games Industry pool) usefulness ratings and standard deviation of the usefulness ratings for the primary categories produced by the LLM models. Table 5.17 shows the combined pool average rating of the primary categories as well as the average frequency in which posts were assigned to this category by the three GPT models.

The second primary category, "Technical Issues & Crashes," proved to be the highest-rated category across both participant pools and the most frequently assigned category by all models across all datasets. These results indicate that real world game developers identify this category as being the most useful, with these findings being re-enforced by the GPT models which found it to describe the largest number of issues reported. This can be seen Table 5.17 which reports the combined pool figures. As seen in Table 5.16, the standard deviation score for this category was the lowest of all categories across both participant pools, signifying the highest level of agreement amongst respondents. The combined pool's standard deviation for the usefulness of this category was 0.60, which was significantly lower than the next lowest score of 0.70. This shows that developers were extremely united in their interpretations of this category's usefulness.

"Gameplay Bugs & Glitches," despite being ranked 2nd and 3rd by rating and frequency respectively, had a notably higher standard deviation than "Technical Issues & Crashes." Indicating that whilst may developers find the category to be useful, there are some who disagree and find it to be none useful.

According to the combined participant pool's results shown in Table 5.16 , the mean usefulness rating for the primary categories was 3.1043, "Very Useful." The mean standard deviation for usefulness ratings for the primary categories was 0.766, indicating that developers were largely united regarding their ratings of these categories. The usefulness ratings of all primary categories fell below 1, whilst there was some variance in standard deviation, this statistic shows that there was no signifiant variance in the developer ratings at all. This collectively shows that Large Language Models are quite capable of generating useful bug report categories for games when provided with sample bug report data.

	Not Useful At All	Slightly Useful	Very Useful	Extremely Useful	Average (1-4)	Standard Deviation
2. Technical Issues & Crashes	1	4	22	71	3.663	0.6051459185
4. Graphics & Rendering Problems	0	20	49	29	3.092	0.7011176892
5. Controls & Input Recognition	0	22	46	30	3.082	0.7238427694
3. Game Progression Blocks	1	11	34	50	3.385	0.7270057265
1. Gameplay Bugs & Glitches.	2	10	36	51	3.374	0.746518635
6. Multiplayer & Connectivity Issues	0	23	37	37	3.144	0.7731271447
8. Audio Issues	3	39	37	19	2.735	0.8024327858
7. User Interface & Experience Problems	2	31	35	31	2.960	0.8399036972
10. Achievements & Rewards Issues	7	41	33	16	2.598	0.8453608247
9. Installation & Update Problems	5	24	34	35	3.010	0.8977852336

Table 5.16: Primary Categories Combined Frequency of Ratings, Average Rating and Standard Deviation (Sorted by Standard Deviation Low to High

Category	Combined Avg. Rating	Avg. Frequency	Ranking by Rating	Ranking by Frequency
1. Gameplay Bugs & Glitches.	3.373737374	44	3	2
2. Technical Issues & Crashes	3.663265306	70	1	1
3. Game Progression Blocks	3.385416667	31.66666667	2	3
4. Graphics & Rendering Problems	3.091836735	21.66666667	5	4
5. Controls & Input Recognition	3.081632653	10.66666667	6	6
6. Multiplayer & Connectivity Issues	3.144329897	19	4	5
7. User Interface & Experience Problems	2.95959596	8.333333333	8	8
8. Audio Issues	2.734693878	3.666666667	9	10
9. Installation & Update Problems	3.010204082	5	7	9
10. Achievements & Rewards Issues	2.597938144	10	10	7

Table 5.17: Primary Categories Combined Average Usefulness Rating (1-4 Not Useful at all - Extremely Useful) and Average Frequency Assigned by Models

5.8.2 Usefulness Ratings & Standard Deviation values across Sub-Categories.

This same process was repeated for the sub-categories produced. Table 5.19 shows the combined participant pool's (Games Industry pool combined with the None-Games Industry pool) usefulness ratings and standard deviation of the usefulness ratings for the sub-categories produced by the LLM models. Table 5.18 shows the combined pool average rating of the sub-categories as well as the average frequency in which posts were assigned to this category by the three GPT models.

According to the combined participant pool's average usefulness ratings of the sub-categories shown in Table 5.19, the average usefulness rating for the sub-categories produced by LLM models was 3.249, which is slightly higher than the average usefulness rating of the primary categories, though not by a meaningful amount. The standard deviation for the average usefulness ratings of the sub-categories was 0.776, which like the primary categories standard deviation for this score, is quite low. For the sub-categories, Table5.18 shows that there was a weaker correlation between the average usefulness rating of the category and the average frequency in which models assigned posts to the category. This indicates that whilst developers may find these sub-categories to be useful, they were not frequently used by the GPT models.

These overall results indicate that like with the primary categories, LLM models are capable of generating useful Sub-Categories for game developers.

Category	Combined Avg. Rating	Avg. Frequency	Ranking by Rating	Ranking by Frequency
1. Specific Mission/Level Crashes	3.39	3.777777778	3	7
2. Device-Specific Issues	3.252525253	4	4	6
3. Performance Issues (FPS Drops/Stutters)	3.12	5.222222222	5	3
4. Crash On Launch/Startup Issues	3.49	22.55555556	2	1
5. In-Game Freeze/Crash	3.489795918	19.22222222	1	2
6. Asset Loading Errors	3.04040404	1.777777778	8	10
7. Memory Management/Leaks	3.288659794	1.888888889	7	9
8. Game Integrity and Corruption Problems	3.288659794	2.111111111	6	8
9. Hardware Compatability Issues	2.97	4.666666667	10	4
10. Post-Update Problems	3.15625	4	9	5

Table 5.18: Sub-Categories Combined Average Usefulness Rating (1-4 Not Useful at all - Extremely Useful) and Average Frequency Assigned by Models

	Not Useful At All	Slightly Useful	Very Useful	Extremely Useful	Average (1-4)	Standard Deviation
5. In-Game Freeze/Crash	0	5	40	53	3.489795918	0.5932425246
4. Crash On Launch/Startup Issues	1	7	34	58	3.49	0.6707458535
1. Specific Mission/Level Crashes	1	12	34	53	3.39	0.7334166619
3. Performance Issues (FPS Drops/Stutters)	0	25	38	37	3.12	0.7782030583
7. Memory Management/Leaks	1	17	32	47	3.288659794	0.7859430331
2. Device-Specific Issues	1	19	33	46	3.252525253	0.7957390865
6. Asset Loading Errors	2	24	41	32	3.04040404	0.8030144208
8. Game Integrity and Corruption Problems	3	14	32	48	3.288659794	0.8243555795
10. Post-Update Problems	5	16	34	41	3.15625	0.8818555839
9. Hardware Compatability Issues	4	30	31	35	2.97	0.899499861

Table 5.19: Sub- Categories Combined Frequency of Ratings, Average Rating and Standard Deviation

5.9 Breakdown of Qualitative Responses

5.9.1 Qualitative Responses regarding Primary Categories

During the survey, respondents were given the option to provide additional feedback for each primary and sub-category they were presented with. In total, there were 230 optional qualitative responses in the combined participant pool, the breakdown of which can be seen in Table 5.20. Some of this additional qualitative feedback offered insights into how the categories could be improved, whilst others were general comments. Similarly to the previous disagreement analysis, these comments were placed into 9 categories based on the information they contain:
Development Impact - These comments speak of the ways and magnitude that a category may impact product development.
Too Generic/ Too High Level- These comments indicate concerns that the category is too high level or too vague to be useful.
Too Specific/ Too Low Level- These comments indicate concerns that the category is too low level or too specific to be useful.
High Importance- These comments indicate that the respondent finds the category to be of high importance.
Low Importance- These comments indicate that the respondent finds the category to be of low importance.
Player Perspective- These comments speak of the ways issues in this category affect players.
Reporting/ Communication- These comments speak, indicate concerns with, and suggest methods in which these issues can be reported by players.
Understanding- These comments indicate a difficulty to understand what the category entails.
Other- This category is for miscellaneous comments that don't fit into the previously described categories and are not frequent enough for a category of their own.

As seen in Table 5.20, the majority of comments were placed into two categories: Low importance, where users reported that a category would not be very useful, and high importance, where they would report a category would be very useful. Of these two categories, comments attributing high importance to primary categories were vastly more occurrent than comments attributing low importance. In order, "Gameplay Bugs & Glitches", "Technical Issues & Crashes", "Game Progression Blocks", and "Installation & Update Problems" were attributed the highest level of importance. Interestingly, despite having the fourth most comments stating its high level of importance, "Installation & Update Problems" was actually rated the least useful category by the combined pool. There were not many comments attributing low importance to the primary categories, though "Achievements & Rewards Issues" had the highest number of these comments. This provides further detail in answering Research Question One, where more information regarding the usefulness of these categories

is now available.

Whilst the majority of comments were regarding the importance of the categories they were discussing, the second largest number of comments were regarding the scope/specificity of the categories. These comments were split into two categories respectively: High Level, where users would report concerns regarding ambiguity, genericness, and overlap between this and other categories and Low Level, where users would report concerns and comments regarding niche-ness and specificity.

Table 5.20 shows that despite having high usefulness ratings and being assigned frequently by the GPT models "Gameplay Bugs & Glitches" and "Technical Issues & Crashes" both had numerous comments indicating that these categories were too high level to be useful. These results suggest that the high frequency of these categories in the GPT datasets is because they are the most generic and encompass a variety of differing bugs and issues and perhaps developers would be better suited if these categories were replaced with more specific ones.

Category	Development Impact	Too Generic/ Too High Level	Too Specific/ Too Low Level	High Importance	Low Importance	Player Perspective	Reporting/ Communication	Understanding	Other	Total
1. Gameplay Bugs & Glitches.	4	13	0	10	0	3	5	2	1	38
2. Technical Issues & Crashes	2	9	0	9	0	7	3	2	1	33
3. Game Progression Blocks	1	1	0	10	0	5	4	1	3	25
4. Graphics & Rendering Problems	0	4	0	6	2	1	2	0	6	21
5. Controls & Input Recognition	1	1	2	4	0	5	1	1	2	17
6. Multiplayer & Connectivity Issues	5	3	1	5	0	3	1	0	3	21
7. User Interface & Experience Problems	2	6	0	2	3	2	3	2	2	22
8. Audio Issues	1	2	0	5	3	2	2	0	1	16
9. Installation & Update Problems	2	3	0	8	0	3	2	0	0	18
10. Achievements & Rewards Issues	0	2	2	1	5	5	0	0	4	19

Table 5.20: Combined Pool Qualitative Feedback for Primary Categories Breakdown

5.9.2 Qualitative Responses regarding Sub-Categories

As qualitative feedback served as an optional addition to the quantitative feedback, users were not required to provide it. Due to this, there was not enough data for meaningful analysis, with only 111 comments provided that was evenly distributed across each category of feedback, there is not enough information to draw conclusions regarding any sub-category from this feedback.

5.10 Discussion

In bug triage, development-oriented decision making involves categorising bug reports based on a variety of factors, one of which is the element of the product affected by the issue/defect Anvik and Murphy, 2011. Our results show that LLMs are capable of identifying the elements of the product (In this case games) effected by bug reports that are provided to the LLMs, generating categories. Our results also show that LLMs are effective at placing bug reports into these categories provided to it. Allowing LLMs to perform the categorisation element of development-oriented triage unsupervised. This has previously been an expensive and time-consuming process for developers.

Historically, when clustering bug reports the process has been to group similar reports together using a clustering method of choice and then prescribing a label to these clusters (Luaphol et al., 2018,Limsettho et al., 2014). The results presented in this study strongly show that Large Language Models are capable of placing bug reports into pre-determined categories, effectively allowing developers to seek out groups of bug reports that approach or detail a desired issue rather than clustering bug reports together based on similarity and then evaluating what these clusters contain. This has interesting implications for the future of bug triage, for example a developer may have a pre-existing group of bug or defect reports that describe a certain problem, the developer will now have the ability to check if future bug reports also present the same problem.

However, these past approaches still have merits. Due to the limited context windows of the varying LLM models, grouping posts together based on similarities may not be applicable on a large scale, meaning that if developers want to create these clusters they may still be better served by traditional methods. These methods could also act complementary to each other, breaking posts down into smaller categories and then grouping them based on similarity to further aid developers.

In addition to this, our results show that despite certain bug reports lacking in formality, structure and proper vocabulary, Large Language Models are still effective at categorising these reports based on the context present in the bug report. These types of reports would not be ideal for previous clustering methods that rely on various methods of calculating and weighting term-frequency to cluster reports together (Luaphol et al., 2018,Limsettho et al., 2014).

Chapter 6

Threats to Validity

6.1 Data Collection Criteria

During data collection, one of the selection criteria for games based on the minimum number of posts/entries in their Steam bug report discussion section is 50. This introduces the possibility that games that provide meaningful bug reports that fail to meet this threshold will be excluded. Potentially excluding useful data.

6.1.1 Optional Qualitative Feedback

One limitation of this study is that qualitative feedback from Game Developers was optional instead of required, this choice was made as to incentivise as many completed surveys as possible by reducing the time required to complete it. Unfortunately, this low amount of data has limited the exploration of what could be done to improve the categories produced.

6.1.2 Generalisability

To keep this study focused and to ensure that the researcher could understand the content of any collected data, its scope was limited exclusively to PC games on the Steam distribution platform for data collection. This does have implications for the generalisability of the results. First, the data collected may be representative of the types of bugs appearing in PC games but not representative of the issues affecting console games (for example, PlayStation or Xbox games). Second, the data collected may be representative of the types of bugs appearing for PC games on the Steam platform but may not be representative of the types of issues affecting games on other platforms that may have different methods for bug reporting. Third, and most importantly, this evaluation of GPT models is representative only of their usage with video game bug report data, and may not be representative of their reliability with bug reports from other types of Software.

6.2 Survey Participant Pools

When recruiting from Prolific, it became apparent that recruiting participants solely from the Games Industry was not feasible due to the number of participants available and the low likelihood of all of them responding. To counter this, additional criteria was developed for a second participant pool that would target people that likely had game development experience, with them explicitly having software development experience and experience playing games.

This introduced the possible issue that this additional criteria would not work and results would be skewed by none-game developers. To counter this, a screening survey was produced that tested the knowledge of participants to ensure that developers had the prerequisite knowledge to provide meaningful feedback.

In the survey, there was not a significant difference in the mean average usefulness rating of the primary categories nor the sub-categories, this can be seen in Tables 7.36 and 7.35 for the None-Game Industry pool and Games Industry Pool for the Primary categories and Tables 7.38 and 7.37 for the Sub-categories, these tables can also be used to view the individual standard deviation for the average usefulness rating of primary and sub-categories in each participant pool.

There was not a significant level of variety between the standard deviation for the average usefulness rating of the primary categories in the Games Industry pool and the None-Games Industry pool, except for three categories. The highest of these being "Achievements & Rewards Issues," which had a score of 0.741 in the None-Games Industry pool and 0.967 in the Games Industry Pool, a difference of 0.226. This category was rated slightly more useful by the Games Industry pool than the None-Games Industry pool, whilst also having a lowest level of agreement in the Games Industry pool than the None-Games Industry pool.

The second of these categories, "Game Progression Blocks," had a notably higher standard deviation in the Games Industry pool than in the None-Games Industry pool, scoring 0.792 and 0.679 respectively. Despite this, it still had the 2nd highest combined pool average rating, indicating that whilst both pools generally found this category very useful, the Games Industry participants were more divided on the matter.

The third category with large variation in standard deviation between pools is "Installation & Update Problems." The category had a Games Industry pool standard deviation of 0.953 and a None-Games Industry pool standard deviation of 0.861, once again indicating that the Games Industry pool was less united on the matter of its usefulness than the None-Games Industry pool. However, unlike the "Game Progression Blocks" category this category ranked lower in usefulness, being the 7th most useful category.

After calculating the mean standard deviation for the usefulness ratings of all categories, the Games Industry pool had a much higher mean standard deviation than the None-Games industry participants, having values of 0.976 and 0.741 respectively. This shows that there was substantially less uniformity in the responses of the Games Industry participants than

the None-Games Industry Participants.

The difference between the standard deviation of the Games Developers pool and the None-Game developers pool was larger in the sub-categories than with the primary categories. With the average difference in Standard Deviation between the two pools being 0.1222 (See Table 5.19).

Overall, the difference in Average Usefulness Ratings and Standard Deviation scores across both participant pools is negligible, though it was worth noting.

Chapter 7

Conclusion

The first research question for this study was "How useful do developers find categories produced by Large Language Models?" To answer this question developers were presented with categories generated by Large Language Models based on a dataset of bug reports and user feedback from games of various genres and asked to rate them based on usefulness. The mean usefulness scores were then calculated for each category, as well as the standard deviation being calculated for each score to indicate how much variance there was amongst the developers regarding the score of these categories. This was done for 10 primary categories and 10 sub-categories based on a selected primary category. Following the rating of the primary and sub-categories, developers were presented with the option to provide additional qualitative feedback regarding each category.

Higher level (more vague) categories were rated as more useful by developers than lower level, more specific categories. These findings indicate a correlation between what developers think will be most frequent and what developers think will be the most useful. However, multiple developers used the optional qualitative feedback question to express concern that the vagueness of these categories would in fact make them less practical for real world applications, which is in conflict with the ratings provided by developers regarding the categories.

The second research question for this study was "How reliable are different popular Large Language Models at categorising unstructured and informal bug reports for games?" To answer this question three popular selected to categorise a sample dataset of unstructured and informal bug reports collected from Steam. Cohen's Kappa, a statistical measure of Inter-Rater Reliability, was used to assess how reliably the models agreed with each other and with a human reviewers classifications of the same data. The findings suggest that even older LLMs which may lack features and advancements as compared to newer models are still capable of reliably completing the task with results comparable to those of a human reviewer. Despite this, there was still a positive correlation between the recency of a model and the Inter-Rater Reliability between that models categorisations and that of the reviewer.

I believe the natural next steps to this investigation is to test other Large Language Models to see how they perform compared to OpenAI's GPT models. The integration of LLM categorisations into existing issue tracking systems also provides ample opportunities for research. As our results indicate that LLMs are capable of performing the categorisation element of Development-Oriented decision making in triage, I believe it would be valuable to explore how capable LLMs are at performing Repository-Oriented Decision Making, where the goal is to reduce the number of bug reports in a repository and prioritise the best, smallest sample of bug reports (Anvik and Murphy, 2011).

The approaches to categorisation used in this study could also be consolidated and packaged into a single tool for developers to use to categorise their user feedback.

Bibliography

Antoniol, Giuliano et al. (2008). "Is it a bug or an enhancement? a text-based approach to classify change requests". In: *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, pp. 304–318.

Anvik, John and Gail C. Murphy (2011). "Reducing the effort of bug report triage: Recommenders for development-oriented decisions". In: 20.3.

Bergmann, Dave (2023). *What is a context window?* URL: `https://www.ibm.com/think/topics/context-window` (visited on 03/03/2025).

Bettenburg, Nicolas et al. (2008). "What makes a good bug report?" In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 308–318.

Brown, Tom et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Chen, Banghao et al. (2025). "Unleashing the potential of prompt engineering for large language models". In: *Patterns*.

Danilova, Anastasia et al. (2021). "Do you really code? designing and evaluating screening questions for online surveys with programmers". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, pp. 537–548.

Fruntke, Lukas and Jens Krinke (2025). "Automatically fixing dependency breaking changes". In: ACM (Association for Computing Machinery).

Hendy, Amr et al. (2023). "How good are gpt models at machine translation? a comprehensive evaluation". In: *arXiv preprint arXiv:2302.09210*.

Herzig, Kim, Sascha Just, and Andreas Zeller (2013). "It's not a bug, it's a feature: how misclassification impacts bug prediction". In: *2013 35th international conference on software engineering (ICSE)*. IEEE, pp. 392–401.

Hossain, Soneya Binta et al. (2024). "A deep dive into large language models for automated bug localization and repair". In: *Proceedings of the ACM on Software Engineering* 1.FSE, pp. 1471–1493.

Jalbert, Nicholas and Westley Weimer (2008). "Automated duplicate detection for bug tracking systems". In: *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. IEEE, pp. 52–61.

Jeong, Gaeul, Sunghun Kim, and Thomas Zimmermann (2009). "Improving bug triage with bug tossing graphs". In: *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 111–120.

Johnson, Jeffrey N and Paul F Dubois (2003). "Issue tracking". In: *Computing in Science & Engineering* 5.6, pp. 71–77.

Kolesnyk, AS and NF Khairova (2022). "Justification for the use of cohen's kappa statistic in experimental studies of NLP and text mining". In: *Cybernetics and Systems Analysis* 58.2, pp. 280–288.

Lee, Jaekwon, Dongsun Kim, and Woosung Jung (2019). "Cost-Aware Clustering of Bug Reports by Using a Genetic Algorithm." In: *J. Inf. Sci. Eng.* 35.1, pp. 175–200.

Limsettho, Nachai et al. (2014). "Automatic unsupervised bug report categorization". In: *2014 6th international workshop on empirical software engineering in practice*. IEEE, pp. 7–12.

Luaphol, Bancha et al. (2018). "Assembling relevant bug report using the constraint-based k-means clustering". In: *2018 International conference on information technology (InCIT)*. IEEE, pp. 1–6.

Plein, Laura et al. (2024). "Automatic generation of test cases based on bug reports: a feasibility study with large language models". In: *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pp. 360–361.

Runeson, Per, Magnus Alexandersson, and Oskar Nyholm (2007). "Detection of duplicate defect reports using natural language processing". In: *29th International Conference on Software Engineering (ICSE'07)*. IEEE, pp. 499–510.

Sandusky, Robert J, Les Gasser, and Gabriel Ripoche (2004). "Bug report networks: Varieties, strategies, and impacts in a F/OSS development community". In: *26th International Conference on Software Engineering-W17S Workshop" International Workshop on Mining Software Repositories (MSR 2004)"*. IET, pp. 80–84.

Statista (2023). *Number of peak concurrent Steam users worldwide from 2015 to 2023*. URL: `https://www.statista.com/statistics/1330211/steam-peak-concurrent-players/` (visited on 01/20/2025).

— (2024a). *Number of games available in the Epic Games Store from 2019 to 2023*. URL: `https://www.statista.com/statistics/1234037/epic-games-store-games-available/` (visited on 01/21/2025).

— (2024b). *Number of games released on Steam worldwide from 2018 to 2023, by developer type*. URL: `https://www.statista.com/statistics/1411839/number-games-released-steam-developer-type/` (visited on 01/20/2025).

— (2025). *Number of games released on Steam worldwide from 2004 to 2024*. URL: `https://www.statista.com/statistics/552623/number-games-released-steam/` (visited on 01/20/2025).

Appendix

Appendix A: Cohen's Kappa Calculation Tables for GPT models & Human Reviewer Comparisons

Appendix A1: Cohen's Kappa Calculation Tables for Old Categories

Appendix A1.1: GPT 3.5 Turbo vs GPT 4

	GPT 3.5 Turbo	GPT 4	Expected Agreement
Frequency 1. Bugs & Glitches	90	93	0.1653333333
Frequency 2. Technical Issues	90	80	0.1422222222
Frequency 3. Server & Multiplayer	12	19	0.004503703704
Frequency 4. UI Problems	1	3	0.00005925925926
Frequency 5. Save Corruption	6	10	0.001185185185
Frequency 6. Support & Community	1	1	0.00001975308642
Frequency 7. Localization	2	2	0.00007901234568
Frequency 8. Content & Progression	6	2	0.000237037037
Frequency 9. Feedback & Suggestions	8	11	0.001738271605
Frequency 10. Optimization	9	4	0.0007111111111
Number of Comments	225		
Sum Expected Agreement	0.3160888889		
Agreements	170		
Observed Agreements	0.7555555556		
Cohen's Kappa	0.6425786327		

Table 7.1: Cohen's Kappa Calculation Table, Old Categories: GPT 3.5 Turbo vs GPT 4

Appendix A1.2: GPT 4 vs GPT 4 Turbo

	GPT 4	GPT 4 Turbo	Expected Agreement
Frequency 1. Bugs & Glitches	93	88	0.1602318114
Frequency 2. Technical Issues	80	74	0.1159057091
Frequency 3. Server & Multiplayer	19	27	0.01004385621
Frequency 4. UI Problems	3	5	0.0002936800063
Frequency 5. Save Corruption	10	12	0.00234944005
Frequency 6. Support & Community	1	0	0
Frequency 7. Localization	2	3	0.0001174720025
Frequency 8. Content & Progression	2	4	0.0001566293367
Frequency 9. Feedback & Suggestions	11	9	0.001938288041
Frequency 10. Optimization	4	4	0.0003132586733
Number of Comments	226		
Sum Expected Agreement	0.2913501449		
Agreements	186		
Observed Agreements	0.8230088496		
Cohen's Kappa	0.7502417461		

Table 7.2: Cohen's Kappa Calculation Table, Old Categories: GPT 4 vs GPT 4 Turbo

Appendix A1.3: GPT 4 Turbo vs GPT 3.5 Turbo

	GPT 4 Turbo	GPT 3.5 Turbo	Expected Agreement
Frequency 1. Bugs & Glitches	88	90	0.1564444444
Frequency 2. Technical Issues	74	90	0.1315555556
Frequency 3. Server & Multiplayer	27	12	0.0064
Frequency 4. UI Problems	5	1	0.0000987654321
Frequency 5. Save Corruption	12	6	0.001422222222
Frequency 6. Support & Community	0	1	0
Frequency 7. Localization	3	2	0.0001185185185
Frequency 8. Content & Progression	4	6	0.0004740740741
Frequency 9. Feedback & Suggestions	9	8	0.001422222222
Frequency 10. Optimization	4	9	0.0007111111111
Number of Comments	225		
Sum Expected Agreement	0.2986469136		
Agreements	159		
Observed Agreements	0.7066666667		
Cohen's Kappa	0.5817608292		

Table 7.3: Cohen's Kappa Calculation Table, Old Categories: GPT 4 Turbo vs GPT 3.5 Turbo

Appendix A1.4: GPT 3.5 Turbo vs Human Reviewer

	GPT 3.5 Turbo	Human	Expected Agreement
Frequency 1. Bugs & Glitches	90	83	0.1475555556
Frequency 2. Technical Issues	90	77	0.1368888889
Frequency 3. Server & Multiplayer	12	8	0.001896296296
Frequency 4. UI Problems	1	3	0.00005925925926
Frequency 5. Save Corruption	6	6	0.0007111111111
Frequency 6. Support & Community	1	10	0.0001975308642
Frequency 7. Localization	2	2	0.00007901234568
Frequency 8. Content & Progression	6	13	0.001540740741
Frequency 9. Feedback & Suggestions	8	12	0.001896296296
Frequency 10. Optimization	9	11	0.001955555556
Number of Comments	225		
Sum Expected Agreement	0.2927802469		
Agreements	159		
Observed Agreements	0.7066666667		
Cohen's Kappa	0.585230288		

Table 7.4: Cohen's Kappa Calculation Table, Old Categories: GPT 3.5 Turbo vs Human Reviewer

Appendix A1.5: GPT 4 vs Human Reviewer

	GPT 4	Human	Expected Agreement
Frequency 1. Bugs & Glitches	93	83	0.1524740741
Frequency 2. Technical Issues	80	77	0.1216790123
Frequency 3. Server & Multiplayer	19	8	0.003002469136
Frequency 4. UI Problems	3	3	0.0001777777778
Frequency 5. Save Corruption	10	6	0.001185185185
Frequency 6. Support & Community	1	10	0.0001975308642
Frequency 7. Localization	2	2	0.00007901234568
Frequency 8. Content & Progression	2	13	0.0005135802469
Frequency 9. Feedback & Suggestions	11	12	0.002607407407
Frequency 10. Optimization	4	11	0.0008691358025
Number of Comments	225		
Sum Expected Agreement	0.2827851852		
Agreements	160		
Observed Agreements	0.7111111111		
Cohen's Kappa	0.597207304		

Table 7.5: Cohen's Kappa Calculation Table, Old Categories: GPT 4 vs Human Reviewer

Appendix A1.6: GPT 4 Turbo vs Human Reviewer

	GPT 4 Turbo	Human	Expected Agreement
Frequency 1. Bugs & Glitches	88	83	0.1442765432
Frequency 2. Technical Issues	74	77	0.1125530864
Frequency 3. Server & Multiplayer	27	8	0.004266666667
Frequency 4. UI Problems	5	3	0.0002962962963
Frequency 5. Save Corruption	12	6	0.001422222222
Frequency 6. Support & Community	0	10	0
Frequency 7. Localization	3	2	0.0001185185185
Frequency 8. Content & Progression	4	13	0.001027160494
Frequency 9. Feedback & Suggestions	9	12	0.002133333333
Frequency 10. Optimization	4	11	0.0008691358025
Number of Comments	225		
Sum Expected Agreement	0.266962963		
Agreements	154		
Observed Agreements	0.6844444444		
Cohen's Kappa	0.5695230396		

Table 7.6: Cohen's Kappa Calculation Table, Old Categories: GPT 4 Turbo vs Human Reviewer

Appendix A2: Cohen's Kappa Calculation Tables for New Categories

Appendix A2.1: GPT 3.5 Turbo vs GPT 4

	GPT 3.5 Turbo	GPT 4	Expected Agreement
Frequency 1. Gameplay Bugs & Glitches	58	42	0.04854910714
Frequency 2. Technical Issues & Crashes	71	67	0.09480628189
Frequency 3. Game Progression Blocks	9	43	0.007712850765
Frequency 4. Graphics & Rendering Problems	21	21	0.0087890625
Frequency 5. Controls & Input Recognition	8	14	0.002232142857
Frequency 6. Multiplayer & Connectivity Issues	21	17	0.007114955357
Frequency 7. User Interface & Experience Problems	15	4	0.001195790816
Frequency 8. Audio issues	4	4	0.000318877551
Frequency 9. Installation & Update Problems	8	2	0.000318877551
Frequency 10. Achievements & Rewards Issues	9	10	0.001793686224
Number of Comments	224		
Sum Expected Agreement	0.1728316327		
Agreements	145		
Observed Agreements	0.6473214286		
Cohen's Kappa	0.5736314572		

Table 7.7: Cohen's Kappa Calculation Table, New Categories: GPT 3.5 Turbo vs GPT 4

Appendix A2.2: GPT 4 vs GPT 4 Turbo

	GPT 4	GPT 4 Turbo	Expected Agreement
Frequency 1. Gameplay Bugs & Glitches	42	32	0.02678571429
Frequency 2. Technical Issues & Crashes	67	72	0.09614158163
Frequency 3. Game Progression Blocks	43	43	0.03685028699
Frequency 4. Graphics & Rendering Problems	21	23	0.009626116071
Frequency 5. Controls & Input Recognition	14	10	0.002790178571
Frequency 6. Multiplayer & Connectivity Issues	17	19	0.006437340561
Frequency 7. User Interface & Experience Problems	4	6	0.0004783163265
Frequency 8. Audio issues	4	3	0.0002391581633
Frequency 9. Installation & Update Problems	2	5	0.0001992984694
Frequency 10. Achievements & Rewards Issues	10	11	0.002192283163
Number of Comments	224		
Sum Expected Agreement	0.1817402742		
Agreements	172		
Observed Agreements	0.7678571429		
Cohen's Kappa	0.7162968556		

Table 7.8: Cohen's Kappa Calculation Table, New Categories: GPT 4 vs GPT 4 Turbo

Appendix A2.3: GPT 4 Turbo vs GPT 3.5 Turbo

	GPT 4 Turbo	GPT 3.5 Turbo	Expected Agreement
Frequency 1. Gameplay Bugs & Glitches	32	58	0.03698979592
Frequency 2. Technical Issues & Crashes	72	71	0.1018813776
Frequency 3. Game Progression Blocks	43	9	0.007712850765
Frequency 4. Graphics & Rendering Problems	23	21	0.009626116071
Frequency 5. Controls & Input Recognition	10	8	0.001594387755
Frequency 6. Multiplayer & Connectivity Issues	19	21	0.007952008929
Frequency 7. User Interface & Experience Problems	6	15	0.001793686224
Frequency 8. Audio issues	3	4	0.0002391581633
Frequency 9. Installation & Update Problems	5	8	0.0007971938776
Frequency 10. Achievements & Rewards Issues	11	9	0.001973054847
Number of Comments	224		
Sum Expected Agreement	0.1705596301		
Agreements	146		
Observed Agreements	0.6517857143		
Cohen's Kappa	0.5801816522		

Table 7.9: Cohen's Kappa Calculation Table, New Categories: GPT 4 Turbo vs GPT 3.5 Turbo

Appendix A2.4: GPT 3.5 Turbo vs Human Reviewer

	GPT 3.5 Turbo	Human	Expected Agreement
Frequency 1. Gameplay Bugs & Glitches	58	63	0.07217777778
Frequency 2. Technical Issues & Crashes	71	90	0.1262222222
Frequency 3. Game Progression Blocks	9	16	0.002844444444
Frequency 4. Graphics & Rendering Problems	21	14	0.005807407407
Frequency 5. Controls & Input Recognition	8	9	0.001422222222
Frequency 6. Multiplayer & Connectivity Issues	21	13	0.005392592593
Frequency 7. User Interface & Experience Problems	15	5	0.001481481481
Frequency 8. Audio issues	4	6	0.0004740740741
Frequency 9. Installation & Update Problems	8	2	0.0003160493827
Frequency 10. Achievements & Rewards Issues	9	7	0.001244444444
Number of Comments	225		
Sum Expected Agreement	0.217382716		
Agreements	154		
Observed Agreements	0.6844444444		
Cohen's Kappa	0.5967945482		

Table 7.10: Cohen's Kappa Calculation Table, New Categories: GPT 3.5 Turbo vs Human

Appendix A2.5: GPT 4 vs Human Reviewer

	GPT 4	Human	Expected Agreement
Frequency 1. Gameplay Bugs & Glitches	42	63	0.05226666667
Frequency 2. Technical Issues & Crashes	67	90	0.1191111111
Frequency 3. Game Progression Blocks	43	16	0.01359012346
Frequency 4. Graphics & Rendering Problems	21	14	0.005807407407
Frequency 5. Controls & Input Recognition	14	9	0.002488888889
Frequency 6. Multiplayer & Connectivity Issues	17	13	0.004365432099
Frequency 7. User Interface & Experience Problems	4	5	0.0003950617284
Frequency 8. Audio issues	4	6	0.0004740740741
Frequency 9. Installation & Update Problems	2	2	0.00007901234568
Frequency 10. Achievements & Rewards Issues	10	7	0.001382716049
Number of Comments	225		
Sum Expected Agreement	0.1999604938		
Agreements	158		
Observed Agreements	0.7022222222		
Cohen's Kappa	0.6277961582		

Table 7.11: Cohen's Kappa Calculation Table, New Categories: GPT 4 vs Human

Appendix A2.6: GPT 4 Turbo vs Human Reviewer

	GPT 4 Turbo	Human	Expected Agreement
Frequency 1. Gameplay Bugs & Glitches	32	63	0.03982222222
Frequency 2. Technical Issues & Crashes	72	90	0.128
Frequency 3. Game Progression Blocks	43	16	0.01359012346
Frequency 4. Graphics & Rendering Problems	23	14	0.006360493827
Frequency 5. Controls & Input Recognition	10	9	0.001777777778
Frequency 6. Multiplayer & Connectivity Issues	19	13	0.004879012346
Frequency 7. User Interface & Experience Problems	6	5	0.0005925925926
Frequency 8. Audio issues	3	6	0.0003555555556
Frequency 9. Installation & Update Problems	5	2	0.0001975308642
Frequency 10. Achievements & Rewards Issues	11	7	0.001520987654
Number of Comments	225		
Sum Expected Agreement	0.1970962963		
Agreements	159		
Observed Agreements	0.7066666667		
Cohen's Kappa	0.6346593845		

Table 7.12: Cohen's Kappa Calculation Table, New Categories: GPT 4 Turbo vs Human

Appendix A3: Cohen's Kappa Calculation Tables for Technical Issues & Crashes Sub-Category

Appendix A3.1: Tables for Sub-Categorisation Models Comparisons using the GPT 3.5 Turbo "Technical Issues & Crashes" sample set

Appendix A3.1.1: GPT 3.5 Turbo vs GPT 4

	GPT 3.5 Turbo	GPT 4	Expected Agreement
1. Specific Mission/Level Crashes	2	9	0.003673469388
2. Device-Specific Issues	9	2	0.003673469388
3. Performance Issues (FPS Drops/Stutters)	1	8	0.001632653061
4. Crash On Launch/Startup Issues	25	17	0.08673469388
5. In-Game Freeze/Crash	29	13	0.07693877551
6. Asset Loading Errors	3	2	0.001224489796
7. Memory Management/Leaks	2	2	0.0008163265306
8. Game Integrity and Corruption Problems	0	4	0
9. Hardware Compatability Issues	0	8	0
10. Post-Update Problems	0	5	0
Number of Comments	70		
Sum Expected Agreement	0.1746938776		
Agreements	33		
Observed Agreements	0.4714285714		
Cohen's Kappa	0.3595450049		

Table 7.13: Cohen's Kappa Calculation Table, Sub-Categories: GPT 3.5 Turbo vs GPT 4

Appendix A3.1.2: GPT 4 vs GPT 4 Turbo

	GPT 4	GPT 4 Turbo	Expected Agreement
1. Specific Mission/Level Crashes	9	5	0.008926800238
2. Device-Specific Issues	2	3	0.001190240032
3. Performance Issues (FPS Drops/Stutters)	8	1	0.001586986709
4. Crash On Launch/Startup Issues	17	25	0.08430866891
5. In-Game Freeze/Crash	13	22	0.05673477485
6. Asset Loading Errors	2	4	0.001586986709
7. Memory Management/Leaks	2	3	0.001190240032
8. Game Integrity and Corruption Problems	4	3	0.002380480063
9. Hardware Compatability Issues	8	1	0.001586986709
10. Post-Update Problems	5	4	0.003967466772
Number of Comments	71		
Sum Expected Agreement	0.163459631		
Agreements	46		
Observed Agreements	0.6478873239		
Cohen's Kappa	0.5790846573		

Table 7.14: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 vs GPT 4 Turbo

Appendix A3.1.3: GPT 4 Turbo vs GPT 3.5 Turbo

	GPT 4 Turbo	GPT 3.5 Turbo	Expected Agreement
1. Specific Mission/Level Crashes	5	2	0.001983733386
2. Device-Specific Issues	3	9	0.005356080143
3. Performance Issues (FPS Drops/Stutters)	1	1	0.0001983733386
4. Crash On Launch/Startup Issues	25	25	0.1239833366
5. In-Game Freeze/Crash	22	29	0.12656219
6. Asset Loading Errors	4	3	0.002380480063
7. Memory Management/Leaks	3	2	0.001190240032
8. Game Integrity and Corruption Problems	3	0	0
9. Hardware Compatability Issues	1	0	0
10. Post-Update Problems	4	0	0
Number of Comments	71		
Sum Expected Agreement	0.2616544336		
Agreements	46		
Observed Agreements	0.6478873239		
Cohen's Kappa	0.5231058571		

Table 7.15: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 Turbo vs GPT 3.5 Turbo

Appendix A3.1.4: GPT 3.5 Turbo vs Human Reviewer

	GPT 3.5 Turbo	Human	Expected Agreement
1. Specific Mission/Level Crashes	2	4	0.001586986709
2. Device-Specific Issues	9	1	0.001785360048
3. Performance Issues (FPS Drops/Stutters)	1	2	0.0003967466772
4. Crash On Launch/Startup Issues	25	17	0.08430866891
5. In-Game Freeze/Crash	29	30	0.1725848046
6. Asset Loading Errors	3	3	0.001785360048
7. Memory Management/Leaks	2	2	0.0007934933545
8. Game Integrity and Corruption Problems	0	5	0
9. Hardware Compatability Issues	0	4	0
10. Post-Update Problems	0	3	0
Number of Comments	71		
Sum Expected Agreement	0.2632414204		
Agreements	41		
Observed Agreements	0.5774647887		
Cohen's Kappa	0.4264943457		

Table 7.16: Cohen's Kappa Calculation Table, Sub-Categories: GPT 3.5 Turbo vs Human

Appendix A3.1.5: GPT 4 vs Human Reviewer

	GPT 4	Human	Expected Agreement
1. Specific Mission/Level Crashes	9	4	0.00714144019
2. Device-Specific Issues	2	1	0.0003967466772
3. Performance Issues (FPS Drops/Stutters)	8	2	0.003173973418
4. Crash On Launch/Startup Issues	17	17	0.05732989486
5. In-Game Freeze/Crash	13	30	0.07736560206
6. Asset Loading Errors	2	3	0.001190240032
7. Memory Management/Leaks	2	2	0.0007934933545
8. Game Integrity and Corruption Problems	4	5	0.003967466772
9. Hardware Compatability Issues	8	4	0.006347946836
10. Post-Update Problems	5	3	0.002975600079
Number of Comments	71		
Sum Expected Agreement	0.1606824043		
Agreements	46		
Observed Agreements	0.6478873239		
Cohen's Kappa	0.5804774285		

Table 7.17: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 vs Human

Appendix A3.1.6: GPT 4 Turbo vs Human Reviewer

	GPT 4	Human	Expected Agreement
1. Specific Mission/Level Crashes	5	4	0.003967466772
2. Device-Specific Issues	3	1	0.0005951200159
3. Performance Issues (FPS Drops/Stutters)	1	2	0.0003967466772
4. Crash On Launch/Startup Issues	25	17	0.08430866891
5. In-Game Freeze/Crash	22	30	0.1309264035
6. Asset Loading Errors	4	3	0.002380480063
7. Memory Management/Leaks	3	2	0.001190240032
8. Game Integrity and Corruption Problems	3	5	0.002975600079
9. Hardware Compatability Issues	1	4	0.0007934933545
10. Post-Update Problems	4	3	0.002380480063
Number of Comments	71		
Sum Expected Agreement	0.2299146995		
Agreements	50		
Observed Agreements	0.7042253521		
Cohen's Kappa	0.6159196291		

Table 7.18: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 Turbo vs Human

Appendix A3.2: Tables for Sub-Categorisation Models Comparisons using the GPT 4 "Technical Issues & Crashes" sample set

Appendix A3.1.1: GPT 3.5 Turbo vs GPT 4

	GPT 3.5 Turbo	GPT 4	Expected Agreement
1. Specific Mission/Level Crashes	1	6	0.001377410468
2. Device-Specific Issues	6	2	0.002754820937
3. Performance Issues (FPS Drops/Stutters)	6	9	0.01239669421
4. Crash On Launch/Startup Issues	23	20	0.1056014692
5. In-Game Freeze/Crash	24	10	0.05509641873
6. Asset Loading Errors	0	1	0
7. Memory Management/Leaks	1	1	0.0002295684114
8. Game Integrity and Corruption Problems	0	2	0
9. Hardware Compatability Issues	3	7	0.004820936639
10. Post-Update Problems	2	8	0.003673094582
Number of Comments	66		
Sum Expected Agreement	0.1859504132		
Agreements	36		
Observed Agreements	0.5454545455		
Cohen's Kappa	0.4416243655		

Table 7.19: Cohen's Kappa Calculation Table, Sub-Categories: GPT 3.5 Turbo vs GPT 4

Appendix A3.1.2: GPT 4 vs GPT 4 Turbo

	GPT 4	GPT 4 Turbo	Expected Agreement
1. Specific Mission/Level Crashes	6	3	0.004132231405
2. Device-Specific Issues	2	2	0.0009182736455
3. Performance Issues (FPS Drops/Stutters)	9	5	0.01033057851
4. Crash On Launch/Startup Issues	20	24	0.1101928375
5. In-Game Freeze/Crash	10	18	0.04132231405
6. Asset Loading Errors	1	2	0.0004591368228
7. Memory Management/Leaks	1	2	0.0004591368228
8. Game Integrity and Corruption Problems	2	2	0.0009182736455
9. Hardware Compatability Issues	7	4	0.006427915519
10. Post-Update Problems	8	4	0.007346189164
Number of Comments	66		
Sum Expected Agreement	0.1825068871		
Agreements	49		
Observed Agreements	0.7424242424		
Cohen's Kappa	0.6849199663		

Table 7.20: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 vs GPT 4 Turbo

Appendix A3.1.3: GPT 4 Turbo vs GPT 3.5 Turbo

	GPT 4 Turbo	GPT 3.5 Turbo	Expected Agreement
1. Specific Mission/Level Crashes	3	1	0.0006887052342
2. Device-Specific Issues	2	6	0.002754820937
3. Performance Issues (FPS Drops/Stutters)	5	6	0.006887052342
4. Crash On Launch/Startup Issues	24	23	0.1267217631
5. In-Game Freeze/Crash	18	24	0.09917355372
6. Asset Loading Errors	2	0	0
7. Memory Management/Leaks	2	1	0.0004591368228
8. Game Integrity and Corruption Problems	2	0	0
9. Hardware Compatability Issues	4	3	0.002754820937
10. Post-Update Problems	4	2	0.001836547291
Number of Comments	66		
Sum Expected Agreement	0.2412764004		
Agreements	46		
Observed Agreements	0.696969697		
Cohen's Kappa	0.6006051437		

Table 7.21: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 Turbo vs GPT 3.5 Turbo

Appendix A3.1.4: GPT 3.5 Turbo vs Human Reviewer

	GPT 3.5 Turbo	Human	Expected Agreement
1. Specific Mission/Level Crashes	1	4	0.0008401596303
2. Device-Specific Issues	6	2	0.002520478891
3. Performance Issues (FPS Drops/Stutters)	6	6	0.007561436673
4. Crash On Launch/Startup Issues	23	18	0.08695652174
5. In-Game Freeze/Crash	24	26	0.1310649023
6. Asset Loading Errors	0	2	0
7. Memory Management/Leaks	1	3	0.0006301197227
8. Game Integrity and Corruption Problems	0	4	0
9. Hardware Compatability Issues	3	2	0.001260239445
10. Post-Update Problems	2	2	0.0008401596303
Number of Comments	69		
Sum Expected Agreement	0.2316740181		
Agreements	47		
Observed Agreements	0.6811594203		
Cohen's Kappa	0.5850191361		

Table 7.22: Cohen's Kappa Calculation Table, Sub-Categories: GPT 3.5 Turbo vs Human

Appendix A3.1.5: GPT 4 vs Human Reviewer

	GPT 4	Human	Expected Agreement
1. Specific Mission/Level Crashes	6	4	0.005040957782
2. Device-Specific Issues	2	2	0.0008401596303
3. Performance Issues (FPS Drops/Stutters)	9	6	0.01134215501
4. Crash On Launch/Startup Issues	20	18	0.07561436673
5. In-Game Freeze/Crash	10	26	0.05461037597
6. Asset Loading Errors	1	2	0.0004200798152
7. Memory Management/Leaks	1	3	0.0006301197227
8. Game Integrity and Corruption Problems	2	4	0.001680319261
9. Hardware Compatability Issues	7	2	0.002940558706
10. Post-Update Problems	8	2	0.003360638521
Number of Comments	69		
Sum Expected Agreement	0.1564797311		
Agreements	42		
Observed Agreements	0.6086956522		
Cohen's Kappa	0.5361055777		

Table 7.23: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 vs Human

Appendix A3.1.6: GPT 4 Turbo vs Human Reviewer

	GPT 4	Human	Expected Agreement
1. Specific Mission/Level Crashes	3	4	0.002520478891
2. Device-Specific Issues	2	2	0.0008401596303
3. Performance Issues (FPS Drops/Stutters)	5	6	0.006301197227
4. Crash On Launch/Startup Issues	24	18	0.09073724008
5. In-Game Freeze/Crash	18	26	0.09829867675
6. Asset Loading Errors	2	2	0.0008401596303
7. Memory Management/Leaks	2	3	0.001260239445
8. Game Integrity and Corruption Problems	2	4	0.001680319261
9. Hardware Compatability Issues	4	2	0.001680319261
10. Post-Update Problems	4	2	0.001680319261
Number of Comments	69		
Sum Expected Agreement	0.2058391094		
Agreements	53		
Observed Agreements	0.768115942		
Cohen's Kappa	0.708013753		

Table 7.24: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 Turbo vs Human

Appendix A3.3: Tables for Sub-Categorisation Models Comparisons using the GPT 4Turbo "Technical Issues & Crashes" sample set

Appendix A3.3.1: GPT 3.5 Turbo vs GPT 4

	GPT 3.5 Turbo	GPT 4	Expected Agreement
1. Specific Mission/Level Crashes	0	5	0
2. Device-Specific Issues	7	2	0.002777226741
3. Performance Issues (FPS Drops/Stutters)	3	9	0.005356080143
4. Crash On Launch/Startup Issues	22	21	0.09164848244
5. In-Game Freeze/Crash	27	11	0.05891688157
6. Asset Loading Errors	1	1	0.0001983733386
7. Memory Management/Leaks	2	2	0.0007934933545
8. Game Integrity and Corruption Problems	2	3	0.001190240032
9. Hardware Compatability Issues	6	9	0.01071216029
10. Post-Update Problems	1	8	0.001586986709
Number of Comments	71		
Sum Expected Agreement	0.1731799246		
Agreements	38		
Observed Agreements	0.5352112676		
Cohen's Kappa	0.4378598848		

Table 7.25: Cohen's Kappa Calculation Table, Sub-Categories: GPT 3.5 Turbo vs GPT 4

Appendix A3.3.2: GPT 4 vs GPT 4 Turbo

	GPT 4	GPT 4 Turbo	Expected Agreement
1. Specific Mission/Level Crashes	5	3	0.002975600079
2. Device-Specific Issues	2	3	0.001190240032
3. Performance Issues (FPS Drops/Stutters)	9	5	0.008926800238
4. Crash On Launch/Startup Issues	21	26	0.1083118429
5. In-Game Freeze/Crash	11	19	0.04146002777
6. Asset Loading Errors	1	2	0.0003967466772
7. Memory Management/Leaks	2	2	0.0007934933545
8. Game Integrity and Corruption Problems	3	3	0.001785360048
9. Hardware Compatability Issues	9	4	0.00714144019
10. Post-Update Problems	8	4	0.006347946836
Number of Comments	71		
Sum Expected Agreement	0.1793294981		
Agreements	47		
Observed Agreements	0.661971831		
Cohen's Kappa	0.5881073241		

Table 7.26: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 vs GPT 4 Turbo

Appendix A3.3.3: GPT 4 Turbo vs GPT 3.5 Turbo

	GPT 4 Turbo	GPT 3.5 Turbo	Expected Agreement
1. Specific Mission/Level Crashes	3	0	0
2. Device-Specific Issues	3	7	0.004165840111
3. Performance Issues (FPS Drops/Stutters)	5	3	0.002975600079
4. Crash On Launch/Startup Issues	26	22	0.1134695497
5. In-Game Freeze/Crash	19	27	0.1017655227
6. Asset Loading Errors	2	1	0.0003967466772
7. Memory Management/Leaks	2	2	0.0007934933545
8. Game Integrity and Corruption Problems	3	2	0.001190240032
9. Hardware Compatability Issues	4	6	0.004760960127
10. Post-Update Problems	4	1	0.0007934933545
Number of Comments	71		
Sum Expected Agreement	0.2303114461		
Agreements	50		
Observed Agreements	0.7042253521		
Cohen's Kappa	0.6157216495		

Table 7.27: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 Turbo vs GPT 3.5 Turbo

Appendix A3.3.4: GPT 3.5 Turbo vs Human Reviewer

	GPT 3.5 Turbo	Human	Expected Agreement
1. Specific Mission/Level Crashes	0	3	0
2. Device-Specific Issues	7	3	0.004050925926
3. Performance Issues (FPS Drops/Stutters)	3	5	0.002893518519
4. Crash On Launch/Startup Issues	22	18	0.07638888889
5. In-Game Freeze/Crash	27	27	0.140625
6. Asset Loading Errors	1	1	0.0001929012346
7. Memory Management/Leaks	2	2	0.0007716049383
8. Game Integrity and Corruption Problems	2	4	0.001543209877
9. Hardware Compatability Issues	6	6	0.006944444444
10. Post-Update Problems	1	3	0.0005787037037
Number of Comments	72		
Sum Expected Agreement	0.2339891975		
Agreements	51		
Observed Agreements	0.7083333333		
Cohen's Kappa	0.6192394863		

Table 7.28: Cohen's Kappa Calculation Table, Sub-Categories: GPT 3.5 Turbo vs Human

Appendix A3.3.5: GPT 4 vs Human Reviewer

	GPT 4	Human	Expected Agreement
1. Specific Mission/Level Crashes	5	3	0.002893518519
2. Device-Specific Issues	2	3	0.001157407407
3. Performance Issues (FPS Drops/Stutters)	9	5	0.008680555556
4. Crash On Launch/Startup Issues	21	18	0.07291666667
5. In-Game Freeze/Crash	11	27	0.05729166667
6. Asset Loading Errors	1	1	0.0001929012346
7. Memory Management/Leaks	2	2	0.0007716049383
8. Game Integrity and Corruption Problems	3	4	0.002314814815
9. Hardware Compatability Issues	9	6	0.01041666667
10. Post-Update Problems	8	3	0.00462962963
Number of Comments	72		
Sum Expected Agreement	0.1612654321		
Agreements	48		
Observed Agreements	0.6666666667		
Cohen's Kappa	0.602575897		

Table 7.29: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 vs Human

Appendix A3.3.6: GPT 4 Turbo vs Human Reviewer

	GPT 4	Human	Expected Agreement
1. Specific Mission/Level Crashes	3	3	0.001736111111
2. Device-Specific Issues	3	3	0.001736111111
3. Performance Issues (FPS Drops/Stutters)	5	5	0.004822530864
4. Crash On Launch/Startup Issues	26	18	0.09027777778
5. In-Game Freeze/Crash	19	27	0.09895833333
6. Asset Loading Errors	2	1	0.0003858024691
7. Memory Management/Leaks	2	2	0.0007716049383
8. Game Integrity and Corruption Problems	3	4	0.002314814815
9. Hardware Compatability Issues	4	6	0.00462962963
10. Post-Update Problems	4	3	0.002314814815
Number of Comments	72		
Sum Expected Agreement	0.2079475309		
Agreements	56		
Observed Agreements	0.7777777778		
Cohen's Kappa	0.7194349732		

Table 7.30: Cohen's Kappa Calculation Table, Sub-Categories: GPT 4 Turbo vs Human

Appendix A1: Cohen's Kappa Calculation Tables for Old Categories **Appendix A4.1: New GPT 3.5 Turbo vs GPT 3.5 Turbo**

	New GPT 3.5 Turbo	Old GPT 3.5 Turbo	Expected Agreement
1. Gameplay Bugs & Glitches.	57	58	0.06588807398
2. Technical Issues & Crashes	65	71	0.09197624362
3. Game Progression Blocks	37	9	0.006636639031
4. Graphics & Rendering Problems	14	21	0.005859375
5. Controls & Input Recognition	7	8	0.001116071429
6. Multiplayer & Connectivity Issues	15	21	0.006277901786
7. User Interface & Experience Problems	3	15	0.0008968431122
8. Audio Issues	5	4	0.0003985969388
9. Installation & Update Problems	12	8	0.001913265306
10. Achievements & Rewards Issues	8	9	0.00143494898
Number of Comments	224		
Sum Expected Agreement	0.1823979592		
Agreements	146		
Observed Agreements	0.6517857143		
Cohen's Kappa	0.5741029641		

Table 7.31: Cohen's Kappa Calculation Table, Recategorisation Test: New GPT 3.5 Turbo vs Old GPT 3.5 Turbo

Appendix A4.2: New GPT 3.5 Turbo vs GPT 4

	New GPT 3.5 Turbo	GPT 4	Expected Agreement
1. Gameplay Bugs & Glitches.	57	42	0.04771205357
2. Technical Issues & Crashes	65	67	0.08679448342
3. Game Progression Blocks	37	43	0.03170838648
4. Graphics & Rendering Problems	14	21	0.005859375
5. Controls & Input Recognition	7	14	0.001953125
6. Multiplayer & Connectivity Issues	15	17	0.005082110969
7. User Interface & Experience Problems	3	4	0.0002391581633
8. Audio Issues	5	4	0.0003985969388
9. Installation & Update Problems	12	2	0.0004783163265
10. Achievements & Rewards Issues	8	10	0.001594387755
Number of Comments	224		
Sum Expected Agreement	0.1818199936		
Agreements	154		
Observed Agreements	0.6875		
Cohen's Kappa	0.6180547098		

Table 7.32: Cohen's Kappa Calculation Table, Recategorisation Test: New GPT 3.5 Turbo vs GPT 4

Appendix A4.3: New GPT 3.5 Turbo vs GPT 4 Turbo

	New GPT 3.5 Turbo	GPT 4 Turbo	Expected Agreement
1. Gameplay Bugs & Glitches.	57	32	0.03635204082
2. Technical Issues & Crashes	65	72	0.09327168367
3. Game Progression Blocks	37	43	0.03170838648
4. Graphics & Rendering Problems	14	23	0.006417410714
5. Controls & Input Recognition	7	10	0.001395089286
6. Multiplayer & Connectivity Issues	15	19	0.005680006378
7. User Interface & Experience Problems	3	6	0.0003587372449
8. Audio Issues	5	3	0.0002989477041
9. Installation & Update Problems	12	5	0.001195790816
10. Achievements & Rewards Issues	8	11	0.001753826531
Number of Comments	224		
Sum Expected Agreement	0.1784319196		
Agreements	148		
Observed Agreements	0.6607142857		
Cohen's Kappa	0.5870266599		

Table 7.33: Cohen's Kappa Calculation Table, Recategorisation Test: New GPT 3.5 Turbo vs GPT 4 Turbo

Appendix A4.4: New GPT 3.5 Turbo vs Human Reviewer

	New GPT 3.5 Turbo	Human	Expected Agreement
1. Gameplay Bugs & Glitches.	57	63	0.07093333333
2. Technical Issues & Crashes	65	90	0.1155555556
3. Game Progression Blocks	37	16	0.01169382716
4. Graphics & Rendering Problems	14	14	0.003871604938
5. Controls & Input Recognition	7	9	0.001244444444
6. Multiplayer & Connectivity Issues	15	13	0.003851851852
7. User Interface & Experience Problems	3	5	0.0002962962963
8. Audio Issues	5	6	0.0005925925926
9. Installation & Update Problems	12	2	0.0004740740741
10. Achievements & Rewards Issues	8	7	0.00110617284
Number of Comments	225		
Sum Expected Agreement	0.2096197531		
Agreements	151		
Observed Agreements	0.6711111111		
Cohen's Kappa	0.5838852373		

Table 7.34: Cohen's Kappa Calculation Table, Recategorisation Test: New GPT 3.5 Turbo vs Human Reviewer

Appendix B: Statista Figures and Statistics for Steam & Epic Games Platform

Appendix B.1: Number of peak concurrent Steam users worldwide from 2015 to 2023

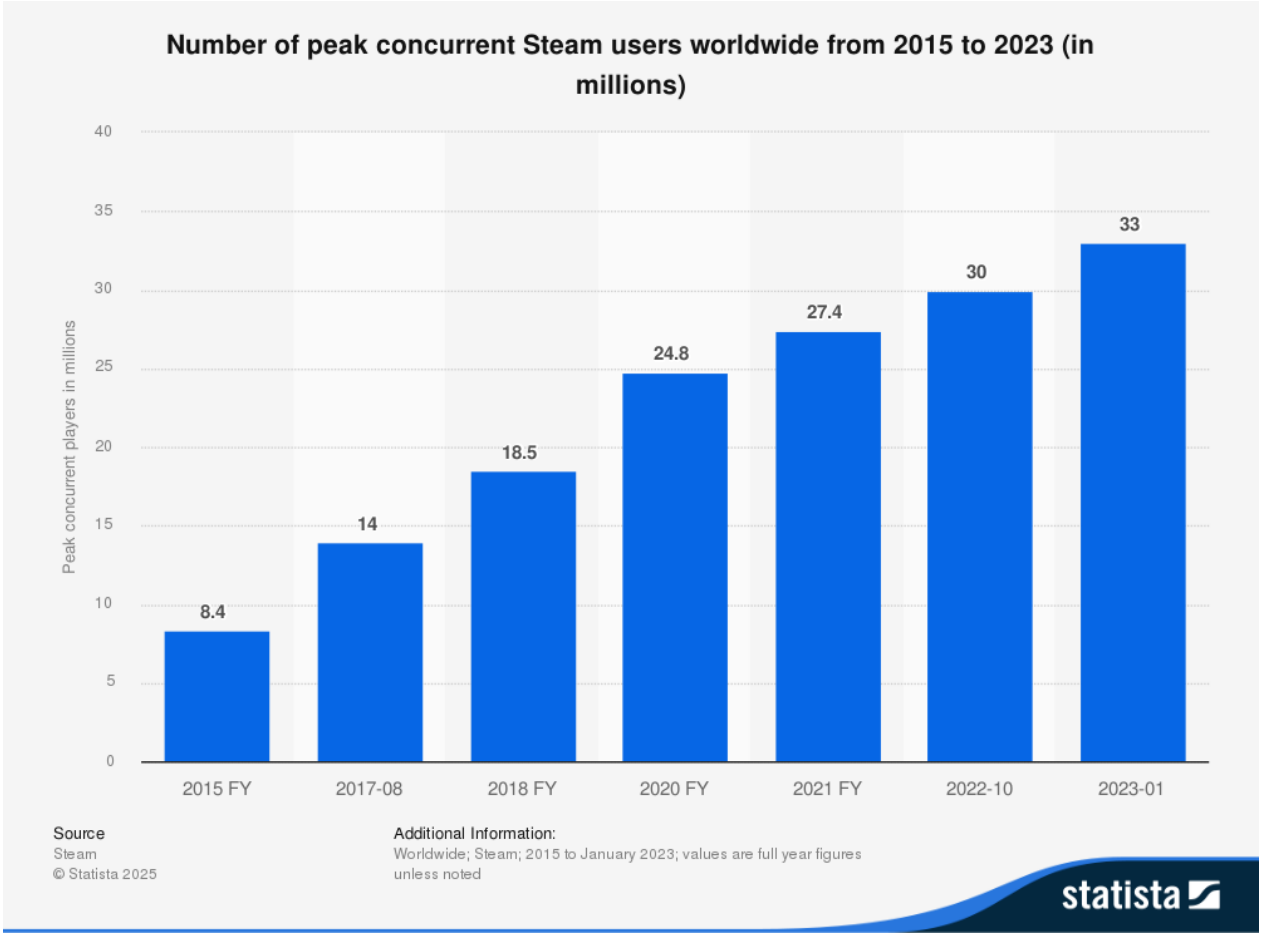


Figure 7.1: Statista: Number of peak concurrent Steam users worldwide from 2015 to 2023

Statista, 2023

Appendix B.2: Number of games released on Steam worldwide from 2004 to 2024. 2025

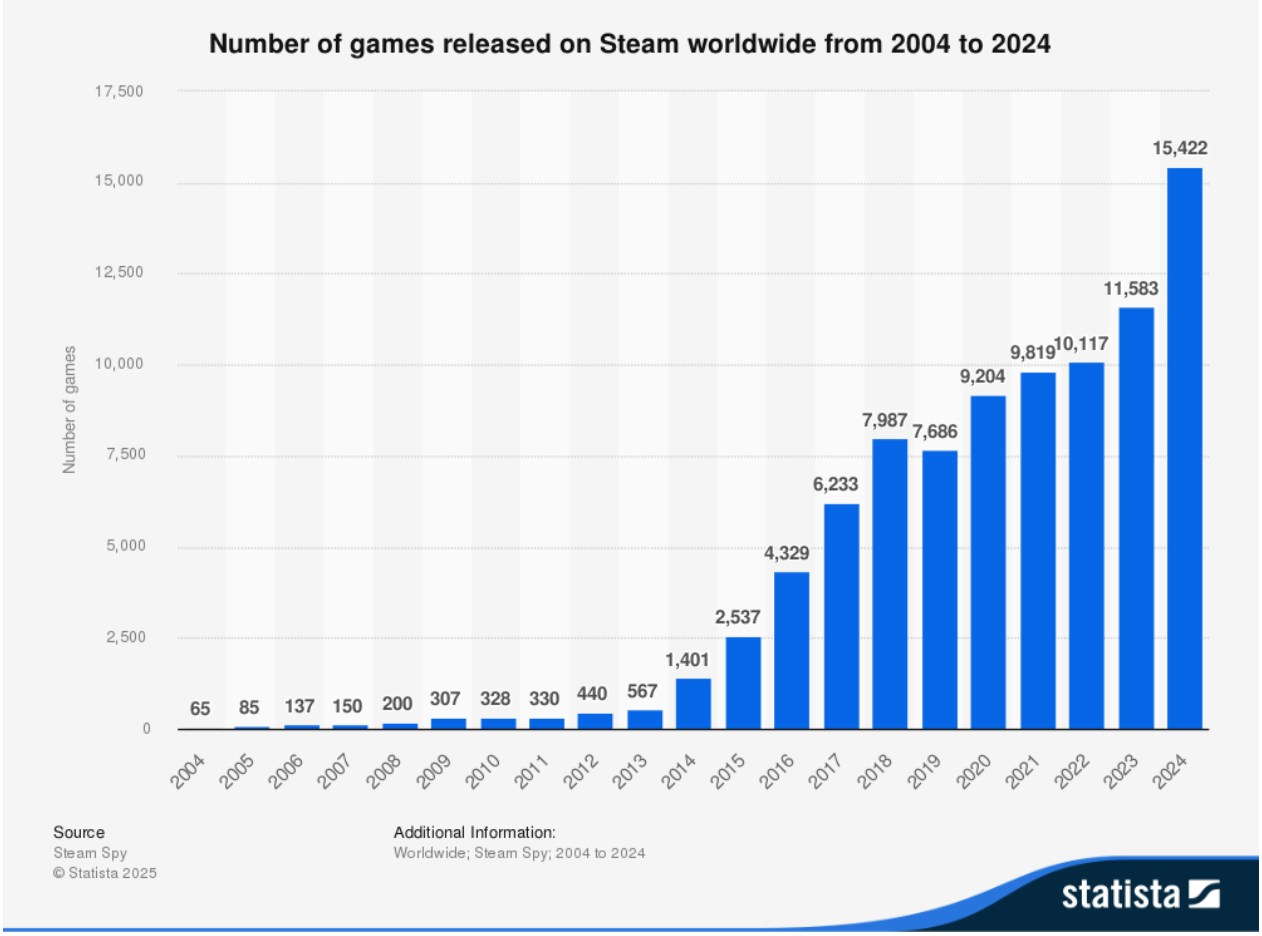


Figure 7.2: Statista: Number of games released on Steam worldwide from 2004 to 2024. 2025

Statista, 2025

Appendix B.3: Number of games released on Steam worldwide from 2018 to 2023, by developer type

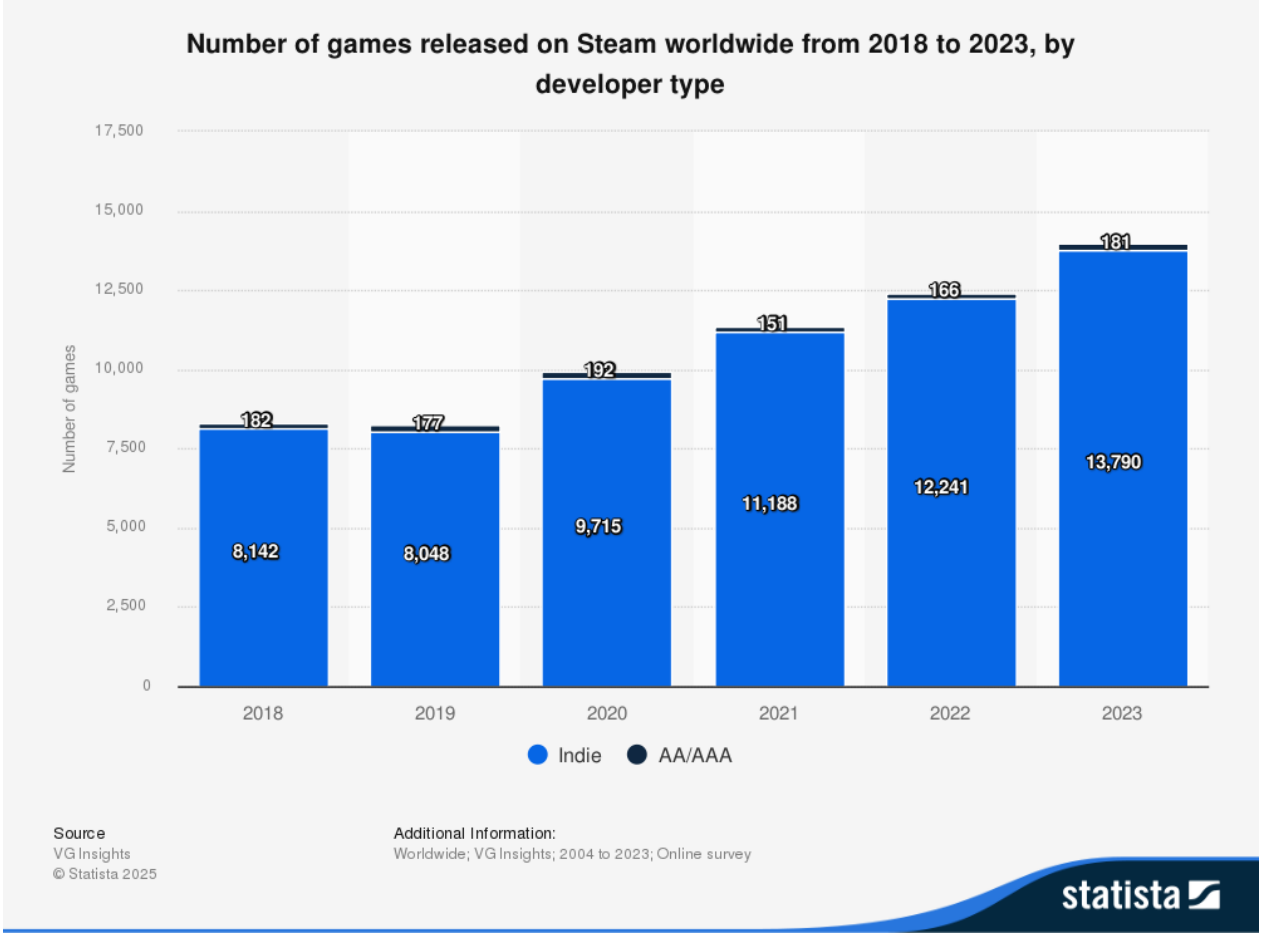


Figure 7.3: Statista: Number of games released on Steam worldwide from 2018 to 2023, by developer type

Statista, 2024b

Appendix B.4: Number of games available in the Epic Games Store from 2019 to 2023

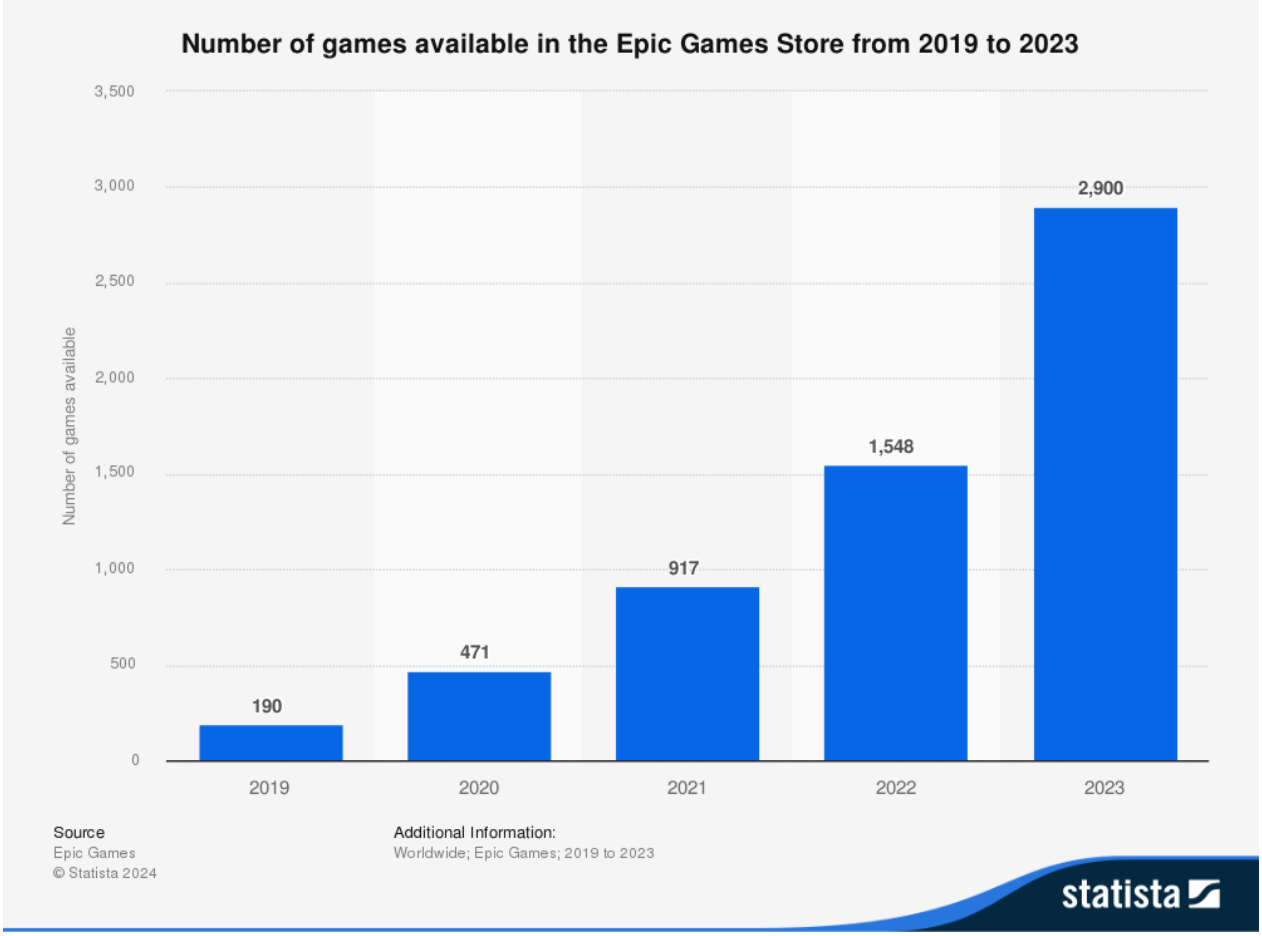


Figure 7.4: Statista: Number of games available in the Epic Games Store from 2019 to 2023

Statista, 2024a

Appendix C: Communication with Steam Support regarding access of Inactive posts.

Figure 7.5: Communication with Steam Support regarding access of Inactive posts.

Appendix D: Survey Results

Appendix D1: Primary Categories Standard Deviation

Appendix D1.1: Primary Categories Games Industry Pool Standard Deviation

	Not Useful At All	Slightly Useful	Very Useful	Extremely Useful	Average (1-4)	Standard Deviation
1. Gameplay Bugs & Glitches.	0	5	13	19	3.378378378	0.7109700777
2. Technical Issues & Crashes	0	2	10	25	3.621621622	0.5859319835
3. Game Progression Blocks	1	4	9	23	3.459459459	0.7916658657
4. Graphics & Rendering Problems	0	7	17	12	3.138888889	0.7130832029
5. Controls & Input Recognition	0	7	17	12	3.138888889	0.7130832029
6. Multiplayer & Connectivity Issues	0	9	13	14	3.138888889	0.787145962
7. User Interface & Experience Problems	1	10	14	12	3	0.8382736443
8. Audio Issues	1	13	15	7	2.777777778	0.7856742013
9. Installation & Update Problems	3	6	12	15	3.083333333	0.9537935952
10. Achievements & Rewards Issues	4	12	11	10	2.72972973	0.9767195135

Table 7.35: Primary Categories Games Industry Frequency of Ratings, Average Rating and Standard Deviation

Appendix D1.2: Primary Categories None-Games Industry Pool Standard Deviation

	Not Useful At All	Slightly Useful	Very Useful	Extremely Useful	Average (1-4)	Standard Deviation
1. Gameplay Bugs & Glitches.	2	5	23	32	3.370967742	0.7669350599
2. Technical Issues & Crashes	1	2	12	46	3.68852459	0.6151364931
3. Game Progression Blocks	0	7	25	27	3.338983051	0.6792360986
4. Graphics & Rendering Problems	0	13	32	17	3.064516129	0.6926100179
5. Controls & Input Recognition	0	15	29	18	3.048387097	0.7279538127
6. Multiplayer & Connectivity Issues	0	14	24	23	3.147540984	0.7647150026
7. User Interface & Experience Problems	1	21	21	19	2.935483871	0.8399494559
8. Audio Issues	2	26	22	12	2.709677419	0.8109551671
9. Installation & Update Problems	2	18	22	20	2.967741935	0.8607525203
10. Achievements & Rewards Issues	3	29	22	6	2.516666667	0.7414325473

Table 7.36: Primary Categories None-Games Industry Frequency of Ratings, Average Rating and Standard Deviation

Appendix D2: Sub-Categories Standard Deviation

Appendix D2.1: Sub-Categories Games Industry Pool Standard Deviation

	Not Useful At All	Slightly Useful	Very Useful	Extremely Useful	Average (1-4)	Standard Deviation
1. Gameplay Bugs & Glitches.	0	5	13	19	3.378378378	0.7109700777
2. Technical Issues & Crashes	0	2	10	25	3.621621622	0.5859319835
3. Game Progression Blocks	1	4	9	23	3.459459459	0.7916658657
4. Graphics & Rendering Problems	0	7	17	12	3.138888889	0.7130832029
5. Controls & Input Recognition	0	7	17	12	3.138888889	0.7130832029
6. Multiplayer & Connectivity Issues	0	9	13	14	3.138888889	0.787145962
7. User Interface & Experience Problems	1	10	14	12	3	0.8382736443
8. Audio Issues	1	13	15	7	2.777777778	0.7856742013
9. Installation & Update Problems	3	6	12	15	3.083333333	0.9537935952
10. Achievements & Rewards Issues	4	12	11	10	2.72972973	0.9767195135

Table 7.37: Sub-Categories Games Industry Frequency of Ratings, Average Rating and Standard Deviation

Appendix D2.2: Sub-Categories None-Games Industry Pool Standard Deviation

	Not Useful At All	Slightly Useful	Very Useful	Extremely Useful	Average (1-4)	Standard Deviation
1. Gameplay Bugs & Glitches.	2	5	23	32	3.370967742	0.7669350599
2. Technical Issues & Crashes	1	2	12	46	3.68852459	0.6151364931
3. Game Progression Blocks	0	7	25	27	3.338983051	0.6792360986
4. Graphics & Rendering Problems	0	13	32	17	3.064516129	0.6926100179
5. Controls & Input Recognition	0	15	29	18	3.048387097	0.7279538127
6. Multiplayer & Connectivity Issues	0	14	24	23	3.147540984	0.7647150026
7. User Interface & Experience Problems	1	21	21	19	2.935483871	0.8399494559
8. Audio Issues	2	26	22	12	2.709677419	0.8109551671
9. Installation & Update Problems	2	18	22	20	2.967741935	0.8607525203
10. Achievements & Rewards Issues	3	29	22	6	2.516666667	0.7414325473

Table 7.38: Sub-Categories None-Games Industry Frequency of Ratings, Average Rating and Standard Deviation

Appendix D3: Survey Qualitative Results

Appendix D3.1: Games Industry Pool Qualitative Data Breakdown-Primary Categories

Category	Development Impact	Too Generic/ Too High Level	Too Specific/ Too Low Level	High Importance	Low Importance	Player Perspective	Reporting/ Communication	Understanding	Other	Total
1. Gameplay Bugs & Glitches.	3	5	0	2	0	2	2	2	1	17
2. Technical Issues & Crashes	1	4	0	2	0	2	2	2	1	14
3. Game Progression Blocks	0	0	0	4	0	2	2	0	2	10
4. Graphics & Rendering Problems	0	3	0	2	1	0	0	0	4	10
5. Controls & Input Recognition	0	0	1	2	0	2	0	0	1	6
6. Multiplayer & Connectivity Issues	1	2	0	3	0	2	0	0	1	9
7. User Interface & Experience Problems	2	2	0	1	1	1	2	1	1	11
8. Audio Issues	0	0	0	3	1	2	0	0	1	7
9. Installation & Update Problems	1	2	0	3	0	1	1	0	0	8
10. Achievements & Rewards Issues	0	0	1	1	1	3	0	0	2	8

Table 7.39: Games Industry Pool Qualitative Feedback for Primary Categories Breakdown

Appendix D3.2: None-Games Industry Pool Qualitative Data Breakdown - Primary Categories

Category	Development Impact	Too Generic/ Too High Level	Too Specific/ Too Low Level	High Importance	Low Importance	Player Perspective	Reporting/ Communication	Understanding	Other	Total
1. Gameplay Bugs & Glitches.	1	8	0	8	0	1	3	0	0	21
2. Technical Issues & Crashes	1	5	0	7	0	5	1	0	0	19
3. Game Progression Blocks	1	1	0	6	0	3	2	1	1	15
4. Graphics & Rendering Problems	0	1	0	4	1	1	2	0	2	11
5. Controls & Input Recognition	1	1	1	2	0	3	1	1	1	11
6. Multiplayer & Connectivity Issues	4	1	1	2	0	1	1	0	2	12
7. User Interface & Experience Problems	0	4	0	1	2	1	1	1	1	11
8. Audio Issues	1	2	0	2	2	0	2	0	0	9
9. Installation & Update Problems	1	1	0	5	0	2	1	0	0	10
10. Achievements & Rewards Issues	0	2	1	0	4	2	0	0	2	11

Table 7.40: None-Games Industry Pool Qualitative Feedback for Primary Categories Breakdown

Appendix D3.3: Games Industry Pool Qualitative Data Breakdown-Sub Categories

Category	Development Impact	Scope/Specificity	Importance	Player Perspective	Reporting/ Communication	Understanding	Other	Total
1. Specific Mission/Level Crashes	1	2	6	0	1	0	1	11
2. Device-Specific Issues	0	0	3	0	0	0	2	5
3. Performance Issues (FPS Drops/Stutters)	0	1	2	2	0	0	0	5
4. Crash On Launch/Startup Issues	0	0	3	0	0	0	0	3
5. In-Game Freeze/Crash	0	1	1	1	0	0	0	3
6. Asset Loading Errors	0	1	1	0	0	0	1	3
7. Memory Management/Leaks	0	1	1	1	0	1	0	4
8. Game Integrity and Corruption Problems	1	1	0	1	0	0	1	4
9. Hardware Compatability Issues	0	1	1	1	1	0	1	5
10. Post-Update Problems	0	4	0	0	0	2	0	6

Table 7.41: Games Industry Pool Qualitative Feedback for Sub-Categories Breakdown

Appendix D3.4: None-Games Industry Pool Qualitative Data Breakdown - Sub Categories

Category	Development Impact	Scope/Specificity	Importance	Player Perspective	Reporting/ Communication	Understanding	Other	Total
1. Specific Mission/Level Crashes	4	0	1	1	1	0	0	7
2. Device-Specific Issues	3	2	1	1	0	1	0	8
3. Performance Issues (FPS Drops/Stutters)	5	0	2	0	0	0	0	7
4. Crash On Launch/Startup Issues	1	1	4	0	0	0	1	7
5. In-Game Freeze/Crash	2	1	1	0	0	0	0	4
6. Asset Loading Errors	0	0	2	1	0	0	0	3
7. Memory Management/Leaks	0	0	5	0	2	0	1	8
8. Game Integrity and Corruption Problems	1	1	1	1	0	0	0	4
9. Hardware Compatability Issues	3	3	2	0	1	0	1	10
10. Post-Update Problems	2	0	2	0	0	0	0	4

Table 7.42: None-Games Industry Pool Qualitative Feedback for Sub-Categories Breakdown

Appendix D3.5: Combined Pool Qualitative Data Breakdown -Sub Categories

Category	Development Impact	Scope/Specificity	Importance	Player Perspective	Reporting/ Communication	Understanding	Other	Total
1. Specific Mission/Level Crashes	5	2	7	1	2	0	1	18
2. Device-Specific Issues	3	2	4	1	0	1	2	13
3. Performance Issues (FPS Drops/Stutters)	5	1	4	2	0	0	0	12
4. Crash On Launch/Startup Issues	1	1	7	0	0	0	1	10
5. In-Game Freeze/Crash	2	2	2	1	0	0	0	7
6. Asset Loading Errors	0	1	3	1	0	0	1	6
7. Memory Management/Leaks	0	1	6	1	2	1	1	12
8. Game Integrity and Corruption Problems	2	2	1	2	0	0	1	8
9. Hardware Compatability Issues	3	4	3	1	2	0	2	15
10. Post-Update Problems	2	4	2	0	0	2	0	10

Table 7.43: None-Games Industry Pool Qualitative Feedback for Sub-Categories Breakdown

Appendix E: Survey Questions

Appendix E.1: Screening Survey Questions

Figure 7.6: Screening Survey

Introduction

. **Hello! Thank you for taking this Survey!**

I'm Callum Hemingway, a MSc. By Research student at Lancaster University in the UK.

This is a recruitment survey to find participants for our primary survey regarding a tool that has been developed to automatically categorise unstructured and informal bug reports for games.

The recruitment survey will take approximately 2 minutes.

Your name, email address and other personal information is not being collected for this survey, however, your Prolific ID will be.

Please begin when you are ready!

Participant Information Sheet

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

What is the study about?

This study seeks to create and determine the value of a tool built to automatically categorise unstructured and informal bug reports for video games.

Why have I been invited?

You have been invited as I am currently trying to gauge the opinions of knowledgeable individuals regarding the categories that have been produced for this study.

I would be very grateful if you would agree to take part in this study.

What will I be asked to do if I take part?

You will be asked to complete a brief, fully anonymous survey. The survey contains some very basic demographic information questions as well as questions regarding the subject of the survey.

Do I have to take part?

No. It's completely up to you to decide whether or not you take part. Your participation is voluntary and you are free to withdraw at any time, without giving any reason.

What if I change my mind?

As explained above, you are free to withdraw at any time prior to submitting your completed survey, as the collected survey data is anonymised by the time it reaches me, I will not be able to pick out your data from the collected set once it is submitted.

Will my data be identifiable?

Your Prolific ID is the only identifiable information being collected, this is so I can match your responses to your Prolific information. Any other identifiable info such as names, email addresses, etcetera, will not be collected by this survey.

How will my data be stored?

Your data will be stored in cloud encrypted files (that is no-one other than me, the researcher will be able to access them) and on password-protected computers.

In accordance with University guidelines, I will keep the data collected for this survey for a minimum of ten years.

How will we use the information you have shared with us and what will happen to the results of the research study?

I will use the data you have shared with only in the following ways:

When writing up the findings from this study, I would like to reproduce some of the views and ideas you shared with me. When doing so, I will only use anonymised quotes.

What if I have a question or concern?

If you have any queries or if you are unhappy with anything that happens concerning your participation in

the study, please contact myself or my supervisors via the following details:

Callum Hemingway, Post-Graduate Researcher
c.p.hemingway@lancaster.ac.uk /
callumhemingway2002@outlook.com

Professor Tracy Hall, Chair in Software Engineering, Supervisor
tracy.hall@lancaster.ac.uk

Doctor Saad Ezzini, Lecturer in Computer Science, Supervisor
s.ezzini@lancaster.ac.uk

If you have any concerns or complaints that you wish to discuss with a person who is not directly involved in the research, you can also contact:

Professor Nigel Davies, Head of School of Computing and Communications:
n.a.davies@lancaster.ac.uk

.

Consent Form

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

By proceeding to the survey you confirm that:

- You have read the participant information sheet and understand what is expected of you within this study
- You understand that any responses/information you give will remain anonymous
- Your participation is voluntary
- You consent for the information you provide to be discussed with my supervisor at Lancaster University
- You consent that the data will be pooled and published and that if quotes are provided they could be published
- You consent to Lancaster University keeping the data collected from this survey for a minimum of ten years.

. By clicking **"Agreed"** I confirm that I am over the age of 18 and have read the attached Consent Form &

Participant Information Sheet and consent to this survey.

- ○ Agreed
- ○ Not Agreed, Exit Survey

Prolific ID

. What is your Prolific ID?
Please note that this response should auto-fill with the correct ID.

${e://Field/PROLIFIC_PID}

Screening Questionaire

Q1. Which of these game engines do you have experience working with (Select ALL engines that apply.)

- ☐ Unity
- ☐ Godot
- ☐ BlazeFx Visual Scripting Engine
- ☐ Unreal Engine
- ☐ Seed Engine
- ☐ Blender
- ☐ Visual Studio Code

- [ ] Other
- [ ] None of the above

Q2. What roles have you held in game development projects?

- [ ] Game Designer
- [ ] Storyboard Artist
- [ ] Programmer
- [ ] Tester
- [ ] Artist/Animator
- [ ] Other
- [ ] None of the above

Q3. Which of these programming languages have you used to develop a game? (The game itself, not things related to the game such as websites or launchers.)

- [ ] Blueprint Visual Scripting
- [ ] C++
- [ ] C#
- [ ] CSS
- [ ] Python
- [ ] Java
- [ ] JavaScript

- [ ] GDScript
- [ ] None of the above

Q4. Choose the answer that best fits the description of a compiler's function.

- ○ Refactoring code
- ○ Connecting to the network
- ○ Aggregating user data
- ○ I don't know
- ○ Translating code into executable instructions
- ○ Collecting user data

Q5. Choose the answer that best fits the definition of a recursive function

- ○ I don't know
- ○ A function that runs for an infinite time
- ○ A function that does not have a return value
- ○ A function that can be called from other functions
- ○ A function that calls itself
- ○ A function that does not require any inputs

. This function is regarding in the following 2 questions.

```
main{
    print(func("hello world"))
}

String func(String in){
    int x = len(in)
    String out = ""
    for(int i = x-1; i >= 0; i--){
        out.append(in[i])
    }
    return out
}
```

Q6. What is the parameter of the function?

- String Out
- String In
- I don't know
- int i = x-1; i >= 0; i--
- Outputting a String
- int x = len(in)

Q7. Please select the returned value of the pseudocode above:

- hello world

- ○ hello world 10
- ○ dlrow olleh
- ○ world hello
- ○ HELLO WORLD
- ○ I don't know
- ○ hello world hello world hello world hello world

Appendix E.2: Primary Survey Questions

Figure 7.7: Primary Survey

Introduction

. **Hello! Thank you for taking this Survey!**

I'm Callum Hemingway, a MSc. By Research student at Lancaster University in the UK.

This is a recruitment survey to find participants for our primary survey regarding a tool that has been developed to automatically categorise unstructured and informal bug reports for games.

The recruitment survey will take approximately 2 minutes.

Your name, email address and other personal information is not being collected for this survey, however, your Prolific ID will be.

Please begin when you are ready!

Participant Information Sheet

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

What is the study about?

This study seeks to create and determine the value of a tool built to automatically categorise unstructured and informal bug reports for video games.

Why have I been invited?

You have been invited as I am currently trying to gauge the opinions of knowledgeable individuals regarding the categories that have been produced for this study.

I would be very grateful if you would agree to take part in this study.

What will I be asked to do if I take part?

You will be asked to complete a brief, fully anonymous survey. The survey contains some very basic demographic information questions as well as questions regarding the subject of the survey.

Do I have to take part?

No. It's completely up to you to decide whether or not you take part. Your participation is voluntary and you are free to withdraw at any time, without giving any reason.

What if I change my mind?

As explained above, you are free to withdraw at any time prior to submitting your completed survey, as the collected survey data is anonymised by the time it reaches me, I will not be able to pick out your data from the collected set once it is submitted.

Will my data be identifiable?

Your Prolific ID is the only identifiable information being collected, this is so I can match your responses to your Prolific information. Any other identifiable info such as names, email addresses, etcetera, will not be collected by this survey.

How will my data be stored?

Your data will be stored in cloud encrypted files (that is no-one other than me, the researcher will be able to access them) and on password-protected computers.

In accordance with University guidelines, I will keep the data collected for this survey for a minimum of ten years.

How will we use the information you have shared with us and what will happen to the results of the research study?

I will use the data you have shared with only in the following ways:

When writing up the findings from this study, I would like to reproduce some of the views and ideas you shared with me. When doing so, I will only use anonymised quotes.

What if I have a question or concern?

If you have any queries or if you are unhappy with anything that happens concerning your participation in

the study, please contact myself or my supervisors via the following details:

Callum Hemingway, Post-Graduate Researcher
c.p.hemingway@lancaster.ac.uk /
callumhemingway2002@outlook.com

Professor Tracy Hall, Chair in Software Engineering, Supervisor
tracy.hall@lancaster.ac.uk

Doctor Saad Ezzini, Lecturer in Computer Science, Supervisor
s.ezzini@lancaster.ac.uk

If you have any concerns or complaints that you wish to discuss with a person who is not directly involved in the research, you can also contact:

Professor Nigel Davies, Head of School of Computing and Communications:
n.a.davies@lancaster.ac.uk

.

Consent Form

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

By proceeding to the survey you confirm that:

- You have read the participant information sheet and understand what is expected of you within this study
- You understand that any responses/information you give will remain anonymous
- Your participation is voluntary
- You consent for the information you provide to be discussed with my supervisor at Lancaster University
- You consent that the data will be pooled and published and that if quotes are provided they could be published
- You consent to Lancaster University keeping the data collected from this survey for a minimum of ten years.

. By clicking **"Agreed"** I confirm that I am over the age of 18 and have read the attached Consent Form &

Participant Information Sheet and consent to this survey.

- ○ Agreed
- ○ Not Agreed, Exit Survey

Prolific ID

. What is your Prolific ID?
Please note that this response should auto-fill with the correct ID.

${e://Field/PROLIFIC_PID}

Screening Questionaire

Q1. Which of these game engines do you have experience working with (Select ALL engines that apply.)

- ☐ Unity
- ☐ Godot
- ☐ BlazeFx Visual Scripting Engine
- ☐ Unreal Engine
- ☐ Seed Engine
- ☐ Blender
- ☐ Visual Studio Code

- [ ] Other
- [ ] None of the above

Q2. What roles have you held in game development projects?

- [ ] Game Designer
- [ ] Storyboard Artist
- [ ] Programmer
- [ ] Tester
- [ ] Artist/Animator
- [ ] Other
- [ ] None of the above

Q3. Which of these programming languages have you used to develop a game? (The game itself, not things related to the game such as websites or launchers.)

- [ ] Blueprint Visual Scripting
- [ ] C++
- [ ] C#
- [ ] CSS
- [ ] Python
- [ ] Java
- [ ] JavaScript

- [ ] GDScript
- [ ] None of the above

Q4. Choose the answer that best fits the description of a compiler's function.

- ○ Refactoring code
- ○ Connecting to the network
- ○ Aggregating user data
- ○ I don't know
- ○ Translating code into executable instructions
- ○ Collecting user data

Q5. Choose the answer that best fits the definition of a recursive function

- ○ I don't know
- ○ A function that runs for an infinite time
- ○ A function that does not have a return value
- ○ A function that can be called from other functions
- ○ A function that calls itself
- ○ A function that does not require any inputs

. This function is regarding in the following 2 questions.

```
main{
    print(func("hello world"))
}

String func(String in){
    int x = len(in)
    String out = ""
    for(int i = x-1; i >= 0; i--){
        out.append(in[i])
    }
    return out
}
```

Q6. What is the parameter of the function?

- String Out
- String In
- I don't know
- int i = x-1; i >= 0; i--
- Outputting a String
- int x = len(in)

Q7. Please select the returned value of the pseudocode above:

- hello world

- ○ hello world 10
- ○ dlrow olleh
- ○ world hello
- ○ HELLO WORLD
- ○ I don't know
- ○ hello world hello world hello world hello world

Introduction

. **Hello! Thank you for taking this Survey!**

I'm Callum Hemingway, a MSc. By Research student at Lancaster University in the UK. This survey is to gain thoughts and opinions regarding the output of an automatic tool that I've created to categorise informal, unstructured bug reports for video games, such as the ones found on Steam.

Our tool was created from a sample of approximately 4000 uncategorised bug reports found on the Steam discussion pages from a variety of games. Categories were generated automatically from a sample of these bug reports and were then used to categorise another sample of such reports. It is our goal to create a tool that will help developers improve their game by helping them understand and quantify the issues that players are having without having to do so themselves.

The survey will take approximately 15 minutes, you're only

required to answer 4 questions (excluding the consent agreement and Prolific ID intake), though please fill out as many as you can!

Your name, email address and other personal information is not being collected for this survey, however, your Prolific ID will be.

Please begin when you are ready!

.

Participant Information Sheet

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

What is the study about?

This study seeks to create and determine the value of a tool built to automatically categorise unstructured and informal bug reports for video games.

Why have I been invited?

You have been invited as I am currently trying to gauge the opinions of knowledgeable individuals regarding the categories that have been produced for this study.

I would be very grateful if you would agree to take part in this study.

What will I be asked to do if I take part?

You will be asked to complete a brief, fully anonymous survey. The survey contains some very basic demographic information questions as well as questions regarding the subject of the survey.

Do I have to take part?

No. It's completely up to you to decide whether or not you take part. Your participation is voluntary and you are free to withdraw at any time, without giving any reason.

What if I change my mind?

As explained above, you are free to withdraw at any time prior to submitting your completed survey, as the collected survey data is anonymised by the time it reaches me, I will not be able to pick out your data from the collected set once it is submitted.

Will my data be identifiable?

Your Prolific ID is the only identifiable information being collected, this is so I can match your responses to your Prolific information. Any other identifiable info such as names, email addresses, etcetera, will not be collected by this survey.

How will my data be stored?

Your data will be stored in cloud encrypted files (that is no-one other than me, the researcher will be able to access them) and on password-protected computers.

In accordance with University guidelines, I will keep the data collected for this survey for a minimum of ten years.

How will we use the information you have shared with us and what will happen to the results of the research study?

I will use the data you have shared with only in the following ways:

When writing up the findings from this study, I would like to reproduce some of the views and ideas you shared with me. When doing so, I will only use anonymised quotes.

What if I have a question or concern?

If you have any queries or if you are unhappy with anything that happens concerning your participation in the study, please contact myself or my supervisors via the following details:

Callum Hemingway, Post-Graduate Researcher
c.p.hemingway@lancaster.ac.uk /
callumhemingway2002@outlook.com

Professor Tracy Hall, Chair in Software Engineering, Supervisor
tracy.hall@lancaster.ac.uk

Doctor Saad Ezzini, Lecturer in Computer Science, Supervisor
s.ezzini@lancaster.ac.uk

If you have any concerns or complaints that you wish to discuss with a person who is not directly involved in the

research, you can also contact:

Professor Nigel Davies, Head of School of Computing and Communications:
n.a.davies@lancaster.ac.uk

.

Participant Information Sheet

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

What is the study about?

This study seeks to create and determine the value of a tool built to automatically categorise unstructured and informal bug reports for video games.

Why have I been invited?

You have been invited as I am currently trying to gauge the opinions of knowledgeable individuals regarding the categories that have been produced for this study.

I would be very grateful if you would agree to take part in this study.

What will I be asked to do if I take part?

You will be asked to complete a brief, fully anonymous survey. The survey contains some very basic demographic information questions as well as questions regarding the subject of the survey.

Do I have to take part?

No. It's completely up to you to decide whether or not you take part. Your participation is voluntary and you are free to withdraw at any time, without giving any reason.

What if I change my mind?

As explained above, you are free to withdraw at any time prior to submitting your completed survey, as the collected survey data is anonymised by the time it reaches me, I will not be able to pick out your data from the collected set once it is submitted.

Will my data be identifiable?

Only your Prolific ID will be stored, any other information regarding your identify such names, email addresses and other such information will not be collected.

How will my data be stored?

Your data will be stored in cloud encrypted files (that is no-one other than me, the researcher will be able to access them) and on password-protected computers.

In accordance with University guidelines, I will keep the data collected for this survey for a minimum of ten years.

How will we use the information you have shared with us and what will happen to the results of the research study?

I will use the data you have shared with only in the following ways:

When writing up the findings from this study, I would like to reproduce some of the views and ideas you shared with me. When doing so, I will only use anonymised quotes as no personal data regarding you will be collected.

What if I have a question or concern?

If you have any queries or if you are unhappy with anything that happens concerning your participation in the study, please contact myself or my supervisors via the following details:

Callum Hemingway, Post-Graduate Researcher
c.p.hemingway@lancaster.ac.uk /
callumhemingway2002@outlook.com

Professor Tracy Hall, Chair in Software Engineering, Supervisor
tracy.hall@lancaster.ac.uk

Doctor Saad Ezzini, Lecturer in Computer Science, Supervisor
s.ezzini@lancaster.ac.uk

If you have any concerns or complaints that you wish to discuss with a person who is not directly involved in the research, you can also contact:

Professor Nigel Davies, Head of School of Computing and Communications:
n.a.davies@lancaster.ac.uk

Consent Form

For further information about how Lancaster University processes personal data for research purposes and your data rights please visit our webpage: https://www.lancaster.ac.uk/research/data-protection

Please take time to read the following information carefully before you decide whether or not you wish to take part.

By proceeding to the survey you confirm that:

- You have read the participant information sheet and understand what is expected of you within this study
- You understand that any responses/information you give will remain anonymous
- Your participation is voluntary
- You consent for the information you provide to be discussed with my supervisor at Lancaster University
- You consent that the data will be pooled and published and that if quotes are provided they could be published
- You consent to Lancaster University keeping the data collected from this survey for a minimum of ten years.

Consent Agreement. By clicking **"Agreed"** I confirm that I am over the age of 18 and have read the attached Consent Form & Participant Information Sheet and consent to this survey.

- ○ Agreed
- ○ Not Agreed, Exit Survey

Prolific ID

. What is your Prolific ID?
Please note that this response should auto-fill with the correct ID.

${e://Field/PROLIFIC_PID}

Demographic Information

. Demographic Information

These questions pertain to your demographic and your

general background information.

Q1. What is your age range?

- ○ 18-24
- ○ 25-34
- ○ 35-44
- ○ 45-54
- ○ 55-64
- ○ 65 or Over

Q2. What is your current standing in the context of Video Game Development?

- ☐ Self-Employed Game Developer (Indie Game Dev)
- ☐ Employed Game Developer (Employed at a Game Studio)
- ☐ Hobbyist Game Developer
- ☐ Student (Studying a course related to Game Development, i.e, Computer Science)
- ☐ _____ Other

Q3. Approximately how many hours have you invested in video game development across your life/career?

(Hobby/informal experience counts.)

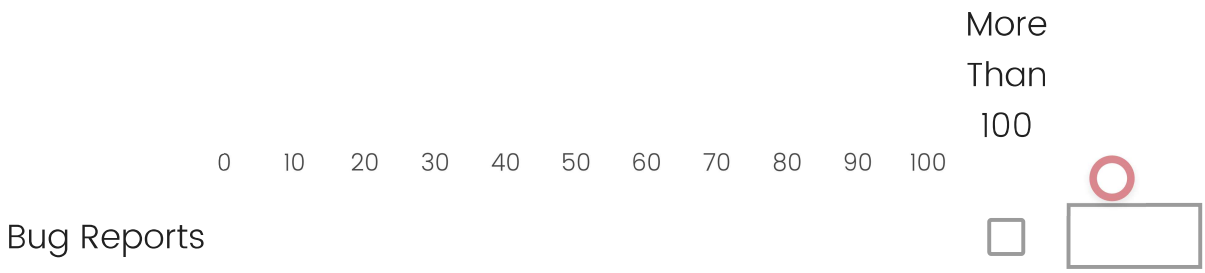
- ○ Less than 100 hours
- ○ 100-500 Hours
- ○ 500-1000 Hours
- ○ 1000-5000 Hours
- ○ 5000-10,000 Hours
- ○ More than 10,000 Hours

Q4. What responsibilities have you undertaken in game development projects?

- ☐ Game Designer
- ☐ Programmer
- ☐ Artist/Animator
- ☐ Sound Designer
- ☐ Project Manager
- ☐ Quality Assurance Tester
- ☐ [ ] Other

Q5. How many bug reports do you have experience acting upon across your entire life/career?

NOTE: A bug report in this context is any comment that details or expresses an issue or problem within the game.



Q6. How have you learned the majority of the skills you use as a Game Developer?

- ○ Self-Taught (Online tutorials, books, etc.)
- ○ Traditionally Taught (University, School, etc.)
- ○ Learned on the job.
- ○ _____ Other

Q7. How many **released** game development projects have you participated in?

_____

Main Categories

. **Rating Automatically Assigned Bug Report Categories (Main Categories)**

The following are automatically generated categories which were applied to a sample set of informal bug reports found on Steam. How useful do you find these categories?

Q8. How would you rate the usefulness of these main bug report categories?

	Not Useful at All	Slightly Useful	Very Useful	Extremely Useful	Unsure/I Don't Know
Gameplay Bugs & Glitches	○	○	○	○	○
Technical Issues & Crashes	○	○	○	○	○
Game Progression Blocks	○	○	○	○	○
Graphics & Rendering Problems	○	○	○	○	○
Controls & Input Recognition	○	○	○	○	○
Multiplayer & Connectivity Issues	○	○	○	○	○
User Interface & Experience Problems	○	○	○	○	○
Audio Issues	○	○	○	○	○

	Not Useful at All	Slightly Useful	Very Useful	Extremely Useful	Unsure/I Don't Know
Installation & Update Problems	○	○	○	○	○
Achievements & Rewards Issues	○	○	○	○	○

. **Further Comments on Main Categories**

The following questions are completely optional.

Q8.A. Do you have any further thoughts or comments about the category: "**Gameplay Bugs & Glitches?**"

Q8.B. Do you have any further thoughts or comments about the category: "**Technical Issues & Crashes?**"

Q8.C. Do you have any further thoughts or comments about the category: "**Game Progression Blocks?**"

Q8.D. Do you have any further thoughts or comments about the category: "**Graphics & Rendering Problems?**"

Q8.E. Do you have any further thoughts or comments about the category: "**Controls & Input Recognition?**"

Q8.F. Do you have any further thoughts or comments about the category: "**Multiplayer & Connectivity Issues?**"

Q8.G. Do you have any further thoughts or comments about the category: "**User Interface & Experience Problems?**"

Q8.H. Do you have any further thoughts or comments about the category: "**Audio Issues?**"

Q8.I. Do you have any further thoughts or comments about the category: "**Installation & Update Problems?**"

Q8.J. Do you have any further thoughts or comments about the category: **"Achievements & Rewards Issues?"**

Sub-Categories

. Rating Automatically Assigned Bug Report Categories (Sub-Categories)

The following are automatically generated sub-categories for the "Technical Issues & Crashes" category which were applied to a sample set of informal bug reports found on Steam. How useful do you find these categories?

Q9. How would you rate the usefulness of these **sub-categories** of the **"TECHNICAL ISSUES & CRASHES"** main category?

	Not Useful at All	Slightly Useful	Very Useful	Extremely Useful	Unsure/I Don't Know
Specific Mission/Level Crashes	○	○	○	○	○
Device-Specific Issues	○	○	○	○	○
Performance Issues (FPS drops/Stutters)	○	○	○	○	○
Crash on Launch/Startup Issues	○	○	○	○	○

	Not Useful at All	Slightly Useful	Very Useful	Extremely Useful	Unsure/I Don't Know
In-Game Freeze/Crash	○	○	○	○	○
Asset Loading Errors	○	○	○	○	○
Memory Management and Leaks	○	○	○	○	○
Game Integrity and Corruption Problems	○	○	○	○	○
Hardware Compatability Issues	○	○	○	○	○
Post-Update Problems	○	○	○	○	○

. **Further Comments on Sub-Categories**

The following questions are completely optional.

Q9.A. Do you have any further thoughts or comments about the sub-category: "**Specific Mission/Level Crashes?**"

Q9.B. Do you have any further thoughts or comments about the sub-category: "**Device-Specific Issues?**"

Q9.C. Do you have any further thoughts or comments about the sub-category: "**Performance Issues (FPS Drops/Stutters)?**"

Q9.D. Do you have any further thoughts or comments about the sub-category: "**Crash on Launch/Startup**

Issues?"

Q9.E. Do you have any further thoughts or comments about the sub-category: "**In-game Freeze/Crash?**"

Q9.F. Do you have any further thoughts or comments about the sub-category: **"Asset Loading Errors?"**

Q9.G. Do you have any further thoughts or comments about the sub-category: "**Memory Management and Leaks?**"

Q9.H. Do you have any further thoughts or comments about the sub-category: "**Game Integrity and Corruption Problems?**"

Q9.I. Do you have any further thoughts or comments about the sub-category: "**Hardware Compatability Issues?**"

Q9.J. Do you have any further thoughts or comments about the sub-category "**Post-Update Problems?**"

Final Thoughts

. **Final thoughts & Information**

Final information collection and thoughts on the categories and sub-categories displayed in this survey.

Q10. How would you rate the overall usefulness of the categories and sub-categories displayed in this survey.

	Not Useful	Slightly Useful	Very Useful	Extremely Useful	Unsure/I Don't Know
Main Categories	○	○	○	○	○
Sample of Sub-Categories	○	○	○	○	○

Q11. On a scale from 1 to five, how likely would you be to use a tool to automatically categorise your informal/unstructured bug reports?

Extremely unlikely	Somewhat unlikely	Neither likely nor unlikely	Somewhat likely	Extremely likely
1	2	3	4	5

Q12. What advice would you give to improve this tool/the categories produced.

Q13. Do you think there should be more or less bug report categories added to this tool?

- ○ More Categories
- ○ Less Categories
- ○ The number of categories should remain the same

Q14. Are the Bug Categories too HIGH level or too LOW level to use?

(High level being too vague, Low Level being too specific)

- ○ Too High Level
- ○ Too Low Level
- ○ Neither too high nor too low level

Q15. Did you find the categories to be self-explanatory and easy to understand?

- ○ No
- ○ Yes

Q16. From your experience which of the main categories do you think will be the MOST frequently assigned?

- ○ Gameplay Bugs & Glitches
- ○ Technical Issues & Crashes
- ○ Game Progression Blocks
- ○ Graphics & Rendering Problems
- ○ Controls & Input Recognition
- ○ Multiplayer & Connectivity Issues
- ○ User Interface & Experience Problems
- ○ Audio Issues
- ○ Installation & Update Problems
- ○ Achievements & Rewards Issues
- ○ Unsure/I Don't Know

Q17. From your experience which of the main categories do you think will be the LEAST frequently assigned?

- ○ Gameplay Bugs & Glitches
- ○ Technical Issues & Crashes
- ○ Game Progression Blocks
- ○ Graphics & Rendering Problems
- ○ Controls & Input Recognition
- ○ Multiplayer & Connectivity Issues
- ○ User Interface & Experience Problems
- ○ Audio Issues
- ○ Installation & Update Problems
- ○ Achievements & Rewards Issues
- ○ Unsure/I Don't Know

Q18. From your experience which of the Technical Issues & Crashes Sub-Categories do you think will be the MOST frequently assigned?

- ○ Specific Mission/Level Crashes
- ○ Device-Specific Issues
- ○ Performance Issues (FPS drops/Stutters)
- ○ Crash on Launch/Startup
- ○ In-game Freeze/Crash
- ○ Asset Loading Errors
- ○ Memory Management and Leaks
- ○ Game Integrity & Corruption Problems
- ○ Hardware Compatability Issues
- ○ Post-Update Problems
- ○ Unsure/I Don't Know

Q19. From your experience which of the Technical Issues & Crashes Sub-Categories do you think will be the LEAST frequently assigned?

- ○ Specific Mission/Level Crashes
- ○ Device-Specific Issues
- ○ Performance Issues (FPS drops/Stutters)
- ○ Crash on Launch/Startup
- ○ In-game Freeze/Crash
- ○ Asset Loading Errors
- ○ Memory Management and Leaks
- ○ Game Integrity & Corruption Problems

- ○ Hardware Compatability Issues
- ○ Post-Update Problems
- ○ Unsure/I Don't Know

Q20. How did you find this survey?

- ○ Social Media (Reddit, Discord, etc.)
- ○ Referred by a friend/colleague etc.
- ○ Invited by email
- ○ Through a survey recruitment service (I.e. Prolific)
- ○ [ ] Other